RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABOU BEKR BELKAID TLEMCEN
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

THÈSE
Présentée pour l'obtention du diplôme de
DOCTORAT
Spécialité : Informatique

Par
Mr MERZOUG Mohammed

La découverte et la sélection des services web

Thèse soutenue publiquement le 2 juin 2018 devant le jury :

- **Pr. CHIKH Azzedine**	Université de Tlemcen	Président
- **Pr. BENSLIMANE Sidi Mohammed**	Université de SBA	Examinateur
- **Dr. BENMAMAR Badr**	Université de Tlemcen	Examinateur
- **Pr. BABA HAMED Latifa**	Université d'Oran	Examinatrice
- **Pr. CHIKH Mohamed Amine**	Université de Tlemcen	Directeur
- **Pr. HUCHARD Marianne**	Université de Montpellier	Co-directrice
- **Dr. HADJILA Fethallah**	Université de Tlemcen	Invité

Année universitaire 2017-2018

Remerciements

Louange à Allah qui m'a donné patience et courage pour mener à bien ce travail de thèse malgré les difficultés rencontrées. Je tiens à remercier en premier lieu Mr Chikh Mohammed Amine, mon directeur de thèse pour ses encouragements, ses conseils et sa sympathie qui m'ont permis de mener à bien cette thèse.

J'aimerais également remercier mon co-directeur de thèse Mme HUCHARD Marianne, Professeur au laboratoire LIRMM à Montpellier pour ses encouragements et son écoute attentive. Je remercie également M. TIBERMACINE Chouki, Maitre de conférence au laboratoire LIRMM pour son aide précieuse, ses conseils, et sa sympathie.

Je remercie très vivement mon ami Monsieur HADJILA Fethallah, Maitre de conférence à la Faculté des Sciences de Tlemcen et membre du laboratoire LRIT qui m'a beaucoup aidé et rendu le déroulement de cette thèse agréable.

J'adresse mes sincères remerciements à Monsieur CHIKH Azzedine, Professeur à l'université de Tlemcen, qui m'a fait l'honneur de présider le jury de cette thèse.

J'exprime ma profonde reconnaissance à Madame BABA HAMED Latifa, Professeur à l'université d'Oran, Monsieur BENSLIMANE Sidi Mohamed, Professeur et directeur de l'Ecole Supérieure en Informatique de Sidi Bel Abbès, et Monsieur BENMAMMAR Badr, Maitre de conférence à l'Université de Tlemcen, pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant de l'examiner et d'en être rapporteurs.

Mes remerciements vont à toutes les personnes que j'ai rencontrées au cours de mon séjour de 11 mois au laboratoire LIRMM à Montpellier, pour leur convivialité et pour toutes les riches discussions qu'elles ont menées.

Je ne saurais oublier de remercier tous mes amis et collègues, particulièrement ceux du Département d'Informatique de l'université de Tlemcen. Je leur exprime ma profonde sympathie et je leur souhaite une bonne continuation.

Je remercie chaleureusement ma famille pour son soutien, son écoute et ses encouragements tout au long de cette thèse.

Je dédie cet humble travail
A mes très chers parents, que Dieu tout puissant les protège
A ma femme, mes enfants et toute ma famille qui sont toujours présents dans mes pensées
A mes amis ainsi que tous les gens qui m'ont aidé de près ou de loin à accomplir ce travail

Résumé

Les applications distribuées convergent rapidement vers l'adoption du paradigme orienté service (SOA). Les services web constituent le moyen le plus convenable pour mettre en œuvre l'architecture SOA. La prolifération rapide des services web disponibles sur le web ainsi que leur hétérogénéité rend très difficile leur découverte et leur sélection.

Il est par conséquent nécessaire d'automatiser les mécanismes de découverte et de sélection des services web appropriés afin de répondre à des besoins fonctionnels et non fonctionnels de l'utilisateur. Ces deux phases constituent les deux problématiques étudiées dans cette thèse. Pour résoudre la première problématique, à savoir la découverte, notre contribution consiste à proposer un algorithme d'agrégation de mesures de similarité qui se base sur une relation de dominance floue 'FDAA' (Fuzzy Dominance Aggregating Algorithm) pour trouver les services pertinents relativement à une requête donnée. Pour résoudre la problématique de sélection des services web, nous proposons trois contributions. La première contribution est basée sur la métaheuristique Harmony Search. La deuxième contribution est basée sur la sélection clonale qui s'inspire du système immunitaire biologique. Et la troisième contribution est une hybridation de la sélection clonale et l'algorithme d'optimisation par essaim particulaire. Les résultats obtenus en expérimentations sont très satisfaisants et encouragent les futurs travaux dans ce domaine de recherche.

Mots clés : Architecture orientée service, Découverte de services, Sélection de services, Fonctions de matching, Qualité de service, Optimisation combinatoire, Métaheuristiques.

Abstract

Nowadays, distributed applications are rapidly converging to the paradigme of service computing. Web services are increasingly standardized, extensive and powerful way to implement SOA architecture. The rapid proliferation of web services available on the web and their heterogeneity makes them very difficult to discover and select.

Hence the need to automate the mechanisms of discovery and selection of appropriate web services to meet the complex requirements of the user. These two phases constitute the two issues studied in this thesis.

To handle the first issue (i.e the service discovery) we propose an aggregation algorithm that fuses a set of similarity measures according to the fuzzy dominance relationship concept to find the relevant services for a user request. To handle the second issue (which is modeled as an optimisation problem) we propose three contributions. The first one leverages the harmony search meta-heuristic to get the near optimal solutions, the second one leverages the clonal selection that is inspired by the immune system. Lastly the third one leverages both the clonal selection and the particule swarm optimization in order to boost the overall performance. The results obtained in experiments are very satisfactory and encourage future work in this field of research.

Key words: Service Oriented Architecture, Service Discovery, Service Selection, Matching Functions, Quality of Service, Combinatorial Optimization, Metaheuristics.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

Sommaire

1.1 Contexte

Avec une évolution exponentielle dans le temps, les SI (Systèmes d'Informations) sont devenus plus complexes et plus hétérogènes en raison de la diversité des besoins, des exigences des clients, et de la grande masse d'information stockée et manipulée par les internautes. Les infrastructures publiques et privées du secteur informatique sont de plus en plus multiplateformes, multifournisseurs et distribuées à grande échelle. Dans une telle situation, nombreux sont les professionnels de l'informatique qui considèrent que l'interopérabilité est un aspect aussi important que la sécurité et la fiabilité pour la gestion de leurs SI et leurs environnements de fonctionnement.

Assurer l'interopérabilité est l'une des clés de succès de développement et de l'intégration des grands systèmes d'information. Cet objectif présente le noyau des travaux de recherches dédiés à l'amélioration des architectures, des plateformes et des technologies de développement des systèmes d'information depuis les méthodes cartésiennes jusqu'aux architectures orientées services, plus connues sous l'abréviation SOA (Service Oriented Architecture) [Josuttis, 2007]. Ces dernières présentent actuellement la vision architecturale des SI modernes.

Dans la littérature, nous pouvons trouver plusieurs définitions relatives à l'architecture orientée services. Dans [Josuttis, 2007], le modèle SOA est défini comme un paradigme permettant d'organiser et d'utiliser des savoir-faire distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoir-faire pour produire le résultat désiré avec des pré-conditions et des buts mesurables.

SOA est un style architectural qui permet de construire des solutions d'entreprises basées sur les services. Au début, sa mise en place a été basée sur les technologies et les middlewares conçus pour le modèle « objets distribués » comme CORBA, RMI et DCOM. Cependant, chacune de ces technologies propose sa propre infrastructure, ce qui impose une forte liaison entre les services fournis et leurs consommateurs.

Selon [Dodani, 2004], "L'architecture orientée services permet l'intégration d'applications et de ressources de manière flexible en : (1) représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, (2) permettant à un service d'échanger des informations structurées (messages, documents, "objets métier"), (3) coordonnant et en organisant les services afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement." La figure 1.1 illustre le modèle général de l'architecture orientée service. Elle montre plusieurs intervenants, et plusieurs types d'activités [Carey, 2008].

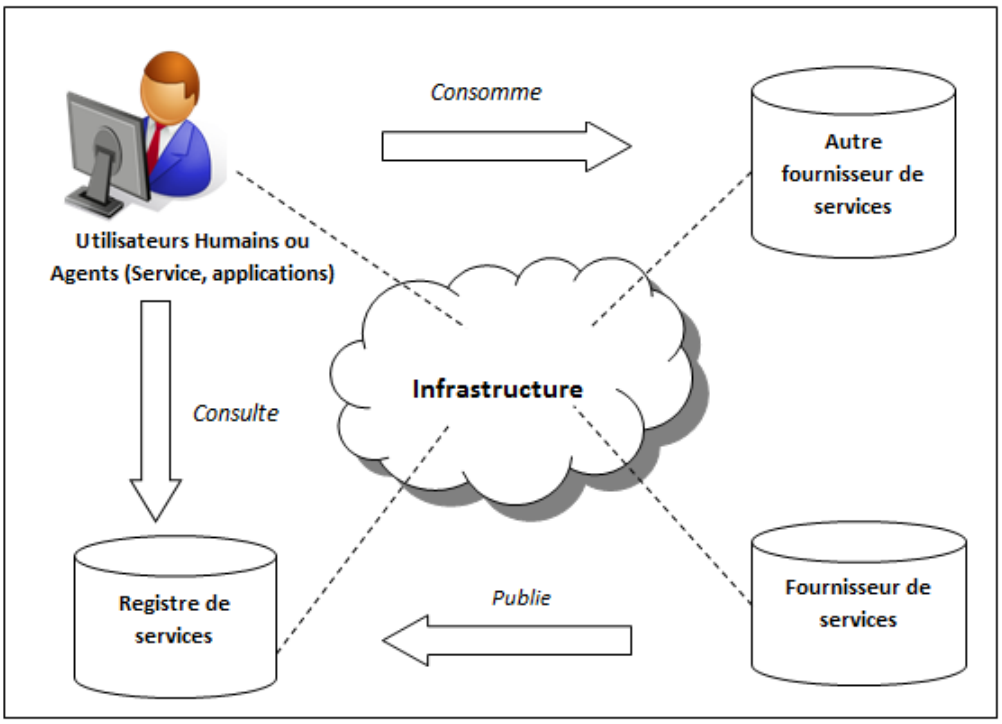

Figure *1.1:* *Le modèle SOA [Lahoud, 2010]*

En plus de la modélisation de son SI interne, l'entreprise d'aujourd'hui a besoin d'une ouverture vers son environnement économique pour accomplir des échanges inter-entreprises avec ses partenaires par le biais d'Internet (B2B : Business to Business), et répondre le plus rapidement possible aux besoins de ses clients (B2C : Business to Consumer).

Actuellement, les solutions EAI (pour Enterprise Applications Integration) sont orientées vers les processus métiers et les échanges inter-entreprises. Elles prennent en compte les nouveaux modèles économiques créés et promus par Internet et ses technologies (TCP/IP, SMTP, HTTP, FTP, XML, etc.). De plus, les progiciels de gestion intégrés de la deuxième génération apportent plusieurs modules organisés autour de la SOA. C'est dans ce contexte que les services web sont apparus pour mettre en place cette architecture.

Les services web constituent la technologie actuelle la mieux placée pour mettre en place l'architecture orientée services, ils sont munis d'un ensemble de standards qui facilitent leur mise en œuvre. Le concept de service web fait référence essentiellement à une application offerte par un fournisseur de service et accessible par des clients à travers des standards et des protocoles internet qui sont :

- SOAP, le protocole permettant de structurer les messages échangés entre les

services web.

- WSDL, la spécification qui définit les interfaces des services web.
- UDDI, une spécification de publication et de localisation de services web.

Les services web possèdent un cycle de vie qui ramifie le modèle d'interaction de l'architecture SOA et en particulier l'étape de consultation de service. Nous avons en l'occurrence les phases de :

- La découverte, qui permet de chercher et trouver les services adéquats pour une requête ;
- La composition, phase qui permet de sélectionner et combiner plusieurs services web pour répondre à une requête, car il est devenu de plus en plus rare de satisfaire cette requête par un seul service ;
- La sélection et réutilisation, phase qui permet de choisir un service pour chaque tâche dans la phase de composition ; dans ce cas, on parle de sélection de services web composants.

1.2 Objectifs

Bien que les applications à base de services web soient récemment proposées, elles connaissent un engouement et focalisent l'attention d'un bon nombre de professionnels dans le domaine des technologies de l'information et de la communication. Les atouts qui caractérisent les architectures orientées services c'est-à-dire l'utilisation de standards et l'indépendance vis-à-vis des technologies d'implémentation, la distribution des services sur des réseaux à large échelle ou encore le faible couplage entre les services favorisent la flexibilité et l'adaptabilité des systèmes adoptant ce modèle.

Pour assurer l'interopérabilité et l'échange inter-entreprises (B2B), nous devons concevoir des mécanismes pour découvrir et sélectionner les services web. Nous notons que l'augmentation considérable du nombre de services sur la toile et le caractère dynamique et volatile de ces entités compliquent de plus en plus ces tâches. Plus nous automatisons ces étapes, plus nous gagnons en termes d'interopérabilité [Benatallah et al., 2005a]. L'objectif de cette thèse est de présenter des approches efficaces pour automatiser ces deux phases (découverte et sélection) répondant aux besoins fonctionnels de l'utilisateur mais aussi à ses préférences et exigences particulières.

Notre idée est d'exploiter les techniques d'optimisation combinatoire et la décision multi-critères pour la découverte et la sélection des services web sémantiques, tout en proposant des mécanismes intelligents pour d'une part optimiser le temps d'exécution et, d'autre part, améliorer la qualité des résultats en termes de précision et rappel.

1.2.1 Problématique de la découverte des services web sémantiques

La découverte des services web représente un axe de recherche émergeant. Selon le W3C, la découverte de services est définie comme suit : "Web service discovery is the act of locating a machine-processable description of a Web service that may have been previously unknown and that meets certain functional criteria. It involves matching a set of criteria with a set of Web service descriptions. The goal is to find an appropriate Web service" [Group et al., 2004]. L'annuaire standard UDDI (Universal Definition

and Description Integration) permet à l'utilisateur la sélection des services web publiés répondant à sa requête, en fournissant un mécanisme de découverte fondée sur une recherche syntaxique par mots-clés. Toutefois, la solution offerte par l'UDDI est limitée par le fait qu'elle est incapable de découvrir des services dont les éléments de description ne sont pas syntaxiquement les mêmes que les mots clés spécifiés par l'utilisateur. La tendance actuelle de la communauté des chercheurs est d'exploiter les technologies du web sémantique, par exemple OWL (Ontology Web Langage) [Bechhofer, 2009], afin d'enrichir les services web de descriptions sémantiques. Ainsi, ces services sont, d'une part, aussi compréhensibles par les agents logiciels que par les utilisateurs humains, et d'autre part, ils permettent une interprétation correcte des informations envoyées et reçues dans le cadre de découverte, d'invocation et de composition [Berners-Lee et al., 2001].

Étant donné un besoin d'un utilisateur, qui peut être présenté sous la forme d'un ensemble de concepts d'entrées, de concepts de sorties, et éventuellement des descriptions informelles de la fonctionnalité du service, nous devons créer des mécanismes qui comparent ces besoins avec l'ensemble des services publiés. Ces mécanismes doivent gérer la sémantique. De plus, ils doivent avoir une bonne performance en termes de rappel, de précision et de temps d'exécution. Dans ce contexte, divers travaux de découverte de services web ont été alors réalisés. Ils se répartissent en trois catégories d'approches : les approches logiques, les approches non-logiques et les approches hybrides. Les approches de découverte non-logiques sont basées généralement sur le calcul de la similarité entre la description des services et la description des requêtes (à base d'ontologies). Par contre les approches de découverte logique sont basées essentiellement sur des approches déductives. Les approches hybrides quant à elles, utilisent une combinaison d'algorithmes logiques et non logiques. L'idée est de remédier à certaines limites de chacune de ces deux classes grâce à différentes combinaisons hybrides qui réussissent là où chacune de ces deux approches échoue.

1.2.2 Problématique de la sélection des services web sémantiques

En général, les requêtes de l'utilisateur (composées de concepts d'entrée et de concepts de sortie) ne sont pas satisfaites par un seul service web mais par une composition de services. Les services constituant ces différentes compositions se distinguent les uns des autres par leurs propriétés non fonctionnelles, comme les critères de Qualités de Service (réputation, fiabilité, durée d'exécution etc.). La sélection à base de Qualités de Services (QoS) consiste à choisir parmi les services web découverts de chaque tâche, ceux qui répondent au mieux aux exigences de l'utilisateur (sur la base des besoins non fonctionnels QoS). La sélection de services web basée sur QoS "QoS-aware web service composition" dépend de la spécification adoptée lors de la définition des critères QoS et du profil QoS du service web. Ce problème est une instance des problèmes combinatoires. C'est une version multidimensionnelle du problème du sac à dos connu pour être NP-difficile [Alrifai et al., 2012]. Par conséquent, toute solution exacte à ce problème nécessite un temps d'exécution exponentiel, et de ce fait, nous ne pouvons garantir les exigences temps réel des utilisateurs.

1.3 Contributions

Les contributions majeures de cette thèse sont :

- **Proposition d'une approche de découverte des services web à base d'agrégation floue de mesures de similarités**

  Pour pallier aux difficultés liées aux approches de découverte loqiques et non-logiques, nous proposons une approche [Mohammed et al., 2016] qui utilise quatre fonctions de matching (similarité) appartenant à la classe non-logique, et une fonction appartenant à la classe logique afin de bénéficier des avantages de chaque fonction. La fusion de ces mesures est réalisée à l'aide d'un algorithme d'agrégation basé sur une relation de fuzzy dominance 'FDAA' (Fuzzy Dominance Aggregating Algorithm). Afin de montrer l'efficacité de l'approche proposée, nous l'avons évaluée à l'aide des critères de rappel et de précision.

- **Proposition de trois approches de sélection des services web composites à base de QoS**

  1. La première approche [Merzoug et al., 2014] est basée sur l'algorithme Harmony search [Geem et al., 2001], qui a prouvé son efficacité dans la résolution des problèmes NP-hard comme le problème du voyageur de commerce. Nous avons proposé une fonction mono-objectif qui prend en considération les préférences de l'utilisateur en termes de QoS afin de sélectionner la meilleure composition.
  2. La deuxième approche est basée sur l'algorithme de sélection clonale (clonAlg) [De Castro and Von Zuben, 2002]. Nous avons adapté l'algorithme clonAlg au problème de la sélection des services web car il converge mieux vers l'optimum en introduisant itérativement de nouvelles solutions candidates qui élargissent l'espace de recherche (par rapport aux algorithmes génétiques qui peuvent stagner autour d'un optimum local). Notre algorithme utilise les propriétés non-fonctionnelles des services web afin de trouver une solution quasi-optimale.
  3. La troisième est basée sur un algorithme hybride (SC-PSO) de sélection clonale et l'optimisation par essaim particulaire (PSO) afin de combiner les avantages des deux algorithmes (la sélection clonale et PSO). Les résultats obtenus sont très satisfaisants.

1.4 Organisation du mémoire

Cette thèse est constituée de deux parties principales : la première est intitulée «État de l'art», elle introduit le contexte dans lequel se place cette thèse et montre un état de l'art des deux problématiques traitées à savoir la découverte et la sélection des services web. La seconde partie du document est intitulée « Contributions », elle présente les approches proposées pour les deux problématiques et présente aussi le prototype développé ainsi que l'ensemble des expérimentations effectuées.

1. Partie I : État de l'art. Cette partie inclut trois chapitres :
   - **Chapitre 2 : Technologies des Services Web.**

     Ce chapitre introduit les principes de la technologie des services Web en définissant le paradigme SOA, ainsi que les protocoles, les langages et les

modèles qui sont en relation avec le concept des services web traditionnels et sémantiques.

- **Chapitre 3 : La découverte des services web sémantiques** Ce chapitre est consacré à la définition de la problématique de découverte des services web. Il vise aussi à analyser les approches proposées dans la littérature en vue d'identifier des mécanismes qui constituent des points forts de ces approches. Une étude comparative ainsi qu'une synthèse sont présentées à la fin du chapitre, dans laquelle nous identifions notamment les limitations de chaque approche étudiée.
- **Chapitre 4 : La sélection des services web composés à base de QoS**

  Ce chapitre est consacré à l'analyse et à la définition de la problématique de sélection des services web dans une composition à base de paramètres de QoS. Une classification des différentes approches existantes relatives à l'optimisation du problème de sélection est présentée. Cette classification regroupe 4 catégories qui sont : les approches de résolution exactes, heuristiques, méta-heuristiques et les approches basées sur la dominance au sens de Pareto. Nous effectuons un survol des approches proposées dans la littérature.

2. Partie II : Contributions. Cette partie regroupe deux chapitres :
   - **Chapitre 5 : Approches proposées.**

     Ce chapitre présente les quatre contributions énoncées plus haut. Pour chacune d'entre elles :
     - nous présentons nos motivations argumentées par rapport aux travaux cités dans l'état de l'art,
     - nous exprimons formellement nos objectifs,
     - nous présentons les algorithmes proposés accompagnés, pour certains d'entre eux, d'exemples illustratifs.
   - **Chapitre 6 : Implémentation et expérimentations.**

     Ce chapitre présente le prototype implémentant l'environnement de découverte et de sélection de services web nommé ***DSS*** (Environnement de **D**écouverte et de **S**élection de **S**ervices). Ce dernier implémente toutes les approches proposées et a été utilisé dans le but de les valider. Cette partie expose aussi une série d'expérimentations que nous avons réalisées sur une base de données réelle **OWL-S TC.2**.

     Nous terminons cette thèse avec un chapitre conclusion et perspectives.
   - **Chapitre 7 : Conclusion et Perspectives.**

     Ce chapitre présente une synthèse des principales idées de nos propositions. A l'occasion, nous reprenons certaines réflexions dans le but de mettre en avant les principales contributions et d'identifier les questions ouvertes et les perspectives de ce travail.

Chapitre 2

Technologies des Services Web

Sommaire

2.1 Introduction

L'évolution des réseaux informatiques combinée à celle des systèmes informatiques a permis la mise en œuvre de systèmes distribués de plus en plus performants. Les systèmes informatiques distribués recouvrent un champ très vaste dont font partie les grilles de calcul, le stockage d'informations dans des réseaux de pairs, le déploiement d'applications web, etc.

L'évolution de plateformes à composants distribués (RMI, DCOM, CORBA,...) a donné naissance à un nouveau mode de développement logiciel appelé SOC.

SOC, acronyme de Service Oriented Computing, désigne ce paradigme de programmation émergeant pour le développement de systèmes d'information distribués, dans lequel les concepts de distribution, d'ouverture, de messagerie asynchrone et de faible couplage tiennent un rôle primordial [Georgakopoulos et al., 2009], [Huhns and Singh, 2005]. L'avantage de SOC est qu'il permet de créer des systèmes d'information en agrégeant des services individuels. En effet, il permet de créer des systèmes

d'information inter- et intra-entreprises de manière relativement aisée en "serviçant" les systèmes existants. Il assure aussi l'interopérabilité et la transparence entre les applications distribuées.

Le modèle orienté service est aussi soutenu par une architecture répondant aux propriétés citées précédemment. Une telle architecture est appelée SOA ou architecture orientée services. SOA possède plusieurs implémentations ; dans notre travail nous nous intéressons uniquement aux services web, qui représentent la concrétisation la plus répandue sur le réseau internet.

Dans ce chapitre nous présentons le concept de Service web ainsi que son cycle de vie et les standards associés. Ensuite nous présenterons le concept de service web sémantique et les principaux modèles sémantiques.

2.2 Service web

Un service web est un composant logiciel indépendant, qui rassemble un ensemble de standards web et de langages dérivés du langage XML. Ces standards permettent sa publication, sa découverte et enfin son invocation à distance. Il expose des fonctionnalités via une interface publique et permet de communiquer avec les autres applications et services web en utilisant des messages XML.

2.2.1 Définition

Selon IBM [1]

Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer.

Selon W3C [2]

Un service web est un composant logiciel identifié par une URI, et conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format exploitable par la machine (WSDL). D'autres systèmes interagissent avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant HTTP (HyperText Transfer Protocol) avec une sérialisation XML en même temps que d'autres normes du Web.

2.2.2 Architecture des services Web

Pour permettre l'interopérabilité entre des applications distantes, indépendamment des systèmes d'exploitation et des langages de programmation, les services web sont décrits avec différentes couches. Nous mettons en relief dans cette section deux types d'architectures. La première est l'architecture dite de référence et traditionnellement utilisée pour les services web isolés. La seconde architecture est plus complète

1. http ://www-106.ibm.com/developerworks/webservices/.
2. www.w3c.org

et est fréquemment utilisée lors de la composition de services web. Elle est appelée architecture étendue ou encore en pile.

2.2.2.1 Architecture de référence

Cette architecture vise trois objectifs importants [Kreger et al., 2001] :

1. Identification des composants fonctionnels,
2. Définition des relations entre ces composants et
3. Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

L'architecture de référence des services web (Figure 2.1.) s'articule autour des trois rôles suivants :

- **Le fournisseur de service** : correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.
- **Le client** : correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un service web.
- **L'annuaire des services** : correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de recherche et de liens (Bind) d'opérations. Pour garantir l'interopérabilité de ces trois opérations, des propositions de standards ont été élaborées pour chaque type d'interaction. Nous citons, notamment les standards émergeants suivants :

- **SOAP** définit un protocole de transmission de messages basé sur XML.
- **WSDL** introduit une grammaire commune pour la description des services.
- **UDDI** fournit l'infrastructure de base pour la publication et la découverte des services.

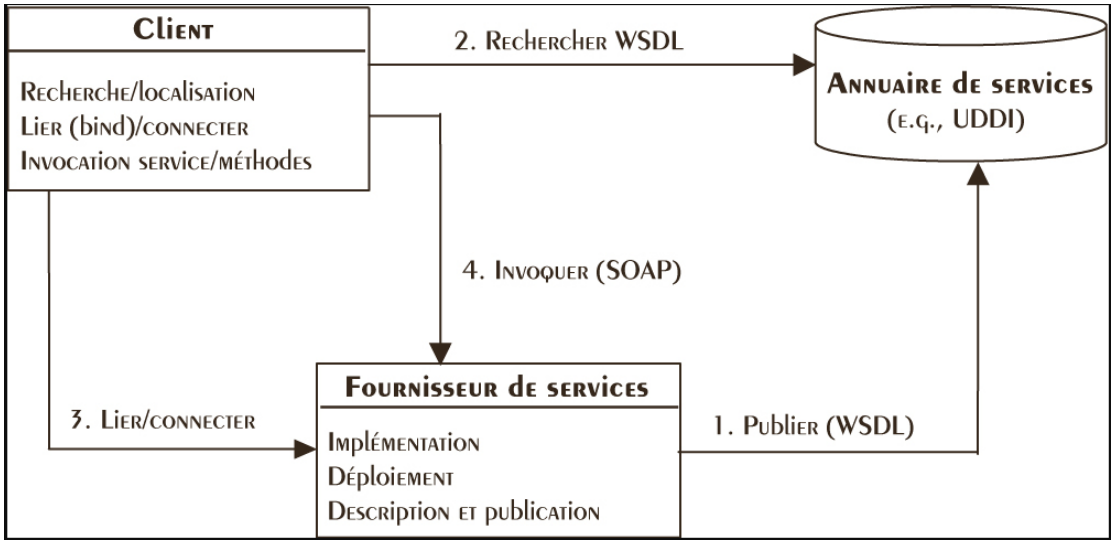

Figure *2.1:* *Architecture de référence des services web*

Cependant, cette infrastructure n'est pas suffisante pour permettre une utilisation effective des services web dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples. Par conséquent nous introduisons dans la suite une nouvelle architecture plus complète et plus adaptée au contexte du web.

2.2.2.2 Architecture étendue

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des services web. La Figure 2.2 décrit un exemple d'une telle pile. La pile est constituée de plusieurs couches. Chaque couche s'appuie sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

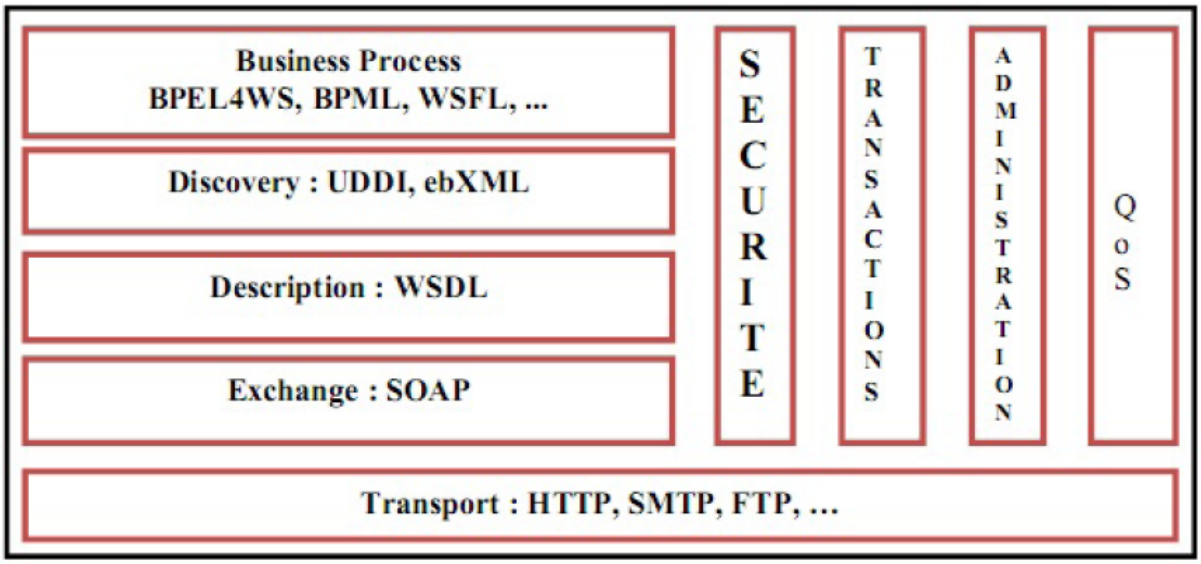

Figure ***2.2:*** *Architecture en Pile des services Web*

Nous apportons une explication de la mise en relief des trois types de couches (Voir la Figure 2.2) :

L'infrastructure de base (Discovery, Description, Exchange) : Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les échanges des messages établis par SOAP, la description de service par WSDL et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI ;

Couches transversales (Security, Transactions, Administration, QoS) : Ce sont ces couches qui rendent viable l'utilisation effective des services Web dans le monde industriel ;

La couche Business Processus (Business Process) : Cette couche supérieure permet l'intégration de services web. Elle établit la représentation d'un « BusinessProcess » comme une composition de services Web. Grâce à cette couche, nous décrivons le flux de contrôle et le flux de données de ces compositions.

2.2.3 Les langages et protocoles utilisés par les services web

Dans cette partie nous décrivons les différentes technologies des services web en nous basant sur leur architecture, sachant que toutes ces technologies sont basées sur XML.

2.2.3.1 Echange avec SOAP-Simple Object Access Protocol

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol) [Mitra et al., 2003]. Ce protocole est normalisé par le W3C.

Le protocole SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP. La norme n'impose pas de choix.

Le choix de l'utilisation de protocoles applicatifs, comme par exemple HTTP, est lié aux problèmes d'interconnexion connus des réseaux. Le but des services web étant d'être réutilisables, après publication, à travers tout le réseau Internet, il faut alors donner les moyens de passer les protections telles que les pare-feux (firewalls). Ces derniers autorisent généralement sans aucune restriction le trafic sur les ports liés aux protocoles tels que HTTP, permettant ainsi le passage sans problème à travers les différents réseaux des messages générés par l'utilisation du protocole SOAP.

Structure d'un message SOAP

La technologie des services web repose principalement sur le protocole SOAP qui est indépendant des langages de programmation ou des systèmes d'exploitation. Le standard SOAP définit trois éléments composant un message : [Gardarin, 2002]

L'enveloppe (Envelope), l'en-tête du message (Header) et le corps du message (Body) (voir Figure 2.3)

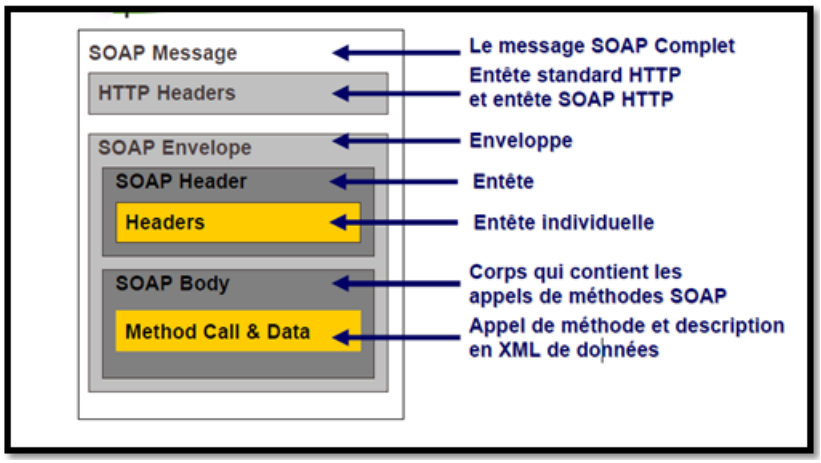

Figure *2.3:* *Structure d'un message SOAP*

Enveloppe SOAP

L'enveloppe SOAP est l'élément obligatoire d'un message SOAP qui englobe les deux autres parties de ce message (Header et Body). Elle contient le nom du message et le nom du domaine, et permet de préciser la version de SOAP utilisée.

L'Entête SOAP

L'en-tête est un élément facultatif dans un message SOAP, marqué par la balise <Header>. Il peut avoir plusieurs fils. Ces fils sont utilisés pour ajouter des fonctionnalités au message SOAP comme l'authentification et la gestion des transactions [De La Rosa-Rosero and Estublier, 2004]. L'en-tête peut utiliser les attributs mustUnderstand et/ou SOAP actor pour déterminer comment le destinataire d'un message doit traiter ce dernier.

- L'attribut acteur de SOAP peut être utilisé pour indiquer le destinataire d'un élément d'en-tête. La valeur de l'attribut d'acteur de SOAP est un URI.
- L'attribut mustUnderstand peut être utilisé pour indiquer si le traitement d'une entrée d'en-tête est obligatoire ou facultative par le destinataire. La valeur de l'attribut de mustUnderstand est "1" ou "0". L'absence de cet attribut est sémantiquement équivalente à sa présence avec la valeur "0".

Le corps SOAP : [Gardarin, 2002]

Le corps du message SOAP est obligatoire, marqué par la balise <Body>. Il permet de transmettre les requêtes et les réponses entre les systèmes. Il est composé d'un ou plusieurs sous-éléments, qui sont :

- L'élément FAULT : il permet d'indiquer les défaillances de transmission des messages SOAP. Il renvoie des informations sur le type d'erreur, une description de l'erreur et l'adresse du serveur SOAP qui a généré l'erreur.
- L'élément MESSAGE : il contient les données à transmettre via le protocole SOAP.

Modèle d'échange de messages en SOAP

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmission d'un message commence (ex. invocation d'un service web), un message SOAP (ou document XML) est généré. Ce message est envoyé à partir d'une entité appelée le SOAP Sender, localisé dans un nœud SOAP. Le message est transmis à zéro ou plusieurs nœuds intermédiaires (SOAP intermédiaires) et le processus finit lorsque le message arrive au SOAP Receiver. Le chemin suivi par un message SOAP est nommé message path, [De La Rosa-Rosero and Estublier, 2004]. La Figure 2.4 montre les éléments qui participent au processus.

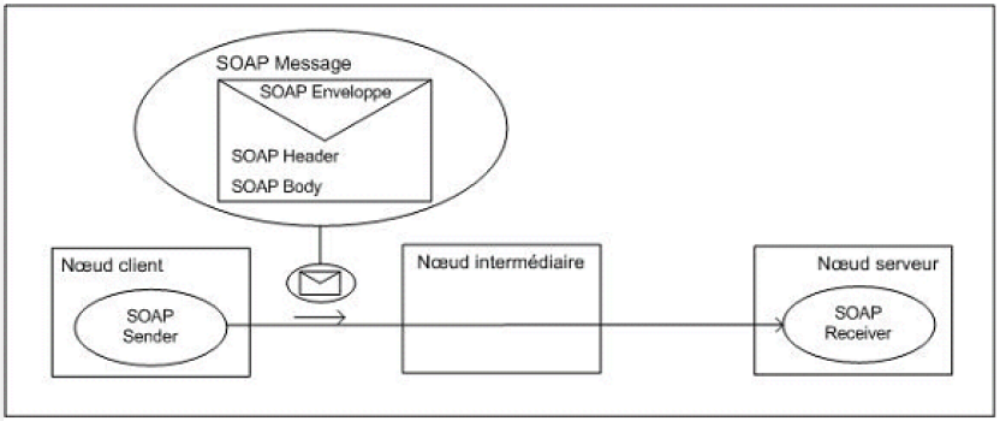

Figure *2.4:* *Modèle d'échange de message en SOAP [Josuttis, 2007]*

En résumé, SOAP est un protocole de communication entre les applications fondé principalement sur le protocole HTTP et sur XML, visant à satisfaire un double

objectif : servir de protocole de communication sur Internet, dans une optique d'intégration d'application, et permettre la communication entre les applications et les services web. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux services web. C'est le rôle du langage WSDL.

2.2.3.2 Description avec WSDL-Web Services Description Language

Le WSDL (Web Services Description Language) [Rampacek, 2006] est un langage basé sur XML, créé dans le but de fournir une description unifiée des services web. Il se présente comme un standard actuel dans ce domaine, et il est, de plus, normalisé par le W3C. Il est issu d'une fusion des langages de description NASSL (Network Accessibility Service Specification Language - IBM) et SCL (Service Conversation Langage - Microsoft). Son objectif principal est de séparer la description abstraite du service de son implémentation même.

La structure d'un document WSDL

Le langage de description WSDL [De La Rosa-Rosero and Estublier, 2004] se comporte donc comme un langage permettant de décrire l'interface visible (ou publiée) du service web. Il décrit, à l'aide du langage de balises XML, les différents éléments du service, [Rampacek, 2006]. Une description WSDL d'un service web est faite sur deux niveaux, un niveau abstrait et un niveau concret. Au niveau abstrait, la description du service web consiste à définir les éléments de l'interface du service web tels que les types de données (Data Types), les messages (Message), les types de ports (Port Type). Ces parties décrivent des informations abstraites indépendantes du contexte de mise en œuvre. On y trouve les types de données envoyées et reçues, les opérations utilisables et le protocole qui sera utilisé.

- Data types : est l'élément qui définit les types de données utilisées dans les messages échangés par le service web.
- Message : spécifie les types d'opérations supportées par le service web ; il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.
- Port Type : est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogue à une définition d'un objet contenant un ensemble de méthodes.

Au niveau concret, le service web est défini grâce aux éléments Bindings, Service et Port. Ces deux derniers décrivent des informations liées à un usage contextuel du service web. On y trouve l'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

- Bindings : Décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (RPC par exemple). Pour un type de port, on peut avoir plusieurs liaisons, pour différencier le mode d'invocation ou de transport de différentes opérations.
- Port : Spécifie une adresse URL qui correspond à l'implémentation du service web par un fournisseur. Elle identifie aussi une ou plusieurs liaisons aux protocoles de transport pour un Port Type donné.
- Service : Spécifie l'adresse complète du service web, et permet à un point d'accès d'une application distante de choisir de multiples catégories d'opérations à exposer pour divers types d'interactions.

En résumé, le document WSDL est indispensable au déploiement de services web et décrit deux éléments essentiels : une description pour l'interface du service web et une autre pour son implémentation. La publication et la recherche d'un service web au sein de l'annuaire UDDI se font via ces documents WSDL.

2.2.3.3 Recherche avec UDDI- Universal Discovery Description and Integration

Un service web doit être référencé afin de pouvoir être retrouvé et utilisé par une autre organisation. Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels. Il existe de nombreux annuaires, mais nous choisissons de décrire seulement le registre UDDI, annuaire dit universel.

L'UDDI définit les mécanismes permettant de répertorier des services Web. Ce standard gère donc l'information relative à la publication, la découverte et l'utilisation d'un service Web. En clair, l'UDDI définit un registre des services Web sous un format XML. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application cliente consulte cet annuaire pour la recherche des informations concernant le service Web désiré, pour une éventuelle interaction ; ainsi l'UDDI a été créé pour faciliter la découverte de services Web en plus de leur publication. Par une API SOAP, on peut interagir avec l'UDDI au moment de la conception et l'exécution des applications afin de découvrir des données techniques et administratives sur les entreprises et leurs services Web.

L'annuaire UDDI repose sur le protocole SOAP. Les requêtes et les réponses sont des messages SOAP [Gardarin, 2002]. L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte.

La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP. L'UDDI peut être vu comme un annuaire contenant [Newcomer, 2002] :

- Les pages blanches : noms, adresses, identifiants et contacts des entreprises enregistrées. Ces informations sont décrites dans des entités de type « BusinessEntity ». Cette description inclut des informations de catégorisation permettant de faire des recherches spécifiques dépendant du métier de l'entreprise.
- Les pages jaunes : donnent les détails sur le métier des entreprises et les services qu'elles proposent. Ces informations sont décrites dans des entités de type «BusinessService».
- Les pages vertes : contiennent des informations techniques du service offert, la manière d'interagir avec le service, une définition du «BusinessProcess» et aussi un pointeur vers le fichier WSDL et une clé unique identifiant le service.

La structure de données du registre UDDI contient cinq types de données :

- BusinessEntity : Les informations concernant l'entreprise qui publie ses services Web dans l'annuaire et le type de services qu'elle offre sont contenues dans la structure «BusinessEntity» et correspondent notamment aux pages blanches concernant l'entreprise.
- BusinesService : L'élément «BusinessService» contient des informations sur les services métiers. Il s'agit des informations de description des services web référencés : le nom du service, sa description, son code. Une entreprise peut

enregistrer plusieurs services. Le type d'informations contenu dans l'élément «BusinessService» correspond aux informations pages jaunes d'une entreprise.

- BindingTemplate (informations de liaison) : Les informations de liaison contiennent des informations techniques sur un service Web. Elles servent également de pages vertes de l'annuaire UDDI. Il s'agit des informations indiquant les références aux «tModels» désignant les spécifications de l'interface pour un service web, ainsi que le point de terminaison (adresse Internet) de ce dernier. Ces informations aident le client à se connecter puis à invoquer le service désiré. Un service peut avoir plus d'un modèle de liaison.
- tModel (informations techniques) : La structure «tModel» a pour rôle de décrire les spécifications techniques des services Web à enregistrer. Elle comporte les informations permettant de connaître les normes auxquelles le service est conforme. Concrètement, un tModel peut pointer sur une description syntaxique du service (document WSDL) ou une description sémantique (document OWLS, une ontologie de domaine,...)
- PublishAssertion (Assertion contractuelle entre partenaires) : met en correspondance deux ou plusieurs structures «BusinessEntity» selon le type de relation (filiale de, département de). Cette structure comporte les assertions contractuelles entre organismes pour les services publiés. Ces assertions représentent un ensemble de règles contractuelles d'invocation de services représentées sous la forme de protocoles entre deux partenaires métiers.

En résumé, UDDI est un standard pour faciliter la collaboration entre partenaires dans le cadre d'échanges commerciaux. Le cœur d'UDDI est un annuaire qui contient des informations techniques et administratives sur les entreprises et les services Web qu'elles publient. Ainsi, il permet de publier et découvrir des informations qui concernent une entreprise et ses services Web.

2.3 Services web sémantiques

Le web sémantique constitue une prolongation et une sorte de révolution de fond du Web actuel qui permet une définition non ambiguë de l'information. Ainsi il permet de favoriser une meilleure coopération entre humain et machine tout en s'ouvrant sur de nouvelles possibilités d'automatisation des tâches requises par l'humain.

Proclamé technologie du futur en 2001 par son créateur Tim Berners-Lee, le Web sémantique propose une nouvelle plateforme permettant une gestion plus intelligente du contenu, à travers sa capacité de manipuler les ressources sur la base de leur sémantique. En réalité, l'intégration de la sémantique au Web n'est pas une nouvelle idée mais au contraire, elle est née avec le Web [Falquet and Mottaz Jiang, 2001].

Le terme services web sémantiques se trouve à la convergence de deux domaines de recherche importants concernant les technologies de l'Internet : le web sémantique et les services web (voir figure 2.5) [Cardoso, 2007]. Cette tâche de convergence est accomplie en rendant les services Web auto-exploitables par machines, et en réalisant l'interopérabilité entre les applications via le web en vue de rendre le web plus dynamique.

De la même façon, le web sémantique est considéré comme un vaste espace d'échange de ressources entre humains et machines, permettant une meilleure exploitation de masses de données disponibles sur le Web. L'objectif est non pas de

permettre aux machines de se comporter comme des êtres humains, mais de développer des langages pour représenter les informations d'une manière traitable, représentable et intelligible par les machines, afin d'améliorer les rapports des utilisateurs avec le web. Il semble donc nécessaire de tendre vers des services intelligibles pour des machines, c'est le concept de service web sémantique [Kadima and Monfort, 2003].

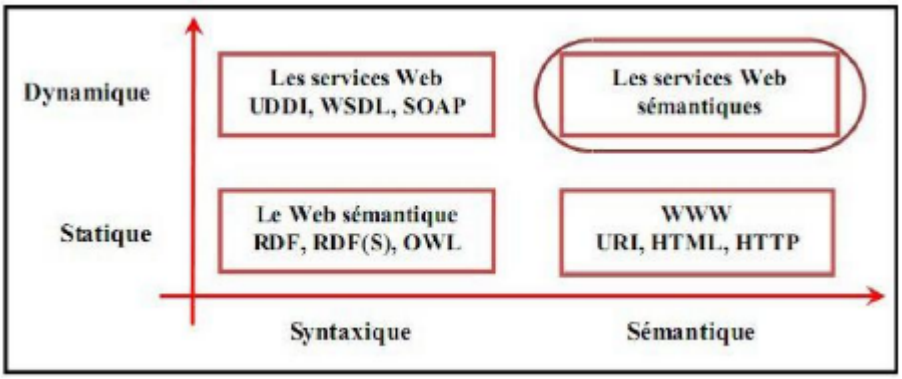

Figure *2.5:* *L'évolution du web [Cardoso, 2007]*

Les avantages potentiels des services web sémantiques ont mené à l'établissement d'un domaine de recherche important, dans le milieu industriel et académique. Plusieurs initiatives sont apparues pour faire ce qu'on appelle l'annotation sémantique des services web, ce qui a produit une variété de descriptions des services Web et leurs aspects relatifs, ce qui en retour a abouti à divers genres de support pour la découverte et la composition.

2.3.1 Langages de description de services Web sémantiques

Les services web sémantiques devraient aussi être décrits sémantiquement afin de pouvoir automatiser leur découverte, sélection, composition, etc. Les langages de description de service web tel que WSDL permettent de décrire les services d'une manière syntaxique. Il existe des approches à base de langage syntaxique comme SAWSDL, WS*-specifications d'une part et des approches à base de modèles sémantiques d'autre part. Par la suite, nous définirons SAWSDL, OWL-S et WSMO.

2.3.1.1 SAWSDL

Semantic Annotations for WSDL and XML Schema (SA-WSDL) [Lausen and Farrell, 2007] est une recommandation W3C établie en 2007 ; elle ajoute des extensions au standard WSDL.

En particulier, elle annote les éléments : operations, input, output, type schemas, et interfaces. Selon [Lausen and Farrell, 2007] "SA-WSDL aims to support the use of semantic concepts analogous to those in OWL-S while being agnostic to the semantic representation language".

La spécification annote un document WSDL 2.0 avec les attributs suivants : modelReference, liftingSchemaMapping et loweringSchemaMapping.

L'attribut modelReference permet d'annoter certains éléments WSDL 2.0. Il est utilisé comme attribut dans les éléments <interface>, <operation> et <fault>.

Il indique le concept équivalent en précisant son adresse. Si nous avons par exemple une opération nommée op1 et qui a le paramètre d'entrée M1 et le paramètre de sortie M2, alors pour associer cette opération avec le concept C1 de l'ontologie Ontologie1, il suffit d'ajouter à l'élément définissant l'opération l'attribut suivant sawsdl :modelReference= "$Ontology1\#C1$".

Les attributs liftingSchemaMapping et loweringSchemaMapping permettent d'associer à un schéma type ou à un élément un concept dans une ontologie partagée.

SAWSDL n'impose pas de langages particuliers pour formaliser les ontologies.

2.3.1.2 OWL-S

OWL-S (Ontology Web Language for Services) [Martin et al., 2004] désigné par DAML-S dans les versions antérieures, est une ontologie de description des services web, dont les objectifs sont de résoudre les ambiguïtés et de rendre la description d'un service compréhensible par une machine. Dans [Ankolekar et al., 2002] les auteurs présentent une ontologie pour les services web dans le but d'automatiser la découverte, l'invocation, la composition et la surveillance de l'exécution des services. Les auteurs reprennent la notion de classes d'OWL et proposent l'ontologie OWL-S. OWL-S fournit trois connaissances essentielles sur les services Web, qui répondent aux questions suivantes :

- que fournit le service Web pour l'utilisateur potentiel ? La réponse à cette question est fournie par le ServiceProfile,
- comment fonctionne le service Web ? La réponse à cette question est fournie par le ServiceModel,
- comment le client peut-il interagir avec le service Web ? La réponse à cette question est fournie par le ServiceGrounding.

OWL-S [Solanki et al., 2004] décrit donc les services web suivant trois points de vue différents qui sont le ServiceProfile, le ServiceModel et le ServiceGrounding représentés dans la figure 2.6.

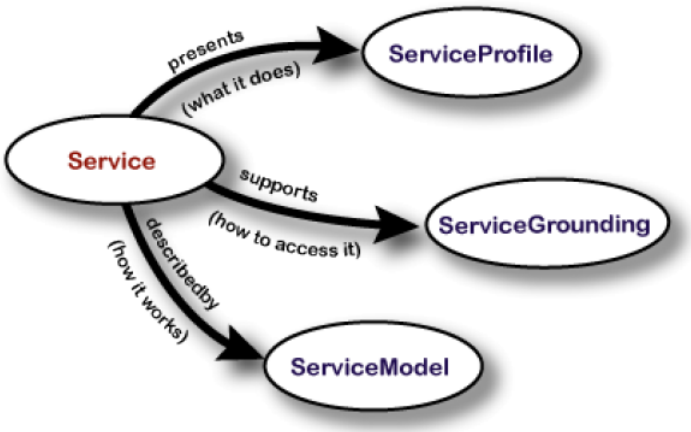

Figure *2.6:* *Niveau supérieur de l'ontologie de OWL-S*

ServiceProfile

Pour décrire un service, OWL-S définit la classe ServiceProfile. La classe ServiceProfile spécifie trois informations :

- Nom du service, contacts et description textuelle du service : le nom du service est utilisé comme identificateur du service, tandis que les informations contacts et la description textuelle sont destinées aux utilisateurs humains.
- Description fonctionnelle du service : Elle spécifie ce que le service exige en termes d'entrées (inputs) attendues et de résultats produits en sortie (outputs). Elle indique également les pré-conditions et les effets du service.
- Classification taxonomique.

ServiceModel

La classe ServiceModel décrit le fonctionnement du service Web. Ceci est fait en exprimant les transformations faites par le service Web sur les données (input à output), et transformations d'état (pré-conditions et effets). Les services Web peuvent être modélisés avec OWL-S en tant que processus grâce à la classe Process. La classe ainsi définie est une sous-classe de ServiceModel. Pour décrire un processus, on spécifie ses entrées, ses sorties et ses états. Les transitions d'un état à un autre sont décrites par les pré-conditions et les effets de chaque processus. Il existe trois types de processus :

1. Les processus atomiques (AtomicProcess) : exécutable en une seule étape, un processus atomique ne peut pas être décomposé de façon plus profonde. Son exécution correspond à une unique avancée dans l'exécution du service, il est directement invoqué par l'utilisateur du service.
2. Les processus composites (CompositeProcess) : un processus composite est constitué par l'assemblage d'autres processus (composite ou non composite). Les processus composites associent des processus à l'aide de structures de contrôle permettant de décrire leur logique d'exécution. Les structures de contrôle sont les suivantes :

   - séquence (Sequence) qui représente une suite ordonnée de processus.

   - exécutions concurrentes de processus qui sont décrites par Split.

   - synchronisation qui peut être décrite par Split+Join.

   - exécution de processus sans ordre particulier avec Unordered.

   - choix décrit par Choice.

   - branchements conditionnels du type si/alors/sinon qui sont décrits par If - Then-Else.

   - Repeat, Iterate et Repeat-Until qui permettent d'effectuer des itérations.
3. Les processus simples (SimpleProcess) : Les processus simples ne sont pas invocables mais comme les processus atomiques, leur exécution s'effectue en une seule étape. Les processus simples sont employés comme éléments d'abstraction, un processus simple peut être employé pour fournir une vue d'un certain processus atomique, ou une représentation simplifiée d'un certain processus composé.

ServiceGrounding

La classe OWL-S ServiceGrounding définit les détails techniques permettant d'accéder au service Web. Les deux premières classes ServiceProfile et ServiceModel d'une description OWL-S s'attachent à abstraire la représentation d'un service Web. ServiceGrounding est la forme concrète d'une représentation abstraite, elle fournit les détails concrets d'accès au service Web, tels les protocoles, les URIs, les messages envoyés, etc.

Les concepteurs de OWL-S ont choisi de combiner OWL-S avec WSDL comme le montre la Figure 2.7. L'objectif de cette combinaison est de tirer profit de chacun d'entre eux. Les types abstraits des messages employés dans les messages WSDL sont définis sémantiquement à l'aide de classes OWL. Les bindings WSDL servent ensuite à définir comment ces messages sont formatés. Les relations entre OWL-S et WSDL sont les suivantes :

- Un processus atomique OWL-S correspond à une opération WSDL.
- Les entrées et sorties d'un processus atomique OWL-S correspondent chacune à un message WSDL (d'entrée ou de sortie).
- Les types des messages d'un processus atomique OWL-S correspondent à l'élément Types d'une description WSDL.

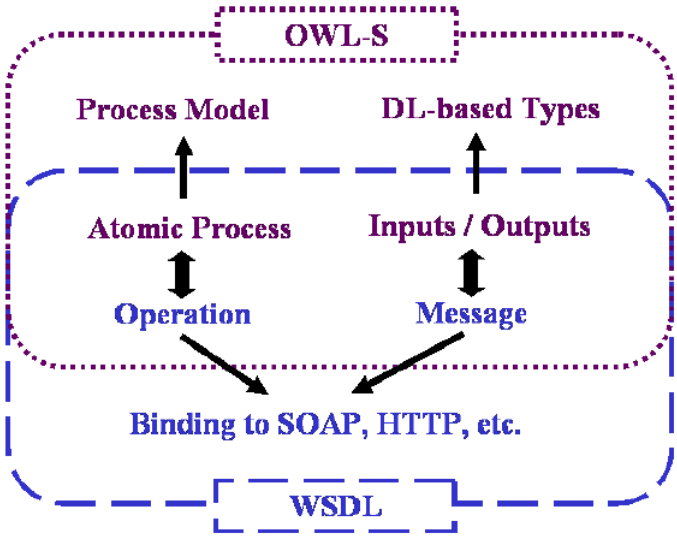

Figure ***2.7:*** *Lien entre OWL-S et WSDL*

2.3.1.3 WSMO

L'ontologie WSMO (Web Service Modeling Ontology) est présentée par [Roman et al., 2005]. Initiée par l'ESSI WSMO workgroup, elle est basée sur le Web Service Modeling Framework WSMF proposé par [Fensel and Bussler, 2002]

WSMO partage avec OWL-S l'objectif d'automatiser les tâches liées aux services. La séparation entre la description des services web sémantiques et les technologies exécutables est un aspect essentiel de WSMO. En effet WSMO fournit un modèle de description d'ontologies indépendant du langage utilisé pour les décrire. Le diagramme de classes UML, illustré par la Figure 2.8, est extrait de [De Bruijn et al., 2008]. Il représente les éléments principaux d'une ontologie WSMO et leurs interactions. Le concept central est le WSMO element qui se spécialise en une ontology ou un webService ou un goal ou un mediator.

- Un élément ontology désigne une ontologie de référence importée par un élément de l'ontologie WSMO afin de décrire ses propriétés.

- Un élément webService est "une unité de calcul capable de répondre à une requête utilisateur" [Roman et al., 2005]. Les webService WSMO sont definis d'une manière uniforme comprenant les éléments suivants : la fonctionnalite du service, les interfaces de description, les propriétés non fonctionnelles, les ontologies importées et les médiateurs utilisés.
  - La fonctionnalité du service, désignée par capability, est décrite en termes d'opérations associées à des pré-conditions, hypothèses (assumptions), post-conditions et effets. Les pré-conditions et les post-conditions décrivent les exigences sur les données avant l'exécution du service et les changements qu'elles subissent après exécution ; les hypothèses et effets décrivent les exigences et les changements relatifs aux services en interaction avec le service décrit.
  - Les interfaces de description décrivent le comportement du service, en d'autres termes l'ordonnancement des opérations.
  - Les propriétés non fonctionnelles d'un webService décrivent les attributs qui ne se rapportent pas directement aux données traitées, comme par exemple la durée d'exécution d'une opération.
  - Les ontologies importées sont utilisées pour référencer les éléments de la description du service.
  - Les médiateurs sont utilisés si deux webServices en interaction importent deux ontologies différentes.
- Un élément goal sert à décrire les souhaits des utilisateurs en terme de fonctionnalités requises. Les objectifs sont une vue orientée utilisateur du processus d'utilisation des services Web, ils sont une entité à part entière dans le modèle WSMO. Un objectif décrit la fonctionnalité, les entrées/sorties, les pre-conditions et post-conditions d'un service web.
- Un élément mediator est utilisé pour résoudre de nombreux problèmes d'incompatibilité, tel que les incompatibilités de données dans le cas où les services web utilisent différentes terminologies, les incompatibilités de processus dans le cas de la combinaison de services Web, et les incompatibilités de protocoles lors de l'établissement des communications.

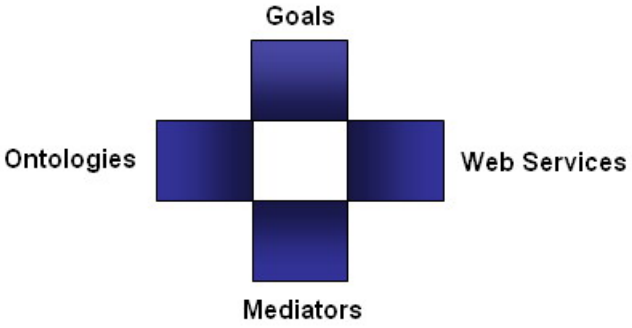

Figure ***2.8:*** *Eléments d'une ontologie WSMO*

Contrairement à OWL-S, WSMO inclut les médiateurs comme des composants centraux de son architecture. Les hétérogénéités rencontrées lors de l'utilisation de services web sont gérées par les types de mediation suivants :

- La médiation de données résout les incompatibilités de représentation des données.
- La médiation de processus est relative à la logique applicative de la composition.
- La médiation de protocoles adapte les différents protocoles de communication utilisés.

Ces types de médiation sont pris en charge par quatre familles de médiateurs :

- Les GG-médiateurs permettent d'effectuer la médiation entre deux objectifs, GG signifiant "goal-goal". Cela signifie qu'ils permettent d'établir des correspondances entre objectifs, en se servant des ontologies d'objectifs disponibles dans le cadre de WSMO.
- Les WG-médiateurs, avec WG pour « web service-goal », établissent les correspondances entre les fonctionnalités offertes par les services Web et les requêtes des utilisateurs, qui sont tous les deux définis comme des objectifs. L'intérêt de ce type de médiateurs est d'aider la découverte et la sélection de services Web.
- Les WW-médiateurs, avec WW pour « web service-web service », établissent les correspondances entre services Web. Leur tâche est de résoudre les conflits au niveau des données, du protocole, et du processus de composition. Ils sont mis en œuvre lors de l'orchestration des services Web au sein d'une composition.
- Les OO-médiateurs, avec OO pour « ontology-ontology », sont destinés à résoudre les conflits entre ontologies. Les autres types de médiateurs énoncés ci-dessus, mais aussi n'importe quel élément de l'architecture WSMO qui pourrait utiliser les ontologies, peuvent utiliser un OO-médiateur pour résoudre un conflit sémantique. Le travail des OO-médiateurs consiste à établir des correspondances entre les terminologies contenues dans les différentes ontologies pour les intégrer en une représentation homogène des données. Cette représentation homogène permet de résoudre les hétérogénéités sémantiques et de répondre aux requêtes soumises par les composants de l'architecture WSMO.

2.4 Conclusion

La technologie des services web permet à des applications de dialoguer à distance via Internet, indépendamment des plates-formes et des langages sur lesquels elles reposent, et en s'appuyant sur des protocoles standards. Dans ce chapitre, nous avons mis en relief les différents langages et protocoles proposés par le consortium W3C permettant aux fournisseurs de décrire leurs services Web. Ensuite, nous avons introduit la notion sémantique dans les services web ainsi que les langages les plus utilisés qui permettent de les décrire sémantiquement, à savoir le langage OWL-S et le modèle WSMO. Nous notons que notre contribution relative à la problématique de la découverte est basée sur la partie "Profile" du standard OWL-S. Le chapitre suivant introduira d'une manière détaillée la problématique de la découverte des services web.

Chapitre 3

La découverte des services web sémantiques

Sommaire

3.1 Introduction

Les services web constituent des composants logiciels interopérables pouvant être réutilisés dans le développement d'applications distribuées. Aider le concepteur ou le développeur dans la recherche de composants nécessaires contribuera à baisser le coût de développement de nouvelles applications. Aussi, plusieurs approches de découverte de services web ont été proposées dans la littérature. En parallèle, le contexte du web a beaucoup évolué grâce à l'évolution du web sémantique et à l'adoption croissante des services web comme technologie d'implémentation des services, et par la suite l'augmentation accélérée du nombre des services web sur internet. Trouver un service web particulier à partir d'un pool de services disponibles est la tâche fondamentale de toute approche de découverte de services Web [Emani, 2009].

McCabe et ses co-auteurs [Booth et al., 2004], ont défini le mécanisme de découverte comme étant "l'acte de localisation d'une description, traitable par machine, d'un service web non connu auparavant décrivant certains critères fonctionnels". Quand à [Chabeb, 2011], il définit la découverte comme le processus d'identifier les services qui peuvent répondre à la requête. Pour cela, une façon de faire serait d'identifier un degré de similitude entre les différents concepts sémantiques qui décrivent le service requis (la requête) et ceux des services offerts (les services publiés). Par exemple pour les services décrits en WSDL, tous les principaux éléments des services doivent être pris en considération dans cette procédure, à savoir : interface, opération, input et output. Toutefois, dans un environnement dynamique de systèmes distribués, une recherche rapide, sémantique et automatique des services web composables est recommandée. La découverte automatique de services web cherche à localiser des services qui peuvent répondre à une requête donnée en procédant à un matching (appariement) des éléments de la requête et ceux des services décrits dans un annuaire.

Selon [Tsetsos, 2007], 3 facteurs influent le processus de la découverte des services web :

- La capacité des fournisseurs de services à décrire leurs services.
- La capacité des demandeurs de service à décrire leurs exigences (leurs besoins).
- L'intelligence et l'efficacité de l'algorithme d'appariement (matching).

Plusieurs efforts ont été définis pour faire face aux défis du problème de découverte. Ils ont été classés dans la littérature selon des critères d'architecture (centralisée, distribuée ou hybride), d'automatisation (manuelle ou automatique) et de niveau de matching (syntaxique ou sémantique). Ce chapitre vise à analyser les approches proposées dans la littérature en vue d'identifier des mécanismes qui constituent des points forts de ces approches. Une étude comparative ainsi qu'une synthèse sont présentées à la fin du chapitre, dans laquelle nous identifions notamment les limitations de certaines approches vis-à-vis du problème de la découverte de services sémantiques.

3.2 Etat de l'art

Selon le critère de matching sémantique et comme mentionné dans [Schumacher et al., 2008], les approches de découverte peuvent être classées en trois classes : l'approche logique, l'approche non-logique et l'approche hybride [Chabeb, 2011]. Les approches logiques exploitent les inférences pour vérifier la compatibilité entre la requête et l'annotation de service (subsomption, test de consistance. . .), alors que les approches non logiques exploitent la sémantique implicite ou informelle des services et la traite avec d'autres techniques, telles que le datamining, la théorie des graphes, la recherche d'informations, les mesures de similarité. La troisième classe mélange les deux premiers types. Dans ce qui suit, nous présentons les principes de chacune de ces approches et un ensemble de travaux ou de plateformes qui l'adoptent.

3.2.1 Approches logiques

Les travaux optant pour cette catégorie d'approches utilisent des descriptions de services et des requêtes spécifiées en langages issus de formalismes logiques, telles que la logique de description et la logique de premier ordre. Ils utilisent également des règles logiques pour la découverte de services Web et exploitent les ontologies pour couvrir leur aspect sémantique. Pour calculer le degré d'appariement, ils recourent à différents procédés et ciblent plusieurs éléments de description de services tout en prenant en considération leur sémantique. Ils optent essentiellement pour trois types d'appariement : IO-matching (Inputs and Outputs matching) [Srinivasan et al., 2006], [Paolucci et al., 2002], PE-matching (preconditions and effects matching) [Schumacher et al., 2008] et IOPE-matching (Inputs, Outputs, preconditions and effects matching) [Jaeger et al., 2005] [Keller et al., 2005] [Kuster and Konig-Ries, 2007].

Paolucci et ses co-auteurs ont défini l'approche WSC (Web Services Capabilities) [Paolucci et al., 2002] qui propose un appariement sémantique des capacités des services web fondé sur l'utilisation de l'ontologie DAML [Ankolekar et al., 2002]. Les services publiés et les requêtes font référence à des concepts DAML qui les décrivent sémantiquement. L'appariement adopté est de type IO-matching. En appariant des concepts sémantiques de type DAML, un service publié est considéré comme un service qui répond bien à une requête lorsque tous les outputs de la requête correspondent

à des outputs du service publié, et tous les inputs du service publié correspondent à des inputs de la requête. Quatre degrés d'appariement sont proposés : EXACT, PLUGIN, SUBSUMES et FAIL. Pour cela, ils comparent les outputs de la requête avec les outputs du service publié, selon l'algorithme suivant :

Matching (R.O,S.O)

// Avec S.O dénote l'output du service S et R.O dénote l'output de la requête.

Retourner

Exact :si les deux concepts ontologiques R.O et S.O sont équivalents ou R.O est un concept plus spécifique de S.O.

Plug-in : si S.O est un concept plus général que le concept R.O

Subsume : si R.O est un concept plus général que le concept S.O

Fail : Aucune relation hiérarchique n'existe entre eux

Si deux services S1, S2 ont le même score, alors nous les classons en comparant leurs inputs avec ceux de la requête en utilisant la fonction matching (S.I,R.I) comme suit :

- Si le score d'appariement des outputs du service S1 est plus grand que le score d'appariement des outputs du service S2 alors c'est le service S1 qui sera sélectionné.
- Dans le cas où le score d'appariement des outputs du service S1 est égal au score d'appariement des outputs du service S2, si le score d'appariement des inputs du service S1 est plus grand que le score d'appariement des inputs du service S2 alors c'est le service S1 qui sera sélectionné.
- Dans le cas où le score d'appariement des outputs du service S1 est égal au score d'appariement des outputs du service S2, si le score d'appariement des inputs du service S1 est égal au score d'appariement des inputs du service S2 alors les deux services seront sélectionnés.

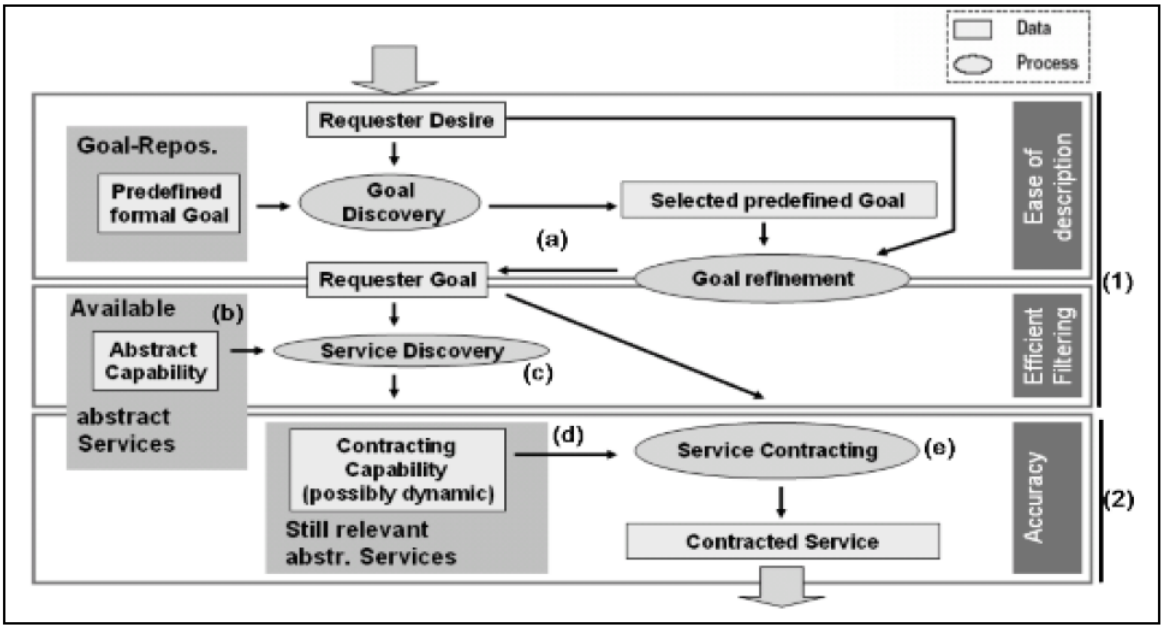

Figure ***3.1:*** *Modèle de localisation automatique et sémantique de services [Keller et al., 2005]*

Dans [Keller et al., 2005], les auteurs proposent un modèle conceptuel (Figure 3.1) pour une découverte sémantique de services (Automatic Location of Services, ALS) qui permet de réutiliser des objectifs (Goals) prédéfinis, des abstractions de services et finalement des services concrets pour répondre à l'objectif d'une requête. Cela s'effectue en deux grandes phases :

- La phase de découverte de services (Figure 3.1.(1)) s'intéresse à :
  - des objectifs représentant ce que l'utilisateur souhaite réaliser après utilisation du service, et ce qu'il recevrait comme output et effets du service (Figure 3.1.(a)),
  - des descriptions abstraites plutôt qu'à des descriptions précises de capacités (Figure 3.1.(b)),
  - l'identification des services candidats possibles (Figure 3.1.(c)).
- La phase de contractualisation de service (Figure 3.1.(2)) s'intéresse aux :
  - descriptions précises des capacités (uniquement celles des services découverts dans la première phase) (Figure 3.1.(d)),
  - informations sur les inputs, pour avoir un service concret (avec tous les inputs, les préconditions et les postconditions) (Figure 3.1.(e)),
  - interactions entre le fournisseur et le demandeur de service.

L'approche proposée distingue cinq degrés d'appariement appelés degrés d'appariement intentionnel :

- *Match* si le service offert satisfait complètement la requête,
- *PossMatch* lorsque le service offert peut satisfaire complètement la requête. A cause d'une incomplétude dans la description disponible, la satisfaction complète de la requête ne peut être garantie. Ce degré d'appariement n'est conclu qu'à la phase de contractualisation,
- *ParMatch* si le service offert satisfait la requête, mais partiellement. Il offre certains aspects requis mais pas la totalité.
- *PossParMatch* lorsque le service offert peut satisfaire partiellement la requête. A cause d'une incomplétude dans la description disponible, ceci ne peut être garanti. Ce degré d'appariement est conclu à la phase de contractualisation,
- *NoMatch* si aucun des précédents cas n'est identifié, l'offre est complètement non pertinente par rapport à la requête.

Les auteurs dans [Jaeger et al., 2005] proposent un matchmaker OWL-S fondé sur le IOPE-matching. Ce matchmaker recherche des correspondances sémantiques entre les paramètres fonctionnels définis dans des descriptions de services web OWL-S et ceux introduits dans la requête [Srinivasan et al., 2006].

La Figure 3.2 résume le processus de l'agrégation. Elle détaille le processus d'appariement qui comporte quatre tâches : l'appariement des inputs (Figure 3.2.(1)), l'appariement des outputs (Figure 3.2.(2)) et l'appariement de la catégorie du service (Figure 3.2.(3)). L'algorithme calcule dans chaque tâche le degré d'appariement. Une quatrième tâche pendant laquelle des contraintes et des fonctionnalités pré-définies par l'utilisateur s'appliquent (Figure 3.2.(4)) avant que les résultats soient agrégés afin de retourner le résultat final (Figure 3.2.(5)). Le résultat est soit un échec s'il y a échec d'appariement dans l'une des précédentes tâches (Figure 3.2.(6)), soit une valeur numérique (dite RANK) déterminée par l'agrégation des degrés d'appariements

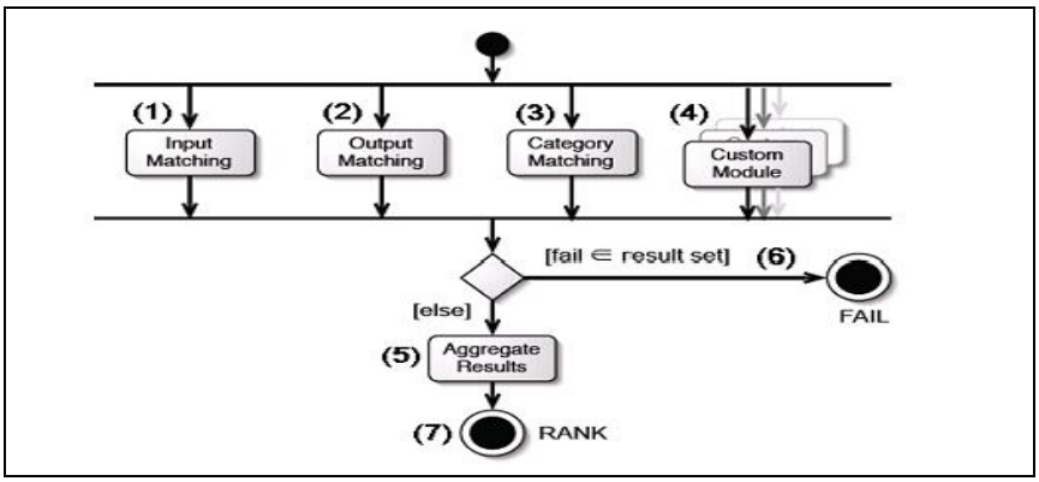

Figure *3.2:* *Agrégation des degrés d'appariement dans OWLS-M [Jaeger et al., 2005]*

de chaque tâche (Figure 3.2.(7)) et représentant le rang du service Web apparié dans l'ensemble des services offerts.

Pour évaluer l'appariement, les auteurs proposent quatre types de degrés d'appariement avec quelques différences d'interprétation d'un élément du profil à un autre :

– Degrés d'appariement pour les input / outputs :

- 0 : *FAIL* : si au moins un input / output requis n'a pas été apparié,
- 1 : *UNKNOWN* : si l'algorithme d'appariement ne peut pas catégoriser un input / output,
- 2 : *SUBSUMES* : si l'input / l'output offert est plus spécifique que l'input / l'output requis,
- 3 : *EQUIVALENT* : en cas d'équivalence entre inputs / outputs.

– Degrés d'appariement pour les catégories de service :

- 0 : *FAIL* : si les concepts des catégories de service ne s'apparient pas,
- 1 : *UNKNOWN* : si l'algorithme d'appariement ne peut pas catégoriser l'un des services,
- 2 : *SUBSUMES* : si le service offert est plus spécifique que le service requis,
- 3 : *EQUIVALENT* : en cas d'équivalence entre catégories de service.

Dans un autre travail de découverte de services tenant compte du contexte (Context Aware Service Discovery, CASD) [Doulkeridis et al., 2006] le module de découverte utilise des ontologies du domaine pour déterminer les catégories des services qui ont une relation sémantique avec la requête de l'utilisateur. Lorsque l'utilisateur formule sa requête (Q) en termes de mots-clés, une autre requête plus riche (Q') est générée et sera attachée à cette dernière. La requête (Q) permet de trouver les services Web qui ont une relation sémantique avec les termes de recherche de l'utilisateur, tandis que la requête (Q') joue le rôle de filtre permettant de sélectionner les services qui correspondent au contexte de l'utilisateur. Les requêtes formulées par les utilisateurs sont exprimées en SPARQL qui est un langage de requête compatible avec les graphes OWL.

Dans [Benatallah et al., 2005b] les auteurs transforment le problème de découverte de services en un problème de réécriture de concepts. Pour cela ils proposent

un algorithme pour découvrir la meilleure couverture sémantique d'une requête par les concepts d'une ontologie exprimée avec la logique de description. La requête est constituée d'un ensemble de concepts d'entrées et de sorties. Les auteurs prouvent que le problème de découverte de la meilleure couverture sémantique, est similaire au problème de calcul, à coût minimal, des transversales minimes d'un hypergraphe pondéré *(computing the minimal transversals with minimum cost of a weighted hypergraph).*

Dans [Küster and König-Ries, 2008], les auteurs ont travaillé sur l'approche présentée dans [Keller et al., 2005], ils proposent des extensions au niveau des degrés d'appariement. Ils ajoutent :

- *RelationMatch* : Le service offert ne fournit pas les services requis mais plutôt des fonctionnalités en relation avec eux. Ceci peut être utile si une coordination peut être planifiée avec d'autres services offerts.
- *ExcessMatch* : Le service offert est capable de fournir les services requis mais des effets additionnels indésirables et non requis par l'utilisateur se produiront.

Les auteurs dans [Srinivasan et al., 2006] appliquent un matchmaking logique qui fait intervenir les entrées, les sorties, les preconditions, et les effets (IOPE) des services.

3.2.2 Approches non logiques

Les approches de découverte qualifiées de non-logiques utilisent des mécanismes syntaxiques, structurels et numériques tels que le concept de distance numérique, l'appariement de graphes structurés, la similitude syntaxique, etc. L'idée est d'exploiter la sémantique implicite plutôt que la sémantique explicite. Pour assurer cela, de tels mécanismes d'appariement utilisent les fréquences des termes, les sous-graphes, etc.

Dans [Schumacher et al., 2008] les auteurs présentent un système de découverte non logique de services Web "iMatcher1" qui utilise un matchmaker syntaxique de profils de services. Ce système puise des services à partir d'un ensemble de profils de services Web décrits en OWL-S. Ces services sont stockés sous forme de graphes RDF (Resource Description Framework) sérialisés dans une base de données RDF, au moyen d'une extension du langage RDQL (RDF Data Query Language), appelée iRDQL [Bernstein and Kiefer, 2005]. Le degré d'appariement de la requête et d'un service est calculé à partir de quatre métriques de calcul de similarité syntaxique : TFIDF (Term Frequency-Inverse Document Frequency) [Sparck Jones, 1972] [Salton and McGill, 1986], la distance de similarité de Levenshtein [Levenshtein, 1966], la mesure du vecteur Cosinus [Garcia, 2006] et la mesure de divergence de Jensen-Shannon [Fuglede and Topsoe, 2004]. Les résultats sont classés en fonction des scores numériques de ces mesures de similarités syntaxiques et d'un seuil défini par l'utilisateur.

Dans [Klein and König-Ries, 2004] les auteurs proposent un matchmaker nommé "DSD-matchmaker" qui réalise la découverte de services Web à travers l'appariement de graphes de description de services. Ces descriptions sont spécifiées au moyen du langage orienté objet de description de service DSD (Diane Service Description) [Kuster and Konig-Ries, 2007] qui spécifie des variables et des ensembles d'objets déclaratifs sans aucune sémantique basée sur la logique. La description est composée de deux parties : une partie statique déclarée dès le début et une partie dynamique construite à base d'informations liées au contexte du service. Le processus d'appariement détermine à partir d'un graphe les variables à satisfaire, et sélectionne, en se basant sur l'état des services, celui qui répond le mieux à la requête parmi l'ensemble de services

découverts, puis il renvoie une valeur numérique représentant le degré d'appariement correspondant. Le fonctionnement du DSD-matchmaker se fait en plusieurs étapes au cours desquelles il effectue des estimations des offres de services. Il détermine d'abord les valeurs concrètes des entrées (inputs) à utiliser pour l'exécution d'estimation. Cela permet également de recueillir des informations pour savoir si une estimation du service proposé est prometteuse, c'est-à-dire si la valeur fournie par cette opération est utile pour décider à quel niveau ce service répond à la requête initiale. Dans une deuxième étape, le DSD-matchmaker n'exécute que les estimations prometteuses et met à jour les descriptions des services offerts à l'aide d'informations collectées dynamiquement. Finalement, dans une troisième étape, l'appariement est effectué sur la base des descriptions mises à jour.

Dans [Klusch et al., 2008], les auteurs expliquent leur approche et proposent d'intégrer des informations dynamiques dans les descriptions des services. Pour effectuer le processus d'appariement, DSD-matchmaker parcourt en parallèle deux descriptions (celle de l'offre en cours et celle de la requête) sous forme d'arbres et compare récursivement les noeuds de ces deux graphes [Küster et al., 2007]. L'algorithme [Klein and König-Ries, 2004], [Klein et al., 2005] utilisé pour ce faire calcule le degré d'appariement de deux noeuds en tant qu'agrégation du degré d'appariement des deux types des concepts sémantiques représentés par les deux noeuds et du degré d'appariement des propriétés de ces deux concepts.

Le matchmaker "URBE (Uddi Registry By Example)" présenté dans [Plebani and Pernici, 2009], assure un matching sémantique de services formalisés avec SAWSDL sans utiliser la logique. Ce système compare :

- les propriétés ontologiques avec des similarités textuelles.
- les types de données associées aux éléments XML (XSD I/O).
- les concepts ontologiques en utilisant une similarité structurelle (basée sur le calcul de longueur de chemins).

De la même manière, les auteurs dans [Li et al., 2007] proposent une approche de découverte à base de l'ontologie WSMO, pour cela ils utilisent la théorie des graphes pour évaluer la similarité sémantique entre deux concepts.

Les auteurs dans [Skoutas et al., 2007] proposent des mesures de similarité basées sur le rappel, et la précision pour apparier les services et la requête. Les auteurs prennent en compte, les paramètres suivants : les entrées, les sorties, les pré-conditions et les effets.

Dans [Boukhadra et al., 2016] les auteurs décrivent une approche efficace pour améliorer la performance et l'efficacité de la découverte et la composition automatique et coopérative des services web dans les systèmes peer to peer (P2P). Ils implémentent une solution distribuée basée sur un algorithme qui permet de découvrir et composer des services web dans des systèmes P2P. L'idée principale de cette approche est de développer une technique de matching hybride qui fonctionne sur les modèles de processus OWL-S afin d'assurer un rappel élevé, aussi elle réduit le nombre de messages échangés et le temps d'exécution pour découvrir et composer des services web dans le réseau P2P. En outre, la technique de matching est capable de calculer une similarité complexe entre les services et la requête de l'utilisateur.

Dans [Chotipant et al., 2014] une fonction de similitude floue (fuzzy) est proposée pour faire le matching entre les termes de la requête et les descriptions de service. En effet, la requête est d'abord enrichie avec un ensemble de synonymes ; par la suite,

une mesure de similitude basée sur la fonction gaussienne est effectuée afin de calculer le score de matching global.

Dans [Cassar et al., 2014], les auteurs utilisent une analyse probabiliste pour la sémantique latente (PLSA) [Hofmann, 1999] et l'allocation de dirichlet latente (LDA) [Blei et al., 2003] pour extraire des facteurs latents issus des descriptions de services sémantiques dans un espace de facteurs latents. Chaque service est représenté comme une distribution de probabilité sur des facteurs latents. Un facteur latent représente un groupe de concepts, plus formellement, c'est une distribution de probabilité par rapport à des concepts sémantiques. Afin d'améliorer l'efficacité du matchmaking, les auteurs utilisent un ensemble de facteurs latents en tant que modèle réduit de services web sémantiques.

La première contribution de [Chabeb, 2011] consiste à proposer un langage de description sémantique des services web nommé YAZAWSDL : Yet Another Semantic Annotation for WSDL (en plus court YASA), qui est une extension de SAWSDL en définissant deux types d'ontologies. Une première ontologie dite "ontologie technique" contenant des concepts définissant des sémantiques -fonctionnelles et/ou non fonctionnelles- de services et une deuxième ontologie dite "ontologie de domaine" contenant des concepts définissant les sémantiques métiers des services. Sa deuxième contribution consiste à proposer une approche de découverte sémantique basée sur un algorithme d'appariement et d'agrégation sémantiques. L'algorithme d'appariement offre des techniques d'appariement qui parcourent la plupart des éléments des deux services, requis et offert.

3.2.3 Approches hybrides

Les approches hybrides utilisent une combinaison de mécanismes logiques et non logiques. L'idée est de remédier à certaines limites de chacun de ces deux mécanismes grâce à différentes combinaisons hybrides qui réussissent là où chacun de ces deux mécanismes échoue.

Dans [Samir et al., 2017], les auteurs proposent un système pour rechercher les services en utilisant les aspects fonctionnels et non fonctionnels. Pour mesurer la similarité fonctionnelle les auteurs apparient la partie abstraite du document WSDL (les opérations, les entrées/sorties) ainsi que certains éléments de la partie concrète du modèle WSDL. Et pour mesurer la similarité non-fonctionnelle les auteurs utilisent les attributs de QoS préférés par l'utilisateur afin de comparer avec les données de QoS offertes par les services. Cette comparaison est réalisée à l'aide de l'erreur des moindres carrées.

OWLS-MX [Klusch et al., 2006] est un matchmaker sémantique hybride qui exploite le raisonnement logique et les techniques de recherche d'information pour réaliser un IO-matching entre des profils de services OWL-S. En plus des degrés de succès et d'échec d'appariement (Exact et Fail), OWLS-MX propose cinq degrés d'appariement : les trois premiers sont uniquement basés sur la logique (Plug-In, Subsumes et Subsumed-By) et les deux derniers sont hybrides. Ils exploitent, en plus de la logique, des valeurs de similarité syntaxique (Logic-based Fail et Nearest-neighbor). Les degrés d'appariement sont définis comme le présente la Figure 3.3.

Les degrés d'appariement de OWLS-MX sont ordonnés en tenant compte de la tolérance d'appariement sémantique, les auteurs de l'approche précisent que plus les données en outputs sont génériques par rapport à celles requises, plus l'appariement sémantique est tolérant. Ainsi, proposent-ils l'ordre suivant, allant du plus haut au

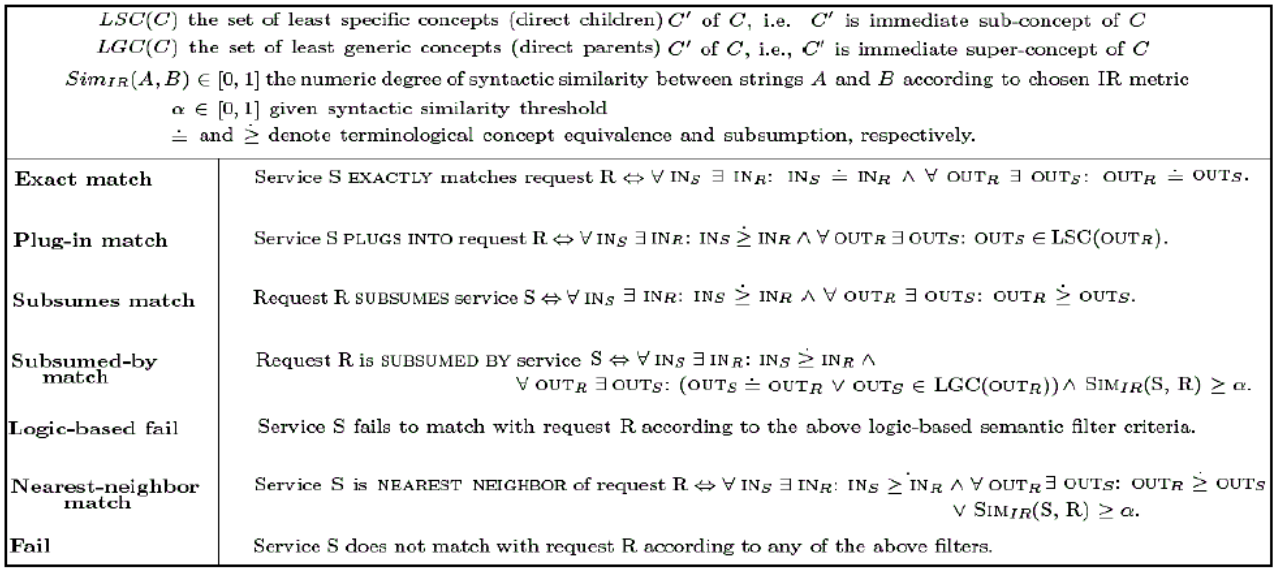

Figure *3.3:* *Les degrés d'appariement dans OWLS-MX [Klusch et al., 2006]*

plus bas degré :

Exact ; Plug-In ; Subsumes ; Subsumed-By ; Logic-based Fail ; Nearest-neighbor ; Fail.

La Figure 3.3 présente les définitions formelles des degrés d'appariement pour la découverte dans l'approche OWLS-MX.

L'algorithme d'appariement reçoit comme requête un service OWL-S et retourne un ensemble ordonné de services jugés pertinents. A chaque service sont associés un degré d'appariement et une valeur de similarité syntaxique par rapport à la requête. L'utilisateur spécifie le degré d'appariement souhaité et le seuil de la valeur de similarité syntaxique.

L'algorithme ne calcule pas uniquement l'appariement logique des inputs et des outputs mais il calcule en parallèle les similarités syntaxiques entre les concepts des inputs et des outputs. L'échec d'un appariement logique au niveau du raisonneur logique de OWLS-MX est toléré si et seulement si la similarité syntaxique correspondante entre le concept du service en cours et de la requête reste supérieure au seuil de similarité souhaité.

Les auteurs de l'approche ont implémenté différentes variantes de l'algorithme générique de OWLS-MX : OWLS-M1, OWLS-M2, OWLS-M3 et OWLS-M4 [Klusch et al., 2006]. Chacune utilise les mêmes degrés logiques mais des métriques de similarité syntaxique différentes. Une autre variante OWLSM0 ne réalise que de l'appariement sémantique logique.

- OWLS-M0, les degrés d'appariement logiques Exact, Plugin et Subsumes sont appliqués comme précédemment définis, alors que le degré hybride Subsumed-By est utilisé sans vérifier aucune contrainte de similarité syntaxique.
- Les variantes hybrides OWLS-M1, OWLS-M2, OWLS-M3 et OWLS-M4 calculent la valeur de similarité syntaxique en se basant respectivement sur les mesures suivantes : LOI (Loss- Of-Information), le coefficient ExtJac (extended Jacquard [31]), la valeur Cos (Cosine [Garcia, 2006]) et la mesure de divergence JS (Jensen-Shannon [Fuglede and Topsoe, 2004]).

OWLS-iMatcher2 [Kiefer, 2009], [Kiefer and Bernstein, 2008], [Kiefer et al., 2007] est une approche hybride de découverte dont la composante logique se base sur un appariement entre les concepts d'inputs/outputs et la composante non logique utilise un appariement basé sur la similarité textuelle des noms et des signatures des services (SimPack [Bernstein et al., 2005]). L'évaluation d'appariement utilise une valeur binaire de pertinence sémantique ainsi que les différentes valeurs de métriques de similarité telles que : Bi-Gram [Kiefer, 2009] et les mesures de Levenshtein [Levenshtein, 1966], [Gusfield, 1997], Monge-Elkan [Monge et al., 1996] et Jaro [Winkler and Thibaudeau, 1991].

L'approche transforme la description structurée du profil du service en un vecteur pondéré de mots clés. Ce vecteur n'inclut pas que les noms utilisés mais aussi des termes qui en dérivent. L'algorithme d'appariement de OWLS-iMatcher2, commence par calculer les valeurs de similarités syntaxiques entre une requête donnée et tous les services disponibles, ensuite, il utilise un modèle mathématique de régression pour prédire l'agrégation d'appariement pour chaque service. Les résultats sont renvoyés, à l'utilisateur, en ordre décroissant d'appariement. L'utilisateur évalue la liste des services ordonnés. Pour cela, le OWLS-iMatcher2 lui offre deux composants : le premier permettant l'évaluation statistique des résultats retournés et le deuxième affichant une visualisation graphique.

Les évaluations expérimentales de OWLS-iMatcher2 ont été réalisées sur la collection de test OWLS-TC2.1. Les tests ont été réalisés sur une même base de services contenant à la fois les requêtes et leurs propres réponses.

Dans l'approche hybride FC-Match (Functional Comparison) [Bertoli et al., 2004], les concepts du service et de la requête à apparier sont présentés en OWL-DL (Web Ontology Language Description Logics). Les concepts utilisés pour décrire un service apportent des informations sur les éléments : Category, Operation, Input et Output.

Un service est perçu comme étant une conjonction d'expressions qualifiées de rôles. Chaque rôle correspond à un paramètre du profil sélectionné, le rôle est décrit par des concepts. Soit S un service, C1 le concept associé au paramètre catégorie du profil de S, C2 le concept associé au paramètre opération, C3 le concept associé au paramètre input et C4 le concept associé au paramètre output. Le concept global associé au profil du service S se note comme suit :

S = ?hasCategory(C1) ?hasOperation(C2) ?hasInput(C3) ?hasOutput(C4)

Cette approche opte également pour un appariement sémantique déductif et un appariement algébrique fondé sur le calcul de similarité syntaxique. Le calcul de la valeur globale d'appariement est effectué en agrégeant le résultat de l'appariement fondé sur la subsomption logique des concepts entre le service et la requête, et la valeur du coefficient de similarité syntaxique.

WSMO-MX [Kaufer and Klusch, 2006] accepte en entrée des services spécifiés en WSML-MX [Klusch et al., 2008]. L'approche d'appariement est une combinaison d'appariement sémantique hybride de OWLS-MX [Klusch et al., 2006], d'appariement de graphes du DSD-Matchmaker [Klein and König-Ries, 2004], et d'appariement intentionnel de service de [Keller et al., 2005]. La spécificité de ce matchmaker est la notion de *derivative*. Une *derivative DT* est un concept ordinaire T "Type" qui est défini dans une ontologie donnée en lui attachant des métainformations. Ces informations permettent de savoir comment apparier T avec un autre type.

Les degrés d'appariement sont calculés par l'agrégation des valuations de quatre éléments :

- l'appariement des types ontologiques (la relation hiérarchique entre concepts logiques)
- l'appariement logique (basé sur des instances) des contraintes spécifiées en F-logic
- l'appariement des noms de relations
- la mesure de similarité syntaxique

Les degrés d'appariement, leur ordre, leur symbole et leur signification par rapport aux états de précondition et postcondition, sont tous présentés dans la Figure 3.4. Soient G le Goal du service (le service requis) et W le service WSMO (le service offert). Les degrés sont cités en ordre décroissant de leur valeur d'appariement, allant de l'équivalence à l'échec. L'équivalence est en haut des degrés, notée equivalence. Ensuite viennent les deux degrés plugin et inverse-plugin qui sont définis différemment par rapport à un état de précondition ou de postcondition. Puis, on retrouve le degré intersection suivi du degré fuzzy similarity. WSMO-MX identifie en avant dernier le degré neutral et en dernier ordre le degré fail qui interprète un échec total de l'appariement.

ordre	symbole	degré d'appariement	pré	post
1	$\equiv$	equivalence	$\mathcal{G} = \mathcal{W}$	
2	$\sqsubseteq$	plugin	$\mathcal{G} \subseteq \mathcal{W}$	$\mathcal{W} \subseteq \mathcal{G}$
3	$\sqsupseteq$	inverse-plugin	$\mathcal{G} \supseteq \mathcal{W}$	$\mathcal{W} \supseteq \mathcal{G}$
4	$\sqcap$	intersection	$\mathcal{G} \cap \mathcal{W} \neq \emptyset$	
5	$\sim$	fuzzy similarity	$\mathcal{G} \sim \mathcal{W}$	
6	$\circ$	neutral	*by derivative specific definition*	
7	$\bot$	disjunction (fail)	$\mathcal{G} \cap \mathcal{W} = \emptyset$	

Figure *3.4:* *Les degrés d'appariement dans WSMO-MX [Kaufer and Klusch, 2006]*

WSMO-MX offre les derivatives types des relations de similarité suivantes :

- Equivalent : les types TW (le Type en service WSMO) et TG (le Type en Goal : Objectif du service) sont équivalents,
- *Sub* : TW est un sous-type de TG,
- *Super* : TG est un sous-type de TW,
- *Sibling* : les types ont un type parent commun direct,
- *Spouse* : les types ont un type descendant commun direct,
- *ComAnc* : les types ont un parent commun non direct,
- *ComDes* : les types ont un descendant commun non direct,
- *Relative* : il existe un chemin dans le graphe non dirigé de l'ontologie qui les relie.

WSMO-MX sélectionne les états de précondition et de postcondition de chaque service offert à partir de sa base de connaissances, ensuite il les apparie un par un avec les états du service requis. Dans le cas où il n'existe pas de préconditions, le

Stratégies de défauts	Vecteur de valuations	
	$val_{pre,webservice}/val_{post,goal}$	$val_{post,webservice}/val_{pre,goal}$
	$(\pi_{\equiv},\pi_{\sqsubseteq},\pi_{\sqsupseteq},\pi_{\sqcap},\pi_{\sim},\pi_{\circ},\pi_{\perp})$	$(\pi_{\equiv},\pi_{\sqsubseteq},\pi_{\sqsupseteq},\pi_{\sqcap},\pi_{\sim},\pi_{\circ},\pi_{\perp})$
assumeEquivalent	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)
none	(0,1,0,0,0,0,0)	(0,0,1,0,0,0,0)
ignore	(0,0,0,0,0,1,0)	(0,0,0,0,0,1,0)
assumeFailed	(0,0,0,0,0,0,1)	(0,0,0,0,0,0,1)

Figure *3.5:* *Valuations d'appariement de relations avec les stratégies de défauts [Kaufer and Klusch, 2006]*

résultat d'appariement de cet élément est fixé à Equivalent par défaut [Kaufer and Klusch, 2006]. Dans ces cas de défauts (absences) d'annotations, WSMO-MX propose d'utiliser des stratégies spécifiques appelées "stratégies de défauts" (voir la Figure 3.5).

Après le traitement des résultats, l'algorithme d'appariement renvoie un vecteur de valuations d'appariements agrégées, avec des annotations sur les résultats du processus d'appariement. Ces annotations sont considérées comme une sorte de retour explicatif à l'utilisateur. Ce retour essaie, en cas de résultats d'appariement insuffisants, de faciliter le raffinement itératif de l'objectif Goal de la part de l'utilisateur. Pour calculer les degrés d'appariement sémantique, l'algorithme utilise des valeurs provenant des trois tables que nous présentons ici.

La table de la Figure 3.5 présente les valuations d'appariement de relations en tenant compte des stratégies spécifiques au cas de défauts. Il existe quatre cas de défauts :

- *assumeEquivalent* : le cas d'équivalence par défaut,
- *none* : le cas du nul,
- *ignore* : le cas où on ignore la relation de similarité,
- *assumeFailed* : le cas d'échec par défaut.

La table de la Figure 3.6 présente les valuations d'appariement de types. On retrouve les huit derivatives types des relations de similarité, à savoir : Equivalent, Sub, Super, Sibling, Spouse, ComAnc, ComDes et Relative. Pour Sub et Super, uniquement les coefficients des degrés d'appariement plugin et inverse-plugin s'intervertissent les valeurs respectivement en précondition et en postcondition. Les vecteurs sont les mêmes en état de précondition et en état de postcondition. Pour les autres relations, on a un même vecteur en état de précondition et en état de postcondition avec un unique coefficient 1 et des zéros en reste du vecteur.

Le coefficient 1 des vecteurs associés à la relation Equivalent correspond au degré d'appariement equivalence. Pour les relations Sibling et ComAnc, ils correspondent au degré intersection. Pour les relations Spouse, ComDes et Relative, ils correspondent au degré fuzzy similarity.

Relation de similarité des types	Vecteur de valuations	
	val_{pre}	val_{post}
	$val_{pre,webservice}/val_{post,goal}$	$val_{post,webservice}/val_{pre,goal}$
	$(\pi_{\equiv},\pi_{\sqsubseteq},\pi_{\sqsupseteq},\pi_{\sqcap},\pi_{\sim},\pi_{\circ},\pi_{\perp})$	$(\pi_{\equiv},\pi_{\sqsubseteq},\pi_{\sqsupseteq},\pi_{\sqcap},\pi_{\sim},\pi_{\circ},\pi_{\perp})$
Equivalent	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)
sub	(0,0,1,0,0,0,0)	(0,1,0,0,0,0,0)
super	(0,1,0,0,0,0,0)	(0,0,1,0,0,0,0)
sibling	(0,0,0,1,0,0,0)	(0,0,0,1,0,0,0)
comAnc	(0,0,0,1,0,0,0)	(0,0,0,1,0,0,0)
spouse	(0,0,0,0,1,0,0)	(0,0,0,0,1,0,0)
comDes	(0,0,0,0,1,0,0)	(0,0,0,0,1,0,0)
relative	(0,0,0,0,1,0,0)	(0,0,0,0,1,0,0)

Figure *3.6:* *Valuations d'appariement pour relations de similarité des types [Kaufer and Klusch, 2006]*

Le résultat d'appariement de type et toutes ces valuations sont agrégés selon un algorithme d'appariement sémantique.WSMO-MX était implémenté en utilisant le raisonneur F-Logic : Onto- Broker.

SAWSDL-MX [Klusch and Kapahnke, 2008] accepte en entrée des services spécifiés en SAWSDL [59]. Ce matchmaker est inspiré par les matchmakers OWLS-MX [Klusch et al., 2006] et WSMO-MX [Kaufer and Klusch, 2006]. La découverte utilise à la fois :

- une approche d'appariement logique basée sur le raisonnement par subsomption,
- une approche d'appariement syntaxique basée sur les techniques de recherche d'information.

Partant du fait qu'une description SAWSDL [Lausen and Farrell, 2007] est une extension d'une description WSDL, l'appariement recouvre les éléments de description suivants :

- interface si on découvre un service spécifié en WSDL 2.0 et portType si on découvre un service spécifié en WSDL 1.1,
- operation, input, output : pour WSDL 1.1 et WSDL 2.0.

Pour réaliser l'appariement des interfaces, le matchmaker effectue des appariements sur des graphes bipartis avec :

- des noeuds représentant des opérations d'un service,
- des arcs valués dont les valeurs sont calculées à partir des degrés d'appariement des opérations reliées par les arcs.

Pour réaliser l'appariement des opérations, le matchmaker utilise différentes techniques :

- basées sur une approche logique en utilisant les annotations sémantiques spécifiées par les attributs modelReference dans les éléments annotés,
- basées sur des approches syntaxiques : Loss-of-Information, Extended Jaccard [31], Cosine [Garcia, 2006] et Jensen-Shannon [Fuglede and Topsoe, 2004],
- basées sur des approches hybrides :
  - "Compensative" : le matchmaker utilise la similarité syntaxique lorsqu'aucun degré d'appariement logique ne peut être associé à un service,
  - "Integrative" : pour remédier aux problèmes liés aux faux positifs. Il ne faut pas uniquement prendre en compte la similarité syntaxique lorsqu'un appariement logique échoue, mais plutôt la considérer comme une contrainte conjonctive à chaque degré d'appariement logique.

Les degrés d'appariement dans une découverte de services avec SAWSDL-MX [Klusch and Kapahnke, 2008] peuvent être distingués en deux catégories :

- Les degrés d'appariement calculés par l'approche logique : Exact, Plug-in, Subsumes et Subsumed-by,
- Les degrés d'appariement calculés par l'approche hybride :Subsumed-by (s'il a besoin de calcul de similarité syntaxique additionnelle), Nearest-neighbour (pour compenser les appariements faux négatifs).

Au moment de l'agrégation des degrés d'appariement et du calcul d'appariement global entre deux services, l'approche opte pour la plus petite valeur. Ceci est appelé le principe d'agrégation "Valeur minimale" (notée Min). Exemple, dans la Figure 3.7, l'agrégation des deux degrés Exact et Plug-in, donne lieu au degré final d'appariement Plug-in. Nous démontrerons, ci-après dans l'examen des approches, que cette agrégation par valeur minimale présente des lacunes et ne permet pas parfois de déceler les meilleurs services Web pour une requête au moment de la découverte.

Dans [Hadjila et al., 2017] les auteurs utilisent la relation du sur-classement (Out ranking) pour fusionner les classements offerts par un ensemble de mesures de similarités. Les expérimentations menées montrent que les seuils de concordances et discordances optimaux convergent vers le modèle de condorcet (algorithme de vote majoritaire).

[Skoutas et al., 2009] utilisent les principes de Probabilistic Skyline *p-Skyline* et proposent un classement pour les données (principes de *Top-k dominating*). En effet, Skoutas et al. ont défini trois façons de calculer le score de dominance *(the dominated score, the dominating score et the dominance score)* pour classer les services selon leurs vecteurs de données.

- le premier score est appelé *the dominated score* ou *le score dominé* : il est basé sur la relation de Pareto *est dominé par* et considère le nombre de services qui dominent un vecteur u dans sa classe.
- le deuxième score est appelé *the dominating score* ou *le score dominant* : il est basé sur la relation de Pareto 'domine'.

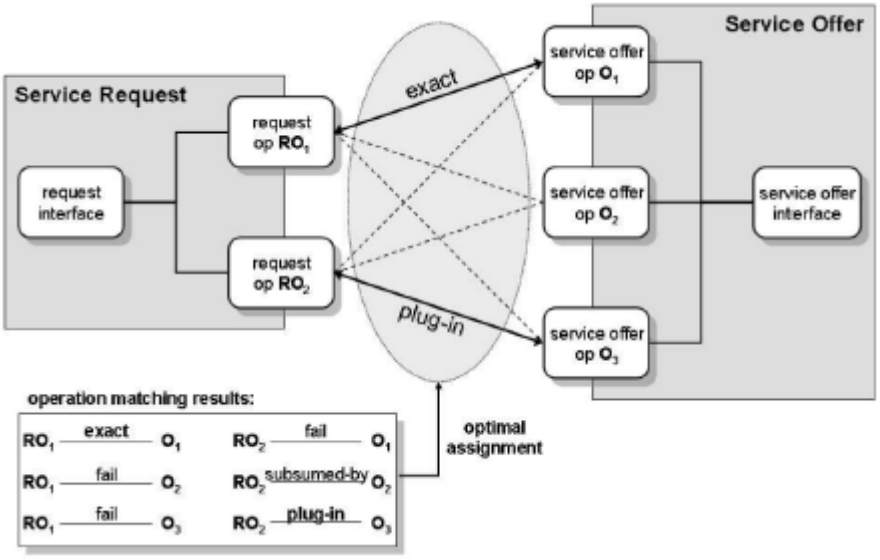

Figure *3.7:* *Exemple d'application du principe d'agrégation Valeur minimale [Klusch and Kapahnke, 2008]*

- Le troisième score est appelé *the dominance score* ou *le score de dominance.*

Les auteurs proposent ensuite 3 algorithmes TKDD, TKTD, et TKM dont chacun présente un classement de services qui utilisent respectivement les trois scores Dominated score, Dominating score et Dominance score.

3.3 Etude comparative

La classification des travaux existants dans la littérature selon le raisonnement adopté pour effectuer le matching est une brique importante dans le processus de leur évaluation. Elle renseigne sur leur efficacité qui dépend essentiellement de la technique de matching qu'elles adoptent. Cependant, elle ne s'avère pas suffisante pour évaluer leur adaptation au contexte des services web. Pour ce faire, d'autres critères doivent être pris en considération. Dans ce qui suit, nous identifions les critères qui nous ont permis de comparer les divers travaux présentés auparavant.

- **Les éléments à apparier**

  La pertinence d'une approche de découverte dépend souvent des éléments à apparier entre la description du service Web requis et les descriptions des services Web offerts (e.g. dans l'extension SAWSDL du standard WSDL : les opérations, les inputs et les outputs).

- **Degrés d'appariement**

  La pertinence d'une approche de découverte dépend directement de la manière avec laquelle on évalue l'appariement entre les éléments de la description du service web requis et les descriptions des services web offerts. Le critère des "degrés d'appariement" est principal pour évaluer une grande partie de la pertinence d'une approche : plus on identifie précisément et fidèlement la similitude entre les éléments des deux services, requis et offert, plus le résultat de découverte est pertinent. Toutefois, il faut souligner que ce n'est pas le nombre de

degrés qui importe, mais c'est plutôt le nombre de degrés significatifs qui fait la différence, vu l'information qu'ils peuvent apporter. Plus les degrés sont bien élaborés et distincts, plus l'interprétation de l'appariement (et par conséquent la découverte) est pertinente.

- **Les propriétés non fonctionnelles (Qualité de Service)**
  Cette caractéristique permet de spécifier si une approche de découverte prend ou non en compte la qualité de service dans le processus de découverte. Considérant les aspects de qualité de service au moment de décider quels services à inclure dans un schéma de découverte des services est important lorsque les exigences fonctionnelles sont satisfaites par plus d'un service.
- **Automatisation**
  D'après notre étude des différentes approches de découverte, nous avons bien constaté que la recherche des services pertinents qui participent dans une composition doit être au moins partiellement (s'il n'est pas entièrement) automatique afin de minimiser l'intervention de l'utilisateur et d'accélérer le processus de découverte. L'exigence d'une assistance humaine est de plus en plus inacceptable dans un tel domaine (de plus en plus automatique). De plus, une interprétation humaine peut ne pas être assez pertinente, cela nécessite un bon niveau d'expertise.
- **Modèle sémantique**
  Désigne l'ontologie ou le langage utilisé pour décrire les services Web. Le formalisme adopté représente ces services conformément à l'ontologie ou au langage utilisé. Pour pouvoir réaliser des appariements sémantiques, la plupart des travaux existants recourent à l'utilisation des ontologies OWL-S ou WSMO comme modèles sémantiques pour décrire les services Web. Cependant, il en résulte que l'approche proposée est exclusivement dépendante de l'ontologie choisie et donc spécifique pour la découverte d'un type précis de services web.
- **Support de plusieurs ontologies**
  La plupart des approches utilisent des ontologies comme modèles sémantiques pour décrire leurs services web. Le critère "support de plusieurs ontologies" est important dans l'évaluation d'une approche. Une approche fermée est une approche qui dépend strictement d'un type particulier d'ontologies et ne tolère pas d'autres types. Une approche ouverte est une approche qui ne dépend pas de types particuliers d'ontologies.
- **Contexte**
  On trouve dans la littérature beaucoup de définitions de ce qu'est un contexte, [Abowd et al., 1999] définissent le contexte comme étant : "toutes les informations pouvant être utilisées pour caractériser la situation d'une entité, où une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes". Par cette définition chacune des deux entités, utilisateur et service Web, a son propre contexte. Le contexte d'un service web peut grouper la localisation du service (restriction d'usage géographique du service), le coût d'utilisation, la catégorie de service, etc. Le contexte de l'utilisateur peut être formé de la localisation de l'utilisateur, son profil,etc.

Le Tableau 3.1 présente une synthèse des résultats de l'étude comparative des travaux que nous avons présentés auparavant dans ce chapitre.

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Paolucci et al., 2002]	Logique	Inputs et Outputs	Exact, Plug-in, Subsume et Fail	/	DAML	Semi-automatique (intervention de l'utilisateur dans le choix du seuil)	/	/
[Keller et al., 2005]	Logique	Inputs, Outputs et Goals	Match, Poss-Match, ParMatch, PossPar-Match, NoMatch	/	Pas d'ontologie : Goals,Services abstraits, Capacités de service	Non spécifié	/	/
[Jaeger et al., 2005]	Logique	Inputs et Outputs	Equivalent, Subsume, Unknown, Fail	/	OWL-S	semi-automatique(Choix de contraintes d'appariement et Sélection du service)	/	/
[Benatallah et al., 2005b]	Logique	Inputs et Outputs	Pas d'échelle de degrés	/	Logique de description	Non spécifié	/	/

Tableau 3.1 – *Continued from previous page*

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Doulkeridis et al., 2006]	logique	Inputs, Outputs, Catégories et Opérations	Pas d'échelle de degrés	/	SPARQL	Automatique	/	Prise en compte du contexte de l'utilisateur
[Küster and König-Ries, 2008]	Logique	Inputs, Outputs et Goals	PossMatch, ParMatch, PossParMatch, NoMatch, RelationMatch, ExcessMatch	/	Pas d'ontologie	Non spécifié	/	/
[Schumacher et al., 2008]	Non-logique	Eléments du profil du service	Pas d'échelle de degrés	/	OWL-S	semi-automatique(Choix du seuil par l'utilisateur)	/	/

Tableau 3.1 – *Continued from previous page*

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Klein and König-Ries, 2004]	Non-Logique	Noms des noeuds et Noms des concepts	Pas d'échelle de degrés	/	Service DSD (Diane Service Description)	semi-automatique(sélection du service)	/	/
[Plebani and Pernici, 2009]	Non-Logique	Inputs et outputs	Pas d'échelle de degrés	/	SAWSDL	Non spécifié	/	/
[Li et al., 2007]	Non-Logique	Noms des noeuds et Noms des concepts	Non spécifié	/	WSMO	Non spécifié	/	/
[Skoutas et al., 2007]	Non-Logique	Inputs, Outputs, préconditions et effets	Pas d'échelle de degrés	/	OWL-S	semi-automatique	/	/
[Boukhadra et al., 2016]	Non-Logique	Inputs, Outputs, préconditions et effets	Exact, Plug-in, Subsume et Fail	/	OWL-S	Automatique	/	/

Tableau 3.1 – *Continued from previous page*

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Chabeb, 2011]	Non-Logique	Inputs, Outputs, préconditions et effets	Exact,Subsume, Subsume-by, Has-same-class, Unclassified, fail	/	YASAWSDL (Yet Another Semantic Annotation for WSDL)	Semi-automatique	/	/
[Klusch et al., 2006]	Hybride	Inputs et Outputs	Equivalent, Subsume, Subsumedby, Logic-based fail, Nearest-neighbor et Fai	/	OWL-S	Semi-automatique (choix du seuil)	/	/
[Kiefer, 2009]	Hybride	Inputs et Outputs	Pas d'échelle de degrés	/	OWL-S	Non-spécifié	/	/

Tableau 3.1 – *Continued from previous page*

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Bertoli et al., 2004]	Hybride	Inputs, Outputs, Catégories et Opérations	Pas d'échelle de degrés	/	OWL-S et OWL-DL	semi-automatique	/	/
[Hadjila et al., 2017]	Hybride	Inputs, Outputs	Pas d'échelle de degrés	/	OWL-S	automatique	/	/
[Samir et al., 2017]	Hybride	Inputs, Outputs	Pas d'échelle de degrés	/	/	semi-automatique	/	Oui
[Kaufer and Klusch, 2006]	Hybride	Service Goals, Préconditions et effets	Equivalent, Plug-in, Inverse Plug-in, Intersection, Fuzzy similarity, Neutral, Disjunction	/	WSMO	semi-automatique (choix du seuil)	/	/

Tableau 3.1 – *Continued from previous page*

Travaux étudiés	**Catégorie de l'approche**	**Les éléments à apparier**	**Degrés d'appariement**	**Les propriétés non fonctinnelles**	**Modèle sémantique**	**Automatisation**	**Support de plusieurs ontologies**	**Contexte**
[Klusch and Kapahnke, 2008]	Hybride	Inputs, Outputs, et Opérations	Equivalent, Subsume, Subsumedby, Nearest-neighbor et Fail	/	SAWSDL	Non spécifié	/	/

Tableau 3.1: *Tableau comparatif des approches de découverte de services Web*

3.4 Conclusion

Dans ce chapitre, nous avons présenté l'état de l'art de la découverte des services web sémantiques, et d'aprés notre étude, nous avons constaté que les approches de matching hybrides ont une large supériorité par rapport aux autres approches (logiques et non logiques), mais leurs difficultés résident dans le choix des heuristiques d'agrégation (qui peuvent se baser sur l'apprentissage automatique, la théorie des probabilités, la logique floue, la théorie de vote, etc).

Dans notre contribution, nous avons utilisé des heuristiques appartenant à la logique floue ainsi que l'apprentissage automatique. Dans le chapitre suivant nous présentons les travaux de résolution de problème de sélection.

Chapitre 4

La sélection des services web composés à base de QoS

Sommaire

4.1 Introduction

Selon le paradigme SOA (Service Oriented Architecture), les applications composites sont spécifiées comme des processus abstraits constitués d'un ensemble de tâches. Pour chaque tâche, on peut découvrir plusieurs services web qui présentent les mêmes fonctionnalités mais qui différent dans leurs paramètres non fonctionnels (les paramètres de qualité de service). On se trouve alors face à un problème de sélection de services web. Pour mieux clarifier notre problématique, considérons l'exemple de planification de voyage dans lequel un utilisateur cherche à trouver 4 types de services, un service de réservation de vol, un service de réservation d'hôtel, un service d'allocation de voiture et un service d'enregistrement dans une conférence. Pour cela il doit sélectionner un seul service (ou entreprise) de chaque catégorie, en utilisant les critères de QoS (réputation, fiabilité, coûts, etc). De plus, l'utilisateur exige des contraintes globales sur chaque critère de QoS, par exemple le coût total du service composite ne doit pas excéder une certaine limite.

La sélection de compositions de services basée sur la QoS "QoS-aware web service composition" vise à trouver la meilleure combinaison de services web qui satisfont les exigences de l'utilisateur (des contraintes en terme de QoS). Ce problème est modélisé comme un problème d'optimisation, il est associé à la variante du problème de sac à dos multidimensionnel à choix multiple (MMKP :multi-dimension multi-choice knapsack problem) [Yu et al., 2007]. Cette classe de problèmes appartient à la classe des problèmes combinatoires qui sont identifiés comme des problèmes NP-difficile [Parra-Hernandez and Dimopoulos, 2005]. Les solutions exactes de ce

problème ont une complexité exponentielle, et donc nous ne pouvons plus garantir les exigences temps réel.

Nous commençons le chapitre par formaliser le problème de la sélection ensuite nous donnons une classification des approches relatives à notre problématique.

4.2 Formalisation du problème

De façon plus formelle, nous modélisons le problème comme suit. Soient :

- $CA = \{S_1, ..., S_n\}$, une composition abstraite qui représente la requête de l'utilisateur, c-à-d les n classes de services S_i à consommer.
- $cg = \{cg_1, .., cg_n\}$, un ensemble de contraintes globales définies par l'utilisateur.
- $C = \{s_{i1}, ..., s_{nj}\}$ une composition concrète, c-à-d nous remplaçons chaque classe S_i par un service concret $s_{ij} \in S_i$
- $q(s_{ij}) = \{q_1, .., q_m\}$ vecteur des qualités du service s_{ij}.
- $Q(C) = \{Q_1(C), .., Q_m(C)\}$vecteur des qualités du service de la composition C.
- $U(C)$ la fonction objectif (nommée aussi fitness) à minimiser ou maximiser. Elle peut être :
  - multi-objectif : dans ce cas, $U(C) = \{U_1..U_k\}$ où U_k représente la $k_{ième}$ fonction à optimiser.
  - mono-objectif : dans ce cas, les m attributs de QoS sont agrégés en une seule valeur.
- Le problème de sélection est alors :
  - Global : si nous cherchons l'ensemble des compositions qui vont satisfaire les contraintes globales c-à-d $Q_k \geq cg_k, \forall k \in \{1..n\}$.
  - Local : si nous cherchons l'ensemble des compositions sans prendre en considération les contraintes globales cg à optimiser, nous cherchons à optimiser les paramètres QoS dans chaque classe abstraite indépendamment des autres. Si nous supposons que le nombre de candidats par classe est L, alors le nombre global de compositions possibles est L^n.

4.3 Etat de l'art

Plusieurs travaux ont été proposés pour résoudre le problème *QoS-aware service composition*. Dans la littérature, on trouve plusieurs classifications des approches selon que la fonction à optimiser est mono-objectif, multiobjectif ou hybride [Hadjila, 2014], selon que la stratégie de sélection utilisée soit locale ou globale [Alrifai et al., 2012] ou selon l'approche utilisée. Dans ce dernier cas on distingue 04 grandes classes [Jatoth et al., 2015], [Khanouche et al., 2016] :

1. Les méthodes exactes (non heuristiques).
2. Les méthodes heuristiques (approximatives).
3. Les méthodes basées sur les méta-heuristiques (approximatives).
4. Les approches basées sur la dominance au sens de Pareto.

Dans ce qui suit nous présentons en détail les travaux de chaque catégorie en s'appuyant sur plusieurs papiers de la littérature du problème de sélection des services composites [KAMAL et al., 2014] et [Jatoth et al., 2015].

4.3.1 Approches exactes

Les méthodes exactes résolvent le problème d'optimisation de manière optimale. Elles utilisent les techniques de la programmation par contraintes ou la programmation dynamique, la programmation linéaire entière (Integer Lineaire Porogramming, ILP) ou encore les techniques de programmation entière et mixte (Mixed Integer Programming ou MIP). Ces approches donnent des résultats optimaux mais elles ont un temps d'exécution exponentiel.

Zeng et al. [Zeng et al., 2003], [Zeng et al., 2004] se concentrent sur la sélection dynamique basée sur la qualité de service. Les auteurs utilisent la planification globale pour trouver les meilleures compositions de service web. Ils utilisent les techniques de programmation entière et mixtes (Mixed Integer Programming ou MIP) [Nemhauser and Wolsey, 1988] pour trouver la composition optimale des services.

Dans la même optique, Ardagna et al. [Ardagna and Pernici, 2005], [Ardagna and Pernici, 2007] étendent le modèle de programmation linéaire pour inclure les contraintes locales. Dans ce modèle, les contraintes globales sont exprimées sur la composition entière, et par l'utilisateur final, tandis que les contraintes locales peuvent être spécifiées par le concepteur de la composition (au niveau des classes).

Alrifai et al. [Alrifai and Risse, 2009] ont proposé une solution en deux étapes : premièrement, ils utilisent la programmation entière et mixte (MIP) pour trouver la décomposition optimale des contraintes QoS globales en contraintes locales. Deuxièmement, ils utilisent une sélection locale distribuée pour trouver les meilleurs services web qui satisfont ces contraintes locales.

Dans [Rosenberg et al., 2009], les auteurs ont proposé un modèle qui traite la sélection comme un problème d'optimisation par contraintes relaxées (Constraint Optimization Problem). Ils ont relaxé le problème par une pondération de contraintes, le but est de trouver une solution maximisant une fonction de contraintes pondérées. L'idée principale dans leur proposition est qu'au lieu d'avoir toutes les contraintes à satisfaire, certaines contraintes sont flexibles et deviennent facultatives.

Gao et al. [Gao et al., 2006] proposent la programmation entière en prenant en compte le conflit entre les services (qui appartiennent à des classes différentes). Les conflits signifient que l'utilisation d'un service i de le classe X n'est pas compatible avec le service j de la classe Y. L'expérimentation montre que l'ajout des conflits, provoque une augmentation de 13 % dans le temps d'exécution de l'approche à base de MIP.

Xu et al. [Xu et al., 2011] proposent une méthode de sélection de services web à base QoS en considérant une variation de valeurs de QoS en fonction de l'intervalle de temps (les fluctuations sont données par les fournisseurs de services). Pour cela l'utilisateur introduit les poids et l'intervalle d'intérêt qui peut englober plusieurs intervalles ayant des valeurs de QoS différentes, mais ils ne gèrent pas les contraintes globales. Pour cela, ils utilisent la programmation dynamique qui découpe la composition en une série de paires de services (02 classes consécutives), l'algorithme cherche la paire concrète qui minimise la fonction objectif.

[Gabrel et al., 2014] proposent une méthode pour trouver la solution pour une composition de services en utilisant un graphe de dépendance et 0-1 LIP. Liu et al. [Liu et al., 2009] proposent une méthode basée sur l'enveloppe convexe (convex-hull) et appliquent la méthode de décision multi-critères, *multiple criteria decision making (MCDM)* pour fusionner les ressources multiples pour les contraintes globales et locales.

L'auteur dans [Maros, 2012] confirme que les méthodes basée sur la programmation entière sont efficaces si le nombre de service L est petit, par contre si L dépasse une certaine limite alors le temps d'exécution n'est plus raisonnable et donc ces méthodes ne passent pas à l'échelle.

4.3.2 Approches heuristiques (approximatives)

Les approches heuristiques utilisent des règles et des fonctions empiriques adaptées à des problèmes d'optimisation spécifiques. Elles essayent de trouver une bonne solution du problème dans un temps acceptable. La solution obtenue n'est pas la solution optimale mais une solution approchée [Yang, 2009].

Les algorithmes heuristiques tirent avantage des particularités du problème, contrairement aux algorithmes exacts qui prennent un grand temps pour obtenir la solution optimale, les algorithmes heuristiques obtiennent des solutions presque optimales dans un temps raisonnable [Jatoth et al., 2015].

Plusieurs travaux sont inspirés des algorithmes heuristiques. [Khan, 1998] propose l'un des premiers algorithmes pour la résolution du problème de sac à dos (et sa version *MMKP-multi-dimensional multiple choice knapsack problem-*). L'auteur propose une heuristique nommée UHE pour sa résolution, UHE utilise une mesure appelée la consommation des ressources agrégées, pour mettre à jour un élément de chaque groupe à chaque tour de sélection. Les auteurs dans [Akbar et al., 2001] ont modifié l'heuristique en créant M-UHE, ils proposent une étape de prétraitement pour trouver une solution réalisable et une étape de post-traitement pour améliorer la valeur totale de la solution, pour cela ils font des améliorations (upgrades) en mettant à jour un élément qui augmente l'utilité totale, ensuite ils dégradent la solution avec un ou plusieurs coups (downgrades), i.e. ils sélectionnent un élément qui diminue la valeur de l'utilité totale. [Akbar et al., 2006] proposent une autre heuristique, C-UHE, et évaluent sa performance et son optimalité contre plusieurs heuristiques, y compris l'algorithme M-UHE. Les résultats montrent que la C-UHE est meilleure par rapport à M-UHE en termes de temps d'exécution. Cependant, les expériences montrent également que M-UHE est la meilleure en termes de degré d'optimalité, tandis que l'optimalité de C-HEU diminue à mesure que le nombre de candidats par classe augmente. Les expériences montrent aussi que l'algorithme C-UHE fonctionne mieux dans le cas où l'objectif à maximiser (par exemple, la valeur d'utilité de la composition de services) n'est pas proportionnel aux besoins en ressources (i.e., les valeurs de QoS des services Web). Et de ce fait elle ne peut être appliquée pour la sélection de services composés à base de qualité (puisque l'utilité dépend des qualités de service).

Une version modifiée de l'algorithme M-UHE, appelée WS-UHE, est conçue par [Yu et al., 2007]. Les auteurs proposent deux modèles pour le problème de sélection de service :

(1) un modèle combinatoire et (2) un modèle de graphe. Un algorithme heuristique est introduit pour chaque modèle : l'algorithme WS-HEU pour le modèle combinatoire et le MCSP-K pour le modèle orienté graphe. La complexité temporelle de WS-UHE est polynomiale, alors que la complexité du MCSP-K est exponentielle. En dépit de l'amélioration significative de ces algorithmes par rapport à solutions exactes, les deux algorithmes ne passent plus à l'échelle (i.e. si le nombre de services Web croit de manière dramatique alors le temps d'exécution n'est plus temps réel). De plus, l'algorithme WS-UHE n'est pas adapté au caractère distribué des services web.

Ceci est dû au fait que WS-HEU applique des améliorations sur une composition de services dont les valeurs de QoS proviennent de plusieurs facilitateurs (Chaque facilitateur gère une classe donnée).

Klein et al. [Klein et al., 2011] utilisent l'algorithme hill climbing (Méthode de Descente) pour réduire la complexité de temps de calcul et comparent leur méthode avec celle utilisant LIP. Qi et al. [Qi et al., 2010] ont présenté une méthode d'optimisation locale et d'énumération pour trouver les candidats locaux et ensuite les combiner pour trouver la solution optimale. Diana et al. [Comes et al., 2010] ont proposé un modèle de recherche heuristique pour la sélection de compositions services basée sur la QoS. Li et al. [Li et al., 2014] ont discuté une approche efficace et fiable pour la sélection de la composition optimale des services a base de paramètres de confiance.

Les auteurs dans [Xia et al., 2011] proposent une sélection globale de composition de services en adoptant 4 structures de flux de contrôle : parallélisme, séquence, choix conditionnels, et les boucles, la requête est formalisée en BPEL, elle possède 5 critères de QoS. L'algorithme est baptisé qssac, il peut donner un résultat proche de l'optimal. Pour réduire le temps d'exécution, ils groupent les services similaires en termes de QoS à l'aide d'un algorithme nommé optics (qui est moins sensible aux bruits et ayant peu de problèmes d'initialisation, optics se base sur la densité), on obtient M classes (et leurs représentants) pour chaque tâche, ensuite l'algorithme énumère toutes les compositions possibles pour 2 ou K tâches du document BPEL, et les trie à l'aide d'une fonction objectif. D'autres heuristiques ont été proposées par [Berbner et al., 2006] : H1 RELAX IP, H2 SWAP,et H3 ANNEAL pour trouver la solution optimale et améliorer l'efficacité dans le problème de sélection des services web. Luo et al. [Luo et al., 2011] ont proposé un algorithme Heuristic HCE pour la composition des web services en tenant compte des contraintes globales de l'utilisateur.

Dans [Kim et al., 2016] les auteurs considèrent la QoS comme des données non déterministes, pour sélectionner les compositions quasi-optimales l'approche adopte la probabilité de satisfaire les contraintes globales comme fonction objectif à maximiser, les auteurs divisent les contraintes globales en contraintes locales en évitant les valeurs aberrantes (outlier). La division utilise les QoS minimales de chaque classe abstraite durant le calcul des contraintes locales.

Plusieurs autres travaux ont proposé plusieurs algorithmes heuristiques [Luo et al., 2008], [Li et al., 2010], [Li and Wen, 2012] et [Li et al., 2013] pour réduire la complexité du temps liée aux contraintes globales et locales dans la sélection de la composition optimale.

4.3.3 Approches à base de métaheuristiques

Contrairement aux algorithmes heuristiques spécifiques à un problème donné, les méta-heuristiques fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes d'optimisation discrète. Les méta-heuristiques sont les méthodes d'optimisation les plus utilisées pour la résolution des problèmes combinatoires NP-difficile. Les méta-heuristiques les plus connues sont les algorithmes d'optimisation par essaim particulaire (PSO :Particule Swarm Optimization) [Eberhart and Kennedy, 1995], les algorithmes d'optimisation par colonie de fourmis (ACO : Ant Colony Optimization) [Dorigo et al., 1996], les algorithmes génétiques (GA : Genetic Algorithm) [Holland, 1975], l'algorithme de recuit simulé (SA : Simulated Annealing) [Kirkpatrick et al., 1983], la recherche tabou (TS : Tabu Search) [Glover, 1986], les algorithmes par colonies d'abeilles artificielles (ABC : Arti-

ficial Bee Colony) [Karaboga, 2005], la recherche coucou (CS : Cuckoo Search) [Yang and Deb, 2009] et la recherche harmonique (HS :Harmony Search) [Geem et al., 2001].

Plusieurs chercheurs ont proposé des approches basées sur des algorithmes génétiques pour la sélection des services web prenant en compte les contraintes globales [Canfora et al., 2005], [Jaeger and Mühl, 2007], [Rojas et al., 2002], [Tang and Ai, 2010], [Yilmaz and Karagoz, 2014] et [Yu et al., 2013]. Dans [Canfora et al., 2005] et [Wang and Hou, 2008], les auteurs présentent un algorithme génétique qui instancie les services abstraits avec des composants concrets. Dans les deux travaux, les points de mutation sont choisis aléatoirement. Le codage des chromosomes utilise la représentation entière pour [Canfora et al., 2005] et la représentation binaire pour [Wang and Hou, 2008].

Le travail de [Canfora et al., 2005] adopte une fonction fitness dynamique qui pénalise les chromosomes lorsqu'ils violent les contraintes globales, et de ce fait l'approche favorise la convergence rapide vers un optimum local. Par opposition le travail [Wang and Hou, 2008], propose 3 fonctions objectif (une pour chaque critère de QoS).

Dans [Jaeger and Mühl, 2007], les auteurs implémentent un algorithme génétique et se sont focalisé sur la capacité d'optimisation de ce dernier en étudiant l'impact d'influence des différents paramètres de l'algorithme proposé comme le taux de mutation. Ils comparent l'algorithme génétique avec celui de 'branch and bound'. Les résultats ont montré que les solutions obtenues dépendent des paramètres de configuration de l'algorithme qui influencent ses performances. Dans [Yu et al., 2013], les auteurs ont proposé un algorithme génétique dans lequel ils ont présenté le problème sous forme d'un arbre. Liu et al. [Liu et al., 2010] ont adopté un algorithme génétique amélioré utilisant l'optimisation par colonies de fourmi pour sélectionner les anticorps de la population initiale afin d'améliorer la convergence. Dans [Xiangbing et al., 2012] les auteurs modélisent le problème des services web en utilisant l'ontologie (WSMO) et appliquent ensuite l'algorithme génétique qui minimise le temps de recherche pour trouver l'optimum. Mais malheureusement, les algorithmes génétiques souffrent des problèmes de passage à l'échelle, (le codage des chromosomes est fixe), en plus la possibilité de stagner dans un optimum local est toujours présente. Pour pallier à ce problème, des chercheurs ont combiné les algorithmes génétiques avec d'autres algorithmes comme la recherche Tabou dans [Bahadori et al., 2009] et [Parejo et al., 2008]. La recherche taboue intervient après le croisement et la mutation, elle sert à choisir le meilleur chromosome (ayant la meilleure fitness) issu du voisinage des enfants, en outre elle permet l'évitement des optima locaux.

Plusieurs autres travaux utilisent PSO (Particule Swarm Optimization) comme [Long and Gui, 2009], [Wang and He, 2010], [Liu et al., 2007], [Ludwig, 2012] et [Wang et al., 2010b]. Dans [Chen and Wang, 2007] les auteurs utilisent PSO [Eberhart and Kennedy, 1995], et confirment qu'il est plus rapide et plus scalable par rapport aux algorithmes génétiques. Dans le même travail les auteurs modifient les services concrets des particules en changeant la vélocité avec des opérateurs tels que l'addition, la soustraction, la multiplication. Dans de nombreux problèmes scientifiques, nous avons besoin de trouver plus d'une solution optimale. La version PSO originale se concentre sur la recherche d'une seule solution. Les algorithmes évolutionnistes s'appuient sur des méthodes de nichage permettant de trouver des solutions multiples pour des problèmes multimodals [Brits et al., 2007]. Dans [Liao et al., 2011] les auteurs présentent un l'algorithme Niche PSO et considèrent le problème de sélection comme un problème multimodal, l'algorithme proposé permet de localiser et de raffiner plusieurs solutions en même temps en considérant de multiples contraintes globales. Liu. et

al [Liu et al., 2011] ont présenté un algorithme d'optimisation d'essaim de particule quantique 'hqpso' pour résoudre le problème de sélection. Dans [Liu and Yin, 2009], les auteurs proposent les réseaux de pétri coloré (CPN) et une version discrète de PSO. [Zhao et al., 2012] adoptent la version discrète de PSO qui est basée sur les systèmes immunitaires artificiels. Une méthode hybride combinant PSO avec la méta-heuristique recuit simulé est proposée dans [Wang et al., 2010a] pour sélectionner la composition optimale basée sur les attributs QoS. La position d'une particule dans PSO représente les solutions potentielles, tandis que la vélocité est utilisée pour modifier une solution. Pour éviter le problème de stagnation prématurée dans une solution (localement optimale), une stratégie de recuit simulé est adoptée pour créer de nouvelles solutions de composition en perturbant la solution initiale.

D'autre travaux utilisent les systèmes immunitaires artificiels (SIA) [Xu and Reiff-Marganiec, 2008], [Yan et al., 2006], [Zhao et al., 2014] comme la sélection clonale ou la sélection négative. Dans [Pop et al., 2009], les auteurs proposent une approche qui combine les techniques de planification et un algorithme de sélection clonale pour trouver la composition optimale en prenant en considération les propriétés sémantiques des services web. [Hadjila et al., 2012] présentent l'algorithme de sélection clonale pour la résolution du problème de sélection. Ils considèrent une fonction monoobjectif qui manipule 5 paramètres QoS. Les auteurs montrent que la sélection clonale s'avère plus intéressante que l'algorithme génétique. [Zhao et al., 2014] proposent une première adaptation de la sélection négative au problème de sélection et présente l'algorithme 'NSA'. Ils proposent ensuite une amélioration de l'algorithme initiale et propos NSA++.

Plusieurs travaux adoptent une optimisation par colonie de fourmis ACO pour résoudre le problème de composition [Yang et al., 2010], [Yunwu, 2009], [Pop et al., 2010] et [Xia et al., 2008]. Ils modélisent le problème comme un graphe direct acyclique pondéré où le point de départ dénote le nid de fourmis, le point cible dénote des sources alimentaires et les contraintes QoS dénotent les poids des arc. Dans [Xia et al., 2008], les auteurs représentent la composition en termes de graphes tels que les nœuds sont des structures d'orchestration et les arcs sont des services, ils utilisent toujours l'ACO, mais ils affirment que la présence d'un seul type de phéromone n'est pas suffisante pour traiter plusieurs attributs de QOS.

Dans [Yang et al., 2010] les auteurs combinent l'ACO et les algorithmes génétiques, pour la sélection, l'algorithme génétique a pour objectif d'ajuster les paramètres d'ACO, qui sera utilisé dans la sélection, les auteurs notent une accélération considérable au niveau du temps d'exécution. [Pop et al., 2010] proposent une méthode hybride basée sur l'optimisation par colonies de fourmi et une structure de graphe de planification en prenant en considération l'aspect sémantique et l'aspect non fonctionnel des services web. [Li and Yan-Xiang, 2010] proposent l'algorithme multi-objectif (MOCACO) en combinant la théorie de chaos avec un algorithme ACO et en tenant en compte les exigences de l'utilisateur en terme de QOS.

Plusieurs travaux [Esfahani et al., 2012], [Jafarpour and Khayyambashi, 2010] et [Merzoug et al., 2014] utilisent l'optimisation par Harmonie pour résoudre le probleme de sélection. Dans [Esfahani et al., 2012] les auteurs proposent un mécanisme de sélection efficace basé sur un algorithme de recherche d'harmonie social en considérant les attributs de QoS. Afin de démontrer l'efficacité de l'algorithme proposé, il est comparé avec l'algorithme génétique et les résultats correspondants ont révélé que le présent algorithme consomme moins de temps en recherchant la solution optimale. [Merzoug et al., 2014] proposent une approche mono-objective basée sur

l'algorithme de recherche d'harmonie, permettant d'obtenir une composition presque optimale, en tirant parti de plusieurs opérateurs tels que la recherche locale, la sélection aléatoire. Dans [Bekkouche et al., 2017] les auteurs proposent une approche basée sur les techniques de planification et la recherche harmonique. Dans leurs approches, l'espace de recherche est modélisée comme un graphe de planification, la recherche de la solution quasi-optimale est basée sur les liens sémantiques qui se trouvent entre les services qui composent la solution ainsi que les attributs QoS. Ils ont aussi comparé la version originale de l'algorithme avec deux autres variantes : l'algorithme IHS (Improved Harmony Search) et l'algorithme GHS (Global Best Harmony Search). Les résultats trouvés sont très prometteurs.

Certains chercheurs ont adopté la recherche par colonies d'abeilles artificielles (ABC) comme dans [Hadjila et al., 2013], [Chifu et al., 2010] et [Kousalya et al., 2011], la recherche par coucou (CS) comme dans [Chifu et al., 2011] ou la recherche par l'auto migration organisée comme dans [Amine, 2017] et [Krithiga, 2012].

Dans [Zhu et al., 2017] les auteurs se focalisent sur la sélection des services web qui satisfont les exigences d'un groupe d'utilisateurs. La solution proposée est constituée de 2 étapes :

Dans la première phase les auteurs retiennent les solutions pareto-optimales à l'aide de l'heuristique nommée "Polynomial Time Approximation Scheme". La deuxième phase permet de filtrer les solutions optimales à l'aide de l'algorithme des colonies d'abeilles.

4.3.4 Les approches basées sur la dominance au sens de Pareto

Le problème de sélection des services web composés à base de QoS est résolu en utilisant l'optimalité de Pareto. Un ensemble Pareto est un ensemble de solutions réalisables qui ont au moins un objectif optimisé tout en gardant les autres inchangés [Chankong and Haimes, 2008].

Dans [Chen et al., 2015], est proposé DPSA (pour Distributed Partial Selection Algorithm) un algorithme de composition de services utilisant une approche de sélection partielle. Cet algorithme procède d'abord à une validation des contraintes locales de QoS où sont éliminés les services candidats violant les contraintes de QoS imposées au niveau de chaque service abstrait de la composition. Ensuite, en se basant sur la relation de dominance, les services candidats non prometteurs en termes de QoS, c'est-à-dire, dominés par d'autres services candidats de la même classe, ne sont pas retenus pour le processus de composition. Ceci permet de réduire l'espace de recherche des compositions. Les services candidats dominants forment l'ensemble Pareto Optimal de services à partir duquel l'ensemble des solutions réalisables est déterminé. En effet, étant donnée une composition abstraite constituée de m classes de services, une solution réalisable est toute composition pour laquelle : i) un service candidat est sélectionné pour chaque classe de services, et ii) toutes les contraintes globales de QoS sont satisfaites. En d'autres termes, les compositions de services qui violent les contraintes globales imposées au niveau du service composite, sont considérées comme non réalisables et sont par conséquent supprimées.

L'analyse des travaux de la littérature montre que le nombre de compositions réalisables augmente exponentiellement avec le nombre de services candidats et le nombre de paramètres de QoS. Ce qui rend l'algorithme DPSA mal adapté à la

composition dans des environnements de services à grande échelle.

Les auteurs dans [Claro, 2006] adoptent 4 fonctions objectifs :

- La minimisation du coût
- La minimisation de la durée,
- La maximisation du chiffre d'affaire,
- La maximisation de la réputation.

Deux expérimentations [Claro, 2006] ont été menées pour évaluer l'algorithme NSGAII ((Nondominated Sorting Genetic Algorithm II). Pour la première expérimentation, le nombre de services de la base varie entre 30 et 60, et le nombre de tâches ou classes varie entre 3 et 5. la taille de la population des chromosomes varie de 10 à 200 individus et le nombre de générations est compris entre 10 et 500. Les résultats montrent que la taille de l'ensemble des skylines varie entre 1 (pour 5 chromosomes) jusqu'à 70 (pour 200 chromosomes). Le temps d'exécution est acceptable puisqu'il est toujours inférieur à 20 secs, pour toutes les populations.

Dans la deuxième expérimentation, la taille des populations a été fixée à 200 et le nombre de générations est égal à 500. Cette expérimentation, montre une corrélation positive entre le nombre de skylines et le nombre de classes/services. Pour 5 classes et 50 services le nombre de skylines=161, et le temps d'exécution=25 sec. Moustafa et al. [Moustafa and Zhang, 2013] et Liang et al. [Feng1/2 et al., 2013] utilisent les algorithmes d'apprentissage par renforcement pour trouver l'ensemble de solutions Pareto optimales qui satisfont les facteurs QoS multiples et les exigences d'utilisateur.

Dans [Trummer et al., 2014], le problème qui consiste à sélectionner les compositions de services Pareto optimales, est traité en utilisant une approche d'approximation polynomiale A-FPTAS (pour Approximations by Fully Polynomial Time Approximation Scheme). En effet, un sous-ensemble de compositions dont les valeurs de QoS sont représentatives de celles des compositions Pareto Optimales, est sélectionné comme une approximation de l'ensemble Pareto de compositions. La metrique $\xi - error$ est utilisée afin d'estimer la précision de l'approximation [Zitzler et al., 2003]. Pour calculer cette métrique, chaque composition de l'ensemble Pareto est associée à la composition de l'ensemble d'approximation ayant des valeurs de QoS comparables. De manière plus précise, la metrique $\xi - error$ est définie comme étant la distance maximale en termes de valeurs de QoS entre une composition Pareto optimale et sa meilleure approximation.

Dans [Gonsalves et al., 2009] les auteurs proposent une optimisation multi objectifs à base d'essaims particulaires, ils considèrent deux fonctions objectifs (le coût et le temps d'attente) et négligent les contraintes globales, ils adoptent la version Gbest de l'algorithme.

Le choix de la meilleure position individuelle et globale est basé sur des règles simples (la dominance), l'expérimentation montre que le front de solutions obtenu, est proche du front de Pareto-optimal réel, mais les auteurs n'ont pas indiqué le nombre de skylines oubliés par erreur, en plus ils n'ont pas donné des comparaisons avec d'autres algorithmes multi-objectifs.

Dans [Chan et al., 2006] les auteurs présentent la notion de k-dominance qui relaxe la notion de Pareto dominance à un sous ensemble de k paramètres, cependant cette approche rend la relation de dominance cyclique (CDR) qui conduit parfois à éliminer certains éléments du Skyline et parfois retourne un ensemble vide. Dans [Lin et al., 2007], les auteurs proposent l'opérateur *Top-k representatif Skyline* qui permet

de sélectionner un ensemble de points, pour qui le nombre de points dominés, par au moins un d'entre eux, soit maximisé. Mais cette approche est plus appropriée pour les données anti-correlées, de plus le *k-representatif Skyline* est considéré comme étant NP-difficile pour un ensemble de données à 3 dimensions.

[Benouaret et al., 2011] proposent une variante de Skyline qui repose sur une extension de Pareto dominance, ils proposent une fuzzification de Pareto dominance appelée '$\alpha-dominance$'. Cette relation permet la comparaison des services de manière plus efficace que la notion de Pareto dominance ; car elle permet de comparer les services qui sont incomparables au sens de Pareto. '$\alpha-dominance$' permet d'étendre la dominance de Pareto avec un degré α. Cette notion sera ensuite utilisée pour déterminer les services qui ne sont pas dominés (avec un degré α) par les autres. Les auteurs utilisent l'algorithme branch and bround Skyline pour calculer le '$\alpha-dominanceSkyline$'. Par contre, les auteurs ne traitent pas l'aspect de la composition ; l'approche qu'ils proposent permet de retourner les services Skyline d'une seule classe. L'intérêt de cette approche est le fait de pouvoir contrôler la taille de services Skyline (étendre ou restreindre le nombre de services) en faisant varier la valeur de α, ceci permet de prendre en considération certains services qui étaient éliminés par la notion de Skyline.

[Alrifai et al., 2010] proposent des approches combinant la sélection multi-objectif (skylines) et mono-objectif. En particulier ils proposent 3 algorithmes à base de skylines Le premier algorithme est nommé « exact skylines », il extrait les services skylines de chaque classe et applique une optimisation globale à base de MIP sur ces résultats. Il faut noter que l'extraction des skylines permet l'élagage de l'espace de recherche, et par la suite, elle assure un gain de temps considérable lors de l'optimisation globale.

Le deuxième algorithme est nommé representative- skylines, il extrait en premier lieu les services skylines de chaque classe, ensuite il réalise un clustering hiérarchique descendant de chaque catégorie de skyline. L'algorithme de clustering applique un découpage binaire descendant des clusters à l'aide l'algorithme k-means, chaque cluster possède un service représentant qui a la plus grande affinité (i.e. la fonction objectif).

La figure suivante présente un clustering hiérarchique des skylines d'une classe donnée pour deux critères le prix (Q1) et le temps d'exécution (Q2).

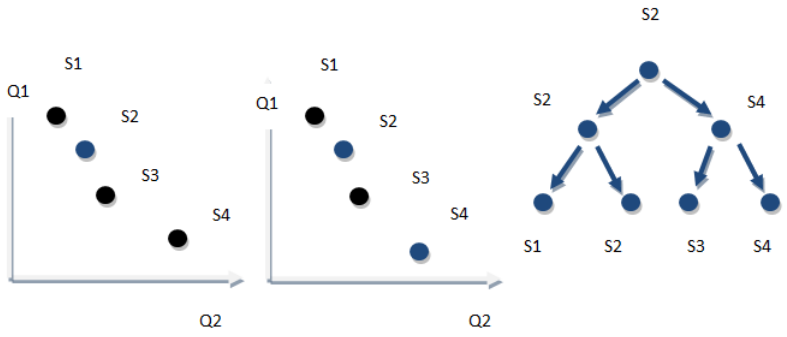

Figure *4.1:* *Clustering hiérarchique des skylines*

La troisième étape de l'algorithme, parcourt les n arborescences (des n classes) niveau par niveau, pour chaque niveau les auteurs appliquent l'algorithme MIP en considérant les représentants des clusters, comme variables. Si le MIP ne trouve pas de solutions, alors il passe au niveau suivant, ce processus est répété tant qu'il n'y a

pas de solutions, ou tant que le dernier niveau de la hiérarchie n'est pas complètement exploré.

Le troisième algorithme nommé *Hybride-Skylines* adapte la recherche hybride pour l'ensemble des skylines extraits de chaque classe. Les expérimentations utilisent 4 types de collections de test, (une collection réelle développée par [Al-Masri and Mahmoud, 2008] et trois synthétiques).

Le tableau 4.1 synthétise les approches présentées dans ce chapitre en les classant selon les critères suivants : l'algorithme d'optimisation utilisé, les attributs QoS traités ((Temps de réponse(Tr), Coût (c), réputation (R), Disponibilité (D), Fiabilité(F), Sécurité (S), Throughput (Th)), et les contraintes QoS.

Travaux étudiés	**Algorithme d'optimisation**	**Attributs QoS traités**	**Contraintes globales QoS**
[Zeng et al., 2004]	Integer Linear Programming (LIP)	C, Tr ,R ,D	Oui
[Ardagna and Pernici, 2005]	Mixed Integer Programming (MIP)	Tr, Th, D, C	Oui
[Alrifai and Risse, 2009]	Mixed Integer Programming (MIP)	C, Tr ,R ,D	Oui
[Rosenberg et al., 2009]	COP (Constraint Optimization Problem)	Tr, F, C, D	Oui
[Gao et al., 2006]	Programmation dynamique	Tr, C, D, F	Non
[Xu et al., 2011]	Programmation dynamique	Non mentionné	Non
[Khan, 1998]	Algorithme UHE	C, Tr, D	Oui
[Akbar et al., 2001]	Algorithme M-UHE	C, Tr, D	Oui
[Yu et al., 2007]	Algorithme WS-UHE	C, Tr, D	Oui
[Klein et al., 2011]	hill climbing	Tr, C, D, F	Non
[Xia et al., 2011]	Algorithme optics	C, Tr, F	oui
[Moustafa and Zhang, 2013]	Apprentissage par renforcement	Non mentionné	Oui
[Feng1/2 et al., 2013]	Apprentissage par renforcement	Non mentionné	Oui
[Luo et al., 2011]	Algorithme HCE	Non mentionné	Non
[Canfora et al., 2005]	Algorithme génétique GA	C, Tr,D, F	Oui
[Jaeger and Mühl, 2007]		C, Tr, F	Oui
[Yu et al., 2013]		C, Tr,D, F	Oui
[Xiangbing et al., 2012]	Algorithme génétique GA	C, Tr, D, F	Non

Travaux étudiés	**Algorithme d'optimisation**	**Attributs QoS traités**	**Contraintes globales QoS**
[Wang and Hou, 2008]	Multi-Objectif GA	C, Tr, F	Oui
[Liu et al., 2011]	Algorithme d'optimisation par essaim particulaire PSO	C, Tr	Oui
[Ludwig, 2012]		F,D, R, Tr, C	Oui
[Zhao et al., 2012]		C, Tr	Oui
[Liu and Yin, 2009]	Version Discrète de PSO	Pas mentionnées	Non
[Liao et al., 2011]	NichePSO	C, Tr	Non
[Pop et al., 2009]	Sélection clonale	C, Tr, F, D	Oui
[Hadjila et al., 2012]	Sélection clonale	C, Tr, F, D, R	Oui
[Zhao et al., 2014]	Sélection négative	C, Tr	Oui
[Yang et al., 2010]	Colonies de fourmis ACO	C, Tr	Oui
[Pop et al., 2010]		D, F, Tr, C	Oui
[Xia et al., 2008]	ACO (multi-phéromones)	Pas mentionnées	Oui
[Li and Yan-Xiang, 2010]	Algorithme multi-objectif (MOCACO)	Pas mentionnées	Oui
[Jafarpour and Khayyambashi, 2010]	L'optimisation par harmonie HS	C, Tr,D, F	Oui
[Esfahani et al., 2012]		Pas mentionnées	Non
[Merzoug et al., 2014]		C, Tr, F, D, R	Oui
[Bekkouche et al., 2017]	L'optimisation par harmonie HS+ Graphe de planification	C, Tr, F, D, R	Oui
[Hadjila et al., 2013]	Colonie d'abeilles artificielles ABC	C, Tr, F, D, R	Oui
[Zhu et al., 2017]		C, Tr, F, D, R	Non
[Chifu et al., 2010]		D, F, Tr, C	Oui
[Kousalya et al., 2011]	Multi-objectif ABC	D, F, Tr, C	Oui
[Krithiga, 2012]	L'algorithme SOMA	Tr, C, D	Oui
[Chifu et al., 2011]	L'algorithme coucou	D, F, Tr, C	Oui
[Chen et al., 2015]	Algorithme DPSA	Non mentionné	Non
[Claro, 2006]	Algorithme NSGAII	Tr, C, D, R	Oui

Travaux étudiés	**Algorithme d'optimisation**	**Attributs QoS traités**	**Contraintes globales QoS**
[Trummer et al., 2014]	Algorithme A-FPTAS	Non mentionné	Non
[Chan et al., 2006]	Dominance au sens de Pareto	Non mentionné	Oui
[Benouaret et al., 2011]		Non mentionné	Oui
[Alrifai et al., 2010]		Non mentionné	Oui

Tableau 4.1: *Classification des approches de sélection des services web composés à base de QoS*

4.4 Conclusion

Nous avons présenté dans ce chapitre, la problématique de sélection de services à base de QoS. Nous avons aussi mis en évidence, un état de l'art illustrant les travaux de recherche sur cette problématique. Plus précisément, nous avons présenté les travaux qui appartiennent aux quatre classes, à savoir les méthodes exactes, les méthodes heuristiques, les méta-heuristiques et les méthodes basées sur la dominance au sens de Pareto. En général, nous pouvons conclure que les méta-heuristiques sont mieux adaptées à cette problématique par rapport aux autres, puisque les méthodes exactes ne passent plus à l'échelle. En plus, les approches heuristiques ne sont pas généralisables pour toutes les sortes de fonctions objectifs. D'autre part, l'optimisation à base de Pareto n'est plus efficace lorsque le nombre de critères de QoS est grand. Par conséquent la meilleure classe d'algorithmes qui gère tous ces compromis (nombre donné d'attributs de QoS, complexité polynomiale, quasi-optimalité de solutions, adaptabilité) sera la classe des méta-heuristiques.

Le chapitre suivant sera consacré aux nos contributions relatives à la sélection et la découverte des services web.

Sommaire

Chapitre 5

Approches proposées

Sommaire

5.1 Introduction

Dans ce chapitre nous décrivons nos approches proposées pour résoudre les problèmes de découverte et sélection des services web sémantiques. Nous commençons d'abord par présenter notre approche de découverte basée sur la relation de fuzzy dominance et l'apprentissage automatique. En particulier, nous décrivons les concepts et

les paramètres relatifs à cette relation de dominance spécialisée ainsi que l'algorithme d'adaptation associé. Ensuite, nous présentons nos contributions pour solutionner le problème de sélection, à savoir une première contribution basée sur un algorithme de recherche d'harmonie (Harmony Search), une deuxième contribution basée sur la sélection clonale (SC) et une troisième contribution qui est une hybridation de la sélection clonale avec l'optimisation par essaim particulaire (SC-PSO). Et enfin nous résumons nos principales contributions, dans la conclusion.

5.2 La découverte des services à base d'agrégation floue de mesures de similarités

Dans la littérature, il existe trois catégories d'approches de découverte sémantique de services web : les approches logiques, les approches non-logiques et les approches hybrides. Les approches de découverte qualifiées de non-logiques sont basées sur le calcul du degré de similarité entre les aspects fonctionnels de la requête et ceux du service (les aspects fonctionnels sont concrétisés avec des concepts d'ontologies, des descriptions textuelles ou des descriptions à base de graphe). Par contre les approches de découverte logique sont basées essentiellement sur des mécanismes déductifs.

D'après [Hadjila, 2014] les inconvénients majeurs des approches logiques sont :

- La complexité exponentielle du matching des concepts (le non passage à l'échelle).
- La présence des faux positifs et faux négatifs due aux scores *subsume* ou *subsumed*.

Le problème primordial des approches non logiques réside dans le choix des techniques de matching de concepts (telles que les mesures de similarités) et/ou les algorithmes d'optimisation de ces matchings.

Selon [Klusch and Kapahnke, 2009] les approches hybrides utilisent une combinaison de classes logiques et non logiques. L'idée est de remédier à certaines limites de chacune de ces deux classes grâce à différentes combinaisons hybrides qui réussissent là où chacune de ces deux approches échoue.

Pour pallier aux difficultés citées ci-dessus, nous avons proposé une approche qui utilise quatre fonctions de matchnig appartenant à la classe non logique, et une fonction appartenant à la classe logique. Ensuite nous avons proposé un algorithme d'agrégation basé sur une relation de fuzzy dominance (voir section 5.2.6). Ce dernier trouvera les services pertinents par rapport à une requête donnée. Le but de cette fuzzification réside dans la discrimination des scores de matching relatifs aux services publiés.

Afin de mieux expliquer nos motivations, nous présentons un exemple dans la section qui suit.

5.2.1 Exemple de motivation

Considérons une requête d'un utilisateur qui consiste en un ensemble de concepts d'entrée Pin_1, Pin_2, ... et des concepts de sortie $Pout_1$, $Pout_2$, ... Nous supposons que tous les concepts appartiennent à une ontologie formelle (OWL). De la même façon, nous supposons que chaque service web est caractérisé par un ensemble de concepts d'entrées et d'un autre ensemble de concepts de sorties.

Afin de satisfaire les exigences de l'utilisateur, nous utilisons des fonctions de matching différentes notées $f_1...f_m$. Chaque fonction est appliquée à un ensemble de paramètres (des entrées ou des sorties). Soit la description RP attachée à la requête, $RP =< RP_I, RP_O > avec\ RP_I = \{Pin_1, Pin_2...\}$ et $RP_O = \{Pout_1, Pout_2...\}$. De la même manière, on définit l'ensemble $SP =< SP_I, SP_O > avec\ SP_I = \{Pin'_1, Pin'_2...\}$ et $SP_O = \{Pout'_1, Pout'_2...\}$ qui représente les paramètres d'E/S d'un service publié S. Chaque fonction de matching f_j calcule le degré de similarité entre les paramètres des services publiés et les paramètres de la requête. Pour chaque service S, si plus d'un paramètre correspond à un paramètre de la requête, le degré de matching le plus élevé est retourné. Par conséquent, le résultat de matching du couple <RP, SP> est donné comme suit :

$$\forall p_i \in RP_I, match_j(p_i, SP_I) = \max_{l=1}^{|SP_I|} f_j(p_i, p'_l) \tag{5.1}$$

Où $p'_l \in SP_I$ et $match_j(p_i, SP_I)$ est le score de matching partiel.

$$\forall p_i \in RP_O, match_j(p_i, SP_O) = \max_{l=1}^{|SP_O|} f_j(p_i, p'_l) \tag{5.2}$$

Où $p'_l \in SP_O$ et $match_j(p_i, SP_O)$ est le score de matching partiel.

Tableau 5.1: *Les scores de matching partiel des services web*

Service	**Paramètre**	f_1	f_2	f_3
A'	*Pin*	0.98	0.92	0.99
	Pout	0.94	0.95	0.89
B'	*Pin*	0.95	0.94	0.97
	Pout	0.93	0.88	0.90
C'	*Pin*	0.85	0.76	0.89
	Pout	0.92	0.77	0.81
D'	*Pin*	0.80	0.71	0.90
	Pout	0.86	0.80	0.84

Dans Tableau 5.1, nous présentons un ensemble de scores de matching partiels. Pour des raisons de simplicité, nous supposons que tous les services ont une seule entrée Pin et une seule sortie Pout. La même hypothèse est faite pour la requête.

La première étape de notre approche de découverte consiste à choisir un algorithme de matching. Pour cela nous avons utilisé cinq mesures de similarité (ou algorithmes de matching) afin de bénéficier des avantages de chaque mesure. Nous nous référons à cette suggestion comme ***Sug1***. Tandis que la deuxième étape concerne le modèle d'agrégation des scores de matching partiel. Afin de calculer le score final, nous pouvons utiliser un schéma de pondération [Dong et al., 2004].

Ce dernier peut être déterminé par l'intermédiaire de l'utilisateur ou de l'apprentissage automatique. De plus, d'autres approches utilisent une agrégation pessimiste telle que le score minimal de matching [Klusch and Kapahnke, 2008]. Nous pouvons remarquer que ce type de techniques d'agrégation n'est pas convenable pour le

matching de services parce que ces dernières permettent une compensation entre les paramètres et entraînent par conséquent une perte d'information.

Pour résoudre ce problème, nous recommandons l'évitement de la compensation des scores de matching partiels, en appliquant la comparaison de ces résultats avant de les agréger. Nous notons cette suggestion comme ***Sug2***. En tenant compte les deux suggestions cités en haut, nous proposons une approche qui utilise une version fuzzifiée de la relation de dominance [Mohammed et al., 2016]. En particulier, un service S1 est meilleur qu'un autre service S2 par rapport à une requête donnée, si S1 domine S2 selon la majorité des résultats obtenus par les algorithmes de matching.

5.2.2 Formalisation du problème

Afin de satisfaire les exigences de l'utilisateur décrites dans ***Sug1***, nous utilisons différents algorithmes de matching ou fonctions de similarité notées $f_1...f_m$. Chaque fonction est appliquée à un ensemble de paramètres (des entrées ou des sorties).

Soit $RP =< RP_I, RP_O >$ *et* $RP_I = \{Pin_1, Pin_2...\}, RP_O = \{Pout_1, Pout_2....\}$.

$SP =< SP_I, SP_O >$ *et* $SP_I = \{Pin'_1, Pin'_2...\}, SP_O = \{Pout'_1, Pout'_2....\}$.

Chaque fonction f_j fournit une liste de services partiellement classée et chaque élément de la liste fournie est étiqueté avec un vecteur de scores de matching ayant **"d"** dimensions. Dans notre exemple d=2. Ce vecteur est appelé **l'instance de matching**, noté par V_{qj}, q indique l'identificateur de service et j représente l'identificateur de fonction de matching. Formellement, la liste partiellement classée de f_j est donnée par :

$$PRL_j = \langle (S_1, V_{1j}), (S_2, V_{2j}), \ldots, (S_{|Data-Set|}, V_{|Data-Set|j}) \rangle \tag{5.3}$$

Où chaque $V_{qj} \in [0,1] \times [0,1] \times ...[0,1]$.

Si la requête de l'utilisateur est décrite par une seule entrée et une seule sortie, alors $V_{qj} \in [0,1] \times [0,1]$.

Etant donné un ensemble de listes partiellement classées $PRL_1...PRL_m$, de m fonctions de matching, on doit produire une liste fusionnée ***FusedList***, telle que les éléments Top-k de FusedList, ont les meilleures performances (en terme de précision et de rappel) .

Pour satisfaire les exigences décrites dans ***Sug2***, nous adoptons une version floue de la relation de dominance [Benouaret et al., 2014]. Le résultat de la fonction de dominance floue est appelé **score de dominance floue**. Afin de classer les services, nous proposons un **score dominant flou**, qui est défini par la formule 5.5. Ces scores sont calculés sur un ensemble d'instances de matching (appartenant à différents services et différentes fonctions de matching f_j). Les sections suivantes donnent plus de détails sur le score de dominance floue et le score dominant flou.

Le Tableau 5.2 décrit les variables les plus importantes utilisées dans cette section.

Tableau 5.2: *La sémantique des variables*

Variable	**Sémantique**
Data-Set	La collection des services
C	Un sous-ensemble de Data-set
L	Le numéro de la requête
RP	Un ensemble de paramètres d'entrée/ sortie qui décrivent l'aspect fonctionnel de la requête i.e $RP =< RP_I, RP_O >$
SP	Un ensemble de paramètres d'entrée/ sortie qui décrivent l'aspect fonctionnel du service publié i.e $SP =< SP_I, SP_O >$
P_i	Un paramètre d'entrée/sortie de la requête
FusedList	La liste agrégée
f_j	La j^{th} fonction de matching
PRL_j	La j^{th} liste source (ou liste partiellement classée)
V_{qj}	L'instance du matching du service q sous la fonction de matching j
$match_j(p_i,\ SP_X)$	Le degré de matching (partiel) entre la requête et les descriptions du service, avec $X = I$ ou $X = O$
m	Le nombre de fonctions de matching
d	Le nombre de paramètres de la requête ($d \geq 1$)
k	La taille préférée par l'utilisateur de la liste fusionnée
k'	Le nombre d'éléments qui appartiennent à PRL_j
λ	Il appartient à $]0, 1]$, il contrôle le score de dominance entre les objets comparés
ξ	Il appartient à $[0, 1]$, il contrôle le score de dominance entre les objets comparés

5.2.3 La relation de Pareto dominance

Soit u et v deux vecteurs de réels, on suppose que u domine v, notée par $u \succ v$, si et seulement si u est au moins aussi bon que v dans toutes les dimensions et strictement meilleure que v dans au moins une dimension , c'est à dire,

$$(\forall i \in \{1, ..d\}, u_i \geq v_i) et (\exists j \in \{1, ..d\}, u_j > v_j) \tag{5.4}$$

D'après la définition, nous pouvons constater que la relation de pareto dominance n'est pas toujours efficace dans la comparaison des objets. En effet, elle ne permet pas la différenciation entre des vecteurs à grande variance.

Pour expliquer cette situation, soit $u = (u_1, u_2) = (0.9, 0.1)$ et $v = (v_1, v_2) = (0.80, 1)$ deux instances de matching. Selon le paradigme de Pareto, nous n'avons ni u domine v, ni v domine u, c'est-à-dire que les instances u et v sont incomparables. Cependant, on peut considérer que v est meilleur que u puisque $v_2 = 1$ est plus grand que $u_2 = 0.1$ et $v_1 = 0.80$ est presque proche de $u_1 = 0.9$.

A partir de cette observation, il est intéressant de *fuzzifier* la relation de Pareto dominance. L'objectif de cette *fuzzification* est de permettre une comparaison numérique entre des instances de matching qui étaient incomparables. Cette comparaison exprime l'extension par laquelle une instance est (plus ou moins) dominée par une autre.

5.2.4 La fonction fuzzy dominance

Étant donné deux vecteurs d-dimensionnels u et v, la fonction fuzzy dominance [Benouaret et al., 2014] calcule le degré avec lequel u domine v :

$$FD(u, v) = (\frac{1}{d}) \sum_{i=1}^{d} EFD(u_i, v_i) \tag{5.5}$$

Où $EFD(x, y)$ est défini comme suit :

$$EFD(x, y) = \begin{cases} 0 & \text{if } (x - y) \leq \varepsilon \\ 1 & \text{if } (x - y) \geq \lambda + \varepsilon \\ (x - y - \varepsilon)/\lambda & \text{Autrement} \end{cases} \tag{5.6}$$

Où $\varepsilon \in [0, 1]$ et $\lambda \in]0, 1]$. Ces paramètres sont choisis de manière heuristique par un expert ou par un algorithme d'apprentissage automatique. L'influence de ε et λ sur la relation de dominance est donnée dans la Figure 5.1.

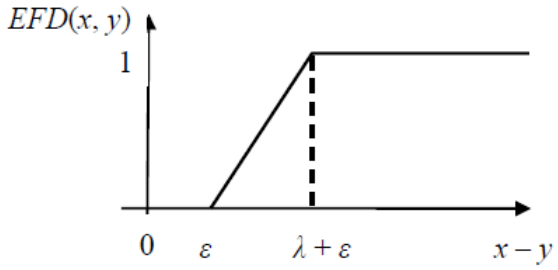

Figure *5.1:* *Les fonctions fuzzy dominance élémentaires*

EFD représente la fonction de dominance floue élémentaire. Elle exprime la mesure dans laquelle x est plus ou moins supérieur à y. La fonction de dominance mentionnée dans la formule 5.5, est simplement une agrégation des différents scores de dominance floue élémentaires. La Figure 5.1 montre les valeurs graduées de la dominance de x sur y, plus précisément :

- Si $(x - y)$ est inférieur à ε, alors x n'est pas supérieur à y.
- Si $(x - y)$ est plus grand que $\lambda + \varepsilon$, alors x est beaucoup plus grand que y.
- Si $(x - y)$ est compris entre ε et $\lambda + \varepsilon$, alors x est beaucoup plus grand que y dans une certaine mesure.

5.2.5 Le score fuzzy dominating

Considérons C l'union des différents Top-k services appartenant aux différentes listes classées (Voir l'exemple de la section suivante pour plus d'explications). Le score fuzzy dominating d'un service S_j calculé sur l'ensemble C, donné comme suit :

$$DS(S_j) = (\frac{1}{m^2} * (|C| - 1)) \sum_{i=1}^{m} \sum_{l \in C, l \neq j} \sum_{i'=1}^{m} FD(V_{ji}, V_{li'}) \tag{5.7}$$

Où m représente le nombre de fonctions de matching et $FD(V_{ji}, V_{li'})$ représente le score de dominance floue de l'instance de matching V_{ji} sur $V_{li'}$(Nous utilisons respectivement les fonctions de matching f_i et $f_{i'}$), $|C|$ est la cardinalité de C.

De façon informelle, cette fonction calcule le score moyen de dominance de S_j sur tous les autres services (de C) et en utilisant toutes les fonctions de matching possibles $f_{i'}$. Les services retenus dans la liste fusionnée sont ceux qui maximisent la formule 5.7.

5.2.6 Algorithme d'agrégation basé sur la dominance floue

Notre algorithme de classement nommé *FDAA* (**Fuzzy Dominance Aggregating Algorithm**), calcule les Top-k services web selon le critère de score fuzzy dominating. Dans ce qui suit, nous présentons le pseudo-code de l'algorithme de classement FDAA :

Algorithme 1 Algorithm FDDA

Entrée(s): $S-S-L=\{PRL_1,...,PRL_m\}$: un ensemble de listes source,

k, k' : Entiers positifs,

λ, ε : Float

Sortie(s): Top-k services web (FusedList) : liste ordonnée

1: **si** $(k' > |PRL_1|)$ **alors**
2: $k' = |PRL_1|$
3: **fin si**
4: **pour tout** $j = 1$ *to* m **faire**
5: **pour tout** (S_q, V_{qj}) *in* PRL_j **faire**
6: $Local\_rank[q] = \sum_{r=1}^{d} V_{qj}(r)$
7: **fin pour**
8: $Sublist_j = extract - Top - k'(PRL_j, Local\_rank)$
9: **fin pour**
10: $C = Sublist_1 \smile Sublist_2 \smile ... \smile Sublist_m$
11: **si** $|C| > 1$ **alors**
12: **pour tout** *chaque service* S_j *in* C **faire**
13: $DS(S_j) = (\frac{1}{m^2} * (|C|-1)) \sum_{i=1}^{m} \sum_{l \in C, l \neq j} \sum_{i'=1}^{m} FD(V_{ji}, V_{li'})$
14: **fin pour**
15: **sinon**
16: $DS(S_1) = 1$
17: **fin si**
18: $FusedList = Une\ liste\ de\ services\ S_j\ de\ C, classés\ selon\ l'ordre\ descendant\ de\ DS(S_j).$
19: **si** $(k > |C|)$ **alors**
20: $k = |C|$
21: **fin si**
22: $Result = Top - k(FusedList)$
23: **retourner** *Result*

L'algorithme est expliqué comme suit :

1. **lignes 1-3** : nous corrigeons k', si elle est supérieure à la taille de l'ensemble de données, elle est remise à $|RPL_1|$.
2. **Lignes 4-7**, nous retenons les $Top-k'$ services de chaque fonction de matching f_j (nous classons les éléments de la liste en fonction de la somme des instances de matching). Par exemple, si un service S a une instance de matching égale à $(0.9, 0.7)$, son score de classement est $0,9+0,7=1,6$. Nous remarquons aussi que plus le score est élevé, meilleur est le classement. La valeur de k' est choisie empiriquement (par défaut k' est la taille de l'ensemble des services pertinents par rapport à une requête).
3. **Ligne 10**, on considère l'ensemble C, constitué de tous les $Top - k'$ services des différentes fonctions de matching.

4. **Lignes 11-17**, nous calculons le score fuzzy dominating de chaque service S_j de C, selon la formule 5.7. Nous devons nous assurer que la taille de C est strictement supérieure à 1, sinon nous n'appliquerons pas cette formule.
5. **lignes 18-22**, nous classons les services de C, selon le score calculé précédemment, et nous ne retenons que les Top-k services. Plus le score de dominance est élevé, plus le rang(classement) du service sera élevé. Ce classement constitue la liste fusionnée.
6. Comme la qualité de la liste fusionnée dépend largement des paramètres de dominance floue ($\varepsilon et \lambda$), nous proposons de les affiner afin d'améliorer le rappel et la précision de la liste globale. À cette fin, nous utilisons l'algorithme d'optimisation des particules en essaim (PSO), pour rechercher les valeurs quasi-optimales de ε et λ. L'algorithme d'optimisation et la fonction objectif sont présentés dans la section suivante.

Considérons maintenant l'exemple suivant, qui consiste en :

$Data - set = \{S_1, S_2, S_3, S_4\}$, trois listes sources : PRL_1,PRL_2,PRL_3, (dans ce cas m = 3). chacune d'elles correspond à une fonction de matching. Nous supposons également que, chacune des trois listes se compose de quatre services. Enfin $k' = 2$ *et* $k = 3$.

Selon les étapes $1, 2, 3$ de l'algorithme FDAA, nous ne gardons que les premiers k' éléments de chaque liste source (ou classée), par conséquent, le processus entier implique $k' * m = 6$ instances de matching qui doivent être fusionnées.

Par exemple si :

$Top - k'(PRL_1) = \langle (S_1, (1, 0.9)), (S_3, (0.8, 0.95)) \rangle$

$Top - k'(PRL_2) = \langle (S_2, (0.85, 0.7)), (S_3, (0.9, 0.6)) \rangle$

$Top - k'(PRL_3) = \langle (S_4, (1, 0.8)), (S_1, (0.5, 0.4)) \rangle$

Ensuite, à partir de ces listes de sources, nous construisons l'ensemble $C = S1, S2, S3, S4$, qui représente tous les $Top - k'$ services. Pour chaque service S_j de C on calcule le score $DS(S_j)$, en supposant que $\lambda = 0,7$, $\varepsilon = 0,3$. Si une instance de matching n'est pas disponible, elle est considérée comme un vecteur nul $(0, 0)$.

Par exemple, pour calculer $DS(S_1)$, nous comparons les instances de matchnig de S_1 avec celles de S_2, S_3, S_4. Puisque S1 a deux instances de matching, S_2 a une seule instance de matching, S_3 a deux instances de matching et S_4 a une seule instance de matching, nous devons calculer huit comparaisons :

$DS(S_1) = 1/27 * (FD((1, 0.9), (0.8, 0.95)) + FD((1, 0.9), (0.85, 0.7)) + FD((1, 0.9), (0.9, 0.6)) + FD((1, 0.9), (1, 0.8)) + FD((0.5, 0.4), (0.8, 0.95)) + FD((0.5, 0.4), (0.85, 0.7)) + FD((0.5, 0.4), (0.9, 0.6)) + FD((0.5, 0.4), (1, 0.))) = 0.017$.

Nous faisons la même chose pour S_2, S_3, S_4, et nous les classons selon l'ordre décroissant de DS. Nous ne retenons que les services $Top - k$ de la liste globale, c'est-à-dire les trois premiers éléments. Dans cet exemple, nous aurons :

$DS(S_2) = 0,013$, $DS(S_3) = 0,037$, $DS(S_4) = 0,031$, par conséquent, on obtient : $FusedList =< S_3, S_4, S_1, S_2 >$. En plus de ça :

$Top - k(FusedList) =< S_3, S_4, S_1 >$, nous calculons ensuite le rappel et la précision en utilisant le benchmark *OWL-S TC*[1].

Dans ce travail, nous utilisons conjointement cinq fonctions de matching. Quatre d'entre elles sont des mesures de similarité [Klusch et al., 2006], [Klusch and Kapahnke, 2008], [Klusch, 2014] : la mesure de perte d'information (Loss-of-Information),

1. projects.semwebcentral.org/projects/owls-tc/

la similitude de Jaccard étendue (Extended Jaccard), la similitude de cosinus (Cosinus) et la similarité de Jensen-Shannon (JS). Chaque mesure de similarité compare toutes les entrées (de la requête et du service) et donne le degré de matching correspondant, la même chose est faite pour les sorties.

La dernière fonction de matching est simplement le raisonnement logique. Celui-ci tire parti du critère de la subsomption pour prendre la décision finale. Il donne cinq scores : Exact, Plugin, Subsume, SubsumedBy et Fail. Tous ces algorithmes sont présentés en détail dans [Klusch and Kapahnke, 2009]. Nous avons choisi ces mesures de similitude parce qu'elles représentent les techniques les plus prometteuses dans la recherche d'information [Klusch and Kapahnke, 2009].

5.2.7 Algorithme d'Optimisation des essaims de particules (PSO)

Le but de cet algorithme est de rechercher l'optimum dans un hyper-volume multidimensionnel. Ceci est réalisé en attribuant des positions initialement aléatoires à toutes les particules dans l'espace et de petites vitesses aléatoires (initiales).

L'algorithme peut être modélisé comme une marche intelligente dans les espaces multi-dimensionnels. La mise à jour de la position d'une particule est basée sur sa vitesse, la position globale la plus connue de toute la population (l'expérience de l'essaim) et la meilleure position connue d'une particule (l'expérience individuelle).

La fonction objectif est évaluée après chaque modification de position. Au fil du temps, les particules se groupent ou se déplacent ensemble autour d'un ou plusieurs optima, grâce à une combinaison de mouvements aléatoires et d'exploitation de bonnes positions connues dans l'espace de recherche.

$$F(\varepsilon, \lambda) = \frac{(2 * MeanRecall(Y) * MeanPrec(Y))}{(MeanRecall(Y) + MeanPrec(Y))} \tag{5.8}$$

Dans notre cas, nous devons maximiser une fonction objectif notée $F(x)$, donnée par la formule 5.8, avec Y : représente les premiers k éléments de la liste agrégée (fusionnée), autrement dit $Y = FDAA(S - S - L, k, k', \varepsilon, \lambda)$.

Le rappel et la précision sont calculés sur les premiers k éléments de la liste fusionnée, en fonction de la pertinence de l'information de la collection des services. Si la collection des services contient L requêtes, alors :

$$MeanRecall(y) = \frac{1}{L} * \sum_{r=1}^{L}(Recall(query_r, Y)) \tag{5.9}$$

$$MeanPrec(y) = \frac{1}{L} * \sum_{r=1}^{L}(Precision(query_r, Y)) \tag{5.10}$$

Algorithme 2 Algorithme PSO

Entrée(s): Pop : *Population des particules*
$S-S-L = \{PRL_1, ..., PRL_m\}$: *ensemble de listes source*
k, k', t, T_Max :entier
$\varepsilon, \lambda, \varepsilon^*, \lambda^*, XPbesti, Pbesti, XGbest, Gbest$: float
$vi(t), \alpha 1, \alpha 2, \beta$: float
Sortie(s): $(\varepsilon^*, \lambda^*)$: *des paires de type* $float$
$f(\varepsilon^*, \lambda^*) : float$

$t = 0$
pour tout *particle i in POP* **faire**
  randomly initialize $p_i(0)$ *and* $v_i(0)$
fin pour
pour tout *particle i inPop* **faire**
  $XPbest_i = P_i(0)$
fin pour
$XGbest = ArgMax_{i \in Pop} F(P_i(0))$
tant que $((t \leq T\_Max)$ *and* $(Not\_convergence))$ **faire**
  pour tout *particle i inPop* **faire**
    compute the performance $F(p_i(t))$
  fin pour
  si $F(p_i(t)) > F(XPbest_i)$ **alors**
    $XPbest_i = p_i(t)$
    $Pbest_i = F(p_i(t))$
  fin si
  si $F(p_i(t)) > F(XGbest)$ **alors**
    $XGbest = p_i(t)$
    $Gbest_i = F(p_i(t))$
  fin si
  $v_i(t+1) = v_i(t) + \alpha 1 * (XPbest_i - p_i(t)) + (1 - \alpha 1) * (XGbest - p_i(t))$ ▷ *Update the current velocity*
  $p_i(t+1) = p_i(t) + v_i(t+1)$ ▷ *Change the current position*
  $\varepsilon = extract - epsilon(P_i(t+1))$
  $\lambda = extract - lambda(P_i(t+1))$
  si $(\varepsilon < 0)$ *or* $(\varepsilon > 1)$ **alors**
    $\varepsilon = random\_value(), such\ that\ \varepsilon \in [0, 1]$
    $updateP_i(t+1)$
  fin si
  si $(\lambda \leq 0)$ *or* $(\lambda > 1)$ **alors**
    $\lambda = random\_value(), such\ that\ \lambda \in]0, 1]$
    $updateP_i(t+1)$
  fin si
  $t = t + 1$
fin tant que
retourner $(XGbest, Gbest)$

L'explication de l'algorithme est décrite comme suit :

- *(Lignes :1-8)*, nous initialisons les positions des particules ainsi que les vitesses initiales, avec des valeurs aléatoires, on remarque que chaque position $p_i(0)$ est une paire (ε, λ) et $v_i(0) \in [0,1] * [0,1]$.

  Nous initialisons chaque meilleure position individuelle $XPbest_i$ avec la position initiale "$p_i(0)$", de la même façon, $XGbest$ est initialisée avec la meilleure position de particule, ensuite nous exécutons la boucle principale.
- *(Lignes : 9-16)*, si la performance courante $F(p_i(t))$ est meilleure que la meilleure performance individuelle $F(XPbest_i)$ rencontrée jusqu'à présent, la meilleure position individuelle notée $XPbest_i$ est mise à jour.
- *(Lignes :17-20)*, si la performance courante $F(p_i(t))$ est meilleure que la meilleure performance de l'ensemble d'essaim $F(XGbest)$ rencontrée jusqu'à présent, alors $XGbest$ est mis à jour.
- *(Ligne 21)*, nous mettons à jour la vitesse actuelle en utilisant deux mouvements (le mouvement social qui signifie que nous nous rapprochons de $XGbest$, et le mouvement basé sur la mémoire, ce qui signifie que nous nous rapprochons de $XPbest_i$).
- *(Ligne 22)*, on calcule la position suivante $p_i(t+1)$.
- *(Lignes :23-34)* Pour chaque particule, nous préservons l'intégrité des contraintes liées à ε et λ.
- *(*ligne 35), nous renvoyons la meilleure solution dénotée $XGbest$ et sa performance $Gbest$.

  La condition de convergence signifie que la meilleure performance globale stagne sur les générations, c'est-à-dire que $|Gbest_{t+1} - Gbest_t| \leq \beta$ (avec β un réel positif). On note aussi que $\alpha 1$ est une valeur aléatoire. Elle est sélectionnée comme une instance aléatoire uniforme dans la plage $[0,1]$: $\alpha 1 \in Uniforme[0,1]$.

5.3 La sélection des services web basée sur les méta-heuristiques

5.3.1 Introduction

La sélection de services web composés est le processus qui consiste à instancier plusieurs tâches abstraites avec des services concrets, tout en optimisant un ensemble de critères de qualité de service (QoS), et en satisfaisant des contraintes globales [Zeng et al., 2003]. Comme nous le montrons dans la Figure 5.2, nous devons sélectionner une instance concrète de service de chaque tâche du workflow abstrait, chaque tâche représente une classe de services web équivalents fonctionnellement (i.e. ayant les mêmes entrées-sorties), mais différents d'un point de vue QoS. Cette instanciation notée c, doit maximiser tous les critères positifs et minimiser tous les critères négatifs, tout en satisfaisant les R contraintes globales (puisque nous avons R critères de QoS). Une contrainte globale peut imposer par exemple, une borne supérieure sur le coût total de la solution c.

Ce problème est reconnu comme étant NP-hard [Parra-Hernandez and Dimopoulos, 2005], et de ce fait il n'y a pas de solution polynomiale qui permet d'acquérir

l'optimum global. Les algorithmes méta-heuristiques deviennent plus appropriés pour résoudre ce type de problème.

Dans cette section, nous proposons trois méta-heuristiques à base de population de solutions qui ont prouvé leur faisabilité et leur efficacité dans la pratique [Brownlee, 2011] :

- La première est basée sur l'algorithme Harmony Search [Merzoug et al., 2014].
- La deuxième s'articule sur un système immunitaire (la sélection clonale).
- La troisième est basée sur un algorithme hybride de sélection clonale et l'optimisation par essaim particulaire (PSO).

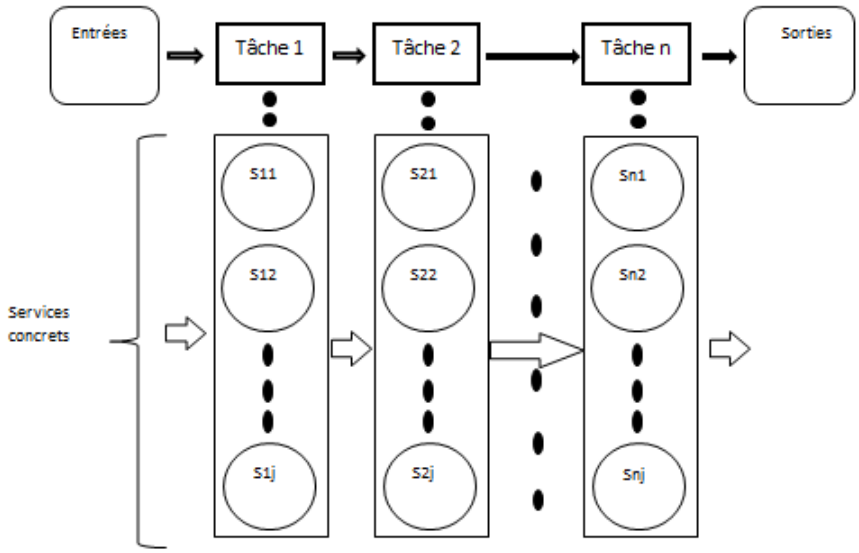

Figure *5.2:* *Le Workflow abstrait*

5.3.2 Formalisation du problème

Pour formaliser le problème de sélection de services web à base de QoS, nous devons concevoir une fonction objectif qui tient compte des critères de QoS, ainsi que les contraintes globales.

Les propriétés non fonctionnelles d'un service web sont exprimées par des attributs de QoS, nous distinguons les attributs génériques et les attributs non génériques [Zeng et al., 2003].

Attributs génériques [Alrifai et al., 2012]

- Le temps de réponse : le temps requis pour compléter une requête par un service web .
- Le coût : c'est le prix qu'un client du service doit payer pour bénéficier du service web.
- La disponibilité : la probabilité pour que le service soit disponible à une certaine période de temps.
- La réputation : c'est une mesure de crédibilité d'un service web.
- La fiabilité : la capacité d'un service de répondre correctement à une requête.

QoS	séquentiel	Parallèle	Boucle	conditionnel
Temps de réponse	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$
Prix	$\sum_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$
Disponibilité	$\Pi_{j=1}^{n} q(S_j)$	$\Pi_{j=1}^{n} q(S_j)$	$\Pi_{j=1}^{n} q(S_j)$	$min_{j=1}^{n} q(S_j)$
Fiabilité	$\min_{j=1}^{n} q(S_j)$	$\min_{j=1}^{n} q(S_j)$	$q(S_j)$	$\min_{j=1}^{n} q(S_j)$

Tableau 5.3: *Les fonctions d'agrégation des propriétés QoS [Alrifai et al., 2012]*

Attributs non génériques

Ce sont des attributs qui sont spécifiques à un domaine donné comme par exemple la bande passante pour des services web multimédia.

L'ensemble des attributs de QoS peut être divisé en deux catégories : attributs négatifs et attributs positifs. Les attributs positifs sont à maximiser (ex.,la disponibilité, la fiabilité,..) tandis que les attributs négatifs sont à minimiser(ex., le temps de réponse, le coût,..).

On suppose qu'on a R attributs de QoS, le vecteur $Q = \{Q_1(S), Q_2(S), \ldots\ldots\ldots, Q_R(S)\}$ représente les attributs de QoS d'un service S, où $Q_i(S)$ détermine la valeur de la $i^{ème}$ attribut de S. Dans notre travail, nous avons utilisé cinq attributs de QoS : le temps de réponse, le coût, la disponibilité, la réputation et la fiabilité [Alrifai et al., 2012].

La valeur QoS d'un service composite (composition) dépend de la valeur des propriétés QoS de ses composants (web services atomiques sélectionnés dans la composition), aussi bien que le workflow utilisé. Il existe plusieurs structures élémentaires de compositions : séquentielle, boucle parallèle (split/ join) et conditionnelle (exclusive split/exclusive join).

Les critères de QoS doivent être normalisés, pour éviter le problème de compensation entre critères. Pour cela, nous utilisons un certain nombre de fonctions d'agrégation. Le Tableau 5.3 montre les fonctions d'agrégation [Alrifai et al., 2012] appliquées à quatre propriétés QoS.

Les contraintes QoS globales peuvent être exprimées en termes de limites supérieures et/ou inférieures sur les valeurs agrégées des différentes propriétés QoS. Ces limites représentent les exigences de l'utilisateur sur les propriétés QoS de la composition (par exemple le coût total de la composition doit être inférieur à 2 $). Nous considérons seulement les propriétés positives pour ne traiter que les limites inférieures.

Les valeurs minimales et maximales du $k^{ème}$ critère de QoS d'une composition c sont :

$$Qmin'(k) = \sum_{j=1}^{n} Qmin(j,k) \tag{5.11}$$

$$Qmax'(k) = \sum_{j=1}^{n} Qmax(j,k) \tag{5.12}$$

$$Qmin(j,k) = min\{Q_k(s_{ji})\}, \forall s_{ji} \in S_j \tag{5.13}$$

$$Qmax(j,k) = max\{Q_k(s_{ji})\}, \forall s_{ji} \in S_j \tag{5.14}$$

Avec :

$Qmin(j,k)$ est la valeur minimale du $k^{ème}$ critère (ex. le coût minimal) appartenant à la classe abstraite j.

$Qmax(j,k)$ est la valeur maximale du $k^{ème}$ critère (ex. le coût maximal) appartenant à la classe abstraite j.

$Qmin'(k)$ et $Qmax'(k)$ sont calculés avec une somme, puisque nous considérons uniquement la séquence.

Pour homogénéiser la normalisation des critères de QoS, nous multiplions les critères négatifs avec -1, et de cette façon tous les critères doivent être maximisés.

La fonction d'utilité d'une instance de services s (respectivement d'une composition c) est donnée comme suit :

$$U(s) = \sum_{k=1}^{R} w_k * (Q_k(s) - Qmin(j,k))/(Qmax(j,k) - Qmin(j,k)) \tag{5.15}$$

$$U'(c) = \sum_{k=1}^{R} w_k * (Q'_k(c) - Qmin'(k))/(Qmax'(k) - Qmin'(k)) \tag{5.16}$$

Avec $w_k \in R^+(\sum_{k=1}^{R} w_k = 1))$ représentent les poids de Q'_k (ou Q_k). Nous avons pris des poids équitables pour tous les critères.

$Q'_k(c)$: représente la valeur agrégée du $k^{ème}$ critère de QoS de la composition c et $Q_k(s)$: représente la valeur du $k^{ème}$ critère de QoS d'un service concret s.

Une composition c est optimale ssi

- Elle est faisable, i.e. elle vérifie toutes les contraintes globales.
- elle a la valeur maximale de la fonction U'.

De manière plus formelle, l'utilisateur cherche à instancier le workflow abstrait de taille n $WA = S1, ..., Sn$ avec une composition concrète $c = s1', ..., sn'$ en choisissant un service concret $sj' \in Sj$ tel que :

1. $U'(c)$ est maximisée.
2. $Q'k(c) \geq Cons(k), \forall Cons(k) \in CONS$ (l'ensemble des contraintes globales).

$Cons(k)$: représente une contrainte globale (une constante numérique) associée au critère k.

Pour intégrer les contraintes globales dans la fonction objectif, nous introduisons une fonction de pénalité notée $P(c)$ qui sert à décroitre le score de $U'(c)$ lorsqu'une ou plusieurs contraintes globales sont violées. Plusieurs fonctions de pénalité sont proposées dans la littérature, [Yeniay, 2005] (nous avons des fonctions statiques, dynamiques, adaptives..).

Pour des raisons de simplicité nous avons choisi une fonction statique, (car les autres n'ont pas donné d'améliorations significatives en expérimentation).

$$P(c) = -\sum_{k=1}^{R} (D_k)^2(c) \tag{5.17}$$

Avec

$$D_k(c) = \begin{cases} 0 & \text{si } Q'_k(c) \geq Cons(k) \\ |Q'_k(c) - Cons(k)| & \text{sinon} \end{cases} \tag{5.18}$$

À partir de cette fonction nous pouvons créer une nouvelle fonction f qui intègre les contraintes globales et les critères de QoS en même temps :

$$f(x) = U'(x) + P(x) \tag{5.19}$$

Notre but, est de chercher la solution x qui maximise la fonction objectif f. Cependant, trouver la solution optimale exige l'énumération de toutes les combinaisons possibles des services candidats. Comme mentionné dans la problématique, ce problème est *NP-difficile*. Si la requête d'un utilisateur concerne m classes de services (m tâches) et chaque classe contient l services candidats, il y a l^m combinaisons possibles à examiner. Pour cela nous proposons trois algorithmes d'optimisation, chacun d'eux est spécifié dans les sections suivantes.

5.3.3 La sélection des services web basée sur l'algorithme Harmony Search

Dans cette section, nous décrivons notre méthode de sélection basée sur l'algorithme Harmony Search (HS). Selon des études précédentes, l'algorithme HS s'est avéré très efficace dans une large gamme de problèmes d'optimisation, tels que la distribution de l'eau et les jeux [Geem, 2009], [Geem, 2007], et a montré une meilleure performance par rapport à d'autres techniques d'optimisation traditionnelles.

5.3.3.1 Principe de l'algorithme Harmony Search

HS est un algorithme d'optimisation évolutionnaire, développé par Geem et al [Geem et al., 2001]. Il a été inspiré par l'improvisation des musiciens de jazz. Selon cette métaphore, chaque musicien correspond à une facette dans une solution candidate (pour un problème donné), et le pitch ainsi que la plage de chaque instrument musical correspond aux limites et aux contraintes de la variable de décision. L'harmonie entre les musiciens est considérée comme une solution candidate à un moment donné, et l'appréciation esthétique du public de l'harmonie représente la fonction de fitness. Les musiciens recherchent l'harmonie dans le temps grâce à de petites variations et des improvisations, ce qui se traduit par une amélioration de la fonction de coût.

L'algorithme HS comporte des concepts simples et peu de paramètres [Kim and Geem, 2015]. Il impose peu d'exigences mathématiques et peut facilement être mis en œuvre. L'idée principale de l'implémentation artificielle est d'utiliser de bonnes solutions candidates déjà découvertes pour influencer la création de nouvelles solutions candidates à la localisation des optimums du problème. Ceci est réalisé en créant des solutions candidates où chaque composant de la solution est soit tiré aléatoirement d'une mémoire des harmonies (HM) et ajusté à partir de la mémoire de solutions, soit attribué de façon aléatoire en fonction des paramètres du problème. Dans l'algorithme HS, la mémoire d'harmonies (HM) est une mémoire où sont stockés tous les vecteurs de solutions candidates (ensembles de variables de décision). Dans chaque évolution de l'algorithme HS, si un vecteur de solution avec une bonne valeur de fitness est généré, il sera sauvegardé en HM et pourrait être utilisé dans les prochaines générations. Les étapes du HS sont les suivantes [Geem et al., 2001] :

- *Étape 1* : Initialiser le problème et les paramètres de l'algorithme.
- *Étape 2* : Initialiser la mémoire d'harmonies.

- *Étape 3* : Improviser une nouvelle harmonie.
- *Étape 4* :Mettre à jour la mémoire d'harmonie.
- *Étape 5* : Répéter Etapes 3 et 4 jusqu'à ce que le critère d'arrêt soit satisfait.

Dans *l'étape 1*, la fonction objectif avec N variables de décision et l'ensemble des valeurs possibles pour chaque variable de décision sont définis. Les paramètres de l'algorithme HS sont également spécifiés dans cette étape. Les paramètres de HS sont les suivants : la taille de la mémoire d'harmonies **(HMS)**, le taux de considération de la mémoire **(HMCR)**, le taux d'ajustement **(PA)** et le critère d'arrêt.

Dans *l'étape 2*, la matrice HM est remplie avec autant de vecteurs de solution générés de manière aléatoire selon HMS.

Dans *l'étape 3*, un nouveau vecteur d'harmonie $X = (x1, x2, ..., xn)$ est généré sur la base de trois règles :

(1) la considération de la mémoire (HMC), (2) l'ajustement (PA) et (3) la sélection aléatoire (RS). Dans HMC, la valeur d'une variable de décision pour le nouveau vecteur est choisie parmi l'une des valeurs de cette variable de décision dans la mémoire d'harmonies spécifiée. Le HMCR, qui varie entre 0 et 1, est le taux de choix d'une valeur à partir des valeurs historiques stockées dans la mémoire tandis que (1 - HMCR) est le taux de sélection aléatoire d'une valeur dans la plage de valeurs possibles. Chaque composante obtenue par HMC est examinée pour déterminer si elle doit être ajustée. En PA, la valeur courante d'une variable de décision est remplacée par une de ses valeurs voisines. Le taux d'ajustement est spécifié par le paramètre PAR. Durant l'improvisation (générant une nouvelle harmonie), HMCR et PAR sont utilisés pour améliorer le vecteur de la solution globalement et localement respectivement.

Dans *l'étape 4*, si le nouveau vecteur d'harmonie est meilleur que le mauvais vecteur d'harmonie dans la HM, évalué en terme de fitness, la nouvelle harmonie est incluse dans le HM et l'harmonie la plus mauvaise existante est exclue du HM.

Dans *l'étape 5*, si le critère d'arrêt (nombre maximal d'improvisations) est satisfait, l'évolution est terminée. Sinon, les étapes 3 et 4 sont répétées. Enfin, le meilleur vecteur de la mémoire en terme de fitness est la solution quasi-optimale.

5.3.3.2 Algorithme HS pour le problème de la sélection des services web

Le pseudo code de l'algorithme HS est donné ci-dessous :

Algorithme 3 Harmony Search Algorithm

Entrée(s): HMS, $HMCR$, PAR, $pitch\_num$, $pitch\_bounds$, $iteration\_max$

Sortie(s): $Best\_Harmony(optimal\ solution)$

1: $HM = Initialize\_Harmony\_Memory(HMS,$
$pitch\_num, pitch\_bounds)$
2: $Evaluate\_Harmony\_set(HM)$
3: **pour tout** $i \leftarrow iteration\_max$ **faire**
4: Harmony=nil
5: **pour tout** $pitch\_j \in pitch\_num$ **faire**
6: **si** $Rand(0,1) \leq HMCR$ **alors**
7: $RandomPitch = Select\_Pitch\_Random\_Harmony(HM, pitch\_j)$
8: **si** $Rand(0,1) \leqslant PAR$ **alors**
9: $Pitch = Pitch\_Adjustment(RandomPitch)$
10: **sinon**
11: $Pitch = RandomPitch$
12: **fin si**
13: **sinon**
14: $Pitch = Get\_Random\_Pitch(pitch\_bounds(pitch\_j))$
15: **fin si**
16: $Update(Harmony, Pitch, pitch\_j)$
17: **fin pour**
18: $Harmony = Evaluate\_Harmony(Harmony)$
19: **si** $Cost(Harmony) \leq Cost(Worst\_Harmony(HM))$ **alors**
20: $HM = HM - (Worst\_Harmony(HM))$
21: $HM = HM \cup (Harmony)$
22: **fin si**
23: **fin pour**
24: $Best\_Harmony = Select\_Best(HM)$
25: **retourner** $Best\_Harmony$

Les entrées requises de l'algorithme, ainsi que son explication sont données ci-dessous :

- HMS : indique le nombre de solutions (ou harmonies) utilisées par l'algorithme
- $pitch\_num$: désigne l'ensemble des classes abstraites T1, ... Tn
- $pitch\_bounds$: indique la plage de valeurs d'une classe abstraite (dans notre cas, l'ensemble des services / classes)
- $HMCR$: c'est une probabilité, qui contrôle l'utilisation de l'information de la mémoire d'harmonie ou la génération d'un pitch aléatoire. En tant que telle, elle contrôle le taux de convergence de l'algorithme et est typiquement configurée $\in [0.7, 0.95]$.
- PAR : est une probabilité qui contrôle la fréquence d'ajustement des pitchs choisis parmi la mémoire d'harmonies, typiquement configurée $\in [0.1, 0.8]$. Des valeurs élevées peuvent provoquer une convergence prématurée.

- L'étape 1 initialise la mémoire en générant de façon aléatoire les solutions. Chaque solution est un vecteur contenant N entiers qui représentent les services sélectionnés (Ligne 1).
- L'étape 2 évalue les solutions en utilisant la fonction objectif f (Ligne 2).
- La boucle principale (étape 3) crée une nouvelle solution (ou harmonie) comme suit (Lignes 3-23) :
  1. Chaque classe abstraite Ti de la solution est initialisée avec une valeur Vi prise d'une harmonie aléatoire $h \in HM$ avec une probabilité $HMCR$ (Lignes 6-7).
  2. Avec une probabilité PAR, la valeur précédente est remplacée par une autre valeur Vi' (tirée de la population) telle que $f(V1, ...V'i, ...Vn)$ soit maximisée (Lignes 8-12).
  3. Si les étapes 1 et 2 ne sont pas exécutées, la classe abstraite Ti est initialisée avec une valeur aléatoire $Zi \in pitch\_bounds(Ti)$ (Ligne 14).
  4. Nous mettons à jour the i^{th} classe abstraite de la solution "Harmony", en utilisant la valeur de $Pitch$ (Ligne 16).
- Dans l'étape 4, nous évaluons la solution construite dénotée "Harmony"(Ligne 18).
- Dans l'étape 5, la mauvaise harmonie de la mémoire est remplacée par "Harmony" si cette dernière est plus efficace en termes de f (Lignes 19-22).
- Dans l'étape 6, nous retournons la meilleure solution de la mémoire ($Best\_Harmony$) (Lignes 24-25).

5.3.4 La sélection des services web basée sur l'algorithme de sélection clonale

Dans cette section nous décrivons notre deuxième approche pour la sélection des services web basée sur l'algorithme de sélection clonale (clonAlg) [De Castro and Von Zuben, 2002]. Nous avons adopté clonAlg pour résoudre le problème de la sélection des services web car il converge mieux vers l'optimum en introduisant itérativement de nouvelles solutions candidates qui élargissent l'espace de recherche (par rapport aux algorithmes génétiques qui stagnent sur l'optima local) [Hadjila, 2014]. Notre algorithme utilise les propriétés non-fonctionnelles des services web afin de trouver une solution quasi-optimale. Une telle solution doit :

1. Optimiser les critères de QoS
2. Satisfaire les contraintes globales non fonctionnelles

5.3.4.1 Principe de l'algorithme de la sélection clonale

Les systèmes immunitaires artificiels (AIS) sont des systèmes informatiques inspirés par les principes et les processus du système immunitaire naturel des vertébrés. Généralement, pour résoudre un problème, chaque solution possible est représentée par un anticorps et le problème est un antigène. Le modèle de ClonAlg [De Castro, Von Zuben, 2000] est le modèle de conception des AIS le plus utilisé. C'est une abstraction des mécanismes de mémorisation des systèmes immunitaires. Son principe est le suivant :

1. Produire un ensemble de solutions de N candidats qui sont définis par le problème à étudier.
2. Produire une nouvelle génération (de taille N) à partir de la population précédente.

   Ceci est fait en créant des copies identiques des cellules. Le nombre de copies est proportionnel aux affinités. Les éléments de cette génération sont appelés clones.
3. Changer la structure des cellules courantes. Le taux de changement (mutation) est inversement proportionnel à leurs affinités. Ceci signifie que les clones ayant une affinité élevée n'ont pas besoin d'être mutés autant que ceux ayant une faible affinité.
4. Sélectionner les M cellules (du résultat de l'étape 4) qui ont la plus grande affinité à l'antigène pour composer la population de l'itération suivante.
5. Remplacer quelques cellules qui possèdent des valeurs d'affinité faible par les nouvelles cellules (nombre de cellules remplacées =N-M).
6. Répéter les étapes 2 à 5 jusqu'à ce qu'un critère d'arrêt donné soit satisfait.

5.3.4.2 Adaptation de l'algorithme de la sélection clonale au problème de la sélection des services web

Dans notre problème de composition du service Web, la sélection clonale est cartographiée comme suit :

1. Une cellule B (ou anticorps) est représentée par une composition de services (c'est à dire une séquence d'instances de services).
2. Un agent pathogène (ou antigène) est représenté par la fonction objectif F (Voir la formule 5.19).

Le pseudo-code de l'Algorithme de sélection clonale (clonAlg) est donné ci-dessous :

Algorithme 4 Algorithme de sélection clonale(ClonAlg)

Entrée(s): 1. la requête **R**= <n=taille-composition, BorneQoS1, BorneQoS2, BorneQoS3, BorneQoS4, BorneQoS5> avec :

- *taille-composition* : représente le nombre de tâches abstraites requises par l'utilisateur.
- *BorneQoS1...BorneQoS5* : représentent les exigences globales de l'utilisateur imposées sur les valeurs de QoS agrégées.

2. Une base de services **B** segmentée en n classes (chaque service est caractérisé par R valeurs de QoS).

Sortie(s): une composition maximisant f (notée c^*)

1: Initialiser les cellules B (anticorps) de la population avec des valeurs aléatoires $Pop = \{a_1^0, a_2^0, ..., a_n^0\}$ avec $a_i = (Ins_1, Ins_2 \ldots Ins_n)$ et $(Ins_k \in \{1, 2, \ldots .nombre\_instances\}$

2: $c^* = Argmax(f(a_i^0))$, $a_i^0 \in Pop$.

3: Initialiser le compteur d'itérations : $t = 1$

4: **tant que** $t \leq T\_Max$ **faire**

5: Pour chaque cellule B notée $a_i^t \in Population$: évaluer son affinité $f(a_i^t)$ (ou sa performance).

6: Calculer l'ensemble *clones_set* pour chaque a_i^t, selon le taux de clonage Tc tel que $Tc(a_i^t) = f(a_i^t)/f(c^*)$

7: Faire *l'hypermutation* pour chaque cellule B de *clones_set* et calculer sa nouvelle affinité $f(a_i^{tt})$ ▷ (variation des cellules clonées)

8: $Union = Population \cup clones\_set$.

9: Trier $Union$ selon l'ordre décroissant des affinités.

10: Mettre à jour la population (création des a_i^{t+1}), en retenant les N premières cellules B de $Union$. ▷ (N est la taille de Population)

11: Créer L cellules B de façon aléatoire ($L = N * insertion\_rate$). ▷ (génération aléatoire de cellules B).

12: Remplacer les L dernières cellules B de Population (en termes d'affinité) par les cellules de l'étape précédente. ▷ (remplacement des mauvaises cellules B par des cellules aléatoires).

13: $c = Argmax(f(a_i^{t+1})), a_i^{t+1} \in Pop$; si $f(c) \geqslant f(c^*) alors\ c^* = c$

14: t=t+1

15: **fin tant que**

16: **retourner** La meilleure cellule B de la population c^*.

Cet algorithme possède plusieurs paramètres de contrôle, tels que le nombre de cellules B, taux de clonage (Tc), taux d'insertion aléatoire ($insertion\_rate$).

Plus le nombre de cellules B et le taux de clonage sont grands, plus le temps d'exécution est élevé et plus les chances d'obtention d'un optimum sont bonnes. Le

taux d'insertion aléatoire est généralement faible, il sert à garantir la variation de la population de cellules.

5.3.5 La sélection des services web basée sur un algorithme hybride de sélection clonale et l'optimisation par essaim particulaire (PSO)

Pour combiner les avantages de la sélection clonale (la convergence rapide vers l'optimum local) et l'optimisation par essaim particulaire PSO (la bonne qualité en termes de fitness des solutions trouvées), nous hybridons ces deux approches pour en construire une méta-heuristique notée SC-PSO. Grosso modo, l'idée de l'algorithme SC-PSO est de diviser la population en deux sous groupes de même taille et chaque algorithme (SC ou PSO) travaillera sur son propre sous groupe. A la fin de l'itération, la meilleure solution trouvée par PSO est injectée dans le sous groupe de SC, et ceci tout en supprimant la mauvaise solution du sous groupe de SC. Cette tâche est répétée jusqu'à la satisfaction du critère d'arrêt.

Le pseudo-code de l'Algorithme de sélection SC-PSO est donné ci-dessous :

Algorithme 5 SC-PSO : Algorithme Hybride de sélection clonale et d'optimisation par essaim particulaire

Entrée(s): 1. la requête **R**= <n=taille-composition, BorneQoS1, BorneQoS2, BorneQoS3, BorneQoS4, BorneQoS5> avec :

- *taille-composition* : représente le nombre de tâches abstraites requises par l'utilisateur.
- *BorneQoS1...BorneQoS5* : représentent les exigences globales de l'utilisateur imposées sur les valeurs de QoS agrégées.

2. Une base de services **B** segmentée en n classes (chaque service est caractérisé par R valeurs de QoS).

Sortie(s): une composition maximisant f (notée c^*)

1: Initialiser les cellules B (anticorps) de la population avec des valeurs aléatoires $Pop = \{a_1^0, a_2^0, ..., a_n^0\}$ avec $a_i = (Ins_1, Ins_2 \ldots . Ins_n)$. ▷ $(Ins_k \in \{1, 2, \ldots . nombre\_instances\})$

2: Initialiser le compteur d'itérations : $t = 1$

3: $Pop1 = \{a_1^0, a_2^0, ..., a_{|Pop|/2}^0\}$, $Pop2 = \{a_{|Pop|/2+1}^0, ..a_{|Pop|}^0\}$

4: $XGbest = Argmax(f(a_j^0))$, $a_j \in Pop1$

5: **tant que** $t \leq T\_Max$ **faire**

6: Pour chaque $a_i^t \in Pop1$: $a_i^{(t)} = modification\_aléatoire(a_i^{(t)})$, $a_i^{(t+1)} = modification\_sociale(a_i^{(t)}, XGbest)$, $sif(a_i^{(t+1)}) > f(XGbest) alors\ XGbest = a_i^{(t+1)}$

7: $Pop1 = \{a_1^{(t+1)} ... a_{|Pop|/2}^{(t+1)}\}$

8: $Worst\_B\_cell = Argmin(f(a_i^t))$, $a_i^t \in Pop2$

9: $Pop2 = Pop2 - \{Worst\_B\_cell\}$

10: $Pop2 = Pop2 \cup \{XGbest\}$

11: Calculer l'ensemble *clones_set* pour chaque $a_i^{(t)} \in Pop2$, selon le taux de clonage Tc tel que $Tc(a_i^{(t)}) = f(a_i^{(t)})/f(c*)$

12: Faire *l'hypermutation* pour chaque cellule B de *clones_set* et calculer sa nouvelle affinité $f(a_i^{\prime(t)})$ ▷ (variation des cellules clonées)

13: $Union = Pop2 \cup clones\_set$.

14: Trier $Union$ selon l'ordre décroissant des affinités.

15: Mettre à jour la population (création de $a^(t+1)_i$), en retenant les M premières cellules B de $Union$. ▷ (M est la taille de Pop2)

16: Créer L cellules B de façon aléatoire ($L = M * insertion\_rate$). ▷ (génération aléatoire de cellules B).

17: Remplacer les L dernières cellules B de $Pop2$ (en termes d'affinité) par les cellules de l'étape précédente. ▷ (remplacement des mauvaises cellules B par des cellules aléatoires).

18: $c = Argmax(f(a_i^{(t+1)})), a_i^{(t+1)} \in Pop2\,;\ sif(c) > f(c^*)\ alors\ c^* = c\,;$

19: t=t+1

20: **fin tant que**

21: **retourner** La meilleure cellule B de la population c^*.

5.4 Conclusion

Nous avons présenté dans ce chapitre nos contributions pour le problème de découverte et la sélection de services web. Nous avons proposé une approche de découverte basée sur la relation fuzzy dominance. Nous avons aussi présenté trois algorithmes pour le problème de sélection des services. Le premier est basé sur l'algorithme Harmony Search, le deuxième est inspiré de la sélection clonale réalisée en systèmes immunitaires artificiels et le troisième est basé sur un algorithme hybride (SC-PSO) de sélection clonale et d'optimisation par essaim particulaire (PSO). Ces algorithmes seront expérimentés et validés dans le prochain chapitre.

Chapitre 6

Implémentation et expérimentations

Sommaire

6.1 Introduction

Dans ce chapitre nous décrivons notre prototype implémentant l'environnement de découverte et de sélection de services web. Nous décrivons en premier lieu, l'architecture générale du système nommé ***DSS*** (Environnement de **D**écouverte et de

Sélection de **S**ervices) en présentant les différents modules ainsi que leurs rôles, ensuite nous présentons les corpus utilisés pour évaluer l'environnement, après nous montrons les différentes expérimentations menées, et finalement nous discutons les résultats obtenus.

6.2 Présentation du système de sélection et de découverte : DSS

Nous avons développé un prototype nommé ***DSS***, il est implémenté avec java *(JDK 1.7.0)* et *netbeans 7.0*. Ce prototype est alimenté avec des documents décrits en OWL-S et enrichis avec des attributs de qualité de services.

6.2.1 Fonctionnalités du système

Le système ***DSS*** (Environnement de Découverte et de Sélection de Services) est présenté dans la Figure 6.1, il utilise plusieurs sortes d'entrées (base de services, ontologies, requête-utilisateur), elles sont décrites comme suit :

6.2.1.1 Répertoire des Services Web Sémantiques

Il contient un ensemble de descriptions syntaxiques et sémantiques de services web. Plus précisément nous avons des documents *WSDL* et *OWLS*, pour chaque instance de service. La description *OWLS* contient des informations fonctionnelles (entrées, sorties, comportement des opérations complexes...) et non fonctionnelles, et des informations d'accès (une association avec l'interface *WSDL*).

6.2.1.2 Collection d'Ontologies

Les raisonnements sémantiques effectués lors de la découverte emploient des ontologies offertes par les collections de tests (Voir Section 6.3). Grâce à ces ontologies, nous pouvons enrichir les descriptions sémantiques des services en plus il sera possible de vérifier des relations de *subsomption* entre concepts.

6.2.1.3 Requête d'Utilisateur

Nous distinguons deux types de requêtes selon l'objectif visé par l'utilisateur :

- Pour le cas de découverte, la requête $R =< entrées - ens, sorties - ens >$, avec :
  - entrées-ens : est l'ensemble des concepts d'entrées.
  - sorties-ens : est l'ensemble des concepts de sorties.
- Pour le cas de sélection la requête R= <taille-comp, BorneQoS1, BorneQoS2, Borne QoS3, BorneQoS4, BorneQoS5>
  - taille-comp : représente le nombre de tâches abstraites requises par l'utilisateur.
  - BorneQoS1...BorneQoS5 : représentent les exigences globales de l'utilisateur imposées sur les valeurs de QoS agrégées.

Le système ***DSS*** est composé des modules suivants :

6.2.1.4 Module d'Analyse des Descriptions de Services Web (Parsing)

Il permet l'extraction des concepts d'entrées et de sorties associées aux descriptions *OWLS*, ceci se fait en utilisant les ontologies de domaine. Ce module récupère aussi les valeurs de QoS associées à chaque service.

6.2.1.5 Module de découverte sémantique de services web

Il permet la comparaison d'une requête-utilisateur (constituée d'un ensemble de concepts d'entrées-sorties) avec un ensemble de descriptions *OWLS* de services. Ce module est implémenté avec un algorithme de classement nommé *FDAA* (Fuzzy Dominance Aggregating Algorithm). *FDAA* calcule les top-k services web selon le critère de score dominant flou (Fuzzy Dominating Score).

6.2.1.6 Module de sélection de service

Ce module effectue une recherche afin d'instancier les tâches abstraites avec des services concrets. Plus précisément, il optimise une fonction objectif, qui agrège des critères de QoS associés à une composition concrète, en plus d'un terme qui représente la pénalité, en cas de violation des contraintes globales. Trois algorithmes sont offerts pour réaliser cet objectif (l'algorithme Harmony Search, l'algorithme de sélection clonale et l'hybridation sélection clonale-PSO).

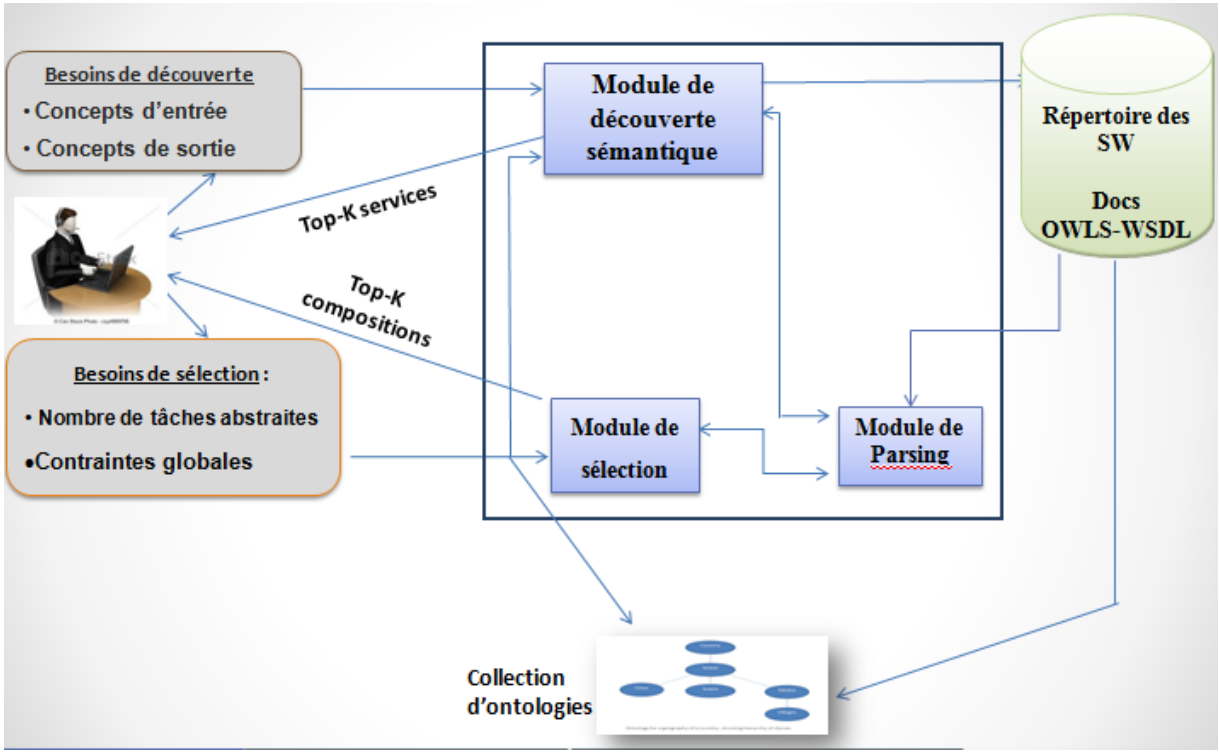

Figure *6.1:* *L'architecture du système proposé*

6.3 Corpus utilisés

Nous décrivons dans ce qui suit, les différentes bases employées pour évaluer les approches proposées. Nous avons 2 bases, chacune d'elles est associée à une problématique donnée (la découverte ou la sélection).

6.3.1 Corpus de Découverte

Nous avons utilisé le corpus *OWLS-TC version 2.2.1*[1]. Ce dernier est développé par le centre allemand pour la recherche en intelligence artificielle (http ://www.dfki.de/scallops). Cette version décrit un ensemble de services web à travers des documents OWLS, elle dispose de 1007 services web segmentés en 7 classes : le domaine de l'éducation, le domaine médical, le domaine de la nourriture, le domaine militaire, le domaine des voyages (tourisme), le domaine de la communication, et le domaine de l'économie.

La majorité de ces services sont extraits de l'annuaire *UDDI* d'IBM, et sont traduits du format *WSDL* en format *OWLS* de façon semi-automatique. La base propose aussi un ensemble de requêtes réparties sur les 7 classes, ces requêtes sont modélisées sous forme de document *OWLS*. Chaque document *OWLS* (service ou requête) comporte dans sa partie *profile* des éléments *profile :hasinput* et *profile :hasoutput*, ces derniers sont employés comme entrées pour le module de découverte des services web. Chaque service web (i.e. document OWLS) est étiqueté manuellement par des experts humains comme étant pertinent ou non par rapport à une requête donnée. En d'autres termes, chaque service web possède une étiquette binaire (pertinent ou non) par rapport à une requête donnée. Ceci permet le calcul des rappels et des précisions des approches proposées. La base offre aussi un ensemble d'ontologies pour décrire les services et les requêtes, chaque classe de services possède une ou plusieurs ontologies.

Dans ce qui suit, nous donnons un exemple d'une description de service web et d'une description de requête à l'aide des éléments *profile :hasinput* et *profile :hasoutput*.

- Requête : cette requête cherche les moyens de diagnostics offerts par un hôpital donné.
  - Concepts d'entrée : $Hospital$
  - Concepts de sortie : $Investigating$
- Service : ce service donne les prix des véhicules ayant trois roues.
  - Nom de service : $3wheeledcar\_price$
  - Concepts d'entrée : $3wheeledcar$
  - Concepts de sortie : *price*

6.3.2 Corpus de Sélection

Il est généré de façon aléatoire, Dans ce cas, l'utilisateur spécifie une séquence de tâches abstraites qui doivent être instanciées avec des services concrets.

La requête de l'utilisateur est constituée de :

- 5 nombres qui représentent les contraintes globales de l'utilisateur (puisque nous avons cinq critères de QoS).

1. projects.semwebcentral.org/projects/owls-tc/

- un entier noté n qui indique le nombre de tâches abstraites.

Par exemple l'utilisateur peut spécifier les contraintes globales suivantes :

- $Coût(sol) \leq borne1$
- $Temps - exécution(sol) \leq borne2$
- $Réputation(sol) \geq borne3$
- $Disponibilité(sol) \geq borne4$
- $Fiabilité(sol) \geq borne5$

Chaque tâche abstraite représente une classe des services. Elle possède plusieurs instances de services similaires d'un point de vue fonctionnel et différents selon le point de vue QoS. Le nombre d'instances varie entre 10 et 100 services.

A chaque instance nous associons un vecteur de 5 paramètres de QoS (le temps d'exécution, le coût, la fiabilité, la disponibilité et la réputation). Les intervalles de valeurs de chaque paramètre sont inspirés de [Yu and Bouguettaya, 2009] (voir le Tableau 6.1).

Tableau 6.1: *Les intervalles des paramètres de QoS de la base de sélection [Yu and Bouguettaya, 2009]*

Critère QOS	**Classe1**	**...**	**Classe n**
Temps d'éxécution	0-300(m.s)	...	0-300(m.s)
Réputation	0-5	...	0-5
Cout	0-30($)	...	0-30($)
Fiabilité	0.5-1.0	...	0.5-1.0
Disponibilité	0.7-1.0	...	0.7-1.0

6.4 Expérimentation

Dans cette section, nous décrivons les expérimentations permettant l'analyse des performances des approches proposées. Dans un premier temps, nous présentons les résultats d'expérimentations relatives à l'approche de découverte.

Ces expérimentations sont menées sur une plateforme, ayant un processeur *Corei3* avec **4** *GO* de *RAM* et sous le système d'exploitation *Windows7*. Pour le cas de la découverte, nous étudions les taux de rappels et de précisions de l'approche proposée. Nous mesurons aussi le temps d'exécution. Tandis que pour le cas de la sélection, nous évaluons les taux d'optimalité et le temps d'exécution pour les différentes approches proposées (l'algorithme Harmony Search, l'algorithme de la sélection clonale et l'hybridation sélection clonale-PSO).

Selon [Van Rijsbergen, 1979], les critères de rappels et de précisions sont définis comme suit :

La précision P : c'est le rapport entre les réponses correctes fournies par le système et le nombre total des réponses, Et plus formellement : Précision= = vp/(vp+fp). Avec :

vp : les vrais positifs. (Un résultat correct, et considéré comme étant valide par le système).

fp : les faux positifs. (Un résultat erroné, mais considéré comme étant valide par le système).

Le rappel R : c'est le rapport entre les réponses correctes fournies par le système et le nombre réel des réponses correctes appartenant au corpus, et plus formellement : $Rappel = vp/(vp + fn)$. Avec :

fn : les faux négatifs. (Un résultat correct, et considéré comme faux par le système).

Le temps d'exécution : c'est la période de temps qui s'écoule entre la validation de la requête de l'utilisateur et la réception des résultats finaux.

Le taux d'optimalité : représente le rapport entre l'affinité (ou fitness) de la composition concrète (ou la solution courante notée c), et l'affinité de la composition optimale réelle notée $c*$, i.e. $f(c)/f(c*)$.

6.4.1 Performances de l'approche de découverte

Nous avons implémenté cinq algorithmes de matching et deux algorithmes d'agrégation, qui sont respectivement [Bianchini et al., 2008] :

- Mesure de similarité *Extended Jaccard (EJ)* ,

$$Sim_{EJ}(S,R) = \frac{\overrightarrow{R}.\overrightarrow{S}}{\|\overrightarrow{R}\|_2^2 + \|\overrightarrow{S}\|_2^2 - \overrightarrow{R}.\overrightarrow{S}} \tag{6.1}$$

- Mesure de similarité *Information loss (IL)*,

$$Sim_{IL}(S,R) = 1 - (\frac{IL_{IN}(R,S) + IL_{OUT}(R,S)}{2}) \tag{6.2}$$

  Avec $IL_x(R,S) = \frac{\left|PC_{R,X} \cup PC_{S,X}\right| - \left|PC_{R,X} \cap PC_{S,X}\right|}{\left|PC_{R,X}\right| + \left|PC_{S,X}\right|}$

- Mesure de similarité *Jensen Shannon (JS)*,

$$Sim_{JS}(S,R) = log2 - JS(S,R) = \frac{1}{2log2}\sum_{i=1}^{n} h(p_{i,R}) + h(p_{i,S}) - h(p_{i,R} + p_{i,S}) \tag{6.3}$$

- Mesure de similarité *Cosinus (Cos)*,

$$Sim_{Cos}(S,R) = \frac{\overrightarrow{R}.\overrightarrow{S}}{\|\overrightarrow{R}\|_2^2 + \|\overrightarrow{S}\|_2^2} \tag{6.4}$$

- Algorithme de *Matching Logique*,

$$Matching\ Logique = \begin{cases} 1 & \text{si le service et la requête sont équivalents } S \equiv R \\ 0.9 & \text{si le service est un cas particulier (direct) de la requête } S \sqsubseteq R \\ 0.8 & \text{si le service est un cas particulier (indirect) de la requête } S \sqsubset R \\ 0.8 & \text{si la requête est un cas particulier (direct) du service } S \sqsupseteq R \\ 0 & \text{Echec de matching} \end{cases}$$

- Algorithme d'agrégation Fuzzy Dominance *(FDAA)*
- Et le modèle *Borda Fuse* (avec une taille de liste source $k' = 50$).

Le dernier algorithme est mis en œuvre pour permettre une comparaison efficace avec notre proposition.

Nous utilisons 10 requêtes pour l'apprentissage des paramètres ε et λ. Les résultats de cette phase d'apprentissage, sont présentés dans la Figure 6.3 et la Figure 6.4.

Les 19 requêtes restantes (29 requêtes dans le total) sont utilisées pour évaluer les différents paramètres *IR* (la recherche d'information) (*MeanPrecision (.), MeanPrec (.), MeanRecall, R-prec, R-rank, P @ N*), ces expériences sont présentées dans la Figure 6.2, le Tableau 6.2, 6.3 et 6.4.

Figure *6.2:* *Temps d'exécution moyen pour toutes les méthodes*

Tableau 6.2: *Précision moyenne pour toutes les méthodes*

Algorithme	**TOP 10**	**TOP 20**	**TOP 30**	**TOP 40**	**TOP 50**	**TOP 60**
EJ	0.81	0.64	0.58	0.51	0.42	0.36
IL	0.81	0.64	0.58	0.50	0.42	0.36
JS	0.8	0.65	0.57	0.49	0.41	0.36
Logique	0.73	0.53	0.48	0.42	0.37	0.31
COSINE	0.81	0.64	0.57	0.48	0.4	0.36
FDAA	**0.85**	**0.71**	**0.62**	**0.54**	**0.45**	**0.4**
BORDA	0.83	0.67	0.58	0.5	0.42	0.38

Tableau 6.3: *Rappel moyen pour toutes les méthodes*

Algorithme	**TOP 10**	**TOP 20**	**TOP 30**	**TOP 40**	**TOP 50**	**TOP 60**
EJ	0.33	0.59	0.73	0.79	0.83	0.85
IL	0.33	0.59	0.74	0.79	0.84	0.86
JS	0.33	0.59	0.72	0.79	0.83	0.86
Logique	0.3	0.46	0.6	0.66	0.72	0.69
COSINE	0.33	0.59	0.72	0.78	0.82	0.86
FDAA	**0.35**	**0.61**	**0.77**	**0.85**	**0.9**	**0.92**
BORDA	0.39	0.58	0.73	0.81	0.84	0.9

Tableau 6.4: *Critères IR pour toutes les méthodes*

Algorithme	**SMR**	**SMP**	**R-P**	**P@20**	**P@30**
EJ	0.686	0.553	0.697	0.64	0.58
IL	0.691	0.551	0.704	0.64	0.58
JS	0.686	0.546	0.696	0.65	0.57
Logique	0.571	0.473	0.562	0.53	0.48
COSINE	0.683	0.543	0.693	0.64	0.57
FDAA	**0.733**	**0.595**	**0.732**	**0.71**	**0.62**
BORDA	0.708	0.563	0.7	0.67	0.58

De plus, nous adoptons les critères d'évaluation de la recherche d'information suivants pour évaluer l'algorithme de matching proposé :

R-précision *($R-prec$ ou $R-P$)* : la précision des mesures associée à un seuil particulier de la liste retournée (seuil= nombre de services pertinents de la requête courante).

Précision à N (P @ N) : mesure la précision après que N articles ont été récupérés.

MeanRecall (.) et MeanPrecision (.) ont été définis dans le chapitre précédent. Ils représentent le rappel moyen (resp. précision moyenne) des requêtes. Et nous mesurons aussi ***le temps d'exécution moyen*** des différentes techniques.

Dans la Figure 6.2, nous montrons le temps d'exécution moyen d'une requête donnée (nous supposons que $k' = k$). L'algorithme *FDAA* est en général moins efficace que les autres algorithmes de matching, mais il est modérément acceptable, en particulier pour $k' \leq 30$. Pour des grandes valeurs, la performance se dégrade, mais elle est toujours inférieure à $80s$. Cette charge temporelle est principalement dû à la complexité de la formule 5.7. Malgré que la complexité de cette dernière est polynomiale $O(m^2 * (|C| - 1))$, elle est en général plus coûteuse que la complexité de COS, EJ, JS, IL et Bordaa.

La même expérience montre également que le modèle *Borda* est la méthode agrégée la plus rapide (environ $700ms$). Ceci est dû au fait que *Borda* est une somme simple des positions des services (sa complexité est $O(k * m + |C| \, log \, |C|)$).

Nous remarquons également que l'approche *logique* est la méthode de matching la plus efficace, parce qu'elle met en œuvre *le test de subsumption* avec l'opérateur logique *ou*. Cette implémentation est assurée grâce à un schéma de codage binaire inspiré de [Caseau et al., 1999].

Les Tableaux 6.2 et 6.3 comparent les différents algorithmes de matching selon plusieurs critères :

Le premier est *la précision moyenne*. Le tableau 6.2 montre que l'algorithme FDAA fonctionne mieux que les autres algorithmes de matching, et pour toutes les valeurs de $k(k = 10, ...k = 60)$, les mesures *IL* et *EJ* sont placées dans la seconde position, avec une performance acceptable. La plus mauvaise approche est l'appariement basé sur la logique (elle implique beaucoup de faux positifs).

Le tableau décrivant *la précision moyenne* montre également que l'algorithme *FDAA* est plus puissant que l'algorithme *Borda*. Cela est dû au fait que les méthodes positionnelles sont généralement moins efficaces que les méthodes basées sur le score.

La même observation est faite pour le rappel moyen. Comme le montre le tableau 6.3, un gain significatif est observé pour l'algorithme *FDAA* par rapport aux approches restantes. Les quatre mesures de similarité ont presque la même performance. Enfin, la plus mauvaise approche est l'approche logique.

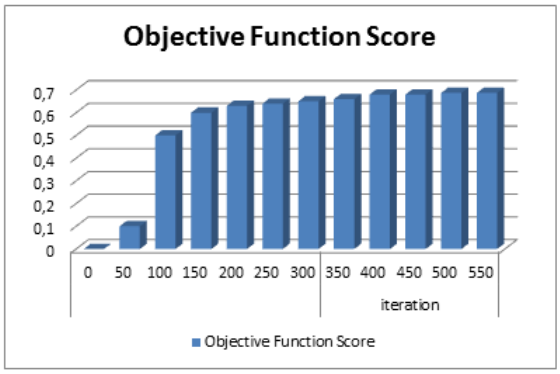

Figure *6.3:* *FDAA performance under PSO*

Pour choisir la meilleure configuration de (ε, λ), nous exécutons plusieurs simulations de l'algorithme *PSO*, chacune d'entre elles utilise une valeur particulière de k'.

La Figure 6.3 montre le processus d'apprentissage des paramètres (ε, λ), pour $k = k' = 30$. Nous observons que la valeur de fitness est obtenue autour de $0,7$. (Ce qui signifie que la précision moyenne vaut $0,62$ et le rappel moyen vaut $0,77$).

Les configurations quasi-optimales pour ε et λ sont données comme suit : $\varepsilon^* \in [0.45, 0.6]$, $\lambda^* \in [0.9, 0.99]$. Ces intervalles fonctionnent très bien pour la majorité des requêtes, sauf pour deux requêtes, où $\varepsilon^* = 0,1$ est plus efficace.

Nous remarquons également que l'algorithme *PSO* converge rapidement vers la solution quasi-optimale (dans 550 itérations). Cette convergence rapide est expliquée comme suit :

1. La forme de la fonction de dominance élémentaire est très simple et possède un minimum de paramètres de contrôle.
2. L'espace de recherche n'est pas très grand (nous avons environ 1000 configurations potentielles à explorer).

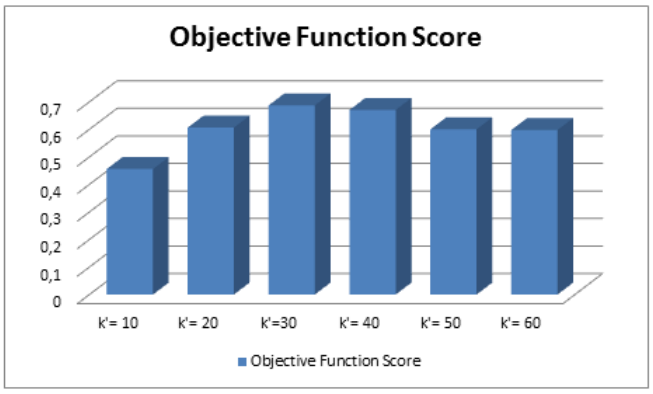

Figure *6.4:* *Effect of k' on FDAA performance*

Puisque la performance de l'algorithme FDAA dépend de la taille des listes sources (dans notre cas c'est k'), nous étudions son impact dans la Figure 6.4, dans cette expérimentation, nous supposons que $k = 30$. En terme de qualité (c'est-à-dire la fonction objective F), on obtient la meilleure performance de FDAA pour $k' = 30$. Ce résultat confirme le résultat de l'expérimentation précédente (voir Figure 6.3).

Lorsque nous utilisons toutes les requêtes du benchmark (i.e 29 requêtes) et en supposant que $k \in \{10, 20, 30, 40, 50, 60, 70\}$, l'ensemble des valeurs optimales de k' converge vers 70. La raison principale est que, dans la plupart des cas, la taille des réponses pertinentes des requêtes est supérieure à 20 et inférieure à 70. Les performances commencent à se dégrader au-delà de 75, parce que le nombre de faux positifs augmente. Cependant, si k' est petit, le temps d'exécution devient très faible, mais les autres critères (rappel, précision moyenne moyenne) restent insuffisants puisque le nombre de faux négatifs est élevé.

Dans le Tableau 6.4, nous rapportons les résultats selon plusieurs métriques : Le SMR (smoothed mean recall) ainsi que la SMP (smoothed mean precision) qui représentent la moyenne des différents rappels moyens (respectivement les différentes précisions moyennes) sur les six premières valeurs de k (i.e k = 10, k = 20, ...k= 60). Ces deux critères résument la qualité des algorithmes de matching indépendamment de k. Comme le montre le Tableau 6.4, la performance de l'algorithme FDAA est supérieure aux autres algorithmes de matching (selon SMR et SMP).

En résumé, l'algorithme FDAA produit un gain de 6% par rapport au SMR individuel le plus élevé (c'est-à-dire la SMR de perte d'information (IL)) et de $7,9\%$ par rapport au SMP individuel le plus élevé.

En ce qui concerne la précision R (c'est-à-dire R-P), l'algorithme FDAA dépasse nettement les autres techniques de matching. On peut dire la même chose pour P@20 et P@30.

Tableau 6.5: *FDAA vs. S3 contest approaches*

Algorithme	**MAP**
FDAA	**0.84**
OWLS-iMatcher2	0.84
JIAC-OWLSM	0.81
SPARQLent	0.71
OPOSSUM	0.57
ALIVE	0.5
OWLS-MX3	0.86

Le tableau 6.5 montre une comparaison entre notre proposition et les différents systèmes qui participent à la compétition S3 2009 [2]. Nous remarquons que cette compétition utilise le même benchmark (c.-à-d. OWLS-TC.2), les résultats présentés couvrent les 29 requêtes.

On suppose dans cette expérimentation que $\lambda* = 0,99, \varepsilon* = 0,6 et k' = 70$. D'après le tableau 6.5, on remarque que l'algorithme FDAA occupe la deuxième place en termes de précision moyenne. La seule approche qui dépasse l'algorithme proposé est *OWLS-MX3*, nous remarquons que ce prototype utilise l'apprentissage *SVM* (machines à vecteurs de support) pour obtenir ces résultats prometteurs, mais les auteurs ne vérifient pas les hypothèses de la théorie de l'apprentissage statistique. En gros, cette théorie suppose que les données utilisées suivent une distribution de probabilité inconnue (cette condition n'est pas vérifiée dans notre collection de test). Deuxièmement, SVM utilise plusieurs paramètres et modèles critiques, tels que la constante de régularisation (appelée RC) et les fonctions du noyau. Et ces paramètres doivent être conformes avec la taille de la base de l'apprentissage et de validation.

Ces paramètres/ modèles influencent largement la performance de SVM, sur les ensembles de données actuelles et futures. En d'autres termes, un mauvais choix peut entraîner une bonne performance sur les données d'apprentissage, mais une mauvaise performance sur des données non encore vues. Par conséquent, leur choix doit être soigneusement justifié par les auteurs de [Klusch and Kapahnke, 2009] (par exemple, pourquoi les auteurs utilisent un noyau à Base Radiale et non pas des noyaux polynomiaux ou d'autres types).

Enfin, la validation croisée utilisée dans [Klusch and Kapahnke, 2009], n'est en général pas suffisante pour dériver les meilleures valeurs pour RC et le paramètre de noyau gamma. En effet, nous avons besoin d'un énorme ensemble de données de validation (c'est-à-dire sa taille doit être suffisamment grande, pour permettre la dérivation des bonnes décisions). Même si les résultats de notre proposition sont très encourageants, nous pouvons constater plusieurs améliorations. Tout d'abord, la taille des listes d'entrée (notée k') peut être acquise en conjonction avec les paramètres de dominance floue. De plus, nous pouvons aussi apprendre la forme de la fonction de dominance floue élémentaire, puisque la courbe proposée sur la figure 1 n'est pas nécessairement optimale.

2. projects.semwebcentral.org/projects/owls-tc/

6.4.2 Performances des approches de sélection

Dans cette partie, nous allons évaluer les performances des trois algorithmes proposées pour résoudre le problème de la sélection des services web. Toutes les expérimentations menées dans cette section sont réalisées sur la base de test décrite en 6.3.2.

6.4.2.1 Performance de l'approche basée sur l'algorithme Harmony Search

Nous avons utilisé 10 classes (i.e n = 10) de services, et chaque classe contient 100 instances, ce qui signifie que le pitchbound = 1..100.

Le nombre total de solutions candidates est donc = 100^{10}. Le nombre d'attributs QoS est fixé à 5. La valeur QoS de chaque attribut est générée par un processus aléatoire uniforme qui respecte les limites spécifiées dans le Tableau 6.1 de la section 6.3.2. Tous les critères ont la même priorité ($W_k = 0,2$).

Nous définissons le *taux d'optimalité*, comme le rapport entre la valeur de fitness (la fonction objectif) de la solution courante et la valeur de fitness de la solution optimale. La valeur de fitness de la solution optimale de ce benchmark est égale à $0,78$, donc le taux d'optimalité d'une solution a est $f(a)/0,78$.

Les paramètres de l'algorithme de recherche d'harmonie (HS) sont modifiés pour obtenir les meilleurs résultats (en terme d'optimalité), après plusieurs expérimentations nous avons choisi cette configuration (qui donne le meilleur résultat) :

- $ConsolidationRate = 0,80$
- $PitchAdjustrate = 0,80$
- $Memory\_size = 140$
- Le nombre maximum d'itérations : IterationMax $\in [100, 2000]$,

La table 6.6 décrit les résultats obtenus par l'approche Harmony Search avec la meilleure configuration des paramètres techniques. Ces résultats montrent que HS dérive un optimum local de faible qualité (puisque la deuxième exécution nous donne un taux d'optimalité égal à 0.28, même si nous accordons plus de temps à l'algorithme HS, les performances ne s'améliorent pas) et cela confirme l'inadéquation (en terme d'optimalité) de cette approche par rapport au problème de sélection des services avec contraintes globales. Ceci peut être justifié comme suit :

L'algorithme HS est par défaut un algorithme d'optimisation continu et sa discritisation n'est pas forcément prometteuse. En effet, HS crée une nouvelle harmonie en composant les pitchs (c'est à dire les services des tâches) à partir de la mémoire et ceci donne des compositions avec une performance (fitness) presque aléatoire, et par conséquent l'algorithme dans sa globalité se comportera de manière quasi-stochastique.

Tableau 6.6: *Optimalité et temps d'éxécution pour l'Algorithme Harmony Search*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.012	309
Exécution 2	200	0.28	636
Exécution 3	300	0.12	925

6.4.2.2 Performance de l'approche basée sur l'algorithme de Sélection Clonale

Sachant que le nombre d'itérations est égal à 100, la table 6.7 nous montre que la taille idéale de la population de la sélection clonale est 140, en même temps, nous notons que la charge du temps additionnelle par rapport à la taille de la population égale à 40 est tolérable (535-480=75 secondes). Notons aussi que le gain en termes d'optimalité est considérable ($76\% - 51\% = 25\%$). Par la suite nous adoptons la taille 140 comme valeur par défaut pour le reste des algorithmes.

Selon la table 6.8, nous constatons une convergence rapide de la sélection clonale avec 140 cellules. En effet après 200/300 itérations, les performances stagnent et ne peuvent dépasser 0.82. Par conséquent, nous pouvons arrêter l'exécution de l'algorithme au niveau de la 300^{eme} itération (c'est à dire 1300 secondes).

Tableau 6.7: *Optimalité et temps d'exécution pour l'algorithme de sélection clonale avec 100 itérations*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	20	0.51	98
Exécution 2	40	0.51	480
Exécution 3	60	0.64	420
Exécution 4	140	0.76	535

Tableau 6.8: *Optimalité et temps d'exécution pour l'algorithme de sélection clonale*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.76	535
Exécution 2	200	0.80	940
Exécution 3	300	0.82	1300
Exécution 4	500	0.82	1650
Exécution 5	1000	0.82	3100

6.4.2.3 Performance de L'hybridation de la Sélection Clonale et de l'Optimisation par Essaim Particulaire

La table 6.9 montre les résultats de l'algorithme hybride SC-PSO pour une taille de population égale à 100, nous constatons que l'algorithme converge vers l'optimum global après 500 itérations, de même nous constatons que les résultats deviennent assez satisfaisants dès l'itération 100 (puisque le taux d'optimalité ≥ 0.87). Enfin, nous observons que le temps d'exécution est abordable puisque l'optimum global est obtenu en moins de 261 secondes. Ces résultats prometteurs sont justifiés comme suit :

L'algorithme PSO est connu avec sa bonne performance sur les problèmes d'optimisation difficile, (en termes d'optimalité) mais il souffre de la lenteur d'exécution en même temps l'algorithme SC possède un temps de convergence rapide mais il coïncide généralement avec un optimum local, donc si nous pouvons injecter dans la population de l'algorithme SC, des solutions pertinentes extraites du PSO, alors les performances du SC peuvent être boostées de manière considérable, et ceci est confirmé par les résultats de la table 6.9.

Pour comparer l'algorithme SC-PSO avec les algorithmes de l'état de l'art [Hadjila, 2014]. Nous adoptons la taille 140 comme valeur par défaut de la taille de la population. Selon les tableaux 6.10, 6.11, 6.12, et les figures 6.5, 6.6, nous constatons que l'algorithme SC-PSO converge vers l'optimum global alors que l'algorithme génétique et l'algorithme des abeilles convergent uniquement vers un optimum local, en plus nous observons que le temps d'exécution de SC-PSO est plus faible que les autres algorithmes. En conclusion, ces résultats confirment que les opérateurs de déplacement à base de mémoire (ceux du PSO) et les opérateurs d'hypermutation (ceux du SC) sont plus prometteurs que les opérateurs de croisement (algorithme génétique) ou de recherche locale (algorithme des abeilles).

Tableau 6.9: *Optimalité et temps d'exécution pour pour l'algorithme hybride sélection clonale-pso avec /Pop/=100*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.87	54
Exécution 2	200	0.96	96
Exécution 3	300	0.97	148
Exécution 4	500	1	261

Tableau 6.10: *Optimalité et temps d'exécution pour pour l'algorithme hybride sélection clonale-pso avec /Pop/=140*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.88	58
Exécution 2	200	0.92	153
Exécution 3	300	0.97	251
Exécution 4	500	1	355

Tableau 6.11: *Optimalité et temps d'exécution pour pour l'algorithme génétique*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.69	358
Exécution 2	200	0.84	719
Exécution 3	300	0.85	1080
Exécution 4	500	0.85	1882

Tableau 6.12: *Optimalité et temps d'exécution pour l'algorithme d'abeilles*

	Itérations	Taux d'optimalité	Temps d'exécution médian(S)
Exécution 1	100	0.35	125
Exécution 2	200	0.76	240
Exécution 3	300	0.38	350
Exécution 4	500	0.83	690

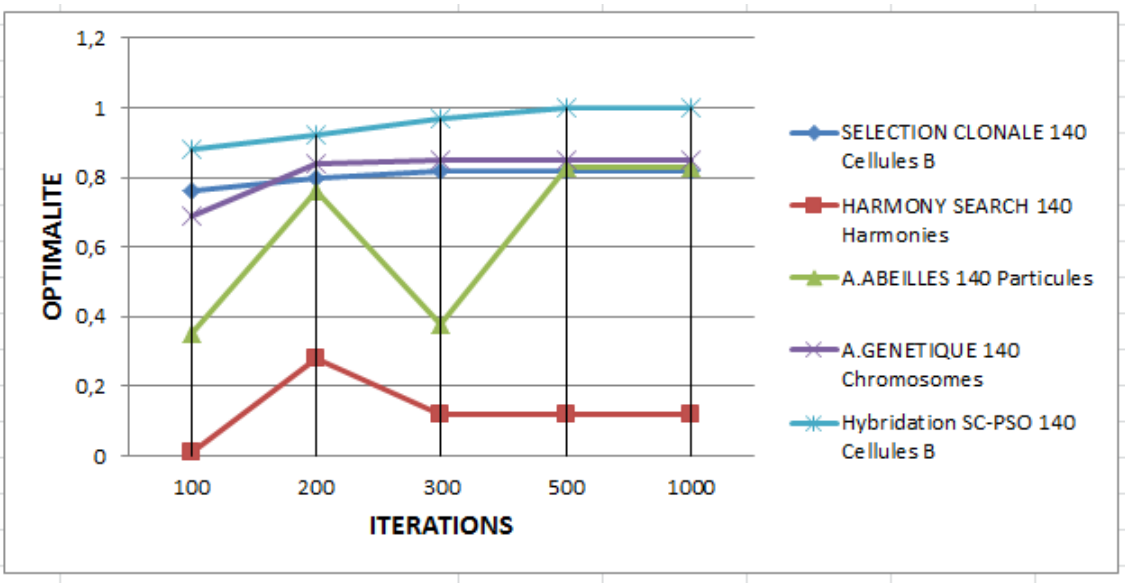

Figure *6.5:* *Le taux d'optimalité obtenu pour les approches de sélection*

La Figure 6.5 résume les performances d'optimalité de nos trois contributions ainsi que l'algorithme génétique et l'algorithme d'abeilles. Nous constatons clairement que l'algorithme SC-PSO est capable d'atteindre l'optimum globale dès l'itération 500, de même la figure 6.6 montre la supériorité de l'algorithme SC-PSO (en termes de temps) par rapport aux quatre approches restantes.

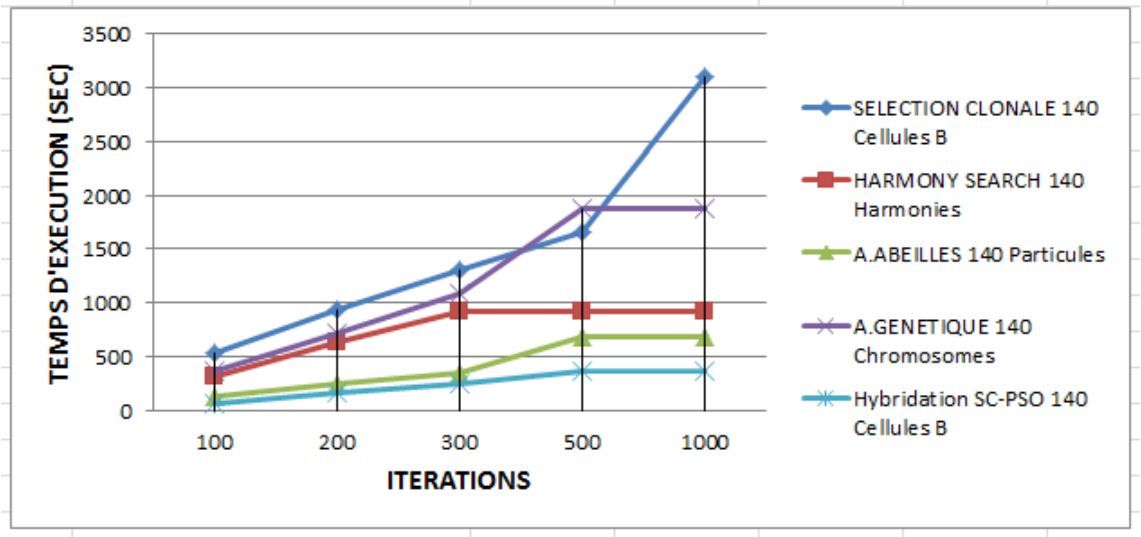

Figure *6.6:* *Le temps d'exécution associé aux approches de sélection*

6.5 Menaces de validité

6.5.1 Validité de construction

La découverte des services web

Les tables 6.2, 6.3, 6.4, 6.5 montrent le rappel, la précision et le MAP. Ces métriques constituent les mesures fondamentales pour évaluer tout système de recherche. Pour le cas K'=30 (qui représente un compromis entre le rappel et la précision), notre approche a montré que le temps d'exécution (Median) par requête est inférieur à 35 secondes. Par conséquent, l'approche est apte à être utilisée dans des cas pratiques.

La sélection des services web

La mesure de performance retenue pour ce problème consiste à agréger les différents critères de QoS, avec une pondération équitable (puisque l'utilisateur n'a pas de préférences à priori). Nous avons aussi utilisé un terme de pénalité pour défavoriser les solutions qui violent les contraintes globales, ce genre de terme est largement utilisé dans les problèmes d'optimisation difficile.

Les tables 6.9, 6.10 montrent les performances réalisées par l'algorithme SC-PSO, les résultats de la Figure 6.9 confirment (en moyenne) que l'optimum global (en termes de QoS) est obtenu en moins de 5 minutes, ce qui est toujours pratique pour l'utilisateur final. En plus une solution quasi-optimale peut être toujours dérivée, avant l'achèvement de la première minute (optimalité=0.87, temps d'exécution=54 secondes).

6.5.2 Validité interne

La découverte des services web

Les données d'expérimentation sont collectées à partir des registres UDDI d'IBM, ensuite elles sont converties en Format OWL-S. Ces données sont réelles et décrivent les applications web des entreprises enregistrées dans l'UDDI. D'autre part, les paramètres techniques $(k', \varepsilon, \lambda)$ de l'approche de Fusion sont largement expliqués dans les Figures 6.3 et 6.4.

La sélection des services web

Les données d'expérimentation sont générées aléatoirement selon la loi uniforme (qui est développée en Java), ceci est dû au manque aux données réelles, et relatives aux compositions de services. Les paramètres techniques de SC-PSO (taille de la population, taille de voisinage) sont clairement justifiés dans les Figures 6.9 et 6.10.

6.5.3 Validité externe

La découverte des services web

Chaque service web de la collection de Test possède une description en termes de concepts d'entrées/sorties (et sans la présence des pré-conditions, post-conditions). Les services sont annotés avec plus de 40 ontologies. La base contient 27 requêtes et 1007 services web répartis sur des domaines différents (éducation, économie, santé, ...), en plus l'approche FDDA fusionne un ensemble de listes sources dont la taille peut aller jusqu'à 1007 services. Avec ces hypothèses, nous estimons que notre approche est compatible avec les contraintes d'environnement réel.

La sélection des services web

Ce corpus est aléatoirement généré grâce à la loi uniforme. Nous disposons de cinq critères de QoS (qui représentent les critères les plus courants dans le domaine des applications web), et dont les bornes sont spécifiées dans la table 6.1. Pour faire face aux contraintes de l'environnement réel, nous avons adopté la requête (i.e les contraintes globales) de [Hadjila, 2014] avec 10 classes abstraites (ce qui est largement suffisant pour les cas pratiques), de même, nous avons varié le nombre de services pour chaque classe entre 10 et 100 éléments. Ceci nous donne au maximum de 10010 compositions possibles à inspecter.

6.5.4 Validité de conclusion

La découverte des services web

Pour chaque algorithme (mesure de similarité ou schéma de fusion), nous avons utilisé 5 exécutions et nous avons retenu le temps d'exécution median. Pour l'apprentissage (Figure 6.3), nous avons exécuté le même processus 10 fois. Dans chaque simulation, nous avons atteint le même optimum (qui est proche de 0.7). Nous avons aussi pris la mesure « perte d'informations » et l'algorithme Borda comme lignes de comparaison (baseline comparison).

La sélection des services web

Pour chaque algorithme(SC, HS, SC-PSO, Abeilles, AG) nous avons réalisé 10 exécutions pour s'échapper des influences aléatoires des conditions de départ. Nous avons aussi pris en considération la même taille de population et le même nombre d'itérations pour tous les algorithmes. Nous avons retenu le temps d'exécution median des 10 exécutions, de même nous avons presque le même taux d'optimalité pour les exécutions du même type d'algorithme.

6.6 Conclusion

Dans ce chapitre, nous avons présenté notre prototype dénommé DSS, ce dernier permet de répondre aux problématiques de découverte et sélection des services web à base de QoS.

Nous avons proposé une approche de découverte qui fusionne quatre fonctions de matching appartenant à la classe non-logique, et une fonction appartenant à la classe logique . La fusion est basée sur un algorithme d'agrégation qui s'articule sur une relation de dominance floue 'FDAA'. L'ensemble des expérimentations menées montrent la supériorité de l'approche proposée en terme de Rappel et Précision.

Pour résoudre la problématique de sélection, nous avons proposé trois approches. La première approche est basée sur l'algorithme Harmony search, la deuxième approche est basée sur l'algorithme de sélection clonale (clonAlg). Et la troisième approche est basée sur un algorithme hybride (SC-PSO) de sélection clonale et l'optimisation par essaim particulaire (PSO) afin de combiner les avantages des deux algorithmes (la sélection clonale et PSO). Les expérimentations menées confirment la capacité des approches HS (Harmony Search), sélection clonale, et hybridation SC-PSO, à obtenir des solutions proches de l'optimale.

Chapitre 7

Conclusion et perspectives

Grâce au développement accéléré des technologies de l'information et l'avènement des services web, l'Internet se transforme d'un simple vecteur d'échange de données, en une plate-forme de composants auto-descriptifs, modulaires, facilement intégrables et faiblement couplés. Récemment, les services web ont émergé comme l'instanciation de-facto de l'approche SOA, permettant ainsi à l'entreprise de se libérer de sa position fermée et d'explorer de nouvelles pistes de collaboration avec d'autres entreprises. Avec l'immense prolifération des services web sur le réseau internet, les utilisateurs auront toujours besoin de techniques efficaces pour automatiser les mécanismes de découverte et de sélection des services web appropriés relativement à une requête utilisateur. C'est dans ce contexte que s'inscrivent les travaux de cette thèse.

7.1 Synthèse

La première partie de cette thèse est réservée aux travaux d'état de l'art, nous avons présenté les principaux standards liés à la technologie des services web, ainsi que leur architecture et leurs avantages.

Après la spécification du problème de la découverte des services web, nous avons élaboré un état de l'art détaillé survolant les différentes approches qui permettent sa résolution. La classification des travaux de l'état de l'art a permis d'une part de dégager les atouts et les limites de chaque approche, et d'autre part de mieux positionner notre approche de découverte pour voir comment il serait possible d'améliorer les critères de performance.

Nous avons aussi analysé, modélisé et formulé le problème de sélection des services web composés à base de QoS. Nous avons présenté les approches d'optimisation dédiées à la résolution de ce problème. Elles sont groupées en 4 catégories qui sont : les approches de résolutions exactes, les heuristiques approximatives, les métaheuristiques et les approches basées sur la dominance au sens de pareto.

La deuxième partie est consacrée aux contributions où nous avons décrit l'ensemble de nos propositions. Concernant la problématique de découverte, nous avons proposé une approche qui fusionne quatre fonctions de matching (similarité) appartenant à la classe non-logique, et une fonction appartenant à la classe logique. Ceci permet l'adoption des avantages de chaque fonction. La fusion est basée sur un algorithme d'agrégation qui s'articule sur une relation de dominance floue 'FDAA' (Fuzzy Dominance Aggregating Algorithm). FDDA permet de trouver les services pertinents par rapport à la requête de l'utilisateur. L'ensemble des expérimentations menées montrent la supériorité de l'approche proposée en terme de Rappel et Précision.

Pour résoudre la problématique de sélection des services web composés à base de QoS, nous avons proposé trois approches. La première approche est basée sur

l'algorithme Harmony search. Ce dernier a prouvé son efficacité dans la résolution des problèmes NP-hard. Nous avons présenté une fonction mono-objectif qui prend en considération les préférences de l'utilisateur en termes de QoS. La deuxième approche est basée sur l'algorithme de sélection clonale (clonAlg). Nous avons adapté l'algorithme clonAlg au problème de la sélection des services web pour mieux trouver les solutions quasi-optimales. En effet, l'élargissement de l'espace de recherche (grâce aux opérateurs de changement aléatoire de services) permet d'orienter la recherche vers les zones prometteuses. Et la troisième approche est basée sur un algorithme hybride (SC-PSO) de sélection clonale et l'optimisation par essaim particulaire (PSO) afin de combiner les avantages des deux algorithmes (la sélection clonale et PSO)

Nous avons aussi validé et mis en oeuvre nos propositions à travers le développement de prototype ***DSS*** (Environnement de **D**écouverte et de **S**élection de **S**ervices). Les expérimentations réalisées à travers ce prototype nous ont permis de constater que, d'une part, l'approche FDDA domine la majorité des travaux de l'état de l'arten termes de MAP (Mean Average Precision) et d'autre part, la recherche harmonique ainsi que la sélection clonale peuvent trouver la solution optimale pour une composition de taille inférieure à 10.

7.2 Perspectives

Comme perspectives à ce travail, nous pouvons citer les pistes suivantes :

- Proposition de nouveaux schémas de fusion qui se basent sur :
    1. Des relations de dominance probabiliste et éventuellement l'apprentissage de distributions de probabilités qui mesure le degré d'adéquation d'un service par rapport à une requête.
    2. La théorie du choix social qui regroupe plusieurs types d'algorithmes de votes (vote majoritaire, jugement majoritaire, vote pluraliste,...)
- L'utilisation de nouveaux corpus de services web qui se basent sur la technologie REST. Nous notons que les services REST sont largement employé par les entreprises par rapport aux services SOAP (en raison de leur performance dû à la politique du cache et leur caractère sans état qui n'exige pas de sessions entre le client et le serveur).
- La prise en compte de l'incertitude et la variabilité de la QoS pendant la sélection des services web. En effet la QoS réelle est toujours fluctuante et dépend largement de l'environnement sous-jacent (par exemple le temps d'exécution d'un service dépend de la charge du réseau et des heures de pointes, le coût d'un service dépend de la saison (réservation d'hôtel)). Donc pour gérer tous ces aspects, il sera judicieux de considérer la QoS comme une variable aléatoire, éventuellement nous pouvons estimer sa distribution de probabilité à partir des réalisations empiriques. En partant de ces faits, le problème de sélection avec QoS incertaine consiste à trouver des compositions qui maximisent la probabilité de satisfaction des contraintes globales, pour réaliser cet objectif, il sera utile de composer les contraintes globales en contraintes locales (restreintes aux taches), cette décomposition est généralement guidée avec des heuristiques. Avec les contraintes locales, il sera facile d'évaluer la probabilité de satisfaction des contraintes globales.

- Actuellement, le déploiement des services web sur les plateformes de cloud computing est de plus en plus répondu, par conséquent le problème de sélection des services web doit être adapté au contexte du cloud computing. En particulier tout système de sélection des services cloud doit :
  1. Prendre la QoS à long terme (et pas de QoS ponctuelle). En effet La QoS à long terme permet aux entreprises de faire le bon investissement dans les infrastructures du I.T (Technologies d'information)
  2. Par opposition à la sélection des services traditionnelle, le modèle économique régissant la sélection des services web cloud suppose que les poids associés aux critères de QoS sont variables et changent avec le temps, en général ils sont modélisés avec des réseaux bayésiens qui montrent leur dynamicité en fonction du temps.
  3. La sélection des services cloud doit être étendue pour tenir en compte des services IAAS (Infrastructure As A Service) en plus des services traditionnels SAAS (Software As A Service). les services SAAS représentent des services web traditionnels accessibles avec un compte sur une plateforme de cloud computing. Par contre, les services IAAS représentent des programmes qui allouent des ressources logiques (ex. systèmes d'exploitation) ou physiques (espace de stockage, RAM, processus, bande passante,...) sur des machines appartenant à la plateforme de cloud computing.

Liste des publications

- Journaux internationaux avec comité de lecture

Merzoug Mohammed, Mohammed Amine Chikh, Hadjila Fethallah : 'Leveraging fuzzy dominance relationship and machine learning for hybrid web service discovery'. International Journal of Web Engineering and Technology. 11(2) : pages 107-132 (2016).

- Conférences internationales avec comité de lecture

Merzoug Mohammed, Mohammed Amine Chikh and Hadjila Fethallah. 'QoS-aware web service selection based on harmony search'. In 4th. International Symposium ISKO-Maghreb "Concepts and Tools for Knowledge Management (KM)", 9-10 Nov. (2014).

Hadjila Fethallah, Merzoug Mohammed and Belabed Amine. 'Semantic Web service Discovery Based on Fuzzy Dominated Scores'. In Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC '15). Article No. 17 , Batna, Algeria — November 23 - 25, 2015 .

Merzoug Mohammed, Mohamed Amine Chikh, Fethallah Hadjila. 'Comparison of Bio-Inspired Algorithms for QoS-aware service selection'. In The 2015 International Conference on Advanced Communication Systems and Signal Processing (ICOSIP '2015). Tlemcen, Algeria- November 8-9, 2015.

Hadjila Fethallah, Smahi Mohamed Ismail, Merzoug Mohamed, Torchane Zeyneb. 'An outranking model for web service discovery'. In 2017 International Conference on Mathematics and Information Technology (ICMIT '2017).

Bibliographie

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer.

[Akbar et al., 2001] Akbar, M., Manning, E., Shoja, G., and Khan, S. (2001). Heuristic solutions for the multiple-choice multi-dimension knapsack problem. *Computational science-ICCS 2001*, pages 659–668.

[Akbar et al., 2006] Akbar, M. M., Rahman, M. S., Kaykobad, M., Manning, E. G., and Shoja, G. C. (2006). Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & operations research*, 33(5) :1259–1273.

[Al-Masri and Mahmoud, 2008] Al-Masri, E. and Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804. ACM.

[Alrifai and Risse, 2009] Alrifai, M. and Risse, T. (2009). Combining global optimization with local selection for efficient qos-aware service composition. In *Proceedings of the 18th international conference on World wide web*, pages 881–890. ACM.

[Alrifai et al., 2012] Alrifai, M., Risse, T., and Nejdl, W. (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2) :7.

[Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T. (2010). Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20. ACM.

[Amine, 2017] Amine, M. C. (2017). *La sélection des services web dans une composition à base de critères non fonctionnels.* PhD thesis, UNIVERSITÉ ABOU-BEKR BELKAID-TLEMCEN.

[Ankolekar et al., 2002] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., et al. (2002). Daml-s : Web service description for the semantic web. *The Semantic Web—ISWC 2002*, pages 348–363.

[Ardagna and Pernici, 2005] Ardagna, D. and Pernici, B. (2005). Global and local qos constraints guarantee in web service selection. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6).

[Bahadori et al., 2009] Bahadori, S., Kafi, S., Far, K., and Khayyambashi, M. (2009). Optimal web service composition using hybrid ga-tabu search. *Journal of Theoretical and Applied Information Technology*, 9(1) :10–15.

[Bechhofer, 2009] Bechhofer, S. (2009). Owl : Web ontology language. In *Encyclopedia of Database Systems*, pages 2008–2009. Springer.

[Bekkouche et al., 2017] Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine, C., Hadjila, F., and Merzoug, M. (2017). Qos-aware optimal and automated semantic web service composition with user's constraints. *Service Oriented Computing and Applications*, pages 1–19.

[Benatallah et al., 2005a] Benatallah, B., Dijkman, R. M., Dumas, M., and Maamar, Z. (2005a). Service composition : Concepts, techniques. *Service-Oriented Software System Engineering : Challenges and Practices*, page 48.

[Benatallah et al., 2005b] Benatallah, B., Hacid, M.-S., Leger, A., Rey, C., and Toumani, F. (2005b). On automating web services discovery. *The VLDB Journal—The International Journal on Very Large Data Bases*, 14(1) :84–96.

[Benouaret et al., 2011] Benouaret, K., Benslimane, D., and Hadjali, A. (2011). On the use of fuzzy dominance for computing service skyline based on qos. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 540–547. IEEE.

[Benouaret et al., 2014] Benouaret, K., Benslimane, D., Hadjali, A., Barhamgi, M., Maamar, Z., and Sheng, Q. Z. (2014). Web service compositions with fuzzy preferences : A graded dominance relationship-based approach. *ACM Transactions on Internet Technology (TOIT)*, 13(4) :12.

[Berbner et al., 2006] Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. (2006). Heuristics for qos-aware web service composition. In *Web Services, 2006. ICWS'06. International Conference on*, pages 72–82. IEEE.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5) :28–37.

[Bernstein et al., 2005] Bernstein, A., Kaufmann, E., Kiefer, C., and Bürki, C. (2005). Simpack : A generic java library for similarity measures in ontologies. *University of Zurich*, page 20.

[Bernstein and Kiefer, 2005] Bernstein, A. and Kiefer, C. (2005). irdql-imprecise rdql queries using similarity joins. In *3rd International Conference on Knowledge Capture (K-CAP)*.

[Bertoli et al., 2004] Bertoli, P., Cimatti, A., and Traverso, P. (2004). Interleaving execution and planning for nondeterministic, partially observable domains. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 657–661. IOS Press.

[Bianchini et al., 2008] Bianchini, D., De Antonellis, V., and Melchiori, M. (2008). Flexible semantic-based service matchmaking and discovery. *World Wide Web*, 11(2) :227–251.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan) :993–1022.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture.

[Boukhadra et al., 2016] Boukhadra, A., Benatchba, K., and Balla, A. (2016). Efficient distributed discovery and composition of owl-s process model in p2p systems. *Journal of Ambient Intelligence and Humanized Computing*, 7(2) :187–203.

[Brits et al., 2007] Brits, R., Engelbrecht, A. P., and van den Bergh, F. (2007). Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation*, 189(2) :1859–1883.

[Brownlee, 2011] Brownlee, J. (2011). *Clever algorithms : nature-inspired programming recipes*. Jason Brownlee.

[Canfora et al., 2005] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005). An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1069–1075. ACM.

[Cardoso, 2007] Cardoso, J. (2007). *Semantic Web Services : Theory, Tools and Applications : Theory, Tools and Applications*. IGI Global.

[Carey, 2008] Carey, M. J. (2008). Soa what ? *Computer*, 41(3).

[Caseau et al., 1999] Caseau, Y., Habib, M., Nourine, L., and Raynaud, O. (1999). Encoding of multiple inheritance hierarchies and partial orders. *Computational Intelligence*, 15(1) :50–62.

[Cassar et al., 2014] Cassar, G., Barnaghi, P., and Moessner, K. (2014). Probabilistic matchmaking methods for automated service discovery. *IEEE Transactions on Services Computing*, 7(4) :654–666.

[Chabeb, 2011] Chabeb, Y. (2011). *Contributions à la description et la découverte de services web sémantiques*. PhD thesis, Institut National des Télécommunications.

[Chan et al., 2006] Chan, C.-Y., Jagadish, H., Tan, K.-L., Tung, A. K., and Zhang, Z. (2006). On high dimensional skylines. In *International Conference on Extending Database Technology*, pages 478–495. Springer.

[Chankong and Haimes, 2008] Chankong, V. and Haimes, Y. Y. (2008). *Multiobjective decision making : theory and methodology*. Courier Dover Publications.

[Chen and Wang, 2007] Chen, M. and Wang, Z.-w. (2007). An approach for web services composition based on qos and discrete particle swarm optimization. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 2, pages 37–41. IEEE.

[Chen et al., 2015] Chen, Y., Huang, J., Lin, C., and Hu, J. (2015). A partial selection methodology for efficient qos-aware service composition. *IEEE Transactions on Services Computing*, 8(3) :384–397.

[Chifu et al., 2010] Chifu, V. R., Pop, C. B., Salomie, I., Dinsoreanu, M., Niculici, A. N., and Suia, D. S. (2010). Selecting the optimal web service composition based on a multi-criteria bee-inspired method. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, pages 40–47. ACM.

[Chifu et al., 2011] Chifu, V. R., Pop, C. B., Salomie, I., Suia, D. S., and Niculici, A. N. (2011). Optimizing the semantic web service composition process using cuckoo search. In *Intelligent distributed computing V*, pages 93–102. Springer.

[Chotipant et al., 2014] Chotipant, S., Hussain, F. K., Dong, H., and Hussain, O. K. (2014). A fuzzy vsm-based approach for semantic service retrieval. In *International Conference on Neural Information Processing*, pages 682–689. Springer.

[Claro, 2006] Claro, D. B. (2006). Spoc-un canevas pour la composition automatique de services web dédiésa la réalisation de devis. *These de Doctorat, Ecole Doctorale d'Angers*.

[Comes et al., 2010] Comes, D., Baraki, H., Reichle, R., Zapf, M., and Geihs, K. (2010). Heuristic approaches for qos-based service selection. In *International Conference on Service-Oriented Computing*, pages 441–455. Springer.

[De Bruijn et al., 2008] De Bruijn, J., Fensel, D., Kerrigan, M., Keller, U., Lausen, H., and Scicluna, J. (2008). The web service modeling ontology. *Modeling Semantic Web Services : The Web Service Modeling Language*, pages 23–28.

[De Castro and Von Zuben, 2002] De Castro, L. N. and Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE transactions on evolutionary computation*, 6(3) :239–251.

[De La Rosa-Rosero and Estublier, 2004] De La Rosa-Rosero, R. and Estublier, J. (2004). *Découverte et Sélection de Services Web pour une application Mélusine*. Université J. Fourier, UFR Informatique et Mathématiques Appliquées.

[Dodani, 2004] Dodani, M. H. (2004). From objects to services : A journey in search of component reuse nirvana. *Journal of Object Technology*, 3(8) :49–54.

[Dong et al., 2004] Dong, X., Halevy, A., Madhavan, J., Nemes, E., and Zhang, J. (2004). Similarity search for web services. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 372–383. VLDB Endowment.

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

[Doulkeridis et al., 2006] Doulkeridis, C., Loutas, N., and Vazirgiannis, M. (2006). A system architecture for context-aware service discovery. *Electronic Notes in Theoretical Computer Science*, 146(1) :101–116.

[Eberhart and Kennedy, 1995] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE.

[Emani, 2009] Emani, S. (2009). A comparative evaluation of semantic web service discovery : Algorithms and engines. *Computer Science Athens : University of Georgia*.

[Esfahani et al., 2012] Esfahani, P. M., Habibi, J., and Varaee, T. (2012). Application of social harmony search algorithm on composite web service selection based on quality attributes. In *Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on*, pages 526–529. IEEE.

[Falquet and Mottaz Jiang, 2001] Falquet, G. and Mottaz Jiang, C.-L. (2001). Navigation hypertexte dans une ontologie multi-points de vue.

[Feng1/2 et al., 2013] Feng1/2, L.-i., Obayashi1/2, M., Kuremoto1/2, T., Kobayashi, K., and Watanabe1/2, S. (2013). Qos optimization for web services composition based on reinforcement learning.

[Fensel and Bussler, 2002] Fensel, D. and Bussler, C. (2002). The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2) :113–137.

[Fuglede and Topsoe, 2004] Fuglede, B. and Topsoe, F. (2004). Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 31. IEEE.

[Gabrel et al., 2014] Gabrel, V., Manouvrier, M., and Murat, C. (2014). Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming. In *International Conference on Service-Oriented Computing*, pages 108–122. Springer.

[Gao et al., 2006] Gao, A., Yang, D., Tang, S., and Zhang, M. (2006). Qos-driven web service composition with inter service conflicts. *Frontiers of WWW Research and Development-APWeb 2006*, pages 121–132.

[Garcia, 2006] Garcia, E. (2006). Cosine similarity and term weight tutorial. *Information retrieval intelligence*.

[Gardarin, 2002] Gardarin, G. (2002). Xml des bases de données aux services web.

[Geem, 2007] Geem, Z. W. (2007). Harmony search algorithm for solving sudoku. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 371–378. Springer.

[Geem, 2009] Geem, Z. W. (2009). *Optimal design of water distribution networks using harmony search*. Lap Lambert Academic Pub.

[Geem et al., 2001] Geem, Z. W., Kim, J. H., and Loganathan, G. (2001). A new heuristic optimization algorithm : harmony search. *Simulation*, 76(2) :60–68.

[Georgakopoulos et al., 2009] Georgakopoulos, D., Papazoglou, M., et al. (2009). *Service-oriented computing*. Number Sirsi) i9780262072960.

[Glover, 1986] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5) :533–549.

[Gonsalves et al., 2009] Gonsalves, T., Yamagishi, K., and Toh, K. (2009). Service cost and waiting time-a multi-objective optimization scenario. In *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the*, pages 369–374. IEEE.

[Group et al., 2004] Group, W. W. et al. (2004). Web services architecture. *http ://www. w3. org/TR/ws-arch/.*

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on strings, trees and sequences : computer science and computational biology*. Cambridge university press.

[Hadjila, 2014] Hadjila, F. (2014). *Composition et interopération des services web sémantiques.* PhD thesis.

[Hadjila et al., 2012] Hadjila, F., Chikh, M. A., and Merzoug, M. (2012). Qos-aware service selection based on clonal selection. In *Proceedings of ICACIS '12 Batna. Algerie. 2012.*

[Hadjila et al., 2013] Hadjila, F., Chikh, M. A., and Merzoug, M. (2013). Qos-aware web service selection based on bees algorithm. In *10eme colloque sur l'optimisation et les systemes d'informations COSI '13. Alger Algerie.*

[Hadjila et al., 2017] Hadjila, F., Smahi, M., and Merzoug, M. (2017). An outranking model for web service discovery. IEEE.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

[Huhns and Singh, 2005] Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing : Key concepts and principles. *IEEE Internet computing*, 9(1) :75–81.

[Jaeger and Mühl, 2007] Jaeger, M. C. and Mühl, G. (2007). Qos-based selection of services : The implementation of a genetic algorithm. In *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, pages 1–12. VDE.

[Jaeger et al., 2005] Jaeger, M. C., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., and Geihs, K. (2005). Ranked matching for service descriptions using owl-s. In *Kommunikation in Verteilten Systemen (KiVS)*, pages 91–102. Springer.

[Jafarpour and Khayyambashi, 2010] Jafarpour, N. and Khayyambashi, M. R. (2010). Qos-aware selection of web service compositions using harmony search algorithm. *Journal of Digital Information Management*, 8(3) :160–166.

[Jatoth et al., 2015] Jatoth, C., Gangadharan, G., and Buyya, R. (2015). Computational intelligence based qos-aware web service composition : A systematic literature review. *IEEE Transactions on Services Computing*.

[Josuttis, 2007] Josuttis, N. M. (2007). *SOA in practice : the art of distributed system design*. " O'Reilly Media, Inc.".

[Kadima and Monfort, 2003] Kadima, H. and Monfort, V. (2003). *Les Web services : techniques, démarches et outils XML, WSDL, SOAP, UDDI, RosettaNet, UML*. Dunod.

[KAMAL et al., 2014] KAMAL, S., IBRAHIM, R., and GHANI, I. (2014). Review on service selection schemes based on user preferences. *Journal of Theoretical & Applied Information Technology*, 70(2).

[Karaboga, 2005] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[Kaufer and Klusch, 2006] Kaufer, F. and Klusch, M. (2006). Wsmo-mx : A logic programming based hybrid service matchmaker. In *Web Services, 2006. ECOWS'06. 4th European Conference on*, pages 161–170. IEEE.

[Keller et al., 2005] Keller, U., Lara, R., Lausen, H., Polleres, A., and Fensel, D. (2005). Automatic location of services. *The Semantic Web : Research and Applications*, pages 1–16.

[Khan, 1998] Khan, M. S. (1998). *Quality adaptation in a multisession multimedia system : Model, algorithms and architecture*. PhD thesis, University of Victoria.

[Khanouche et al., 2016] Khanouche, M. E., Amirat, Y., Chibani, A., Kerkar, M., and Yachir, A. (2016). Energy-centered and qos-aware services selection for internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3) :1256–1269.

[Kiefer, 2009] Kiefer, C. (2009). *Non-Deductive Reasoning for the Semantic Web and Software Analysis*. PhD thesis, University of Zurich.

[Kiefer and Bernstein, 2008] Kiefer, C. and Bernstein, A. (2008). The creation and evaluation of isparql strategies for matchmaking. In *European Semantic Web Conference*, pages 463–477. Springer.

[Kiefer et al., 2007] Kiefer, C., Bernstein, A., Lee, H., Klein, M., and Stocker, M. (2007). Semantic process retrieval with isparql. *The Semantic Web : Research and Applications*, pages 609–623.

[Kim and Geem, 2015] Kim, J. H. and Geem, Z. W. (2015). *Harmony Search Algorithm : Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA2015)*, volume 382. Springer.

[Kim et al., 2016] Kim, M., Oh, B., Jung, J., and Lee, K.-H. (2016). Outlier-robust web service selection based on a probabilistic qos model. *International Journal of Web and Grid Services*, 12(2) :162–181.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simmulated annealing. *science*, 220(4598) :671–680.

[Klein et al., 2011] Klein, A., Ishikawa, F., and Honiden, S. (2011). Efficient heuristic approach with improved time complexity for qos-aware service composition. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 436–443. IEEE.

[Klein and König-Ries, 2004] Klein, M. and König-Ries, B. (2004). Coupled signature and specification matching for automatic service binding. In *Web Services*, pages 183–197. Springer.

[Klein et al., 2005] Klein, M., Konig-Ries, B., and Mussig, M. (2005). What is needed for semantic service descriptions ? a proposal for suitable language constructs. *International Journal of Web and Grid Services*, 1(3-4) :328–364.

[Klusch, 2014] Klusch, M. (2014). Service discovery. In *Encyclopedia of Social Network Analysis and Mining*, pages 1707thallah–1717. Springer.

[Klusch et al., 2006] Klusch, M., Fries, B., and Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922. ACM.

[Klusch and Kapahnke, 2008] Klusch, M. and Kapahnke, P. (2008). Semantic web service selection with sawsdl-mx. In *Proceedings of the Second International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 416*, pages 2–16. CEUR-WS. org.

[Klusch and Kapahnke, 2009] Klusch, M. and Kapahnke, P. (2009). Owls-mx3 : an adaptive hybrid semantic service matchmaker for owl-s. In *Proceedings of 3rd International Workshop on Semantic Matchmaking and Resource Retrieval (SMR2), USA*.

[Klusch et al., 2008] Klusch, M., Kapahnke, P., and Kaufer, F. (2008). Evaluation of wsml service retrieval with wsmo-mx. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 401–408. IEEE.

[Kousalya et al., 2011] Kousalya, G., Palanikkumar, D., and Piriyankaa, P. (2011). Optimal web service selection and composition using multi-objective bees algorithm. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 193–196. IEEE.

[Kreger et al., 2001] Kreger, H. et al. (2001). Web services conceptual architecture (wsca 1.0). *IBM Software Group*, 5 :6–7.

[Krithiga, 2012] Krithiga, R. (2012). Qos-aware web service selection using soma. *GJCST-E : Network, Web & Security*, 12(10).

[Kuster and Konig-Ries, 2007] Kuster, U. and Konig-Ries, B. (2007). Semantic service discovery with diane service descriptions. In *Proceedings of the 2007 IEEE/-WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, pages 152–156. IEEE Computer Society.

[Küster and König-Ries, 2008] Küster, U. and König-Ries, B. (2008). Evaluating semantic web service matchmaking effectiveness based on graded relevance. In *Proceedings of the Second International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 416*, pages 32–46. CEUR-WS. org.

[Küster et al., 2007] Küster, U., König-Ries, B., Klein, M., and Stern, M. (2007). Diane : A matchmaking-centered framework for automated service discovery, composition, binding, and invocation on the web. *International Journal of Electronic Commerce*, 12(2) :41–68.

[Lahoud, 2010] Lahoud, E. A. (2010). *Composition dynamique de services : application à la conception et au développement de systèmes d'information dans un environnement distribué*. PhD thesis, Université de Bourgogne.

[Lausen and Farrell, 2007] Lausen, H. and Farrell, J. (2007). Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C*, 69.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[Li et al., 2007] Li, H., Du, X., and Tian, X. (2007). A wsmo-based semantic web services discovery framework in heterogeneous ontologies environment. In *International Conference on Knowledge Science, Engineering and Management*, pages 617–622. Springer.

[Li et al., 2010] Li, J., Zhao, Y., Liu, M., Sun, H., and Ma, D. (2010). An adaptive heuristic approach for distributed qos-based service composition. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 687–694. IEEE.

[Li et al., 2014] Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., and Chen, D.-r. (2014). An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, 269 :238–254.

[Li et al., 2013] Li, M., Zhu, D., Deng, T., Sun, H., Guo, H., and Liu, X. (2013). Gos : a global optimal selection strategies for qos-aware web services composition. *Service Oriented Computing and Applications*, 7(3) :181–197.

[Li and Yan-Xiang, 2010] Li, W. and Yan-Xiang, H. (2010). A web service composition algorithm based on global qos optimizing with mocaco. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 218–224. Springer.

[Li and Wen, 2012] Li, Y.-Q. and Wen, T. (2012). An approach of qos-guaranteed web service composition based on a win-win strategy. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 628–630. IEEE.

[Liao et al., 2011] Liao, J., Liu, Y., Zhu, X., Xu, T., and Wang, J. (2011). Niching particle swarm optimization algorithm for service composition. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE.

[Lin et al., 2007] Lin, X., Yuan, Y., Zhang, Q., and Zhang, Y. (2007). Selecting stars : The k most representative skyline operator. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 86–95. IEEE.

[Liu et al., 2009] Liu, D., Shao, Z., Yu, C., and Fan, G. (2009). A heuristic qos-aware service selection approach to web service composition. In *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, pages 1184–1189. IEEE.

[Liu et al., 2010] Liu, H., Zhong, F., Ouyang, B., and Wu, J. (2010). An approach for qos-aware web service composition based on improved genetic algorithm. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 1, pages 123–128. IEEE.

[Liu et al., 2007] Liu, J., Li, J., Liu, K., and Wei, W. (2007). A hybrid genetic and particle swarm algorithm for service composition. In *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on*, pages 564–567. IEEE.

[Liu and Yin, 2009] Liu, X. and Yin, Z. (2009). Web service composition with global constraint based on discrete particle swarm optimization. In *Web Mining and Web-based Application, 2009. WMWA'09. Second Pacific-Asia Conference on*, pages 183–186. IEEE.

[Liu et al., 2011] Liu, Y., Miao, H., Li, Z., and Gao, H. (2011). Qos-aware web services composition based on hqpso algorithm. In *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, pages 400–405. IEEE.

[Long and Gui, 2009] Long, J. and Gui, W. (2009). An environment-aware particle swarm optimization algorithm for services composition. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4. IEEE.

[Ludwig, 2012] Ludwig, S. A. (2012). Applying particle swarm optimization to quality-of-service-driven web service composition. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 613–620. IEEE.

[Luo et al., 2011] Luo, Y.-s., Qi, Y., Hou, D., Shen, L.-f., Chen, Y., and Zhong, X. (2011). A novel heuristic algorithm for qos-aware end-to-end service composition. *Computer Communications*, 34(9) :1137–1144.

[Luo et al., 2008] Luo, Y.-s., Qi, Y., Shen, L.-f., Hou, D., Sapa, C., and Chen, Y. (2008). An improved heuristic for qos-aware service composition framework. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 360–367. IEEE.

[Maros, 2012] Maros, I. (2012). *Computational techniques of the simplex method*, volume 61. Springer Science & Business Media.

[Martin et al., 2004] Martin, D., Paolucci, M., McIlraith, S., Burnstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T. R., Sabou, M., Solanki, M., et al. (2004). Bringing semantics to web services : The owl-s approach.

[Merzoug et al., 2014] Merzoug, M., Chikh, M. A., and Hadjila, F. (2014). Qos-aware web service selection based on harmony search. In *ISKO-Maghreb : Concepts and Tools for knowledge Management (ISKO-Maghreb), 2014 4th International Symposium*, pages 1–6. IEEE.

[Mitra et al., 2003] Mitra, N., Lafon, Y., et al. (2003). Soap version 1.2 part 0 : Primer. *W3C recommendation*, 24 :12.

[Mohammed et al., 2016] Mohammed, M., Amine, C. M., and Fethallah, H. (2016). Leveraging fuzzy dominance relationship and machine learning for hybrid web service discovery. *International Journal of Web Engineering and Technology*, 11(2) :107–132.

[Monge et al., 1996] Monge, A. E., Elkan, C., et al. (1996). The field matching problem : Algorithms and applications. In *KDD*, pages 267–270.

[Moustafa and Zhang, 2013] Moustafa, A. and Zhang, M. (2013). Multi-objective service composition using reinforcement learning. In *International Conference on Service-Oriented Computing*, pages 298–312. Springer.

[Nemhauser and Wolsey, 1988] Nemhauser, G. L. and Wolsey, L. A. (1988). Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20 :8–12.

[Newcomer, 2002] Newcomer, E. (2002). *Understanding Web Services : XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional.

[Paolucci et al., 2002] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002). Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347. Springer.

[Parejo et al., 2008] Parejo, J. A., Fernandez, P., and Cortés, A. R. (2008). Qos-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, 2(1) :55–66.

[Parra-Hernandez and Dimopoulos, 2005] Parra-Hernandez, R. and Dimopoulos, N. J. (2005). A new heuristic for solving the multichoice multidimensional knapsack problem. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 35(5) :708–717.

[Plebani and Pernici, 2009] Plebani, P. and Pernici, B. (2009). Urbe : Web service retrieval based on similarity evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 21(11) :1629–1642.

[Pop et al., 2009] Pop, C. B., Chifu, V. R., Salomie, I., and Dinsoreanu, M. (2009). Immune-inspired method for selecting the optimal solution in web service composition. In *International Workshop on Resource Discovery*, pages 1–17. Springer.

[Pop et al., 2010] Pop, C. B., Chifu, V. R., Salomie, I., Dinsoreanu, M., David, T., and Acretoaie, V. (2010). Ant-inspired technique for automatic web service composition and selection. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on*, pages 449–455. IEEE.

[Qi et al., 2010] Qi, L., Tang, Y., Dou, W., and Chen, J. (2010). Combining local optimization and enumeration for qos-aware web service composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 34–41. IEEE.

[Rampacek, 2006] Rampacek, S. (2006). Sémantique, interactions et langages de description des services web complexes. *Doctorate dissertation, University of Reims ChampagneArdenne*.

[Rojas et al., 2002] Rojas, I., González, J., Pomares, H., Merelo, J., Castillo, P., and Romero, G. (2002). Statistical analysis of the main parameters involved in the design of a genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(1) :31–37.

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1) :77–106.

[Rosenberg et al., 2009] Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., and Dustdar, S. (2009). An end-to-end approach for qos-aware service composition. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 151–160. IEEE.

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

[Samir et al., 2017] Samir, S., Sarhan, A., and Algergawy, A. (2017). Context-based web service discovery framework with qos considerations. In *Research Challenges in Information Science (RCIS), 2017 11th International Conference on*, pages 146–155. IEEE.

[Schumacher et al., 2008] Schumacher, M., Helin, H., and Schuldt, H. (2008). *CASCOM : intelligent service coordination in the semantic web*. Springer Science & Business Media.

[Skoutas et al., 2009] Skoutas, D., Sacharidis, D., Simitsis, A., Kantere, V., and Sellis, T. (2009). Top-k dominant web services under multi-criteria matching. In *Proceedings of the 12th international conference on extending database technology : advances in database technology*, pages 898–909. ACM.

[Skoutas et al., 2007] Skoutas, D., Simitsis, A., and Sellis, T. (2007). A ranking mechanism for semanticweb service discovery. In *Services, 2007 IEEE Congress on*, pages 41–48. IEEE.

[Solanki et al., 2004] Solanki, M., Cau, A., and Zedan, H. (2004). Augmenting semantic web service descriptions with compositional specification. In *Proceedings of the 13th international conference on World Wide Web*, pages 544–552. ACM.

[Sparck Jones, 1972] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1) :11–21.

[Srinivasan et al., 2006] Srinivasan, N., Paolucci, M., and Sycara, K. (2006). Semantic web service discovery in the owl-s ide. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 6, pages 109b–109b. IEEE.

[Tang and Ai, 2010] Tang, M. and Ai, L. (2010). A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE.

[Trummer et al., 2014] Trummer, I., Faltings, B., and Binder, W. (2014). Multi-objective quality-driven service selection—a fully polynomial time approximation scheme. *IEEE Transactions on Software Engineering*, 40(2) :167–191.

[Tsetsos, 2007] Tsetsos, V. (2007). Semantic web service discovery : Methods, algorithms, and tools. In *Semantic web services : theory, tools and applications*, pages 240–280. IGI Global.

[Van Rijsbergen, 1979] Van Rijsbergen, C. (1979). Information retrieval. dept. of computer science, university of glasgow. *URL : citeseer. ist. psu. edu/vanrijsbergen79information. html*, 14.

[Wang and Hou, 2008] Wang, J. and Hou, Y. (2008). Optimal web service selection based on multi-objective genetic algorithm. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, volume 1, pages 553–556. IEEE.

[Wang and He, 2010] Wang, L. and He, Y.-x. (2010). Web service composition based on qos with chaos particle swarm optimization. In *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1–4. IEEE.

[Wang et al., 2010a] Wang, P., Chao, K.-M., and Lo, C.-C. (2010a). On optimal decision for qos-aware composite service selection. *Expert Systems with Applications*, 37(1) :440–449.

[Wang et al., 2010b] Wang, W., Sun, Q., Zhao, X., and Yang, F. (2010b). An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems*, 3(sup01) :18–30.

[Winkler and Thibaudeau, 1991] Winkler, W. E. and Thibaudeau, Y. (1991). An application of the fellegi-sunter model of record linkage to the 1990 us decennial census. *US Bureau of the Census*, pages 1–22.

[Xia et al., 2011] Xia, Y., Chen, P., Bao, L., Wang, M., and Yang, J. (2011). A qos-aware web service selection algorithm based on clustering. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 428–435. IEEE.

[Xia et al., 2008] Xia, Y.-m., Chen, J.-l., and Meng, X.-w. (2008). On the dynamic ant colony algorithm optimization based on multi-pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 630–635. IEEE.

[Xiangbing et al., 2012] Xiangbing, Z., Hongjiang, M., and Fang, M. (2012). An optimal approach to the qos-based wsmo web service composition using genetic algorithm. In *International Conference on Service-Oriented Computing*, pages 127–139. Springer.

[Xu and Reiff-Marganiec, 2008] Xu, J. and Reiff-Marganiec, S. (2008). Towards heuristic web services composition using immune algorithm. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 238–245. IEEE.

[Xu et al., 2011] Xu, L., Shi, L., Wang, R., and Jennings, B. (2011). A multiple criteria service composition selection algorithm supporting time-sensitive rules. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 718–721. IEEE.

[Yan et al., 2006] Yan, G., Jun, N., Bin, Z., Lei, Y., Qiang, G., and Yu, D. (2006). Immune algorithm for selecting optimum services in web services composition. *Wuhan University Journal of Natural Sciences*, 11(1) :221–225.

[Yang, 2009] Yang, X.-S. (2009). Harmony search as a metaheuristic algorithm. In *Music-inspired harmony search algorithm*, pages 1–14. Springer.

[Yang and Deb, 2009] Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE.

[Yang et al., 2010] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A dynamic web services composition algorithm based on the combination of ant colony algorithm

and genetic algorithm. *Journal of Computational Information Systems*, 6(8) :2617–2622.

[Yeniay, 2005] Yeniay, Ö. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1) :45–56.

[Yilmaz and Karagoz, 2014] Yilmaz, A. E. and Karagoz, P. (2014). Improved genetic algorithm based approach for qos aware web service composition. In *Web Services (ICWS), 2014 IEEE International Conference on*, pages 463–470. IEEE.

[Yu and Bouguettaya, 2009] Yu, Q. and Bouguettaya, A. (2009). *Foundations for efficient web service selection*. Springer Science & Business Media.

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1) :6.

[Yu et al., 2013] Yu, Y., Ma, H., and Zhang, M. (2013). An adaptive genetic programming approach to qos-aware web services composition. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1740–1747. IEEE.

[Yunwu, 2009] Yunwu, W. (2009). Application of chaos ant colony algorithm in web service composition based on qos. In *Information Technology and Applications, 2009. IFITA'09. International Forum on*, volume 2, pages 225–227. IEEE.

[Zeng et al., 2003] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421. ACM.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5) :311–327.

[Zhao et al., 2012] Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., and Fan, Y. (2012). An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing*, 12(8) :2208–2216.

[Zhao et al., 2014] Zhao, X., Wen, Z., and Li, X. (2014). Qos-aware web service selection with negative selection algorithm. *Knowledge and Information Systems*, 40(2) :349–373.

[Zhu et al., 2017] Zhu, W., Yin, B., Gong, S., and Cai, K.-Y. (2017). An approach to web services selection for multiple users. *IEEE Access*, 5 :15093–15104.

[Zitzler et al., 2003] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers : An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2) :117–132.