Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire

THESE
Présentée

A L'UNIVERSITE DE TLEMCEN
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de
DOCTORAT
Spécialité : Informatique
Par
Mr Hadjila FethAllah

Composition et interopération des services web sémantiques

Soutenue en fevrier 2014 devant le jury :

-	**M.FEHAM**	**Professeur à l'université de Tlemcen**	**Président**
-	**B.ATMANI**	**Maître de Conférences à l'université d'Oran**	**Examinateur**
-	**L.BABA AHMED**	**Maître de Conférences à l'université d'Oran**	**Examinateur**
-	**F.DIDI**	**Maître de conférences à l'université de Tlemcen**	**Examinateur**
-	**K.FERAOUN**	**Maître de conférences à l'université de SBA**	**Examinateur**
-	**M.A.CHIKH**	**Professeur à l'université de Tlemcen**	**Directeur de thèse**

Année universitaire 2013-2014

Remerciements

Tout d’abord, Je remercie Allah qui m’a donné la force et la vie pour accomplir cette tâche, qui au début paraissait une mission difficile.

Je tiens à remercier en premier lieu, Mr Chikh Mohammed Amine, mon directeur de thèse pour son encouragement, son expérience, ses conseils et sa sympathie qui m’ont permis de mener à bien cette thèse.

Je suis très honoré par la présence de Mr Feham Mohammed, qui a accepté de présider le jury de ma thèse, je suis également très honoré par la présence de Mr Atmani Baghdad, Mr Kamel Feraoun, Mme Baba Ahmed Latifa, et Mme Didi Fedoua qui ont accepté d'être les rapporteurs de cette thèse.

Qu'ils trouvent ici, mes plus vifs remerciements pour l'effort qu'ils ont fait pour lire mon manuscrit et l'intérêt qu'ils ont porté à mon travail.

Je remercie ma famille de m’avoir donné le courage d’accomplir cette thèse.

J’adresse mes sincères remerciements à mes collègues : Mr Belabed Amin et Mr Merzoug Mohammed, Pour leurs conseils et soutien continu durant cette thèse.

Dédicaces

A ma chère famille :

Mon père, ma mère, Ilias, Abdelhafid, Amina I, Sarrah, Bouchra, Amina II, Nadjlaa.

A mes amis :

Merzoug Mohammed, Belabed Amin.

Je vous aime !

Résumé

La technologie des services web se répand de plus en plus sur les réseaux à grande échelle. Son ultime objectif est d'assurer l'interopérabilité des applications distribuées, et de créer des services ayant une valeur ajoutée. La réalisation de ces buts nécessite l'automatisation des phases de découverte, de composition, et de sélection de services web, ces trois étapes constituent les majeures problématiques étudiées dans cette thèse. Notre première contribution consiste à proposer deux algorithmes de découverte basés sur les mesures de similarité sémantique. Notre seconde contribution consiste à composer plusieurs services web pour satisfaire un besoin complexe. Pour cela, nous proposons deux metaheuristiques pour le résoudre : La première emploie l'optimisation à base d'essaims particulaires, et la deuxième utilise les principes d'essaims particulaires avec les boids de reynolds. Pour la troisième problématique, nous avons proposé cinq algorithmes de sélection de services à base de QOS. La première approche est basée sur les algorithmes génétiques, la deuxième est inspirée du comportement d'abeilles artificielles (bees-algorithm), la troisième est basée sur les essaims particulaires, la quatrième emploie la sélection clonale, et la dernière hybride le clustering hiérarchique des services skylines avec le parcours systématique des hiérarchies. Les résultats obtenus en expérimentations sont très satisfaisants et encouragent les futurs travaux dans cette piste de recherche.

Mots clés : Architecture orientée service, Composition de services, Sélection de services, Mesures de similarité sémantique, Web sémantique, ontologies, Qualité de service, Optimisation combinatoire, Metaheuristiques.

Abstract

The web service technology plays an increasingly important role in open and large scale networks. In fact, its ultimate purpose is to ensure the interoperability of distributed applications and to form new value added services. To achieve these aims, we must automate the discovery, the composition and the selection of web services. These phases constitute the major issues of this thesis. Our first contribution consists in proposing two algorithms for discovering web services, these proposals use semantic similarity measures to match the services with the request. Our second contribution consists in composing several services, in order to satisfy complex requirements. To this end, we proposed two metaheuristics based on swarm particles optimization. To solve the QOS aware service selection problem, we have proposed five algorithms, the first approach uses the genetic algorithms, the second one employs the bees algorithm, the third one adopts the swarm particle optimization, the fourth one uses the clonal selection, and the last one hybridizes the skylines hierarchical clustering with the exhaustive search. The experimentations showed high efficiency and effectiveness of the suggested approaches.

Keywords: Service oriented architecture, Service composition, Service selection, Semantic similarity measure, semantic web, ontologies,QOS, Combinatory Optimization, Metaheuristics.

ملخص

تعرف خدمات الواب انتشارا واسعا على مستوى الشبكة العنكبوتية ، تسمح هذه التكنولوجيا بتحقيق التفاعل بين التطبيقات الموزعة, كما تسمح بتشكيل خدمات ذات قيمة مضافة. تعد عمليات انتقاء, بحث و تركيب خدمات الواب من المشاكل الرئيسية لهذا النموذج التطبيقي. لحل هاته المشاكل نقترح عدة مقاربات , تسمح المقاربة الأولى والثانية ببحث الخدمات باستعمال مقاييس التماثل الدلالي, أما عملية التركيب فيتم تحقيقها عن طريق خوارزميات الأسراب,بينما نستعمل عدة خوارزميات من اجل انتقاء الخدمات المركبة (خوارزمية الأسراب، الخوارزميات الوراثية، خوارزمية النحل، خوارزمية الاستنساخ، التقسيم الهرمي التصاعدي) .عدة تجارب تمت مباشرتها من اجل تقييم المقاربات،النتائج المتحصل عليها تعتبر جد مشجعة وبالإضافة إلى ذلك، فإنها تظهر تفوق مقترحاتنا بالمقارنة مع بعض الطرق الأخرى.

الكلمات الرئيسية : نموذج SOA ،تركيب الخدمات، انتقاء الخدمات الواب، الواب الدلالي، مقاييس التماثل الدلالي،انتلوجيا ، جودة الخدمة، التحسين التوفيقي،ميتاؤريستيك.

Table des Matières

Table des figures

Liste des tableaux

Liste des algorithmes

Partie I :

Etat de L’art.

Chapitre 1 :

Introduction générale

Sommaire

I) Contexte

L'objectif principal des entreprises actuelles est de faire face aux changements rapides de l'environnement (i.e. la concurrence, l'adaptation aux nouveaux marchés, l'apprentissage et l'adaptation aux besoins des clients…). Pour cela, elles doivent assurer l'intégration et l'interopérabilité de leurs applications au niveau interne (EAI) ou Enterprise application Intégration et au niveau partenaire (B2B) i.e. Business To Business. Ces applications composites sont généralement appelées processus métiers. Pour gérer et automatiser le cycle de vie de ces processus métiers, les entreprises adoptent l'architecture SOA (service oriented architecture) et en particulier la technologie des services web.

L'architecture orientée service est née pour répondre aux inconvénients des technologies orientées composants, telles que CORBA [OMG, 2008] (Common Object Request Broker Architecture), Java RMI [Downing, 1998] (Remote Method Invocation), DCOM (Distributed Component Object Model). [Horstmann et Kirtland, 1997].

Nous notons que ces technologies (ou middlewares), ont pour objectif de partager et de réutiliser des codes existants, en faisant des appels de procédures distantes. Mais ces tentatives n'ont pas pu passer à l'échelle, et elles n'étaient utilisées que localement (à l'intérieur des entreprises).

Leurs principaux inconvenients étaient, le couplage fort des composants, la complexité d'utilisation et de mise en œuvre de ces middlewares, le non support de certains protocoles du web (http, ftp…), et dans certains cas la non standardisation de ces middlewares [Emmerich, 2002].

Contrairement aux architectures orientées composants, l'architecture SOA [Erl, 2004] est caractérisée par son couplage faible, son indépendance par rapport aux plateformes, et sa grande capacité d'intégration et de réutilisation.

Le couplage faible signifie que la dépendance entre les entités est très réduite, et ceci permet une large capacité de réutilisation et d'intégration de ces composants.

Plusieurs definitions ont été proposées, pour ce paradigme, Selon [Bieberstein et al, 2005] "A service-oriented architecture is a framework for integrating business processes and

supporting IT infrastructure as secure, standardized components – services, that can be reused and combined to address changing business priorities".

Le consortium OASIS[1] propose une autre définition "L'architecture orientée service est un paradigme permettant d'organiser et d'utiliser des savoirs distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs pour produire le résultat désiré avec des pré-conditions et des buts mesurables." (´établie dans le modèle de référence OASIS) [OASIS, 2007 a].

Et selon [Dodani, 2004], "L'architecture orientée service 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 I.1 illustre le modèle général de l'architecture orientée service (SOA).elle montre plusieurs intervenants, et plusieurs types d'activités [Carey, 2008].

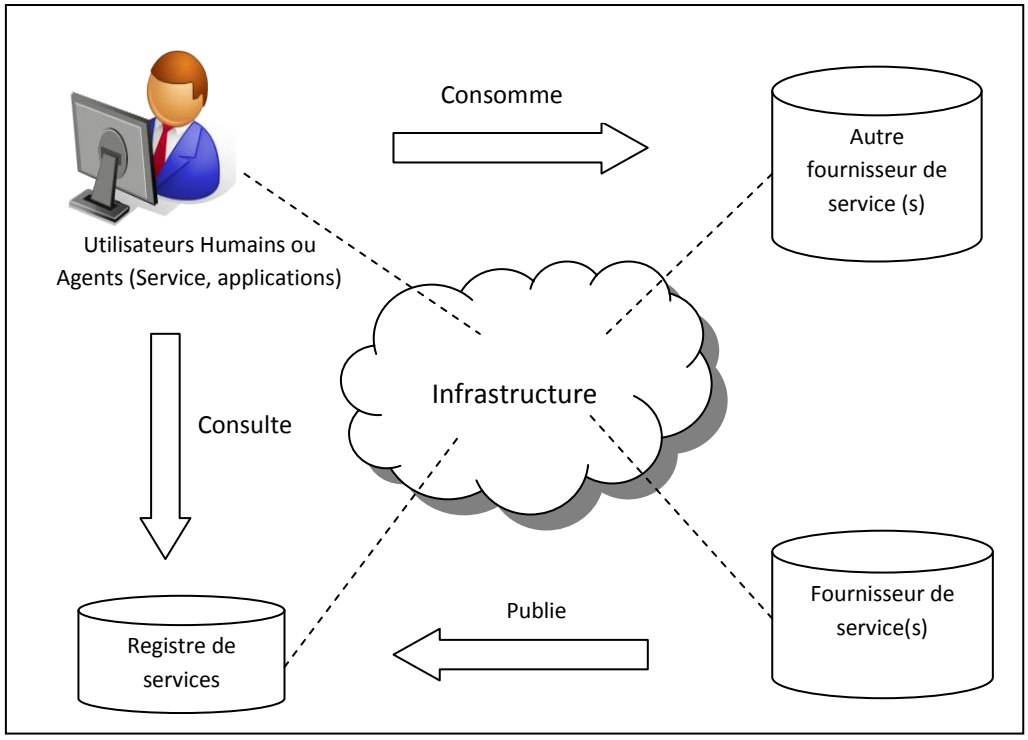


Figure I.1. Le Modèle SOA [Abi Lahoud, 2010]

[1] 1http://www.oasis-open.org/

De façon générale, nous avons : les fournisseurs de services, les registres ou annuaires de services, les consommateurs ou utilisateurs de services. Les fournisseurs créent et décrivent leurs services, et les publient dans un ou plusieurs annuaires. Les annuaires, stockent les descriptions de plusieurs services. Ils sont accessibles par des consommateurs humains ou logiciels, les consommateurs découvrent les services, qui satisfont leurs besoins (cette étape peut être ramifiée en plusieurs sous étapes et en particulier le matching, la composition et la sélection), et les invoquent à travers les interfaces publiées par leurs fournisseurs respectifs.

Les services web constituent une concrétisation des principes du SOA, ils sont munis d'un ensemble de standards qui facilitent leur mise en œuvre. SOAP [Gudin et al, 2003] est un protocole permettant de structurer les messages échangés entre les composants logiciels, WSDL [Chinnici et al, 2007] est une spécification qui définit les interfaces des services. UDDI [Clement et al, 2004] est une spécification de publication et de localisation de services.

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 : la phase de découverte qui cherche les services adéquats à une requête, la phase de composition qui assemble plusieurs services pour satisfaire une requête complexe, et la phase de sélection qui choisit une ou plusieurs compositions en fonctions de critères fonctionnels et non fonctionnels.

II) Objectifs de la Thèse

Pour assurer l'interopérabilité et l'échange inter-entreprises(B2B), nous devons concevoir des mécanismes pour découvrir, composer 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, complique de plus en plus ces tâches.

Plus nous automatisons ces étapes, plus nous gagnons en termes d'interopérabilité [Benatallah et al, 2005 a]. L'objectif de cette thèse est de présenter des approches intelligentes pour automatiser ces trois phases (découverte, composition, et sélection).

Dans ce qui suit, nous définissons de manière précise ces trois problématiques.

Problématique 1 : Découverte de Services Web Sémantiques

Étant donné un besoin d'un client, 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, en plus ils doivent avoir une bonne performance en termes de rappel, de précision et de temps d'exécution.

Problématique 2 : Composition Automatique des Services Web Sémantiques sans Etats

Dans la plupart des cas, un seul service ne peut répondre aux besoins des clients, et par conséquent, nous sommes obligés de composer plusieurs services afin de satisfaire une requête. Si nous considérons l'exemple de planification d'un voyage, un utilisateur doit consommer plusieurs types de services (qui sont offerts par plusieurs organisations). Par exemple, le premier type de service permet de réserver l'hôtel, le deuxième permet la réservation du billet d'avion, le troisième permet le paiement en ligne, et le quatrième permet la consultation de la météo.

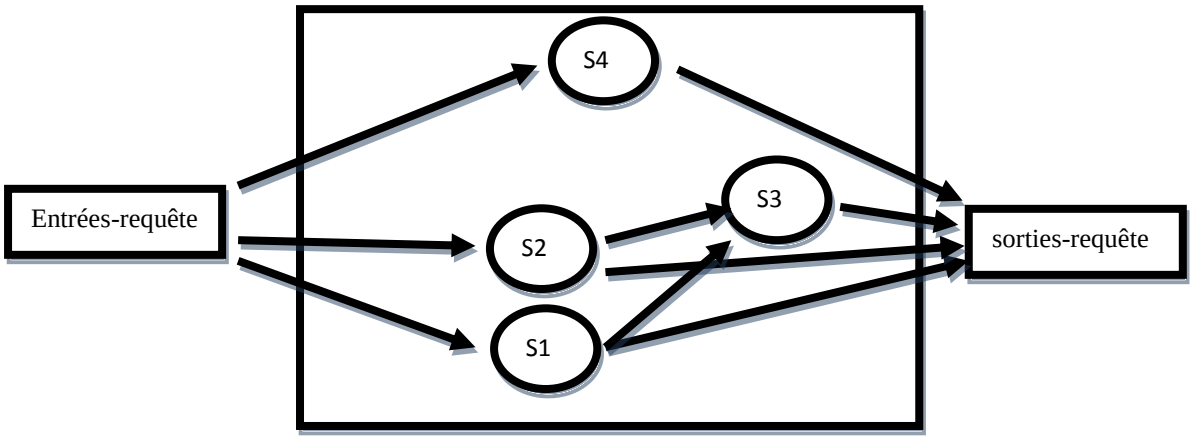


Figure I.2. Exemple de composition.

L'utilisateur fournit un ensemble de concepts en entrée (nom, prénom, date d'arrivée/départ, lieu de départ/d'arrivée) et attend un ensemble de concepts en sorties (des messages de confirmation, des informations de la météo…). Il est clair qu'il n'y a pas de

service individuel qui répond à ce besoin, mais l'assemblage de quatre services différents permet de générer une composition ayant une valeur ajoutée.

Selon [Pathak , 2007] la composition de services web (WSC) est un processus qui permet d'assembler plusieurs services afin de créer une fonctionnalité, ayant une valeur ajoutée. Plusieurs avantages peuvent être tirés en adoptant la composition, en premier lieu nous réduisons le coût et le temps de développement. En second lieu, nous pouvons assurer une bonne adaptation aux changements d'environnement, et des besoins des clients (grâce aux compositions dynamiques), En outre, les services composites peuvent être utilisés comme des services de base dans d'autres compositions.

Le problème de composition peut être étendu, pour prendre en compte la qualité de service, et les contraintes métiers, en plus du besoin fonctionnel.

Il est utile de noter, que cette thèse ne traite pas les notions de contexte, d'aspects de sécurité, et d'aspects transactionnels des services web durant la composition.

Nous notons aussi, qu'il y a deux types de services web: les services avec et sans états, les services avec états stockent un contexte pour chaque requête, ce contexte est modifié par les opérations du service, ce type de services est modélisé généralement avec des formalismes tels que les réseaux de petri et les automates d'états finis ou leurs extensions. Par contre les services sans états ne stockent pas de contexte pour les requêtes, et par conséquent ils traitent les requêtes de manière indépendante. Cette thèse considère uniquement les services web sans états, et le flux de contrôle séquentiel. Le chapitre 4 présente le problème avec plus de détail.

Problématique 3 : Sélection de Services Web Composés à Base de Qualité de Service.

Nous notons que cette problématique, constitue un raffinement de la composition de services web. Plus précisément nous considérons, dans ce cas un workflow abstrait, cad une composition dont les tâches ne sont pas encore affectées aux services, et nous voulons la concrétiser. En d'autres termes nous cherchons des instances concrètes qui peuvent remplacer les nœuds abstraits, de telle sorte que la qualité de service du résultat est optimisée, et en plus les contraintes globales des utilisateurs sont satisfaites, ces contraintes portent généralement sur les facteurs de QOS, tels que le coût, la disponibilité…

Cette problématique est présentée avec détail dans le chapitre 5.

Il est clair que cette sélection est une version multidimensionnelle du problème du sac à dos [Alrifai et risse, 2009], et 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 réelles des utilisateurs.

III) Contributions de la Thèse

Les principales contributions de cette thèse sont résumées comme suit :

- Proposition de deux approches de découverte sémantique de services web. Ces approches se basent sur les mesures de similarité sémantique, elles sont évaluées à l'aide des critères de rappel et de précision.
    - La première contribution, adopte l'indexation conceptuelle comme moyen de représentation de services web et la mesure cosinus (mesure orientée espaces vectoriels) comme mécanisme de comparaison et de classement des résultats.
    - La deuxième adopte une les mesures de similarité basés sur les arcs, et en particulier, celle de [Wu et palmer, 1994], cette dernière permet la comparaison des services web avec la requête, et classe les résultats selon leur degré de proximité sémantique.
- Proposition de deux approches de composition sémantique de services web sémantiques.
    - La première approche est basée sur les essaims particulaires [Eberhart et Kennedy, 1995], en particulier nous discrétisons l'algorithme, en proposant deux mouvements : les premier est un mouvement aléatoire, et le deuxième est social (un rapprochement vers la meilleure position du groupe).
    - La deuxième approche, hybride les essaims particulaires et les boids de reynolds [Reynolds, 1987]. L'algorithme baptisé BPSO (Boid Particle Swarm Optimization) permet de créer et d'optimiser les compositions de services sans états. nous notons que cet algorithme constitue une amélioration du modèle PSO (Particle Swarm Optimization).
- Proposition de cinq approches pour la sélection de services web composites à base QOS.

- La première approche utilise les algorithmes génétiques [Holland, 1962], pour faire une optimisation mono-objective et globale (dans l'espace de recherche).
- La deuxième approche utilise l'optimisation par essaim particulaire (PSO) [Eberhart et kennedy, 1995] pour faire une optimisation mono-objective et globale.
- La troisième approche utilise la sélection clonale [De castro et Von zuben, 2000] pour faire une optimisation mono-objective et globale.
- La quatrième approche utilise l'algorithme à base d'abeilles [Pham et al, 2006] pour faire une optimisation mono-objective et globale.
- La cinquième contribution est une version hybride, elle effectue une recherche (ou filtrage) multi objective dans chaque classe, après elle groupe hiérarchiquement les services de chaque classe en utilisant l'algorithme hiérarchique de Ward [Ward, 1963]. Chaque groupe possède un service représentant. En dernier lieu nous effectuons une recherche mono-objective (par niveaux) sur les représentants des groupes.

IV) Organisation de la Thèse

Cette thèse est constituée de deux parties principales : la première est intitulée « Etat de l'art », elle introduit le contexte dans lequel se place cette thèse et montre un état de l'art des problématiques traitées. La seconde partie du document est intitulée « Contributions », elle présente les approches proposées pour la découverte, la composition et la sélection de services.

Première Partie : « Etat de l'Art»

Cette partie est organisée en 04 chapitres :

- Le chapitre 2 « la technologie des services web » introduit les principes du paradigme SOA, ainsi que les protocoles, les langages et les modèles qui sont en relation, avec la technologie des services web traditionnels et sémantiques.
- Le chapitre 3 « la découverte sémantique de service » définit formellement la problématique de recherche sémantique de services, présente l'état de l'art ainsi que les avantages et les inconvénients de chaque catégorie d'approches.

- Le chapitre 4 « La composition de services » décrit en détail le problème de la composition de services, ainsi que les différents types d'approches de composition qui existent dans la littérature, nous montrons aussi les avantages et les inconvénients de chaque classe d'approches.
- Le chapitre 5 « la sélection de services web composés à base QOS» introduit formellement la problématique, nous montrons aussi un survol sur les approches proposées dans la littérature.

Deuxième Partie : « Contributions »

Cette partie est organisée en deux chapitres :

- Le chapitre 6 « approches proposées» Dans ce chapitre, nous présentons formellement nos besoins et nos objectifs, nous décrivons nos algorithmes de découverte, de composition et de sélection de services. Nous démarquons aussi nos approches, par rapport aux méthodes d'état de l'art.
- Le chapitre 7 « implémentation et expérimentations » présente la mise en œuvre de nos approches en décrivant notre environnement de composition de services, nommé EDCSS « Environnement de Découverte, de Composition et de Sélection de Services». Nous montrons également des expérimentations de performances de ces algorithmes.

Et enfin, nous terminons cette thèse avec un dernier chapitre.

- Le chapitre 8 « Conclusion et Perspectives » 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, d'identifier les questions ouvertes et les perspectives de ce travail.

Partie I :

Etat de l'Art.

Chapitre 2 :

La technologie des services web

Sommaire

I) Introduction

Plusieurs paradigmes de développement de logiciel ont été proposés pour satisfaire le besoin de réutilisation, le paradigme SOA est l'un des modèles les plus prometteurs, en effet il adopte les avantages des autres paradigmes (l'orienté objet et l'orienté composant), tels que la séparation interface/implémentation (l'encapsulation), la modularité, et ajoute d'autres concepts, tels que le couplage faible et l'interopérabilité[2].

Le paradigme SOA possède plusieurs implémentations (OSGI [OSGI alliance, 2005], services Web [Curbera et al, 2002] …).Dans cette étude, nous intéressons uniquement aux services Web, qui représentent la concrétisation la plus répandue.

En plus du respect des principes du SOA, les services Web présentent plusieurs avantages tels que l'adaptation aux caractéristiques du web, et le bénéfice d'un large support de la part des industriels.

Dans la suite de ce chapitre, nous présenterons tout d'abord le concept de service web ainsi que son cycle de vie, la pile des fonctionnalités sous-jacentes, et les standards associés, ensuite nous nous introduisons la notion de services web sémantiques qui étendent les services web traditionnels, nous présentons les principaux modèles sémantiques pour gérer leur cycle de vie.

II) Services Web

II.1) Definitions

Selon IBM[3]

"Web services are a new breed of web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be anything from simple requests to complicated business processes"

Cette définition affirme que les services web sont accessibles par d'autres à travers le web, en utilisant des protocoles et des formats standards.

Selon le W3C[4]

[2] L'interopérabilité est le fait que plusieurs systèmes, qu'ils soient identiques ou radicalement différents, puissant communiquer sans ambiguïté et opérer ensemble. (Source: Wikipedia).

[3] IBM Web services tutorial. Online: http://www-106.ibm.com/developerworks/webservices/.

[4] www.w3c.org

A Web service is a software system identified by a URI and designed to support interoperable machine-to-machine interaction over a network. It has an interface defined and described in a machine-processable format (wsdl) . Its definition can be discovered by other software systems. Other systems may then interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Cette définition cite les caractéristiques les plus importantes d'un service web à savoir l'identifiant (uri), l'interface (wsdl), et l'indépendance par rapport aux plateformes.

La figure II.1 schématise les facettes générales d'un service web, en premier lieu un service web est décrit par une interface XML nommée WSDL, il peut échanger des documents XML avec d'autres services à l'aide du protocole SOAP, il peut être recherché dans un annuaire tel que l'UDDI

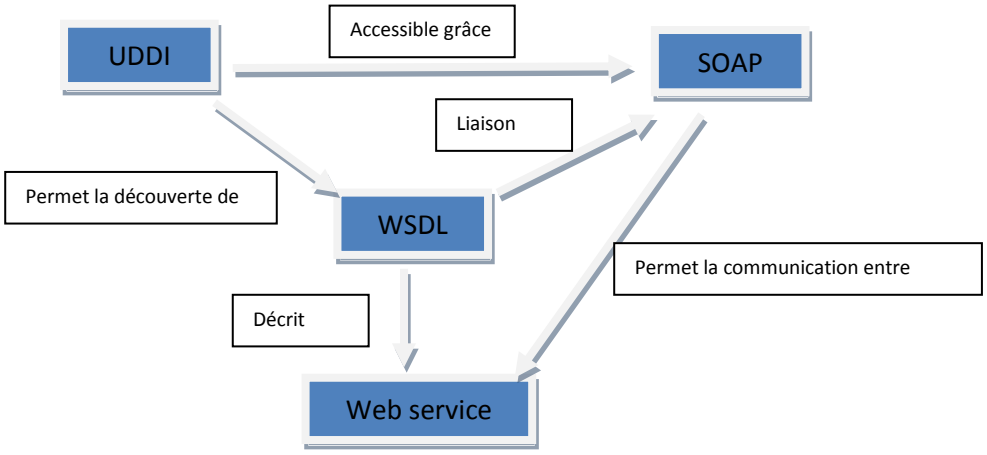


Figure II.1. Les différentes relations entre les spécifications des services web [Erl, 2004]

II.2) Pile Protocolaire

L'architecture des services web est décrite à l'aide d'une pile de langages, de protocoles et de modèles, nous distinguons des couches horizontales et des couches verticales.

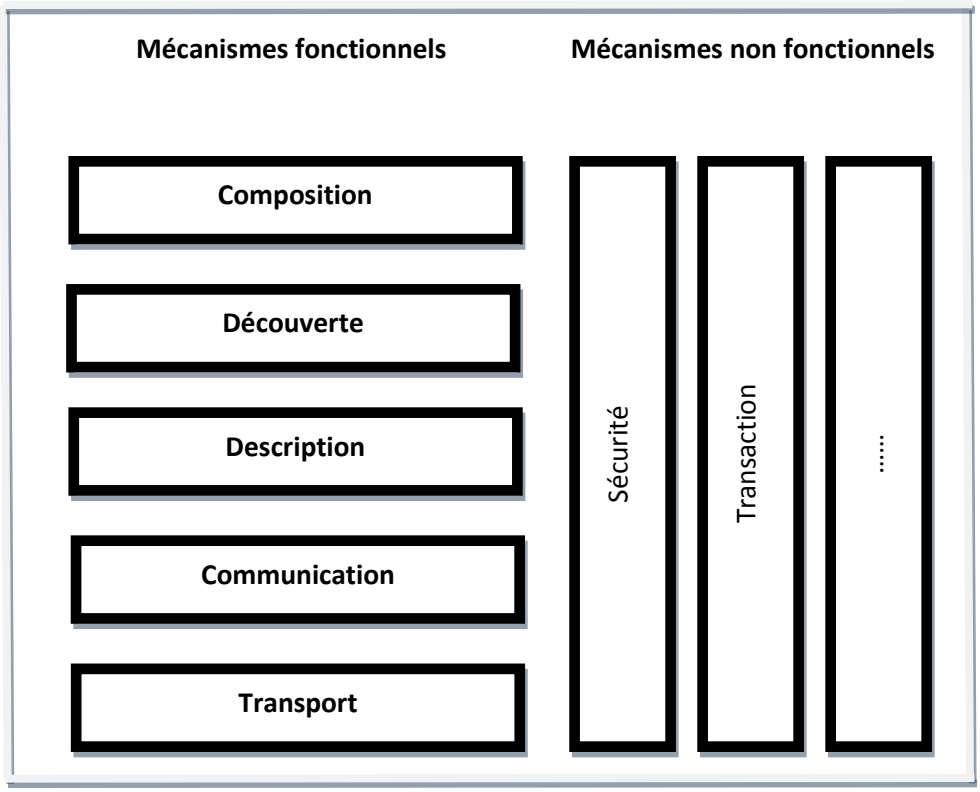


Figure II.2. La pile des services web [Pathak, 2007].

II.2.1) Couche de Transport

Cette couche s'intéresse aux protocoles de transport de bas niveau, ces derniers vont transporter les requêtes et les réponses échangées entre services.

Le protocole le plus utilisé (et recommandé par le consortium « Web Service Interoperability») est l'http, mais d'autres implémentations peuvent utiliser un autre ensemble de protocoles tels que : FTP, SMTP, JMS (java messagerie services)…

II.2.2) Couche de Communication (Messagerie)

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients, elle caractérise aussi le mode d'échange (s'il est bloquant ou non). Le protocole soap est adopté comme un standard pour la messagerie entre les services web.

- **SOAP**

SOAP (Simple Object Access Protocol) [Gudin et al, 2003] est un protocole d'échange de message indépendant des plateformes, c'est un produit de Microsoft et IBM. Sa première version a été acceptée par le W3C (Word Wide Web Consortium) en 2000.

Il est constitué de deux parties : une enveloppe XML, et un entête d'un protocole de transport. La spécification du protocole SOAP ne donne aucune indication sur le mécanisme de transport du message.

SOAP fait une séparation entre le message (i.e. le document XML) et le moyen de transport utilisé. Actuellement, SOAP utilise les protocoles HTTP, SMT ou pour assurer le transport. L'enveloppe XML « envelope » contient à son tour deux sous éléments : un entête « «header » et un corps appelé body

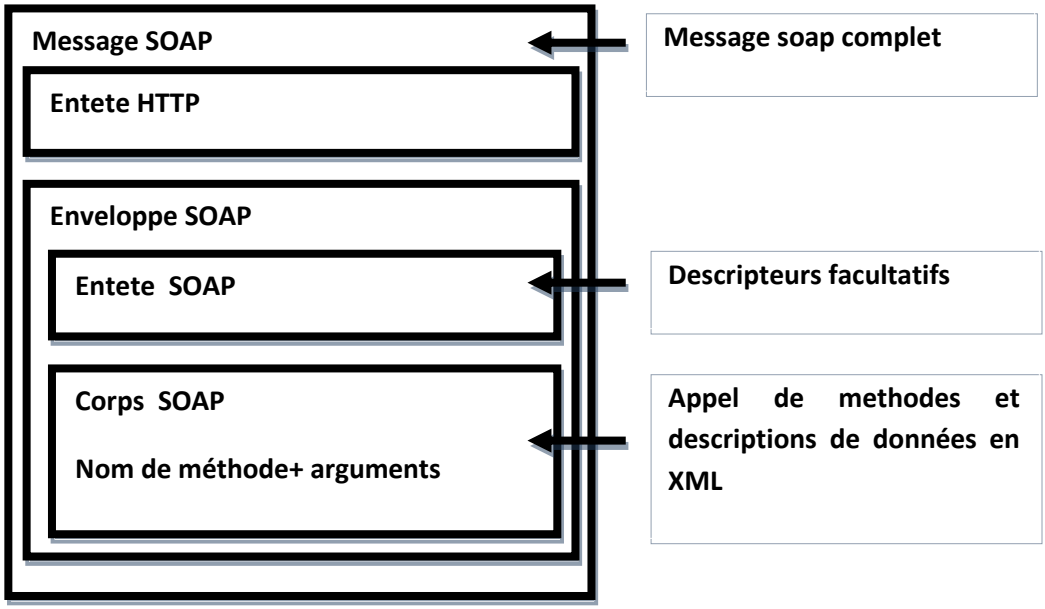


Figure. II.3 Entete du protocole SOAP

L'entête est facultatif, il peut contenir des informations de sécurité (telles que les signatures électroniques), des informations transactionnelles, des informations de traçabilités…

La partie body est obligatoire elle contient les éléments suivants:

• soit le nom de méthode, avec les données correspondantes, ou un simple document XML, pour le cas d'une requête.

• les valeurs de retour ou un message d'erreur pour le cas d'une réponse.

II.2.3) Couche de Description de Service

Beaucoup d'efforts ont été proposés pour décrire les capacités d'un service web, nous distinguons les descriptions syntaxiques et les descriptions sémantiques, chacune d'elles offre des facettes décrivant soit la structure et/ou le comportement du composant.

Nous commençons par la description de l'interface WSDL[Chinnici et al, 2007] qui peut être qualifiée comme étant syntaxique et structurelle. WSDL est devenu le standard incontournable pour l'invocation des services web.

II.2.3.1) Description Syntaxique

II.2.3.1.a) Description Structurelle (l'Interface WSDL)

Le « Web Service Description Language » (WSDL) [Chinnici et al, 2007] est un standard du W3C qui permet de définir la structure abstraite et concrète d'un service. C'est un document XML qui décrit la signature des opérations offertes par le service (nom d'opérations, noms et types des paramètres d'entrées/sorties), il définit aussi des liaisons concrètes pour ces opérations telles que le protocole de transport, l'URI du service, le style du service, et les règles d'encodage employées pour les paramètres d'entrées/sorties.

Nous notons que cette interface décrit juste la structure du service (la partie quoi) et non pas le comportement (la partie comment).Le document WSDL comporte 05 sortes d'éléments XML : <types>, <message>, <portType>, <binding> et <service>. Voir La figure II.4

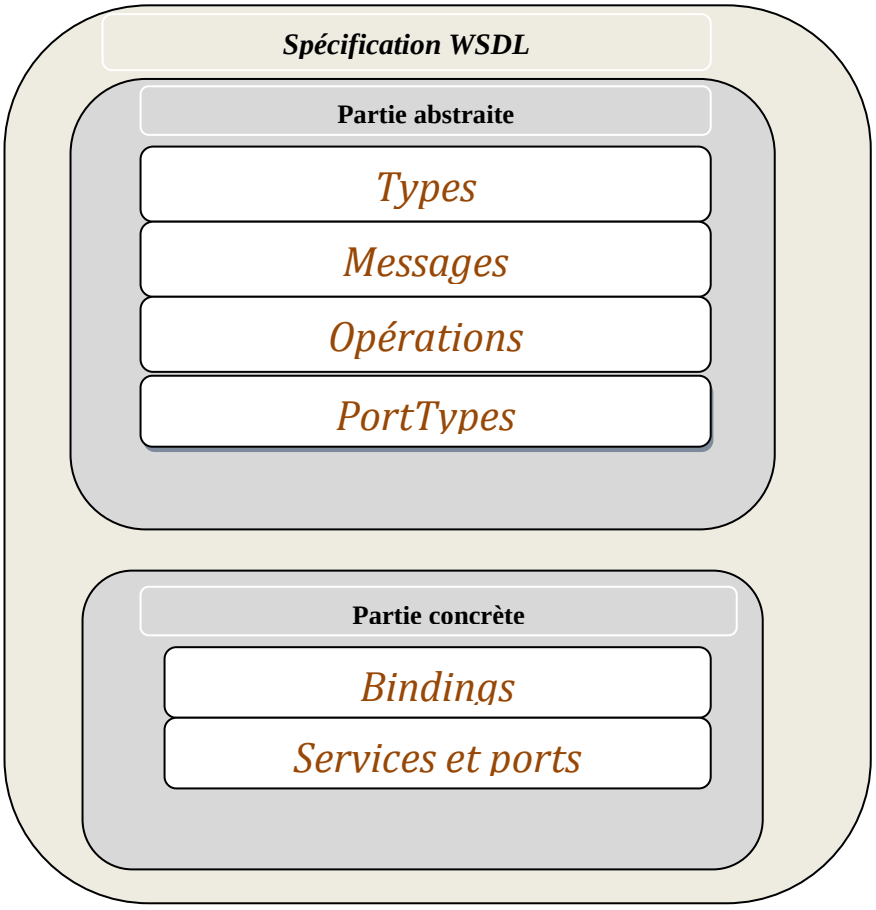


Figure II.4. Les éléments de l'interface WSDL.

<types> : il définit les types des messages échangés par le service, en utilisant XML schéma ces types peuvent être simples (entiers, réels, chaines de caractères) ou complexes (tableaux, enregistrements,….).

<message> : cet élément définit les messages échangés par le service (entrées ou sorties). Il est composé d'un ou plusieurs éléments nommés <part>. Un <part> est associé à une donnée décrite dans <types>.

<portType> : il représente une collection d'opérations. Une <operation> est une action abstraite accomplie par le service. Elle contient éventuellement un sous élément <input> décrivant le message d'entrée, et un sous élément <output> décrivant le message de sortie.

<binding> : cet élément associe un portType avec des informations concrètes telles que le protocole de transport (http, ftp…), les règles d'encodage de données, et le style du service (RPC ou DOC), le style RPC impose un format précis aux messages SOAP (i.e. le nom de la méthode et les arguments pour la requête, et la valeur de retour pour la réponse).ce style est adapté aux échanges synchrones.

Le style DOC autorise n'importe quel document XML bien formé, comme requête ou réponse, ce style est adapté aux échanges asynchrones.

<service>, cet élément est constitué d'un ensemble de <port>.

Un <port> est une association entre un <binding> et un point d'accès i.e. l'url qui indique l'adresse du service web. Nous pouvons avoir plusieurs ports par services, (un même binding et des URLs différents, ou des « binding» différents et éventuellement des URLs différents).

La version 1.0 de l'interface WSDL a été créée en 2000 par IBM, Microsoft et Ariba , ensuite le consortium W3C a publié la version WSDL 1.1 en mars 2001 comme une note [Christensen et al,2001].

La spécification WSDL 1.2 est une simplification de la version WSDL1.1, mais elle n'est pas adoptée comme une norme par le W3C. En juin 2007, la version WSDL 2.0 a été créée comme une recommandation W3C (i.e. norme). WSDL 2.0 a apporté quelques changements à la version WSDL 1.2 , ils sont résumés comme suit :

<portType> est devenu <interface>

<port> est remplacé par <endpoints>

<message> est supprimé et l'élément <xs :element> est utilisé pour spécifier les données échangées.

II.2.3.1.b) Description Comportementale

L'interface WSDL, n'est pas suffisante pour décrire les interactions complexes de certains services web. Pour cela la communauté de recherche a développé plusieurs langages pour décrire le comportement d'un ou plusieurs services en interaction.

De façon générale la facette comportementale d'un service consiste à décrire l'ordre d'invocation des opérations d'un service [Abi Lahoud, 2010]. Un service peut avoir plusieurs descriptions comportementales et chaque description est une vue sur un comportement possible du service dans une communication donnée. Nous distinguons deux types de modèles comportementaux : l'orchestration et la chorégraphie.

Le Modèle d'Orchestration

L'orchestration est une approche centralisée pour commander l'exécution d'une composition (voir la figure II.5), dans ce type toutes les communications sont routées par un processus central appelé moteur d'orchestration ou moteur de workflow. Les services composants ne doivent pas être modifiés pour exécuter le workflow. Le formalisme BPEL [Andrews et al, 2003] est devenu un standard de facto pour l'orchestration des services web. [Papazoglou et Dubray, 2004].

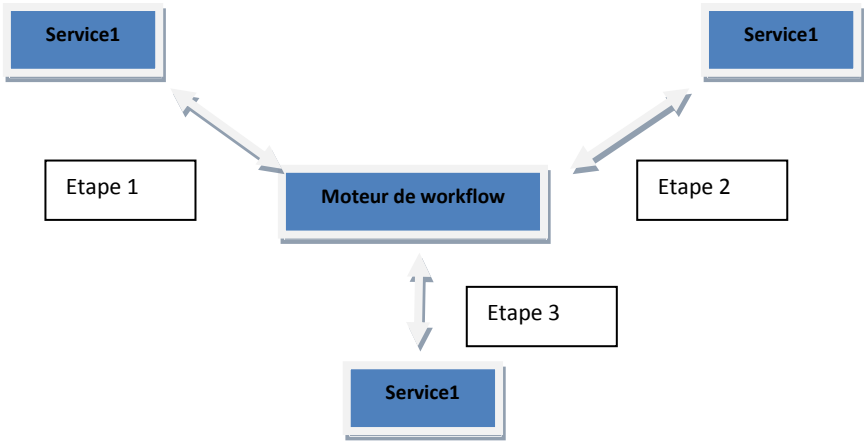


Figure II.5. Le modèle d'orchestration

Le Modèle de Chorégraphie

C'est une approche décentralisée pour l'exécution des compositions (voir la figure II.6), nous n'avons pas de chef d'orchestre central, par contre, chaque partie de la composition

possède un contrat ou des accords avec le reste des services qui dicte la manière de collaborer. [peltz, 2003]

La spécification WSDCL [Kavantzas et al, 2005] (Web Service Description Choregraphy Language) est une recommandation W3C pour les chorégraphies, nous notons que ce type de compositions n'est pas largement utilisé, et il y a peu d'implémentations de la spécification WSDCL. D'autres langages tels que : WSCI [W3C, 2002 a], WSCL[W3C, 2002 b] sont aussi offerts pour décrire les chorégraphies.

La figure suivante montre un exemple de chorégraphie.

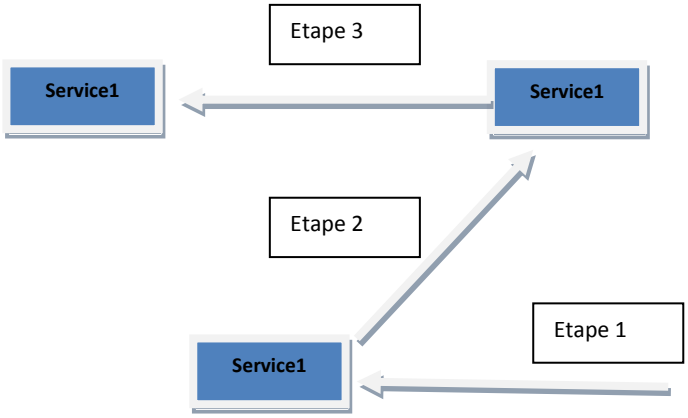


Figure II.6. Le modèle de chorégraphie

II.2.3.2) Description Sémantique

L'interface WSDL, ainsi que les langages de chorégraphie/orchestration cités précédemment ne sont pas suffisants pour décrire de manière formelle les capacités d'un service.

Par exemple, il est possible de trouver deux opérations (i.e. deux interfaces WSDL) qui ont les mêmes paramètres d'entrées/sorties (les noms et les types), mais ils font des fonctions différentes, nous pouvons trouver aussi des opérations ayants des paramètres différents, mais qui font des fonctions similaires.

Pour lever ces ambigüités la communauté de recherche a introduit plusieurs spécifications sémantiques qui fournissent des descriptions formelles des capacités (entrées, sorties, pré-conditions, effets…).

Nous notons 04 spécifications WSDL-S[Akkiraju et al, 2005] , SAWDL, OWLS [Martin et al, 2004], WSMO. Les deux premières sont des annotations sémantiques du standard

WSDL, alors que les deux dernières sont des ontologies de services (elles ne sont pas directement liées à WSDL).

- **WSDL-S**

Le but de WSDL-S [Akkiraju et al, 2005] (prédécesseur de SAWSDL) est d'annoter l'interface WSDL, Son méta-modèle [Miller et al, 2004] permet l'ajout de trois éléments <category>, <precondition>, <effect> et deux attributs modelReference et schemaMapping. Les éléments ajoutés enrichissent la sémantique de WSDL (i.e. les pré-conditions et les effets des opérations).

De façon similaire les attributs modelReference et schemaMapping pointent des concepts ontologiques.

L'élément <category> est un sous- élément de <portType>. Il donne la classe d'un service (la fonctionnalité), en utilisant une certaine taxonomie, il est introduit lors de la publication du service dans l'annuaire.

L'élément <pré-condition>, sous-élément de <operation>, il précise les pré-conditions devant être vérifiés avant l'exécution de l'opération.

L'élément <effect>, sous-élément de <operation>, il précise les changements qui résultent de l'exécution de l'opération.

L'attribut modelReference est associé à un <xs :element> (qui fait partie d'une grammaire XML) et aux éléments <operation>, <precondition> et <effect>. Il pointe sur le concept de l'ontologie de référence.

L'attribut schemaMapping est associé à un <xs :element>. Il fournit les correspondances entre la grammaire XML et les ontologies utilisées.

- **OWLS**

OWL-S : Ontology Web Language for Services OWL-S [Martin et al, 2004](successeur de DAMLS [Ankolekar et al, 2002]) est une ontologie de haut niveau pour la description des services web sémantique

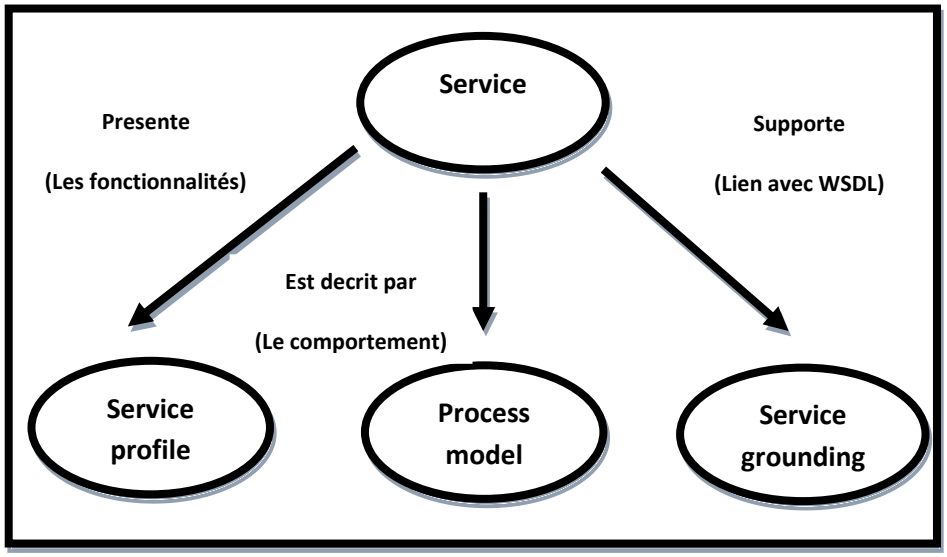


Figure II.7. Les concepts de base d'une ontologie OWL-S

Son objectif est d'assurer l'automatisation du cycle de vie du service, i.e. les phases de découverte, de composition, de sélection, d'invocation, et de surveillance [Ankolekar et al, 2002].

Comme montré dans la figure II.7 , owls est constitué de 03 concepts :

Le service profile, le process model et le service grounding. Ces trois super-concepts permettent de décrire l'aspect fonctionnel et non fonctionnels du service, de décrire la composition de services (dans le cas d'une interaction complexe), et de relier cette description avec l'interface WSDL.

Le « service profile » a pour objectif d'automatiser la découverte et la sélection, il fournit une description de haut niveau d'un service et de son fournisseur. Le service profile contient trois types d'informations :

a) une description du service et de son fournisseur (informations de contact),

b) l'aspect fonctionnel du service et en particulier : les entrées, les sorties, les pré-conditions, les post-conditions des opérations.

c) des propriétés non fonctionnelles comme la réputation, la catégorie du service, et la qualité de service de façon générale.

Le « process model » définit le flux de contrôle et données d'une composition de services.

Nous distinguons trois types de processus : atomiques, composites et simples :

Processus atomique. C'est un sous-concept de process model. Il décrit une opération primitive (non décomposable) et pouvant être invoquée à distance.

Processus composite : il décrit des opérations complexes (constituées de services atomiques et composites).

Processus simples : c'est une représentation abstraite d'un service atomique ou composite.

Le « service grounding » : décrit les moyens d'accès au service en spécifiant le protocole de communication, le format des messages, l'encodage des paramètres. Il connecte le « process model » avec la description WSDL sous-jacente.

- **WSMO**

WSMO : Web Service Modeling Ontology est une meta-ontologie introduite par [Roman et al, 2005], elle est basée sur le Web Service Modeling Framework WSMF proposé par Fensel et Bussler [Fensel et Bussler, 2002]. Le but de WSMO est d'automatiser le cycle de vie des services web (découverte, composition, sélection, substitution…).Il est constitué de 04 éléments les services web, les buts (goal), les ontologies, et les médiateurs, chacun d'eux est décrit avec un formalisme basé sur le langage WSML[5] [Bruijn et al, 2005].. Ce modèle permet de réaliser un couplage faible entre les services web en utilisant un ensemble de médiateurs. Ces derniers assurent les tâches d'intégration d'ontologies, de découverte des services, de composition…Le noyau de l'ontologie WSMO est montré dans la figure II.8.

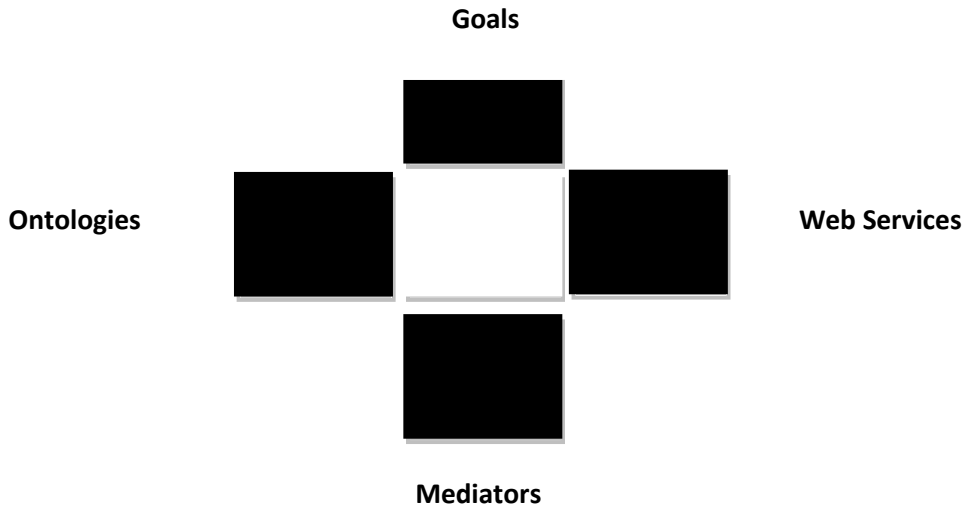


Figure II.8. Les éléments fondamentaux de WSMO

[5] Web service modeling language

- **SAWSDL**

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

En particulier elle annote les éléments : opérations, input, output, type schemas, et interfaces. Selon [Farrell et Lausen, 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 schema type ou à un élément un concept dans une ontologie partagée.

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

[Lecue, 2008] Présente un tableau comparatif des modèles WSMO, OWLS et SAWSDL, de façon générale, nous pouvons constater les points suivants :

- Wsmo et owls sont des ontologies de haut niveau alors que sawsdl est une extension du format wsdl.
- Wsmo et owls sont liés à des formalismes particuliers tels que wsml ou owl, alors que sawsdl est indépendant de tout langage de définition d'ontologies.

	Ontologie de services				Extension des standards des services web	
	OWLS		WSMO		Sa-wsdl	
fonctionnalité	Service profile	IOPE Categorie de services Propriétés non fonctionnelles	Service capability	IOPE Propriétés non fonctionnelles	Annotations semantiques internes et externes	Model rfefernce, schéma mapping
processus	Process model	Processus atomiques, simples et composites	Chorégraphie, orchestration	Transitions avec gardes	×	
invocation	Service grounding	Liaison avec wsdl, soap	Service grounding	Liaison avec wsdl ,soap	Liaison avec wsdl ,soap	
Environnement d'exécution	×		Wsmx		Meteor-s	

Table II.1. Comparaison des principaux standards sémantiques des services web

II.2.4) Couche de Découverte

UDDI (Universal Description, Discovery and Integration) [Clement et al, 2004] (pour la version 3.02) est une spécification d'annuaires de services Web : cette norme W3C propose un ensemble de structures à publier par les fournisseurs de services. Ces structures sont formalisées en XML, elles proposent 03 types d'informations (voir la figure II.9) :

Les pages blanches qui décrivent les informations de contacts sur les entreprises

Les pages jaunes qui décrivent des informations de classification de services

Les pages verte qui donnent des informations techniques (telles que l'accès) des services.

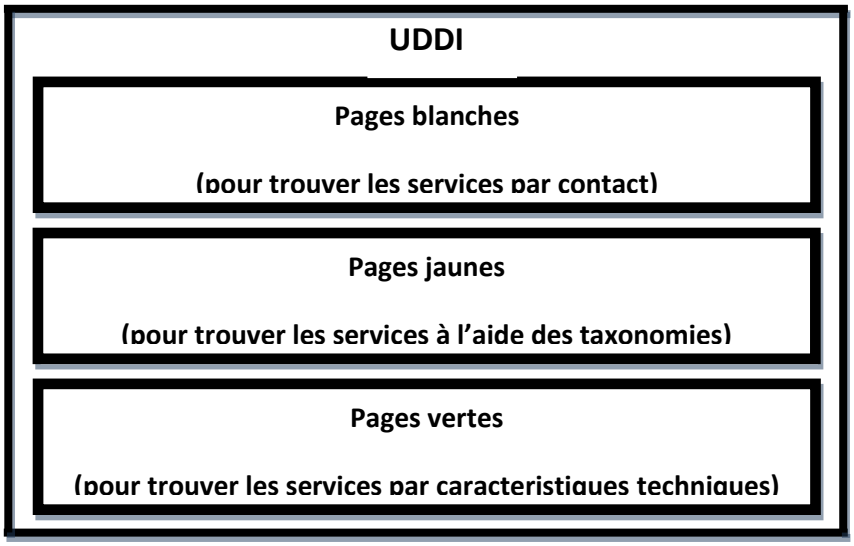


Figure II.9. Le contenu de l'annuaire UDDI

La norme UDDI offre aussi une API aux applications clientes, pour rechercher des services et/ou ses fournisseurs, ajouter ou modifier des services ou des entreprises. Nous notons que l'accès à l' UDDI est réalisé par des services Web. Nous distinguons des d'annuaires UDDI publiques et privés l'annuaire UDDI publique est implémenté sous forme d'un réseau de nœuds UDDI, ces nœuds sont synchronisés, chacun d'eux est possédé par une entreprise donnée (telles que IBM, Microsoft, NTT, et SAP)

La publication d'un service chez une entreprise Propage automatiquement ses informations (pages blanches jaunes et vertes) aux différents nœuds UDDI. L'accès à l'ensemble des informations des registres peut se faire à n'importe quel nœud UDDI. Ce type d'annuaire est gratuit. L'annuaire UDDI privé n'est accessible que sur l'intranet d'une entreprise particulière (ou d'un groupe d'entreprises).Les structures de données d'un UDDI sont décrites comme suit (voir la figure II.10) :

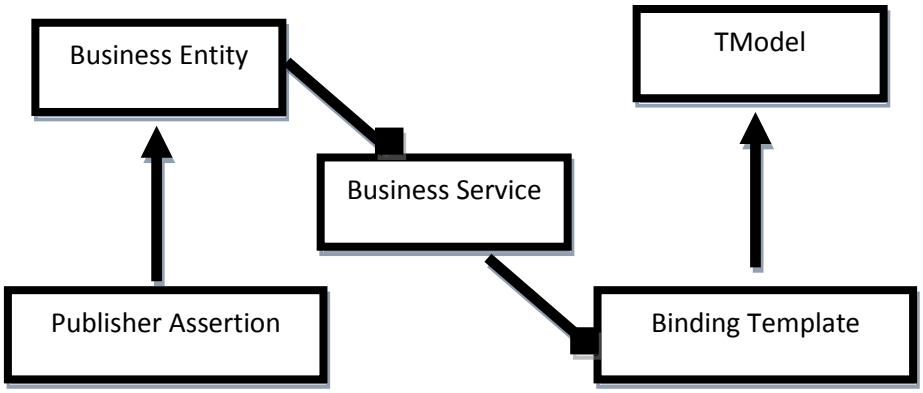


Figure II.10. Les structures de l'annuaire UDDI.

La partie « pages blanches » est constituée de deux éléments <businessEntity> et <publisherAssertion>.

<businessEntity> donne des informations de contact sur l'organisation fournissant les services (les noms des dirigeants, les emails, les numéros de téléphones, le site web…) .

<publisherAssertion> spécifie les liens d'affiliations entre deux entreprises (mère et fille). Lorsque deux éléments <businessEntity> référencent la même <publisherAssertion>, nous parlons de relation d'affiliation entre ces <businessEntity>.

La partie « pages jaunes » est constituée des éléments <businessService>.

<businessService> décrit un ensemble de services fournis par une organisation.(le nom du service, la description textuelle, les informations de classification). Un <businessService> est un sous élément de <businessEntity>.

La partie « pages vertes » est constituée des éléments <bindingTemplate> et <tModel>.

<bindingTemplate> spécifie des informations techniques, telles que l'URI du service. Un <bindingTemplate> est un sous élément de <businessService>.

<tModel> définit des structures d'informations techniques telles que l'interface WSDL, les taxonomies industrielles, les ontologies…, Un élément <tModel> peut être réutilisé par plusieurs éléments <bindingTemplate>.

II.2.5) Couche de Composition

Plusieurs langages de compositions de services ont été proposés dans la littérature, le langage BPEL [Andrews et al, 2003] est l'un des plus importants formalismes.

- **BPEL**

BPEL4WS ou BPEL [Andrews et al, 2003], [OASIS, 2007 b] est un langage qui s'articule sur deux formalismes prédécesseurs: le Web Services Flow Language (WSFL) [Leymann, 2001] d' IBM et le Web Services for Business Process Design (XLANG) de Microsoft. WS-BPEL 2.0 est devenu un standard OASIS en 2007. BPEL permet de modéliser les processus métiers en termes d'orchestration, en décrivant le flux de données (les variables) et le flux de contrôle (les activités simples et composés, les partenaires…), son but est de créer une fonctionnalité complexe qui réutilise les services existants. Il permet aussi de donner une vue centralisée de l'exécution de la composition. Nous distinguons de types de processus BPEL : abstrait et exécutable.

Le premier décrit les interactions publiques de messages entre les partenaires, sans, obligation de montrer le comportement interne. Par opposition le second montre l'ordre d'exécution des activités, les partenaires invoqués, les messages échangés et les mécanismes de gestion des erreurs potentielles. BPEL réutilise l'interface WSDL selon les méthodes suivantes :

• tout processus BPEL est considéré comme un service Web. Par conséquent, l'interface publique du processus (le type et le nom de ses paramètres d'entrée/sortie) est décrite avec WSDL.

• les types de données WSDL sont utilisés par un processus BPEL pour décrire les informations qui transitent entre les différentes activités du workflow.

• WSDL peut être utilisé pour référencer les différents services Web requis par le processus.

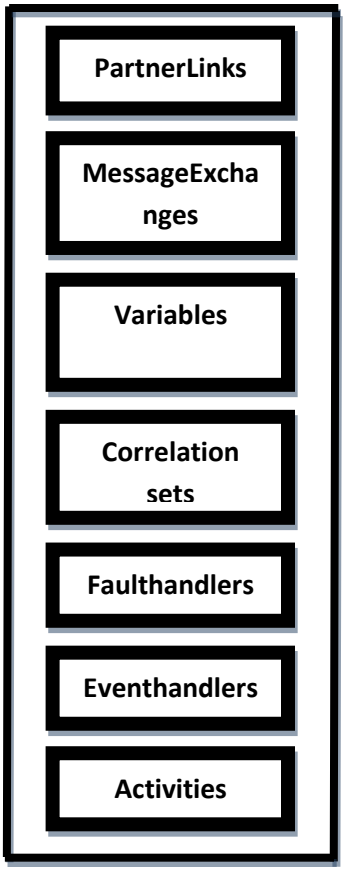


Figure II.11. Structure d'un document BPEL

Comme montré dans la figure II.11 La spécification BPEL offre plusieurs éléments, ils sont décrits comme suit :

PartnerLinks : ils contiennent un ou plusieurs éléments PartnerLink, ce dernier représente un échange de message bilatéral, plus précisément il spécifie le partenaire qui travaille avec la composition.

Chaque partenaire est décrit par son nom (en tant que service indépendant), et son rôle dans le processus.

Nous pouvons avoir un rôle, pour chaque service qui appartient à la composition et qui exécute une activité.

Variables : ils servent à stocker des données ou des messages échangés afin de les réutiliser ultérieurement.

CorrelationSets : une orchestration peut être exécutée plusieurs fois. Chaque exécution est nommée instance. Pour d'acheminer les données à une instance particulière, nous utilisons l'élément <correlationSets>, il permet de spécifier des identifiants aux messages échangés.

L'élément <faultHandlers> permet de gérer les exceptions au niveau du <catch>.

L'élément <eventHandlers> représente les événements pouvant survenir au cours de l'exécution et leurs associe des traitements.

Les activités de base

Receive : c'est une attente bloquante d'un message entrant

Reply : retourne un message suite à une réception d'un autre message (à l'aide de la primitive receive).

La combinaison receive-reply communique avec une opération request-response d'un portType du processus.

Invoke : elle permet l'appel d'un partenaire (service) de type one way ou request-response.

Assign : cette activité permet la mise à jour des variables.

Throw : génère une exception à l'intérieur du processus, elle envoyée au catch de fault handler.

Wait : permet l'attente pour une période donnée.

Empty : c'est l'opération rien-faire , elle est utile pour la synchronisation.

Activités structurées : ces activités permettent le parallélisme, la séquence, les branchements, et les boucles, elles peuvent être imbriquées.

Sequence : elle définit une suite d'activités

flow : il définit des activités en parallèle

switch : il définit un branchement décisionnel

if : il représente un autre cas de branchement décisionnel

while: elle modélise une boucle conditionnelle.

Pick : il bloque l'activité jusqu'à la réception d'un message ou l'expiration d'une période de temps, après cela l'activité est exécutée.

Scope : il définit une activité imbriquée ayant ses propres données, gestionnaires d'exception ou de compensation.

Compensate : indique une tâche particulière devant être citée dans un élément appartenant à faultHandlers.

Cette activité permet par exemple, l'annulation des transactions.

De façon générale, un processus BPEL commence par une activité « receive » et se termine par une activité « send ».

Dans l'annexe B, nous présentons un exemple d'une composition BPEL, elle permet de réaliser plusieurs sortes de flux de contrôle : la séquence, le parallélisme, et les choix conditionnels

II .2.5) Couches Verticales (Sécurité et Transactions)

Plusieurs spécifications [Nadalin et al, 2004], [Box et al, 2004] ont été proposées par le W3C pour gérer les problèmes de sécurité, de transactions, de gestion de messages.

Par exemple la spécification « WS-addressing » [Box et al, 2004] permet l'adressage (et le routage) des messages SOAP, en introduisant des entêtes tels que 'To','From','ReplyTo', 'FaultTo'… dans l'enveloppe SOAP.

Notons que ces aspects (sécurité, gestion, transactions...) ne sont pas traités dans cette thèse.

III) Conclusion

Nous avons présenté dans ce chapitre, les principales spécifications qui supportent la technologie des services web, nous avons montré le noyau, i.e. le protocole SOAP, l'interface WSDL, et l'annuaire UDDI, nous avons présenté aussi les mécanismes d'extensions tels que les formats de composition (BPEL, WSCI, WSCL…). Nous avons montré aussi les formalismes de description sémantiques tels qu'OWLS, WSMO, SAWSDL. Nous notons que ces formalismes sémantiques, sont primordiaux pour l'automatisation du cycle de vie des services.

Chapitre 3 :

La découverte des services web

Sommaire

I) Introduction

Le W3C définit la découverte de services 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” [W3C, 2004 b].

[Keller et al, 2004] Définissent la découverte comme la localisation automatique des services répondant à une requête utilisateur.

[Booth et al, 2004] décrivent le processus de découverte, comme étant la localisation d'une description compréhensible par la machine d'un service éventuellement inconnu au préalable et correspondant à certains critères fonctionnels.

[Toma et al, 2005] Définissent la découverte comme le processus qui prend en entrée une requête utilisateur et retourne une liste de ressources ou services, pouvant combler éventuellement le besoin décrit.

Ces définitions mettent l’accent sur le mécanisme de comparaison de la requête avec les services, ainsi que son degré d’automatisation. Selon notre point de vue la découverte de services vise à comparer une requête d’un utilisateur avec les capacités d’un service web, et trie les résultats selon un certain mécanisme.

Les capacités peuvent être fonctionnelles (les descriptions d’interfaces : syntaxiques ou sémantiques, les descriptions comportementales) ou non fonctionnelles (la qualité de service).

Plusieurs critères peuvent être utilisés pour catégoriser les approches de découverte [Mohebbi et al, 2010], nous avons :

- le critère de centralisation/distribution des annuaires.
- Le principe de l’algorithme de matching (syntaxique, sémantique (logique, non logique), hybride, comportemental, non fonctionnel…)
- Le critère d’automatisation

II) Critères de Découverte

II.1) Critère d'Architecture (Centralisation /Distribution)

Ce critère concerne la manière de stockage et de localisation des descriptions de services dans le réseau. Selon ce point de vue, les systèmes peuvent être centralisés ou décentralisés [Garofalakis et al, 2004], [Klush et Kapahnke, 2008].

Les systèmes centralisés reposent sur un seul annuaire qui peut être géré par un matchmaker. Les systèmes décentralisés stockent les descriptions de services de manière distribuées (tels que les réseaux Pair-à-Pair (P2P)).

Nous pouvons noter, par exemple que [Patil et al, 2004] est une approche distribuée alors que [Clement et al, 2004] est de nature centralisée.

II.2) Critère d'Automatisation

Ce critère concerne le degré d'intervention de l'utilisateur dans la découverte, nous distinguons les approches manuelles et automatiques [Garofalakis et al, 2004]. La découverte manuelle est initiée et assistée par un utilisateur humain. La découverte automatique n'implique pas l'intervention de l'utilisateur, pendant la recherche.

II.3) Critère de Matchmaking

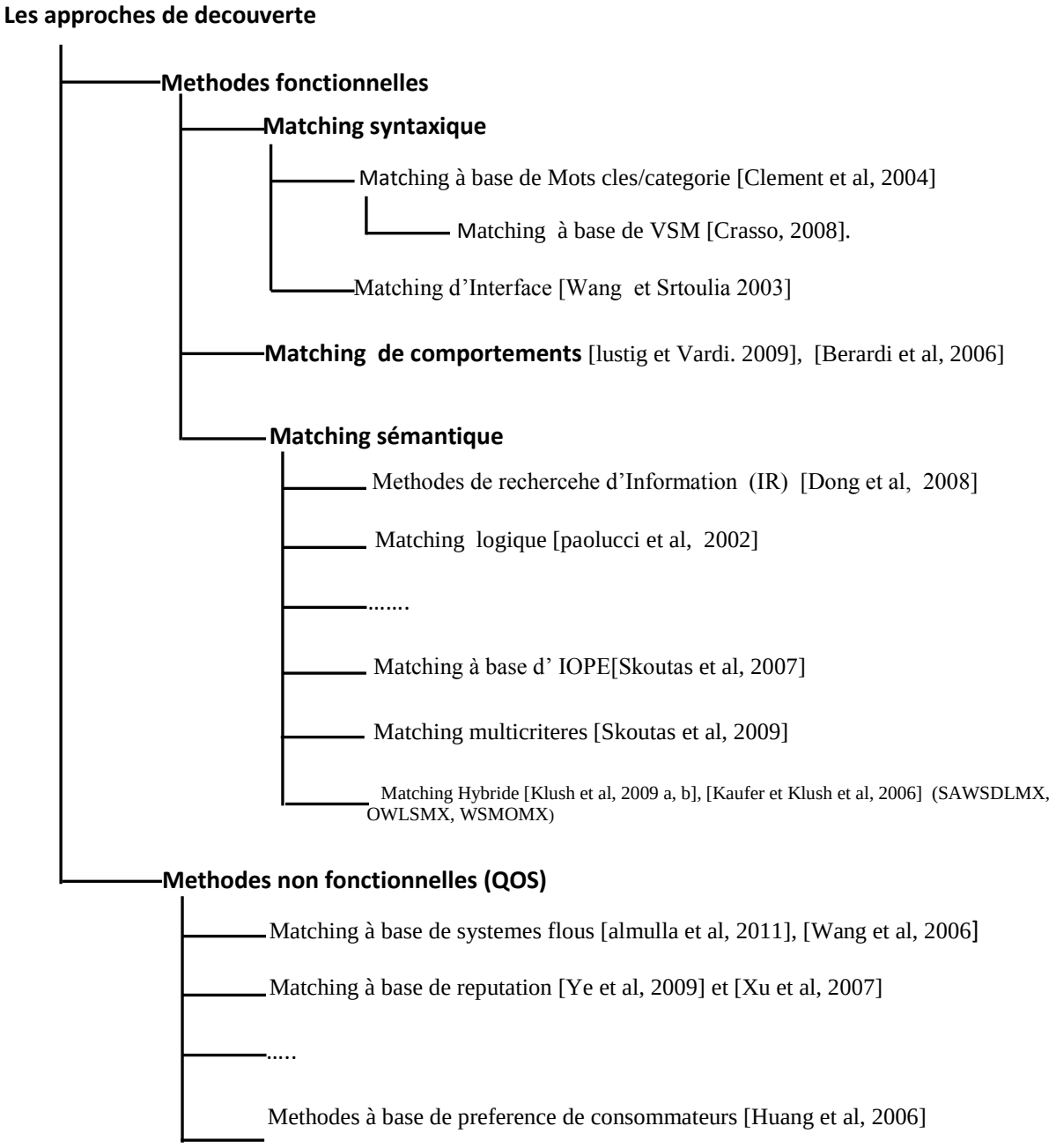
Ce point de vue concerne les données à comparer ainsi que l'algorithme d'appariement de la requête avec la description du service. On distingue les approches fonctionnelles et les approches non fonctionnelles:

Les approches non fonctionnelles comparent les valeurs de QOS des services web et les classent selon le besoin de l'utilisateur. Les algorithmes de comparaison s'inspirent des méthodes de décision multicritères (telles que Topsis [Hwang et Yoon, 1981], Electre [Roy, 1968]….), des systèmes inférentiels flous…

Les approches fonctionnelles comparent les descriptions d'interfaces ou de comportement des services web, elles peuvent être syntaxiques (WSDL), sémantiques (OWLS, SAWSDL, WSMO) ou comportementales.

Dans ce qui reste, nous s'intéressons uniquement au critère de matchmaking (i.e. L'algorithme de matching ainsi que les descriptions à comparer).

Dans la Figure III.1 nous montrons une classification des approches de découverte, en utilisant le critère de matching



FigureIII.1. Quelques approches de découverte

III) Approches de Découverte

III.1) Approches Fonctionnelles :

Les capacités fonctionnelles d'un service peuvent inclure, des informations telles que les entrées, les sorties, la catégorie du service, le comportement, les annotations informelles et éventuellement les préférences. Plusieurs approches ont été proposées pour la découverte fonctionnelle de services, on distingue les approches sémantiques, les approches syntaxiques (non sémantiques), et les approches hybrides.

Les approches syntaxiques, emploient les techniques de recherche d'informations pour comparer des interfaces syntaxiques telles que le document WSDL.

Les approches sémantiques emploient la logique, le datamining, et même les mesures de similarité pour comparer des interfaces sémantiques (OWLS [Martin et al, 2004], SAWSDL [Farrell et Lausen, 2007],WSMO [Roman et al, 2005]).

III.1.1 Approches Syntaxiques (ou Basées sur les Interfaces Syntaxiques)

Ces approches utilisent généralement l'interface WSDL, comme description de services, et adoptent les techniques de recherche d'informations et éventuellement le clustering pour le matching.

La recherche dans l'annuaire UDDI [Clement et al, 2004] est l'exemple typique d'une découverte syntaxique, les mots clés de la requête sont comparés avec les attributs enregistrés, nous pouvons faire une recherche par nom de services ou nom d'entreprise, ou sa catégorie.

D'autres attributs tels que la localisation ou le binding template ou les tmodels peuvent être utilisés. [Garofalakis et al, 2004].

[Liang et al, 2004] emploie le thesaurus wordnet [Fellbaum, 1998] pour enrichir les mots clés de la requête avec des synonymes. Ceci permet l'augmentation du rappel et l'extension des résultats. Dans [Li et al, 2004] les auteurs adoptent la recherche par mots clés sur un réseau P2P.

Query by Example (QBE) permet à un utilisateur de rechercher des services en se basant sur un exemple (des mots-clés) [Crasso et al, 2008]. La requête comprend des mots-clés décrivant la fonctionnalité, le nom d'opération et les noms de paramètre. Cette méthode

adopte le modèle de représentation vectoriel ou Vector Space Model (VSM) et la mesure cosinus pour le processus de matching. Les inconvénients de cette méthode sont :

- la mauvaise affectation de catégorie de services affaiblit le rappel.
- L'utilisation des caractères jockers, affaiblit la précision, en plus la non prise en charge de la synonymie dans les documents wsdl réduit le taux de rappel.

[Ernst et al, 2006] traitent les problèmes de substituabilité et de composabilité de services. Les auteurs supposent que les informations concernant des exécutions passées des services sont disponibles, et proposent un algorithme comportant deux phases. La première phase calcule la similarité des entrées d'un service et des sorties d'un autre, Ces valeurs sont extraites d'un journal d'exécutions passées ensuite le système décide si les deux services sont composables, Sinon, il les marque en tant qu'éventuellement substituables.

Si les services sont considérés comme équivalents, les opérations similaires sont marquées comme interchangeables après découverte de leurs paramètres d'entrées - sorties et génération des éventuelles correspondances. Ces traitements sont efféctués, durant la deuxième phase. Les expérimentations conduites par les auteurs confirment l'absence de faux positifs i.e. que la précision =100%. Les auteurs suggèrent le renforcement des tests en utilisant des services concrets (les paramètres d'entrées-sorties ont des valeurs réels). De plus, ils envisagent d'employer une technique d'apprentissage automatique pour améliorer l'appariement des paramètres d'entrées/ sorties.

Il est utile de noter que la substitution n'est pas symétrique (i.e. si un service A peut substituer le service B, alors l'inverse n'est pas toujours vrai).

L'approche [Algergawy et al, 2010] utilise deux algorithmes: le matching basé sur les niveaux et le matching des schémas. Le premier compare syntaxiquement les éléments concrets de WSDL, et le deuxième compare les éléments abstraits du document WSDL.

Les limites de ce type d'approches (i.e. syntaxiques) sont nombreuses :

- L'ambigüité présente dans le langage naturel est l'inconvénient principal de ces méthodes, en effet les méthodes syntaxiques ont un faible taux de rappel et de précision.

- La nécessite de l'intervention de l'utilisateur : l'utilisateur doit intervenir pour corriger les décisions et gérer les éventuels conflits et ambigüités rencontrées lors du matching.

III.1.2) Approches Sémantiques

Plusieurs interfaces sémantiques ont été créées pour assurer la découverte et la composition de services WSMO [Roman et al, 2005], OWL-S [Martin et al, 2004], et SAWSDL [Farrell et Lausen, 2007], toutes les approches de cette catégorie adoptent les ontologies pour le matching de la requête avec les services. Nous notons que ces approches, sont plus complexes et plus fiables que les techniques syntaxiques.

Nous distinguons 03 classes d'approches sémantiques, les approches logiques, non logiques et hybrides. Les approches logiques exploitent les inférences pour vérifier la compatibilité entre la requête et le 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, le matching de graphes, la recherche d'informations, les mesures de similarité. La troisième classe mélange les deux premiers types.

III.1.2.1) Approches Logiques

[Paolucci et al, 2002] présentent l'une des premières approches sémantiques pour la découverte de services, pour cela, ils comparent les sorties de la requête avec les sorties du service offert, selon l'algorithme suivant :

matching (R.O,S.O) {
// Avec S.O dénote la sortie du service S et R.O dénote la sortie de la requête.
Retourner
Exact :si les deux concepts ontologiques R.O et S.O sont équivalents.
Plug-in : si S.O subsume R.O
Subsume : si R.O subsume S.O
Fail: aucune relation d'équivalence ou d subsomption n'existe entre eux
}

R.O et S.O sont des concepts exprimés avec le langage Daml (prédécesseur de OWL [W3C, 2004 a]), les services et la requête sont formalisés avec Daml-s [Ankolekar et al, 2002].

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

L'algorithme est simple en termes d'implémentation, mais il présente plusieurs défauts : en premier lieu, il possède un faible rappel et une faible précision à cause de la naïveté des règles utilisées, en deuxième lieu il n'est pas possible comparer les services de la même classe (même score).

[Li et Horroks , 2003] étendent l'algorithme de [Paolucci et al, 2002] en ajoutant un cinquième score intitulé intersection, ce dernier permet de repérer les services ayant des concepts qui chevauchent avec la requête. Les auteurs emploient Racer [Haarslev et Möller, 2003] comme moteur de raisonnement sur les descriptions DAMLS.

[Benatallah et al, 2005 b] 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 avec le problème de calcul, à coût minimal, des transversales minimes d'un hypergraphe pondéré (computing the minimal transversals with minimum cost of a weighted hypergraph).

Les résultats expérimentaux de cette technique [Rey et al, 2003] sont satisfaisants, mais nécessitent des améliorations pour prendre en compte un grand nombre d'ontologies, éventuellement hétérogènes.

[Mandel et McIlraith, 2003] présentent une extension du langage BPEL. Les auteurs remplacent la référence statique aux services invoqués, par une description OWL-S de type service profile. Ils mettent en place un SDS, un service de découverte sémantique décrit par un service profile OWL-S. SDS utilise le raisonneur JTP (Java Theorem Prover) pour construire des requêtes DQL.

Pendant l'analyse du document BPEL, le service SDS intervient et envoie des requêtes DQL générées à partir des services profile du processus BPEL vers un annuaire contenant des

services décrits en OWL-S. ainsi chaque service sera découvert juste avant son invocation durant l'exécution du workflow.

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

III.1.2.2) Approches Non Logiques

Les approches logiques possèdent plusieurs inconvénients, la complexité élevée du raisonnement affaiblit les chances de scalabilité, en outre les raisonneurs logiques ont un faible rappel (beaucoup de faux négatifs), puisque la subsomption ne couvre pas tous liens sémantiques, et en dernier nous constatons que la majorité des services web actuels ne sont pas annotés avec des langages logiques formels.

D'autres part, nous constatons que la présence des descriptions informelles ou la sémantique implicite (la fréquence des termes) au niveau des services, peut être efficace dans la découverte, en plus la complexité du matching est moins élevée que celles des approches logiques. En général, ces techniques utilisent le matching de graphes, la linguistique, le data mining, les techniques de recherche d'information, et les mesures de similarité.

- [Dong et al, 2004] proposent un système nommé woogle, il permet de trouver des opérations de services similaires à un prototype donné. Pour cela il utilise plusieurs sources d'évidences. Il propose aussi les services composables avec un service donné.

Le système calcule les similarités entre les descriptions textuelles des opérations de services et entre les identifiants des paramètres d'entrées - sorties. Les auteurs utilisent un algorithme de clustering pour grouper les noms de paramètres, dans des classes sémantiques, qui seront par la suite, utilisées pour calculer la similarité inter paramètres (Inputs/Outputs).

Différents types de similarités sont combinés en utilisant une fonction d'agrégation linéaire, les poids de cette fonction sont assignés manuellement, (après l'analyse des résultats de différentes expérimentations). Les auteurs proposent l'apprentissage des poids en fonction du feed_back des utilisateurs dans les travaux futurs. Cette approche utilise des similarités sémantiques et syntaxiques.

L'algorithme de clustering adopte le principe suivant : « les paramètres expriment souvent le même concept s'ils apparaissent souvent ensemble". Des règles d'association sont définies entre les identifiants des paramètres. La règle d'association entre deux identifiants t1 et t2 est désignée par t1→ t2 (a,b) ou le support a est la probabilité que le paramètre t1 apparait dans

une entrée ou une sortie, et la confidence b est la probabilité que le paramètre t2 apparait dans une entrée ou une sortie sachant qu'elle contient t1.

Il est utile de noter que ces règles ne sont pas symétriques : t1→ t2 et t2 →t1 peuvent avoir des valeurs de supports et de confidences différentes. Les auteurs emploient la fusion (merging) et la scission (splitting) pour améliorer le clustering et garantir une meilleure cohésion. Les auteurs utilisent un autre principe, pour éliminer les bruits des concepts. Un terme associé à un concept est considéré comme bruit, si dans la moitié de ses occurrences il n'est pas associé à d'autres termes du même concept.

Les expérimentations ont confirmé que, la cohésion rigide empêche les grands clusters étroitement associés de se fusionner. En plus la fusion précoce peut être inadéquate car elle empêchera une éventuelle fusion ultérieure.

Le moteur Woogle, propose trois types de recherche : la recherche par mots clés classique, la recherche par prototype « template search » et la recherche par composition « composition search ». Les deux dernières utilisent l'algorithme de clustering. « Template search » considère les mots clés de l'utilisateur comme des paramètres d'opérations, et applique l'algorithme de clustering pour la recherche d'un service équivalent. Le second type de recherche retourne l'éventuelle composition de services qui répond au besoin de l'utilisateur si les services existants ne sont pas satisfaisants.

- [Li et al, 2007] proposent une approche de découverte à base du framework WSMO, ils utilisent la théorie de graphe pour évaluer la similarité sémantique entre deux concepts.

- Le matchmaker URBE présenté dans [Plebani et 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 donnés associées aux éléments XML (XSD I/O)
- ✓ les concepts ontologiques en utilisant une similarité structurelle (basée sur le calcul de longueur de chemins)

Selon [Klusch et al, 2009 b], URBE effectue un appariement de graphes bipartis des opérations de services (de manière similaire à SAWSDL-MX1 et SAWSDL-MX2). Ensuite il classe les résultats, en utilisant un score qui agrège la similarité structurelle et textuelle.

Nous notons que, URBE a gagné la compétition des matchmakers S3[6] (2008 et 2009) en termes de précision moyenne (0.72). URBE a dépassé même la précision offerte par les matchmakers adaptatifs tels que SAWSDL-mx2. Mais le temps d'exécution moyen est un peu long (20 sec) par rapport aux matchmakers adaptatifs.

- Lumina [Li et al, 2006] est développé dans le cadre du projet de météor-s, il convertit les descriptions WSDL-S (le prédécesseur de SAWSDL) en structures enregistrées dans l'annuaire UDDI, le matching sémantique de LUMINA se base sur le calcul de longueurs de chemins qui relient les concepts d'une ontologie de domaine. LUMINA fait aussi une recherche syntaxique par simples mots clés.
- WXplorer de [stroulia et yang, 2004] propose deux fonctions de similarité : une similarité structurelle qui compare uniquement les types des arguments d'opérations, et une similarité sémantique qui compare les noms d'opérations et d'arguments à l'aide du thesaurus wordnet [Fellbaum, 1998]. Et pour cela les auteurs définissent 04 scores sémantiques : exact match, plug-in and subsumes match
- [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 preconditions, les effets.

Pour apparier les entrées /sorties ils proposent les formules suivantes :

$$\mathbf{recall=(C_R,C_A)=|x\ tel\ que\ x\in (ascendants(C_R)\cap ascendants(C_A))|/|y\ tel\ que\ y\in ascendants(C_R)|}$$

$$\mathbf{prec=(C_R,C_A)=|x\ tel\ que\ x\in (ascendants(C_R)\cap ascendants(C_A))|/|y\ tel\ que\ y\in ascendants(C_A)|}$$

C_A dénote un concept de service et C_R un concept de la requête, les formule calculent les concepts (éventuellement les propriétés) communes au niveau des ascendants des deux concepts.

Pour apparier les preconditions et les effets, les auteurs utilisent les concepts et les mesures suivantes:

φ_{Aef} représente l'ensemble des effets du service

φ_{Ref} représente l'ensemble des effets de la requête

φ_{Apr} représente l'ensemble des preconditions du service

[6] Annual International Contest S3 on **Semantic Service Selection** – Retrieval Performance Evaluation of Matchmakers for Semantic Web Services,http://www-ags.dfki.uni-sb.de/_klusch/s3/

φ_{Rpr} représente l'ensemble des preconditions de la requête

Notons que, chaque effet ou precondition est une formule logique.

$Recall_{ef}$ (φ_{Ref} , φ_{Aef}) noté RC_{ef}

C'est la proportion des effets de la requête qui sont disponibles dans ceux du service.

$Precision_{ef}$ (φ_{Ref} , φ_{Aef}) noté PR_{ef}

C'est la proportion des effets de service devant satisfaire un effet de la requête.

$Recall_{pr}$ (φ_{Rpr} , φ_{Apr}) noté RC_{pr}

C'est la proportion des preconditions de service qui sont satisfaites par la requête.

$Precision_{pr}$ (φ_{Rpr} , φ_{Apr}) noté PR_{pr}

C'est la proportion des preconditions de la requête qui doivent satisfaire les preconditions du service.

Les auteurs agrègent les 04 sous-scores de matching selon la fonction suivante :

$$Rec=(W_{in}.\ RC_{in} + W_{out}.\ RC_{out} + W_{pr}.\ RC_{pr} + W_{ef}.\ RC_{ef})/(\ W_{in} + W_{out} + W_{pr} + W_{ef})$$

$$Prc=(W_{in}.\ PR_{in} + W_{out}.\ PR_{out} + W_{pr}.\ PR_{pr} + W_{ef}.\ PR_{ef})/(\ W_{in} + W_{out} + W_{pr} + W_{ef})$$

Nous notons que cette approche n'est pas expérimentée.

III.1.2.3) Approches Hybrides (Logiques et Non Logiques)

Ces dernières implémentent plusieurs filtres, certains d'entre eux sont purement logiques alors que d'autres se basent sur les techniques de recherche d'information, ou le datamining…, Nous notons que certains filtres compensent la défaillance des autres, et de ce fait un service web sera accepté s'il satisfait au moins un filtre. Nous notons aussi que les scores des filtres logiques sont plus élevés que ceux des filtres à base de techniques de recherche d'information (en termes de pertinence).

- **LOG4SWS.KOM** [Schulte et al, 2010] est un matchmaker sémantique et hybride, il associe des valeurs numériques aux scores discrets exact, plug-in, subsumes, subsumed-by, intersection et fail. Pour cela, il utilise la profondeur des relations de subsomption, ce système utilise aussi les similarités à base de wordnet [Fellbaum, 1998] pour comparer les noms des paramètres et des opérations.

- **[Syeda-Mahmood et al, 2005]** présentent un matchmaker sémantique et hybride qui utilise des annotations WSDL-S. les concepts utilisés sont annotés avec des ontologies dépendantes et indépendantes du domaine. L'agrégation des résultats individuels est fixée, elle adopte toujours le score minimal.
- **FUSION** [Kourtesis et Paraskakis, 2008] Combine le raisonnement logique et la recherche par mots-clés, dans un premier stade, il effectue un pré- filtrage à base de mots clés, ensuite il continue avec un raisonnement logique.
- **OWLSMX (version 1)** [Klusch et al, 2006].

Ce système [Klusch et al, 2006], applique 06 filtres successifs pour chaque paire (requête-service), ces filtres sont : Exact, Plug-in, Subsumes Subsumed-by , Hybrid-Subsumed-by, et nearest neighbor.

Les 04 premiers filtres sont purement logiques alors les deux derniers sont hybrides parce qu'ils combinent le raisonnement logique et les mesures de similarités textuelles (ou syntaxiques)

Exact match. Service S s'apparie avec une requête R avec un degré EXACT ssi ($\forall$ IN*S*$\in$ entréesS, $\exists$ IN*R*$\in$ entréesR : IN*S* $\equiv$ IN*R* $\wedge$ $\forall$ OUT*R* $\in$ sortiesR, $\exists$ OUT*S* $\in$ sortiesS, : OUT*R* $\equiv$ OUT*S*. Ceci veut dire que les entrées- sorties du service (I/O) sont parfaitement équivalents à celles de la requête (au sens logique), OUTR est une sortie de la requête.

Plug-in match. un service S s'apparie avec une requête R avec un degré PLUGINTO ssi ($\forall$ IN*S*$\in$ entréesS, $\exists$ IN*R*$\in$ entréesR : IN*S* $\geq^{\cdot}$ IN*R* $\wedge$ $\forall$ OUT*R*$\in$ sortiesR, $\exists$ OUTS $\in$ sortiesS : OUT*S* $\in$ LSC(OUT*R*). Avec LSC(OUT*R*) représente les enfants directs de OUT*R (* LSC veut dire « least specific concepts »)

Ce score accepte les services dont les entrées sont plus générales que celles de la requête, en plus les sorties du service doivent être des fils directs des sorties de la requête.

Subsumes match. Un service S s'apparie avec une requête R avec un degré Subsume ssi ($\forall$ IN*S*$\in$ entréesS, $\exists$ IN*R*$\in$ entréesR: IN*S* $\geq^{\cdot}$ IN*R* $\wedge$ $\forall$ OUT*R* $\in$ sortiesR $\exists$ OUT*S*$\in$ sortiesS: OUT*R* $\geq^{\cdot}$ OUT*S*. Ce filtre accepte des services dont les sorties sont subsumées par les sorties de la requête (ce filtre englobe celui du plug-in).

Subsumed-by match.

Un service S s'apparie avec une requête R avec un degré Subsumed-by ssi : $\forall$ IN*S* $\in$ entréesS $\exists$ IN*R* $\in$ entréesR: IN*S* $\geq^{\cdot}$ IN*R* $\wedge$ OUT*R* $\in$ sortiesR $\exists$ OUT*S*$\in$ sortiesS (OUT*S* $\equiv$ OUT*R* $\vee$ OUT*S* $\in$ LGC(OUT*R*)) Avec LGC(OUT*R*) représente les parents directs de OUT*R (* LGC veut dire « least generic concepts »).

Ce filtre tolère les services dont les sorties sont légèrement différentes par rapport aux sorties de la requête, et en particulier il accepte les parents directs des sorties spécifiées dans la requête.

Logic-based fail.

Aucun des filtres précédents n'est activé (échec des filtres basés sur le raisonnement logique)

Hybrid Subsumed-by match.

Un service S s'apparie avec une requête R avec un degré Hybrid-Subsume-by ssi $\forall$ INS$\in$ entréesS, $\exists$ IN$R\in$ entréesR: INS $\geq^{\cdot}$ INR $\wedge$ $\forall$ OUTR$\in$ sortiesR $\exists$ OUT$S\in$ sortiesS: (OUTS $\equiv$ OUTR $\vee$ OUTS $\in$ LGC(OUTR)) $\wedge$ SIMIR(S, R) $\geq \alpha$. Ce filtre est une spécialisation de Subsumed-by match, en particulier il ajoute une contrainte de similarité textuelle, pour minimiser les faux positifs.

Nearest-neighbor match.

Un service S s'apparie avec une requête R avec un degré NEAREST NEIGHBOR ssi (SIMIR(S, R) $\geq \alpha$). SIMIR est implémentée avec l'une des 04 mesures de similarité (cosine ,extended jaccard, loss of information, jensen shannon information divergence), pour plus de détail sur ces mesures voir [Klusch et al, 2006].

Fail match.

Il n'y a pas de relation entre la requête et le service (échec total)

L'ordre de pertinence des scores est le suivant :EXACT < PLUG-IN < SUBSUMES < SUBSUMED-BY < LOGIC-BASED FAIL < Hybrid SUBSUMED-BY< NEAREST-NEIGHBOR < FAIL.

L'algorithme générique du système OWLSMX (version1) est donné comme suit :

Algorithme de Matching

//l'algorithme cherche des services publiés S qui s'apparient avec la requête R.

//il retourne un ensemble de paires (S, degreDeMatch, SIMIR(R, S)) avec un degré //d'appariement « degreeOfMatch » différent de FAIL, et une similarité syntaxique //dépassant le seuil de l'utilisateur α.

1: fonction MATCH(Requete R, α)

2: Var: Resultat, DegreDeMatch, Filtres-hybrides = {HYBRID-SUBSUMED-BY, NEAREST-NEIGHBOUR}

3: pour tout (S, dom) $\in$ Candidats-entrées-S(entréesR) et (S, dom') $\in$

Candidats-sorties-S(sortiesR) faire

4: DegreDeMatch ← MIN(dom, dom')

5: si DegreDeMatch ≥ NEAREST-NEIGHBOUR et (DegreDeMatch ∉ Filtres-hybrides ou SIMIR(R,S) ≥ α) alors

6: Resultat := Resultat ∪ { (S, DegreDeMatch, SIMIR(R,S)) }

7: fin si

8: fin pour

9: retourner Resultat

10: end fonction

La fonction Candidats-entrées-S(entréesR) retourne le score minimal, de tous les appariements des entrées de S avec les entrées de la requête.

Candidats-sorties-S(sortiesR) retourne score minimal de tous les appariements des sorties de R avec celles de S.

Le système OWLSMX (version 1) contient 05 matchmakers:

OWLSM0: il est purement logique, il applique les 04 premiers filtres

OWLSM1 : il est hybride, il applique tous les filtres, en considérant la mesure Loss of Information [klusch et al, 2006].

OWLSM2: il est hybride, il applique tous les filtres, en considérant la mesure Extended Jaccard. [klusch et al, 2006].

OWLSM3: il est hybride, il applique tous les filtres, en considérant la mesure Cosine [klusch et al, 2006]..

OWLSM4: il est hybride, il applique tous les filtres, en considérant la mesure Jensen-Shannon Information Divergence [klusch et al, 2006].

La comparaison (à l'aide des métriques rappel/précision) des mesures de similarités textuelles sur la collection de test OWLS-TC v2[7]a montré les résultats suivants :

- Si on considère juste les entrées/sorties comme descripteurs, alors le classement est : cosine> extended jaccard>jensen shannon
- Si on considère tout le contenu du document owls [8]alors le classement est : jensen shannon > extended jaccard> cosine

La comparaison (à l'aide des métriques rappel/précision) des matchmakers les plus performants a donné le classement suivant :

[7] projects.semwebcentral.org/projects/owls-tc/

[8] Tous les elements du document owls

- Cosine(IR)> OWLSM3> OWLSM0.
- Tous matchmakers hybrides sont très performants par rapport au matchmaker logique OWLSM0.parce que les filtres hybrides permettent la minimisation des faux positifs et des faux négatifs.
- L'emploi des simples mesures de similarité (telle que Cosine(IR)) dépasse les performances du matchmaker OWLSMX, et surtout lorsqu'on enrichit le contenu textuel en entrée (hasInput, hasOutput, serviceName, and textDescription)

Les inconvénients majeurs de OWLMX sont :

- La présence des faux positifs qui est due aux motifs suivants :
    - Le degré de granularité l'ontologie de domaine peut provoquer l'existence de longues chaines de subsomption entre deux concepts malgré leur divergence sémantique. Cette chaine active le filtre subsume.
    - le filtre subsumed by peut provoquer aussi des faux positifs à cause des relations liants les sorties de la requête et les sorties de services. Cet effet peut être masqué avec l'introduction du filtre hybrid-subsumed-by.
    - La contrainte « all-quantified logical matching » entraine la tolérance des services qui ont des entrées manquantes ou des sorties superflues, et ceci peut causer l'acceptation des faux positifs.
- La présence des faux négatifs qui est due aux motifs suivants :
    - Les insuffisances de modélisation de l'ontologie peuvent provoquer des faux négatifs. Par exemple l'appariement de deux concepts frères (ayant le même parent), cause l'échec logique malgré leur similarité sémantique. En outre la généricité des entrées de la requête par rapport à celles du service provoque la même erreur. Ce phénomène peut être allégé avec l'introduction de la similarité textuelle i.e. l'emploi du filtre nearest neighbor.
    - La contrainte « all-quantified logical matching » impose le respect de la même relation de subsomption pour tous les constituants de la requête ou le service, mais parfois certains concepts peuvent avoir des relations inversées (spécialisation au lieu de subsomption) , et ceci peut causer

des faux négatifs. Ce cas peut être allégé avec l'introduction de la similarité textuelle i.e. l'emploi du filtre nearest neighbor.

- **La version OWLS-MX2** [Klush et al, 2009 a] est introduite dans le but de minimiser les faux positifs de OWLS-MX, et pour cela ils introduisent des mesures textuelles pour les filtres subsumes and plug-in (similairement au filtre hybrid subsumed-by), ceci peut masquer tous les inconvénients hérités par OWLS-M0. Les expérimentations montrent qu'OWLS-MX2 (M3) est plus performant que l'OWLS-MX en termes de rappel et de précision. Et il dépasse COSINE(IR) dans 40% des cas expérimentés et COSINE(IR) le dépasse dans 60% des cas.

- **Le matchmaker OWLS-MX3** [Klusch & Kapahnke 2009 c] est similaire à SAWSDL-MX2 dans le sens où il applique un apprentissage hors ligne à base de SVM. Les concepts de services sont annotés avec OWL-DL. Les majeures différences avec SAWSDLMX sont :
  - Le matching structurel de SAWSDL-MX2 s'applique sur des documents WSDL (lies à SAWSDL) en ignorant les annotations sémantiques, alors que OWLSMX3 ignore la partie grounding de OWLS
  - SAWSDL-MX2 apparie les opérations en créant des affectations bijectives (en utilisant les graphes bipartis) alors que OWLSMX3 apparie les paramètres d'opérations en créant des affectations non bijectives (multiples) ce qui provoque des faux positifs surtout si un paramètre subsume plusieurs concepts de la partie homologue.

- **[Skoutas et al, 2009]** proposent une approche à base de plusieurs critères de matching, ces auteurs constatent que la majorité des approches de découverte ne prennent pas en compte des critères de matching variés, et le peu qui font le matching multicritère adoptent des schémas d'agrégation peu efficaces.

D'autre part la présence de plusieurs attributs à comparer (entées, sorties, pre-conditions effets, qos …), complique le classement des résultats, et peut provoquer des phénomènes de compensations.

Pour pallier à ces problèmes, les auteurs proposent trois indices et par conséquent trois algorithmes pour agréger les scores des vecteurs, et classer les services web en utilisant plusieurs critères (des mesures de similarités ou des raisonnements logiques).

Ces indices sont notés U.dds, U.dgs, U.ds (U est un service web donné).

Le premier algorithme est nommé TKDD (Top K DominateD), il classe les services à base de l'indice U.dds.

Sachant que u est un vecteur (instance) contenant des scores de tous les paramètres décrivant un service U et à l'aide d'un critère (algorithme) de matching particulier, (i.e. U peut posséder plusieurs instances u selon le nombre de critères de matching). Voir la figure III.2.

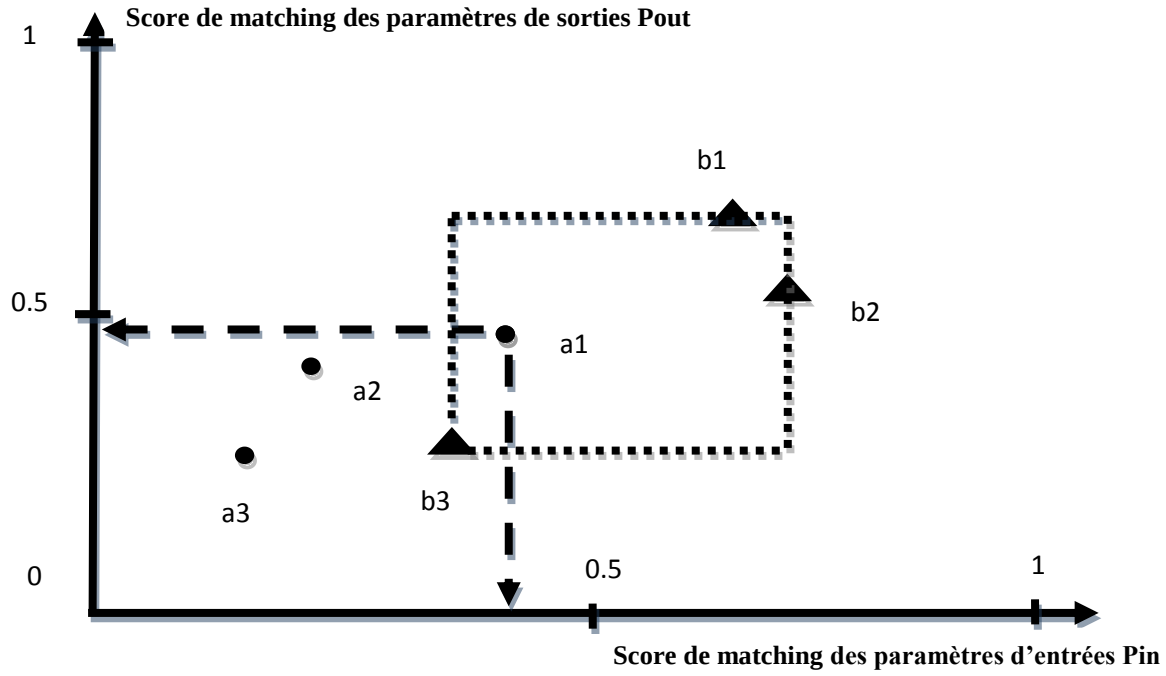


Figure III.2. Les services représentés avec les scores **Pin** et **Pout**

Par exemple le service B est représenté par 03 instances b1=(0.7,0.7), b2=(0.73,0.6), b3=(0.4,0.45). Chaque instance contient deux scores, ils représentent les similarités entre les entrées (respectivement les sorties) de la requête et le service.

u.dds= $\sum_{V \neq U}$(|{v ∈V tel que v>u}|/|V|).....(1)

La formule 1 considère la moyenne d'instances (des services V) qui dominent u

U.dds=[$\sum_{u \in U}$ u.dds]/|U|.....(2)

La formule 2 (l'indice U.dds) donne la moyenne des scores u.dds obtenus pour le service U.

Le deuxième algorithme est nommé TKDG (Top K DominatinG).il classe les services à base de l'indice U.dgs

u.dgs= $\sum_{V \neq U}$(|{v ∈V tel que u>v}|/|V|).....(3)

U.dgs=[$\sum_{u \in U}$ u.dgs]/|U| ..... (4)

La formule 3 Considère la moyenne d'instances des services V qui sont dominés par u et la formule 4 (L'indice U.dgs) fournit la moyenne des scores u.dgs obtenus pour le service U.

Le troisième algorithme est nommé TKM (Top K doMinance), il classe les services à base de l'indice U.ds

u.ds=u.dgs-λ*u.dds .....(5)

U.ds=[$\sum_{u \in U}$ u.ds]/|U| ..... (6)

L'indice u.ds (formule 5) est une combinaison des deux critères précédents, il favorise u pour chaque instance dominée, et le pénalise pour chaque instance qui le domine, (λ est fixé empiriquement par les auteurs).

U.ds (formule 6) est la moyenne des scores de dominance des instances de U.

Les expérimentations sont appliquées sur le corpus OWLS-TC v2 elles, montrent que TKM est le plus performant en termes de rappel et de précision, TKDD présente aussi de bonnes performances. Le classement final de ces algorithmes est donné comme suit :

TKM (20)>TKM(5)>TKDD>TKDG (20 et 5 sont les valeurs de λ)

En termes de temps d'exécution TKDD est le plus efficace :TKDD>TKDG> TKM (x)

TKM excelle par rapport à toutes les versions de OWLSMX (en termes de rappel et de précisons) :

TKM (20)>TKM(5)> { OWLSM1, OWLSM2, OWLSM3, OWLSM4}>OWLSM0

TKM excelle aussi par rapport à la majorité des algorithmes d'agrégation de rang (en termes de rappel et de précisons), tels que, Borda [Aslam et Montague, 2001], condorcet [Montague et Aslam, 2002] Comb Sum et Comb MNZ [Fox et Shaw, 1993] , OutRank [Farah et Vanderpooten, 2007].

Larks [sycara et al, 2002] est un matchmaker hybride qui a fortement inspiré OWLMX. Il permet de comparer les preconditions- effets-entrées-sorties des services. Il ne propose pas de filtres subsumes ou subsumed by ou nearest neighbor.

Il adopte un formalisme propriétaire pour la description des capacités de services. Il est utile de noter que LARKS n'est jamais expérimenté.

- **WSMO-MX**

WSMO-MX [Kaufer et Klusch, 2006] accepte en entrée des services spécifiés en WSML [Bruijn et al, 2005]. L'approche d'appariement est une combinaison d'appariement sémantique hybride de OWLS-MX [Klusch et al, 2006], d'appariement de graphes, orienté objet du DSD-Matchmaker [Klein et 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 appelé "Type" qui est défini dans une ontologie donnée en lui attachant des méta-informations.

Ces informations permettent de savoir comment apparier T avec un autre type. Les degrés d'appariement sont calculés par l'agrégation des affectations ou « 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 pré-condition et post-condition, sont tous présentés dans la table III.1.

Soient **G** le **G**oal du service (le service requis) et **W** le service **W**SMO (le service offert). Les degrés sont cités, par ordre décroissant de leur valeur d'appariement, allant de l'équivalence à l'échec.

L'équivalence est le plus haut degré,(on la note équivalence).

Ensuite viennent les deux degrés plugin et inverse-plugin qui sont définis différemment par rapport à un état de pré-condition ou de post-condition.

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	≡	Equivalence	$G = W$	
2	⊑	Plugin	$G \subseteq W$	$W \subseteq G$
3	⊒	Inverse-plugin	$G \supseteq W$	$W \supseteq G$
4	⊓	Intersection	$G \cap W \neq \emptyset$	
5	~	Fuzzy similarity	$G \sim W$	
6	∘	Neutral	By derivative specific definition	
7	⊥	Disjunction (fail)	$G \cap W = \emptyset$	

Table III.1. Les degrés d'appariement dans WSMO-MX

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.

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)

TableIII.2. Valuations d'appariement de relations avec les stratégies de défauts

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)

TableIII.3.Valuations d'appariement pour relations de similarité des types

WSMO-MX sélectionne les états de pré-condition et de post-condition 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 pré-conditions, le résultat d'appariement de cet élément est fixé à Equivalent par défaut [Kaufer et Klusch 2006]. Dans ces cas de défauts (absences) d'annotations, WSMO-MX propose d'utiliser des stratégies spécifiques appelées "stratégies de défauts" (voir la table.III. 2).

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 III.2 présente les valuations d'appariement de relations, en tenant compte des stratégies spécifiques aux 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 III.3 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 post-condition. Les vecteurs sont les mêmes en état de pré-condition et en état de post-condition. Pour les autres relations, on a un même vecteur en état de pré-condition et en état de post-condition avec un unique coefficient 1 et des zéros en reste du vecteur à chaque fois.

Le coefficient 1 des vecteurs associés à la relation Equivalent, correspond au degré d'appariement équivalence. Pour les relations Sibling et ComAnc, il correspond au degré intersection. Pour les relations Spouse, ComDes et Relative, il correspond au degré de similarité floue.

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**

SAWSDL-MX (première version) [Klusch et Kapahnke, 2008] accepte en entrée des services spécifiés en SAWSDL [Farrell et Lausen, 2007]. Ce matchmaker est inspiré par les matchmakers OWLS-MX [Klusch et al, 2006] et WSMO-MX [Kaufer et 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 est une extension d'une description WSDL, l'appariement recouvre les éléments de description suivants :

– l'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 nœuds représentant des opérations d'un service,

– des arcs valués dont les valeurs sont calculées à partir des degrés d'appariement des opérations.

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

– basées sur des approches syntaxiques (ou textuelle) : Loss-of-Information, Extended Jaccard, Cosine et Jensen-Shannon. Ces scores appartiennent à l'intervalle [0,1]. En effet, chaque signature sémantique d'une opération de service est transformée en paire de vecteurs de mots-clés pondérés. nous aurons 02 vecteurs : un vecteur pour les entrées et un autre pour les sorties, ces vecteurs sont comparés avec les vecteurs de la requête en utilisant les mesures de similarité citées ci-dessus.

– basées sur une approche logique en utilisant les annotations sémantiques spécifiées par

les attributs modelReference dans les éléments annotés (les entrées et les sorties d'opérations). L'approche logique offre 05 scores (05 filtres) leurs sémantiques sont définies comme suit :

Exact match: une opération offerte Oo s'apparie avec une opération Or avec un degré exact match ssi (∃ une affectation injective *Min*: ∀*m* ∈ *Min*: *m*1 ∈ *in*(O_O) ∧ *m*2 ∈ *in*(O_R) ∧ *m*1 ≡ *m*2) ∧ (∃une affectation injective *Mout*: ∀*m* ∈ *Mout*: *m*1 ∈ *out*($O_{R)}$ ∧ *m*2 ∈ *out*(O_O) ∧ *m*1 ≡ *m*2). Ceci veut dire qu'il y a une correspondance exacte entre toutes les sorties la requête, et les sorties du service, et toutes les entrées du service et les entrées de la requête. On note que les affectations doivent être injectives mais pas forcément bijectives, puisque qu'on peut tolérer des sorties de services superflues, ou des entrées de requête superflues.

Plug-in match: une opération offerte Os s'apparie avec une opération Or avec un degré Plug-in match ssi (∃une affectation injective *Min*: ∀*m* ∈ *Min*: *m*1 ∈ *in*(O_O) ∧ *m*2 ∈ *in*(O_R) ∧ *m*1 ⊇ *m*2) ∧ (∃une affectation injective *Mout*: ∀*m* ∈ *Mout*: *m*1 ∈ *out*(O_R) ∧ *m*2 ∈ *out*(O_O) ∈ *m*2 ∈ *lsc*(*m*1)). Ce filtre allège les contraintes du premier score, en admettant que les entrées de services soient plus générales que celles de la requête, en plus les sorties du service peuvent être des fils directs d celles de la requête

Subsumes match: une opération offerte Oo s'apparie avec une opération Or avec un degré Subsumes match (∃une affectation injective *Min*: ∀*m* ∈ *Min*: *m*1 ∈ *in*(O_O) ∧ *m*2 ∈ *in*(O_R) ∧ *m*1 ⊇ *m*2) ∧ (∃une affectation injective *Mout*: ∀*m* ∈ *Mout*: *m*1 ∈ *out*(O_R) ∧ *m*2 ∈ *out*(O_O) ∈ *m*1 ⊇ *m*2). Ce filtre allège les contraintes du deuxième score, en admettant des sorties de service plus spécialisées que les sorties de requêtes (par opposition au filtre *plug-in* qui accepte juste les fils directs). Et de ce fait *plug-in* est un cas particulier de *subsumes* match.

Subsumed-by match: une opération offerte Oo s'apparie avec une opération Or avec un degré Subsumed by ssi (∃une affectation injective *Min*: ∀*m* ∈ *Min*: *m*1 ∈ *in*(O_O) ∧ *m*2 ∈ *in*(O_R) ∧ *m*1 ⊇ *m*2) ∧ (∃une affectation injective *Mout*: ∀*m* ∈ *Mout*: *m*1 ∈ *out*(O_R) ∧ *m*2 ∈ *out*(O_O) ∈ *m*2 ∈ *lgc*(*m*1)). Le principe de *subsumed-by* est d'accepter des services qui offrent des sorties un peu proches par rapport à celles demandées par l'utilisateur, et en particulier, nous pouvons accepter les pères directs des sorties de la requête.

Fail : Le degré de matching fail est vrai si aucun des scores précédents n'est activé.

Les ensembles lgc(C) and lsc(C) dénotent les subsumants les plus spécifiques de C (parents directs) et les subsumés les plus génériques de C (fils directs).

L'ordre de pertinence de ces filtres est le suivant: exact > plug-in > subsumes > subsumed-by > fail.

- Le Matchmaker SAWSDL-MX combine les deux types d'approches (similarité syntaxique ou textuelle et les filtres logiques), en effet les services ayant le même score logique seront classés avec la similarité textuelle.
- Au moment de l'agrégation des degrés d'appariement des différentes opérations (i.e. le calcul d'appariement global de deux services), l'approche opte pour la plus petite valeur. Ceci est appelé le principe d'agrégation "Valeur minimale" (notée Min).

Par exemple, dans la Figure III.3 l'agrégation des deux degrés Exact et Plug-in, donne lieu au degré d'appariement final Plug-in.

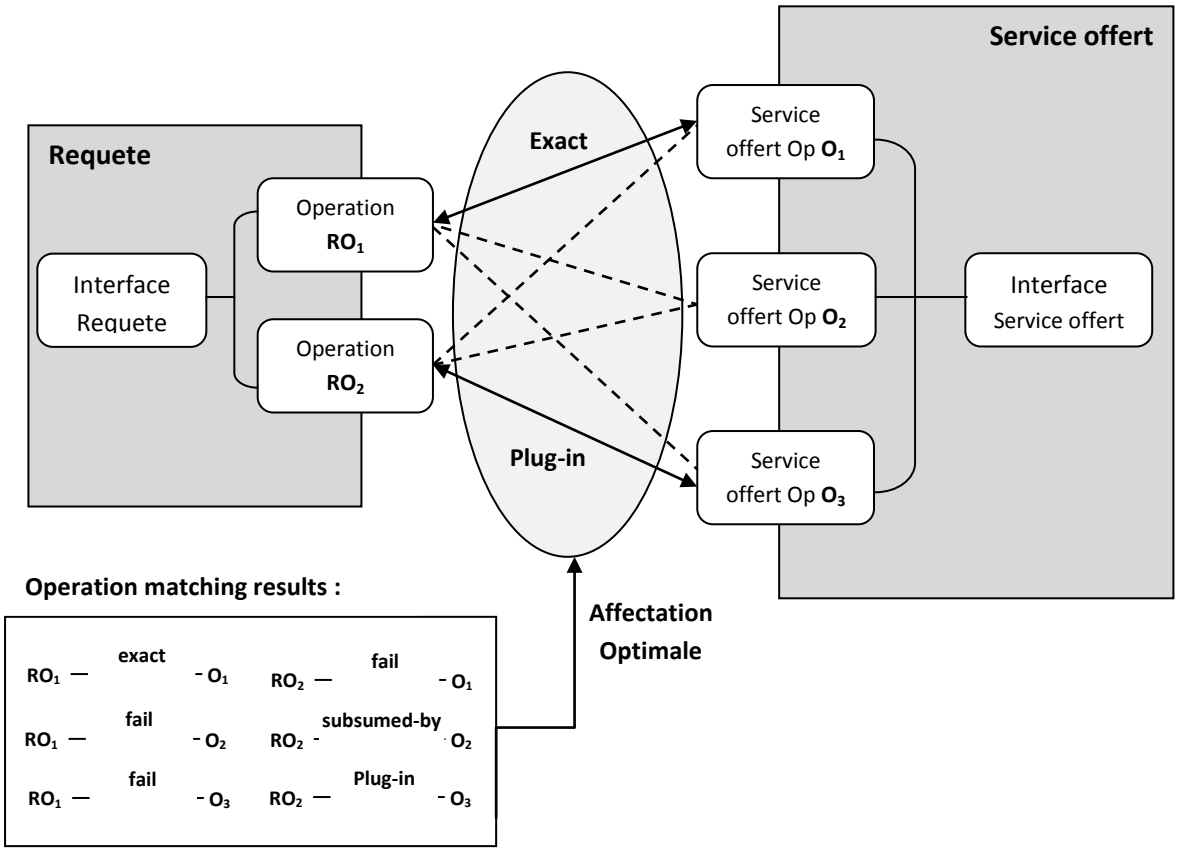


Figure III.3. Exemple d'application du principe d'agrégation de Valeur minimale [Klusch et Kapahnke, 2008]

- Il est important de noter que cette version SAWSDLMX1 se concentre sur les annotations sémantiques de la signature d'opérations et ne prend pas en compte la structure d'arbre du document WSDL.
- Nous notons aussi que la combinaison des filtres logiques et mesures textuelles de SAWSDLMX1 est figée et n'est pas adaptative, en plus le choix du meilleur seuil de la similarité textuelle est fastidieux et dépend de la requête et du service offert.
- Tous ces inconvénients nous poussent à créer un système qui doit être indépendant de la collection de test, en plus il doit être adaptable (cad qu'il est souhaitable d'apprendre la bonne décision à partir des résultats fournis par les méthodes individuelles :logique, similarité textuelle, similarité structurelle).
- Pour gérer ce problème la version SAWSDLMX2 [Klucsh et al, 2009 b] propose un nouveau filtre nommé similarité structurelle, pour comparer les arbres XML, ce filtre est basé sur WSDL-Analyzer présenté dans [Zinnikus et al, 2006], ce dernier calcule la distance d'édition des arbres XML des services à comparer, cet outil vérifie la compatibilité des type d'éléments, en plus du calcul de la similarité lexicale et textuelle (à base de jetons) des éléments.
- La version SAWSDLMX2 applique trois algorithmes de matching pour une paire (requête-service), ces algorithmes sont les filtres logiques, la similarité textuelle, la similarité structurelle, les trois scores servent comme entrées pour entrainer un classificateur de type SVM (support vector machines).
- Ce dernier apprend la décision finale en utilisant des informations de pertinence de la paire (requête – service), et de ce fait SAWSDLMX2 devient plus indépendant de la collection de test et des seuils choisis.
- Les expérimentations montrent que SAWSDLMX2 est plus performant en termes de précision et rappel, que le matchmaker logique, ou les mesures structurelles ou textuelles prises individuellement.
- En plus, il est au moins aussi performant que SAWSDLMX1. Sa précision moyenne est 0.68, et son temps de réponse moyen est 7.9 sec (selon la compétition de sélection de services S3 édition 2009).

- **iMatcher [Kiefer et bernstein, 2008]** est un matchmaker hybride qui calcule la proximité entre les descriptions OwlsProfile, en adoptant d'une part les similarités textuelles, et d'autre part la subsomption logique.

La table III.4 [Mohebbi et al, 2010] montre une comparaison entre un ensemble d'approches de découverte de services. Plusieurs critères sont utilisés dans cette comparaison, ils sont définis comme suit:

IO : prise en compte des entrées - sorties

PE : prise en compte des preconditions et des effets

Propriétés non fonctionnelles : prise en compte de la qualité de service

Autres : prise en compte de propriétés telles que le nombre d'opérations, le nombre de paramètres, leurs types…

Support de dom (degré de matching) : ce critère est vérifié si le matching partiel du service avec la requête est pris en charge.

Matching multi-stage : ce critère est vérifié si on agrège plusieurs sous-scores de matching (de plusieurs descripteurs) en un seul score final.

Exactitude : ce critère est satisfait si on considère plusieurs descripteurs de matching (IOPE), plusieurs étapes de matching, ainsi que la prise en charge des matchings partiels

Support de plusieurs ontologies : ce critère est satisfait si le client et le fournisseur utilisent plusieurs sortes d'ontologies

Support de l'UDDI : ce critère est satisfait si on prend en charge le matching syntaxique

Scalabilté : la scalabilité est assurée si on peut prendre en charge plusieurs ontologies à la fois, ainsi que le matching syntaxique.

Temps de réponse : c'est le temps requis pour traiter une requête.

Performance : elle est mesurée avec le rappel, et la précision, plus ces deux métriques sont bonnes, plus les performances sont meilleures.

Le symbole √ veut dire que le critère est satisfait par l'approche, alors que * veut dire l'inverse.

Catégorie	Approches	Critères											
		Eléments utilisés				*Support Du DoM*	*Matching Multi Stage*	*exactitude*	*Support De +eurs Ontologies*	*Support d'UDDI*	*Scalabilité*	*Temps d'exécution*	*Performance*
		IO	*PE*	*Propriétés Non Fonctionnelles*	*autres*								
Logique	[kifer et al,2004]	√	√	*	*	*	*	faible	√	*	Moyen	Moyen - élevé	Moyen
	[Paolucci et al, 2002]	√	*	*	*	*	*	faible	*	*	Faible		
Non-Logique	[Li et al,2007]	√	*	*	√	√	√	Moyen	√	√	Elevé	Faible	Faible
	URBE [Plebani et Pernici, 2009]	√	*	*	√	√	√	Moyen	*	√	Moyen		
Hybride	WSMO-MX [Kaufer et Klusch, 2006]	√	√	*	√	√	√	Elevé	*	*	Faible		
	OWLS-MX2 [Klusch et al, 2009 a]	√	*	*	*	√	√	Moyen	*	*	Faible		
	iMatcher [Kiefer et al, 2008]	√	*	*	√	√	√	Moyen	*	*	Faible		
	SAWSDL-MX2 Klusch et al, 2009 b]	√	*	*	√	√	√	Moyen	*	*	Faible		
	Lumina [Li et al,2006]	√	*	*	*	*	*	Faible	*	√	Moyen	Elevé	Elevé
	FUSION [Kourtesis et Paraskakis, 2008]	√	*	√	√	*	√	Moyen	*	√	Moyen	Non disponible	Moyen

Table III.4. Comparaison de quelques approches de découverte de services web

III.1.3) Approches Comportementales

Les services atomiques sont généralement composés pour satisfaire les besoins des clients ou des entreprises. Pour rechercher ces compositions (workflows). les méthodes de découverte doivent prendre en charge la notion de comportement des processus, (ou les

cheminements d'exécution), ce comportement doit être modélisé avec des moyens formels tels que, les réseaux de petri, les automates d'états finis, les processus d'algèbre.

Plusieurs modèles à base d'automates COLOMBO [Berardi et al, 05 b] et Romain [Berardi et al, 2005 a] ont été proposés pour la modélisation du comportement, d'autres tentatives [yun et al, 2008] proposent des processus d'algèbre nommés calcul de systèmes communicants (CCS[9]) [Milner ,1982]. D'autres auteurs appliquent la retro ingénierie pour transformer les interfaces de services (exprimés avec BPEL ou WSDL) en spécifications à base de processus d'algèbre (lotos[10] [Eijk et Diaz,1989] pour l'approche [Ferrara, 2004], et CCS pour l'approche [Salaun et al, 2004]).

Ces spécifications seront utilisées, par la suite, pour vérifier des propriétés exprimées en logique temporelle.

Les approches à base d'automates sont présentées avec plus de détail dans la section composition à base de workflow dynamique.

III.2) Approches Non Fonctionnelles (Prenant en Compte la Qualité de Service (QOS))

III.2.1) Approches à Base de Réputation et Confiance

[Ye et al, 2009] se basent sur la réputation comme moyen de découverte de services.

De façon générale, les auteurs adoptent le feed-back des communautés d'utilisateurs pour sélectionner les services.

Ces approches ont plusieurs inconvénients :

- Pas de modèle précis pour les propriétés de sélection de services
- Ils supposent que les valeurs de propriétés de service, sont facilement calculables, et même ces moyens ne sont pas évalués.
- Les fonctions l'agrégation ne sont pas détaillées.

Le système proposé dans [Huang et al, 2006] prend en compte les exigences, et les préférences des utilisateurs de sélectionner les services convenables à l'aide des systèmes flous. Dans [Lo et al, 2010], les auteurs proposent une version floue de la méthode de

[9] Calculus of Communicating Systems

[10] Language of Temporal Ordering Specifications

décision multi objective Topsis pour sélectionner les services, ils considèrent 05 groupes de critères de QOS : les critères transactionnels, de performance (runtime), les critères de coûts, les critères de gestion de configuration, et les critères de sécurité.

Topsis est l'acronyme de «The Technique for Order Preference by Similarity to Ideal Solution » (TOPSIS) [Hwang et Yoon, 1981] elle est considérée comme l'une des méthodes de décision multicritères les plus populaires, pour déterminer la meilleure solution, La méthode TOPSIS calcule la distance la plus courte de la solution idéale positive(PIS) et la distance la plus longue de la solution idéale négative (NIS).

Le PIS est solution fictive qui maximise les critères positifs et minimise les critères de coût, le NIS est solution fictive qui maximise les critères de coût et réduit au minimum les critères positifs. La méthode de TOPSIS est devenue populaire à cause de sa rigueur théorique, et de son adoption du raisonnement rationnel. [Lo et al, 2010].

Dans [Wang et al, 2006] les auteurs supposent qu'il y a Q décideurs qui notent les M services à classer avec les attributs non fonctionnels, pour cela ils, proposent une représentation floue des N critères de QOS.ils proposent aussi un algorithme de 9 étapes qui calcule une matrice de décision floue et pondérée.

[Almulla et al, 2011] proposent système inferentiel flou qui comporte trois étapes

- Fuzzification : des valeurs de Qos selon 03 sous-ensembles flous (faible, medium, élevé),
- Calcul de la matrice des corrélations entre attributs de QOS afin de les pondérer dans la formule de classement final.
- Defuzzification : les auteurs adoptent une version discrète de la méthode de centre de gravité (COG).

Les auteurs estiment que le calcul des distances de corrélation est plus performant que le l'emploi des pondérations à base d'entropie [Xiong et Fan, 2007], en effet, les résultats obtenus par les distances de corrélation sont plus optimistes que celles des méthodes entropiques (i.e. que le score des Top k services obtenu par les distances de corrélations est plus grand que celui de la méthode entropique).

III.2.2) Les Extensions d'UDDI

Certaines approches ajoutent des améliorations au standard UDDI, telles que l'ajout d'un langage de requête structuré (une variante du SQL) , [Dong et al, 2004] proposent l'ajout d'un facilitateur de qualité (broker) qui sépare l'UDDI et l'utilisateur, ce facilitateur gère 03 critères de qualité de service : la fiabilité, la performance, et le coût, et offre 03 scores de matching : gold, silver et bronze.

Le modèle de données de l'uddi est étendu pour prendre en compte la QOS. L'appariement requête-service est fait à travers un calcul d'une corrélation floue.

IV) Conclusion

Nous avons présenté dans ce chapitre les travaux existants sur le problème de découverte, nous avons mis en évidence les avantages et les inconvénients de chaque type d'approches.

Dans le chapitre suivant, nous étudions un problème plus général qui consiste à combiner plusieurs services pour satisfaire les exigences de l'utilisateur.

Chapitre 4 :

La composition des services web

Sommaire

I) Introduction

La composition de services est considérée comme l'une des motivations les plus importantes du paradigme SOA, plusieurs définitions ont été proposées pour le concept de composition, nous citons dans ce qui suit les plus fameuses :

« La composition peut être définie comme le processus de sélection, de combinaison et d'exécution de services en vue d'accomplir un objectif donné". [Martin et al, 2004]. Dans [Benatallah et al, 2005 a], les auteurs considèrent la composition de services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services. Ces auteurs ont défini un cycle de vie englobant six activités [Benatallah et al, 2002 a] (voir la figure IV.1) :

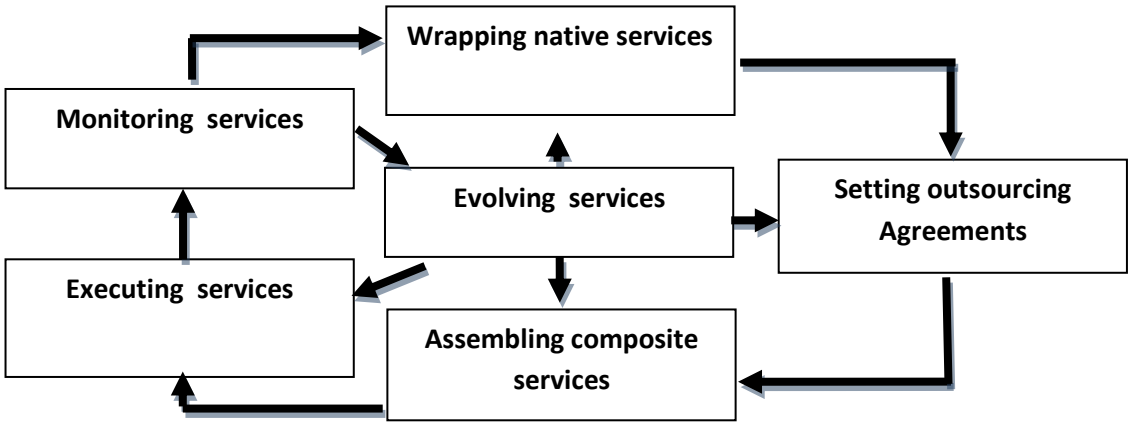


Figure IV.1 Cycle de vie d'une composition de services [benatallah et al, 2002 a]

L'encapsulation de services natifs (Wrapping services).

Cette tâche assure la médiation, i.e. elle garantit que tout service peut être appelé lors d'une composition, (quel que soit le modèle de ses données, le protocole d'interaction,…).

L'établissement d'accord d'externalisation (Setting outsourcing agreements).

Cette tâche consiste à négocier les obligations contractuelles entre les services.

L'assemblage de services composants (Assembling composite services).

Cette tâche permet de décrire, les services à composer de manière abstraite. Cette composition commence par l'identification des services à utiliser et fournit la spécification des échanges inter services selon le protocole partagé.

L'exécution de services composants (Executing services).

Cette activité exécute les spécifications de la composition.

Le contrôle de l'exécution de services composites (Monitoring services).

Cette étape permet la vérification de plusieurs aspects et propriétés, par exemple, elle contrôle l'accès aux services, vérifie les changements de statut, les échanges de messages. Ceci permet la détection des erreurs (violations de règles), de mesurer les performances des services appelés et de prédire des exceptions.

L'évolutivité des services (Evolving services).

Cette étape assure l'évolution de la composition, (i.e. modification des services invoqués, utilisation de nouveaux services, prise en compte des retours de la phase de contrôle).

Dans ce qui suit, nous nous focalisons uniquement sur le mécanisme permettant la combinaison de services pour satisfaire un objectif donné. Plusieurs types d'approches, ont été proposés dans la littérature pour répondre au problème de composition des services. De manière générale nous distinguons les méthodes basées sur les workflows, la planification en intelligence artificielle, les méta-heuristiques, l'hybridation planification-meta-heuristiques. Certaines classes peuvent contenir plusieurs sous classes (telles que les workflows dynamiques).

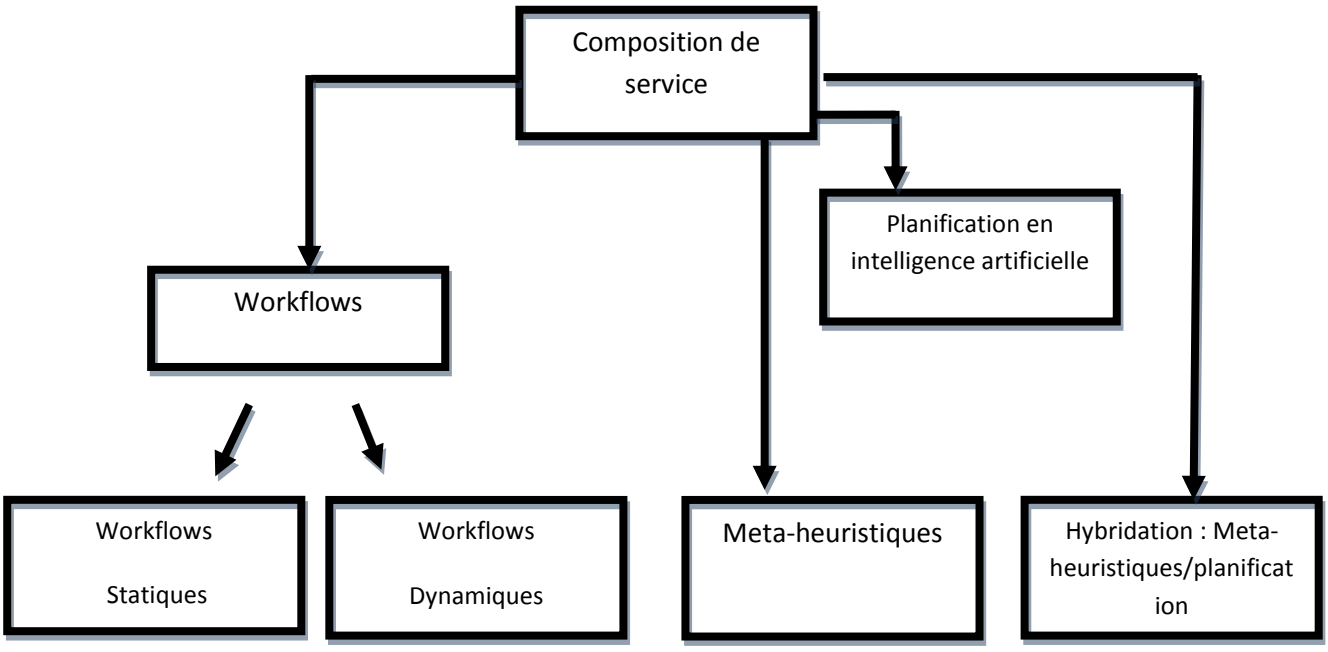


Figure IV.2. Classification des approches de composition

II) Etat de l'art

II.1) Planification en Intelligence Artificielle

La planification en I.A [Russell et Norvig, 2009] est le processus qui crée une suite d'actions qui mènent à un état final à partir d'un état initial. Le plan peut être vu comme une représentation de l'organisation des actions dans l'espace et le temps.

Le problème de planification peut être défini comme un cinq-uplet (S,A,R,s0,G), avec :

- S : est l'ensemble des états possibles du système considéré, s0 est l'état initial du système
- G est l'état final que l'on souhaite obtenir (l'état but). s0 et G appartiennent à S.
- A : est l'ensemble des actions possibles qui permettent le changement d'états.
- R est l'ensemble des transitions d'état possibles ; R est donc inclus dans S×A×S et représente les pré-conditions et de post-conditions de chaque action, les post conditions sont souvent appelées effets. Les pré-conditions doivent être réalisées pour que l'action puisse être appliquée et les post-conditions sont forcément réalisées après une action.

Résoudre le problème de planification revient à trouver un enchaînement d'actions permettant de relier l'état s0 à l'état G

Dans le contexte des services web, la planification et vue comme un six-uplet(W,S,A,R,s0,G) la nouvelle dimension W représente les services web, chaque service web est une collection d'actions.

Dans la suite nous présentons des méthodes de composition qui se basent sur la planification.

- **Calcul Situationnel**

[McIlraith et Son, 2002] étendent le langage Golog pour la composition de services Web. Golog est un langage de programmation logique basé sur le calcul de situations. Ce dernier consiste à modéliser l'état du monde comme un arbre de situations, en partant d'une situation initiale E0 et évoluant vers de nouvelles situations par l'application de différentes actions .

La construction du plan se fait en appliquant une recherche en largeur d'abord dans l'espace d'états. Les auteurs considèrent deux types de services : les services primitifs et les

services complexes. Les services primitifs sont divisés en deux types : ceux qui altèrent l'environnement « world-altering services» et ceux qui produisent juste des informations « information-gathering services », les services complexes sont des compositions de services individuels (primitifs).

La requête de l'utilisateur est constituée d'une procédure générique, ainsi que les contraintes (par exemple des choix non déterministes au niveau des situations) ces entrées sont modélisées avec le calcul de situations. Durant la composition, les agents planificateurs emploient des pré-conditions et des post-conditions (effets) des actions en plus des constructeurs procéduraux et les contraintes de l'utilisateur.

L'approche ne prend pas en compte la concurrence et les choix conditionnels.

- **Planning Domain Definition Language (PDDL)**

[Ghallab et al, 1998] est un langage de formalisation des connaissances de planification, il est considéré comme un standard de facto pour les planificateurs, ce fait a motivé la communauté de chercheurs en services web à l'adapter au contexte de composition de services. Dans cette optique [MCDermott, 2002] introduit un nouveau type de connaissance pour chaque action appelée « *value of an action* », Cette connaissance persiste et n'est pas traitée comme une proposition ordinaire (qui peut être retirée). Plus précisément elle permet de distinguer la transformation ou l'acquisition des informations d'une part et le changement d'état d'autre part, qui sont produits par l'exécution d'un service web. L'information représentée par les paramètres d'entrées/sorties, est supposée être réutilisable. Ainsi, les valeurs des données fournies par les services web peuvent être dupliquées pour l'exécution de plusieurs services Web.

- **Planification à Base de règles (Rule-Based Planning)**

[Medjahed et al, 2003] proposent une approche de composition à base de règles, en d'autres mots la méthode utilise des règles de composabilité pour déterminer si deux services sont composables. Pour cela, ils étendent le langage WSDL pour annoter les services. La méthode comporte quatre phases : La première, est une phase de spécification, elle permet un haut niveau de description de la composition décrite en utilisant un langage de spécification de services composés appelé CSSL (Composite Service Spécification Language). La seconde, la phase d'appariement (matching), utilise des règles de composabilité pour générer des plans

de composition qui sont conformes aux spécifications de la requête de l'utilisateur. La troisième est la phase de sélection. Si nous avons plusieurs plans au niveau de la deuxième phase, l'utilisateur du service choisit un plan en fonction de différents paramètres tels qu'un paramètre de qualité de composition (tel que le coût). La phase finale permet la génération d'un diagramme d'activité UML associé à l'orchestration.

Les auteurs définissent plusieurs ensembles de règles de composabilité. Nous avons les règles de composabilité syntaxique et les règles de composabilité sémantique.

Les règles syntaxiques incluent :

- La vérification de la compatibilité des modes de fonctionnement, cad est ce que les deux services ont le même mode d'envoi de message (one-way) ou (request-response).
- La vérification de la compatibilité des protocoles de transport des services Web.
- La vérification de nombre de paramètres

Les règles sémantiques incluent les sous-ensembles suivants :

• la sémantique statique (du message et de l'opération): deux services Web sont composables si et seulement si le message de sortie d'un service est compatible avec le message d'entrée de l'autre service, en termes de type de données, d'unité, de langage, de catégorie…

• la sémantique dynamique (de l'opération) : elle vérifie la compatibilité entre les pré-conditions et les post-conditions de deux opérations en termes d'implication et d'équivalence, les auteurs définissent 05 scores.

• la qualité de service : ces règles vérifient la compatibilité des valeurs de critères de QOS de deux services (les attributs de sécurité tels que l'authentification, les attributs métiers tels que la réputation, et les attributs de performance tels que le temps d'exécution, la disponibilité…).

• la pertinence de la composition : elle vérifie si une composition de services crée une valeur ajoutée, pour cela les auteurs comparent la composition avec des prototypes stockés auparavant (les prototypes sont des graphes dont les experts ont approuvé leur pertinence).

Nous notons que la contribution majeure de cette approche est la définition des règles de composabilité, ces règles couvrent plusieurs types d'aspects de composition : la syntaxe la sémantique statique, la sémantique dynamique, la qualité de service, et la pertinence de la composition.

SWORD [Ponnekanti et Fox, 2002] est un système d'élaboration de services Web composés à base de règles. Contrairement à [Medjhed et al, 2003], SWORD utilise le modèle entité-relation (ER) [Gardarin, 1993] à la place des formalismes traditionnels des services web (WSDL, et OWLS). SWORD, modélise les services web avec des pré-conditions et des post-conditions. Les pré-conditions et les post-conditions emploient les entités et les relations comme prédicats, un service Web est représenté sous la forme de clause de Horn. Une clause de Horn est une disjonction de littéraux qui a au plus un littéral positif. (Une implication qui a au plus une conclusion).

Clause de horn : ¬a ∨ ¬b ∨.... ∨ ¬c ∨ h

L'utilisateur spécifie l'état initial et l'état final de la composition, ensuite le système expert génère le plan.

Il est utile de noter que les compositions générées par SWORD peuvent donner des résultats non déterministes, cad une même entrée qui provoque plusieurs résultats .ceci est dû à la non dépendance fonctionnelle entre les pré-conditions et les post conditions. Pour éviter ce problème, il suffit de garantir la relation de dépendance fonctionnelle.

[Galves de olivera et parente de olivera, 2010] proposent un planificateur sémantique qui emploie une recherche régressive dans l'espace d'états.

Les auteurs prennent en compte 04 critères de QOS (réputation, temps d'exécution, disponibilité, prix) mais ne gèrent pas les contraintes globales au niveau des expérimentations.

Un service est ajouté dans un plan candidat, si la relation de subsomption est vérifiée entre leurs concepts. Les auteurs considèrent plusieurs heuristiques pour élaguer l'arbre de recherche (et ordonner la liste des plans candidats), nous avons entre autres : le nombre de concepts non satisfaits, le score de la QOS globale, la taille du plan, le nombre concepts, le nombre de concepts éliminés (satisfaits), le nombre de concepts fournis comme sorties.

Les expérimentations montrent qu'il y a une faible augmentation de temps (50 msec), lorsqu'on prend la QOS en plus de la sémantique, en outre la valeur de la fonction fitness varie entre 0.6 et 0.75 pour un benchmark contenant 17 requêtes et un nombre de services variant entre 100 et 10000.

[Wu et al, 2011] Comparent quatre algorithmes de planification, en termes de temps d'exécution, et de nombre de plans trouvés, selon les expérimentations, forward A* et forward A sont plus mauvais en termes de temps d'exécutions que backward A* et backward A (ils ne sont pas scalables), en plus back ward A est plus efficace que back ward A* surtout pour les corpus ayant une grande taille.

En ce qui concerne le nombre de solutions, back ward A est plus performant que back ward A*.

Dans [Wu et al, 2003], [Sirin et al, 2004] le planificateur SHOP2 [Nau et al, 2003] est utilisé pour la composition automatique de services web qui sont formalisés avec DAML-S. SHOP2 (simple hierarchical ordered planner) est un planificateur de type réseau de tâches hiérarchiques (ou Hierarchical Task Network).

« Hierarchical Task Network ou HTN » est une approche qui crée des plans par décomposition de tâches, en plus des éléments standards de la planification (cad l'état initial, l'état final, les opérateurs), HTN ajoute la notion de méthodes et de réseau initial de tâches. Chaque méthode possède des contraintes qui spécifient les conditions de son applicabilité.

HTN fait une décomposition récursive du réseau de tâches initial, et en respectant les contraintes, jusqu'à l'obtention de tâches primitives (les opérateurs),

Selon [Wu et al, 2003], HTN est mieux adapté aux compositions de services, car il est scalable (i.e. il peut traiter des problèmes volumineux en termes de taille grâce à l'élagage basé sur les méthodes) en plus il est plus efficace que Golog [McIlraith et Son, 2002].

[Sirin et al, 2003] présentent une méthode semi-automatique pour la composition. Les auteurs considérent uniquement les entrées-sorties des services et la qualité de service (pas de preconditions ou d'effets), l'approche utilise les scores Exact, PlugIn, Subsume et Disjoint, pour évaluer la proximité sémantique des concepts d'entrées-sorties. L'approche supporte les flux de contrôle séquentiels, conditionnels, et concurrents.

La composition semi-automatique est parfois très prometteuse par rapport aux approches automatiques, tout d'abord, elle renforce l'exactitude et la fiabilité des plans générés, en plus elle capture les besoins réels de l'utilisateur.

- **Méthodes Basées sur la Preuve Automatique.**

[Waldinger, 2001] a développé un système qui génère les services web composés à partir de preuves de théorèmes. L'approche est basée sur la déduction automatique de preuves. La requête de l'utilisateur est décrite comme une formule devant être prouvée (théorème). Initialement, les services web candidats et les besoins de l'utilisateur sont décrits dans la logique du premier ordre en forme d'implications ou d'équivalences, ensuite, une preuve est générée par l'outil de preuve de théorème SNARK. Enfin, la description de la composition de services est extraite de la preuve.

[Lämmermann, 2002] propose une synthèse structurelle de programme (SSP, Structural Synthesis of Programme) pour la composition de services. SSP est une approche déductive pour la synthèse de programme à partir de spécifications. Il définit un langage logique dans lequel les spécifications sont définies par des interfaces. Les interfaces représentent les services web ou la requête. Ces interfaces sont constituées de variables typées, de constantes, de liaisons entre les variables et d'un axiome. L'axiome comporte uniquement les propriétés structurelles, i.e. les informations des entrées/sorties, l'auteur étend son langage en y ajoutant des "variables de contrôle" qui permettent l'expression des pré-/post-conditions. Les interfaces des services web sont considérées comme des hypothèses et l'interface de la requête est considérée comme une formule à prouver. SSP infère à partir des interfaces prédéfinies une "preuve" de l'interface qui représente la requête, et construit une composition des services Web à partir de la preuve.

Lämmermann considère les services web composés comme des programmes et utilise le fait que les programmes sont des preuves.

[Rao et al, 2003], [Rao, 2004], [Rao et al, 2006] utilisent les preuves de théorème en logique linéaire pour la composition automatique de services web sémantiques (linear logic theorem proving). Cette méthode utilise deux représentations pour les services Web. La première est une représentation externe utilisant le langage OWLS. La seconde représente les services Web par des axiomes extra-logiques dans la logique linéaire, (de manière similaire

aux travaux de [Waldinger, 2001] qui adopte la logique classique). La logique linéaire est considérée comme une logique de ressources plus qu'une logique de vérité, elle considère deux types de formules : consommables (elles sont utilisables une seule fois dans une branche de démonstration) et non consommables (elles sont précédées par le constructeur ! et peuvent être utilisée plusieurs fois dans une branche de démonstration). Les formules non consommables permettent de représenter les entrées et les sorties des services, par contre les formules consommables permettent la représentation des pré-conditions et des post-conditions. La logique linéaire permet aussi de formaliser les attributs des services Web, en incluant la qualification et la quantification des valeurs des attributs non-fonctionnels.

De façon générale, un service atomique est représenté par un séquent axiome : a|-b

Tel que 'a' représente les informations de QOS qui seront consommées lors de l'utilisation du service, et 'b' représente une implication linéaire modélisant les entrées les sorties, les pré-conditions, et les post conditions du service.

Les auteurs considèrent aussi des axiomes additionnels pour représenter la similarité sémantique entre les entrées-sorties, cette similarité peut avoir 03 valeurs : exact, plugin et disjoint.

La logique linéaire possède des fondements dans le pi-calcul. Le pi-calcul est la fondation formelle de certains langages de composition de services Web [Cámara et al, 2006] , [Lucchi et Mazzara, 2007]. La possibilité de considérer la preuve de logique linéaire comme un processus de pi-calcul a été pour la première fois étudiée par [Abramski, 1994], et ensuite étalée par [Bellin et Scott, 1994].

Contrairement au reste d'approches [Sirin et al, 2003] et [medjahed et al, 2004] qui utilisent les attributs non-fonctionnels seulement pour filtrer les plans générés, [Rao et al, 2004] Considèrent les attributs non-fonctionnels pendant le processus de preuve de théorème, i.e. durant la génération de plans. Les fonctionnalités des services et les attributs non-fonctionnels sont traduits par des axiomes logiques. La distinction entre les fonctionnalités et les attributs non-fonctionnels est permise par les séquents de la logique linéaire.

Il est utile de noter que, la recherche de preuve dans la logique linéaire est indécidable [Rao et al, 2003]. En d'autres mots, il existe des situations, pour lesquelles on ne peut trouver une démonstration, soit parce que l'espace de recherche est grand, auquel cas on arrête le

processus de recherche, soit parce que la preuve n'existe pas, auquel cas les méthodes de preuve automatique ne pourront pas s'arrêter pour confirmer l'absence de preuve.

II.2) Composition à Base de Workflows

Les méthodes à base de workflows modélisent les compositions avec des graphes, elles peuvent être statiques ou dynamiques. Dans le cas statique, l'utilisateur établit manuellement un modèle abstrait des tâches, en plus des dépendances. (i.e. le flux de contrôle, le flux de données…). Chaque tâche contient une clause ou question qui est employée pour rechercher le service concret qui l'accomplit. Ce travail est fait pendant la conception. Durant l'exécution, le système de composition instancie chaque nœud (ou tâche), en cherchant un service concret.

Par contre les méthodes à base de workflows dynamiques génèrent les graphes de tâches instanciées à la volée (pendant le runtime).Dans ce qui suit nous détaillons les deux types d'approches.

II.2.1) Composition à Base de Workflows Statiques

la plateforme Eflow [Casati et al, 2000] permet de modéliser manuellement le workflow (par un graphe), les nœuds représentent des invocations de services (simples ou composites) ou bien des décisions de contrôle de flux (if,join…), ou bien des événements à envoyer ou à recevoir.

Eflow spécifie aussi des régions transactionnelles du graphe cad celles qui doivent être exécutées de manière atomique. Les arcs représentent les dépendances, l'instanciation est faite en utilisant des caractéristiques du fournisseur (mais sans prise en charge de la QOS).

Meteor-s [Aggarwal et al, 2004] est un projet créé par le laboratoire LSDIS (large scale distributed information system) de l'université de Georgia USA, il permet de réaliser la publication, la découverte et la composition de services web sémantiques, meteors comporte deux parties : front end et back-end.

Le front end permet l'annotation et la publication de services.

Le back end permet la découverte, la composition et la sélection. Il comporte plusieurs modules : « abstract process designer » ou concepteur de processus abstrait, « discovery

engine » moteur de découverte, « constraint analyser » l'analyseur de contraintes, et « binder » le lieur.

Le premier module crée des workflows abstraits à l'aide du langage bpel. Les nœuds représentent des patrons non instanciés, chacun d'eux est représenté avec des concepts d'entrée, de sorties, la catégorie sémantique de l'opération, les preconditions, les post conditions les valeurs permises de chaque critère de qualité de service. On note que ces informations sémantiques vont enrichir les documents WSDL.

Le deuxième module effectue des recherches sur l'uddi en utilisant les entrées, les sorties, les catégories de l'opération, les pre-conditions, les post-conditions (annotées avec des ontologies), afin de satisfaire les besoins du workflow abstrait.

Le troisième module sélectionne les services finaux à partir des résultats générés par le deuxième module, en utilisant la qualité de service imposée dans le workflow abstrait et les contraintes métiers (telles que les préférences sur les fournisseurs).

Le module binder instancie le workflow abstrait avec les résultats du module 03 et crée un document BPEL exécutable.

L'architecture dirigée par les modèles [Kleppe et al, 2003], est une approche qui utilise les modèles pour l'ingénierie de logiciel, l'idée principale est de séparer la conception (construction de modèles) de l'architecture.

Dans certains cas, on est capable de générer le code exécutable (partiellement ou totalement) à partir d'un modèle, nous notons aussi que l'UML [Fowler, 2003] est devenu le standard de facto de l'ingénierie dirigée par les modèles.

[Orriens et al, 2003] proposent une approche pour la construction de compositions dynamiques, pour cela ils adoptent UML comme langage de modélisation de compositions, et proposent des règles de transformation vers le langage BPEL.

[Grønmo, Solheim, 2004] transforment les documents WSDL en diagrammes UML, et par la suite, ils convertissent les compositions en UML en spécifications BPEL, et proposent un outil open source qui réalise ces tâches.

[Gannod et al, 2006] construisent des documents OWLS en exécutant deux étapes, la première étape génère une description OWL-S à partir d'un diagramme d'activité UML.

La deuxième étape utilise les constructeurs du service profile et process de OWLS et un ensemble de documents WSDL, afin de générer la partie grounding de OWL-S. ce travail est étendu dans [Gannod et Timm, 2007] afin de prendre en compte les constructeurs de contrôles structurés avec Object Constraint Language (OCL) [OMG, 2007]

II.2.2) Composition à Base de Workflows Dynamiques

La composition à base de workflows dynamiques crée le modèle de tâches et l'instanciation des services de façon automatique. [Rao et Su, 2004].

Cette composition est qualifiée comme étant orientée processus ou comportement, tous les services web considérés possèdent des états (statefull web services).[lecue, 2008]

Contrairement aux services web sans états (stateless web services), dont le schéma d'interaction est trop simplifié (i.e. de type requête réponse), les services web avec états ont une interaction complexe, et les résultats finaux ne sont pas complètement prévisibles (non déterministes).

Cette interaction est généralement modélisée avec des systèmes formels, tels que les automates d'états finis, les réseaux de petri…,).[lecue, 2008]

L'interaction d'un service avec état représente un protocole multi-phases, qui peut englober des actions internes, des échanges de messages avec les partenaires, des accès aux bases de données….

L'intérêt des méthodes formelles (systèmes formels) est de donner la possibilité de vérifier automatiquement des propriétés d'un système (code logiciel, modèle conceptuel…). Ils peuvent être adoptés pour générer les workflows dynamiques. Ils peuvent aussi vérifier et prouver la compatibilité de la composition avec les besoins de l'utilisateur.

- **Composition à Base d'Automates**

Plusieurs modèles basés sur les automates d'états finis, ont été proposé pour la composition de services avec états, nous avons Les modèles COLOMBO [Berardi et al, 2005 b] et Romain [Berardi et al, 2005 a], [Berardi et al, 2006], le modèle « mealy machines » et

« moore machines » [Bultan et al, 2003], [Lustig et Vardi, 2009]. Nous distinguons des automates, qui peuvent effectuer uniquement des actions de communication [Pistore et al, 2005 c], d'exécuter uniquement des actions internes [Berardi et al, 2006] ou d'exécuter les deux types d'actions [Berardi et al, 2005 b]. Parfois, ces automates sont enrichis par l'ajout de conditions et d'effets sur les transitions.

Les modèles « mealy machines » et « moore machines » [Bultan et al, 2003], [Lustig et Vardi, 2009] [pistore et al, 2005 c] considèrent uniquement des opérations de communications (envoi et réception de messages). Il est utile de noter que le modèle « mealy machine » prend en compte la communication asynchrone, i.e. la prise en compte des files d'attentes pour stocker les messages d'entrées.

a- Modèle Romain

Le modèle romain adopte les hypothèses suivantes :

- Chaque service web du corpus est modélisé avec un automate d'état fini, dont les arcs sont des appels aux opérations du service, et les états sont des données qui représentent le métier du service.
- La requête Pt est aussi modélisée sous forme d'un automate d'état fini,
- La résolution du problème revient à créer un médiateur (ou contrôleur ou délégateur) [Berardi et al, 2003], [Berardi et al, 2005 a], [Muscholl et al, 2007], [Ragab et al, 2008] qui délègue toutes les tâches (ou arcs) de la requête aux opérations associées aux services du corpus. En d'autres termes nous cherchons à générer le comportement de l'automate cible, en utilisant les opérations des automates du corpus. Le corpus contient n services (ou automates) P1, ..., Pn. Pour ce faire les chercheurs adoptent soit :
    - Les relations de simulation ou bisimulation :
        - La relation de simulation [Musholl et al, 2007] (notée $\leq$) ou de bi-simulation (la simulation dans les deux sens) entre Pt et un sous ensemble de services du corpus.
        - Pi $\lesssim$ Pj veut dire que la trace générée par Pi est incluse dans la trace générée par Pj.
        - Le délégateur de Pt existe si et seulement si (Pt $\lesssim$ P1x ..x Pn)
        - Pi x Pj représente le produit asynchrone des automates.

- La vérification des relations de simulation est Exp-time complete.
- Il est utile de noter que le nombre d'instances des automates du corpus peut être borné ou infini.

- La satisfiabilité d'une formule logique qui encode l'existence d'un médiateur pour la requête Pt [Berardi et al, 2003 ; Berardi et al, 2005 a], cette formule est codée avec une logique propositionnelle dynamique, la complexité de ce problème de satisfiabilité est 2Exp-time.
- La théorie des jeux
- Les langages formels.

La figure suivante montre une relation de simulation entre le processus cible P et les composants p1,p2

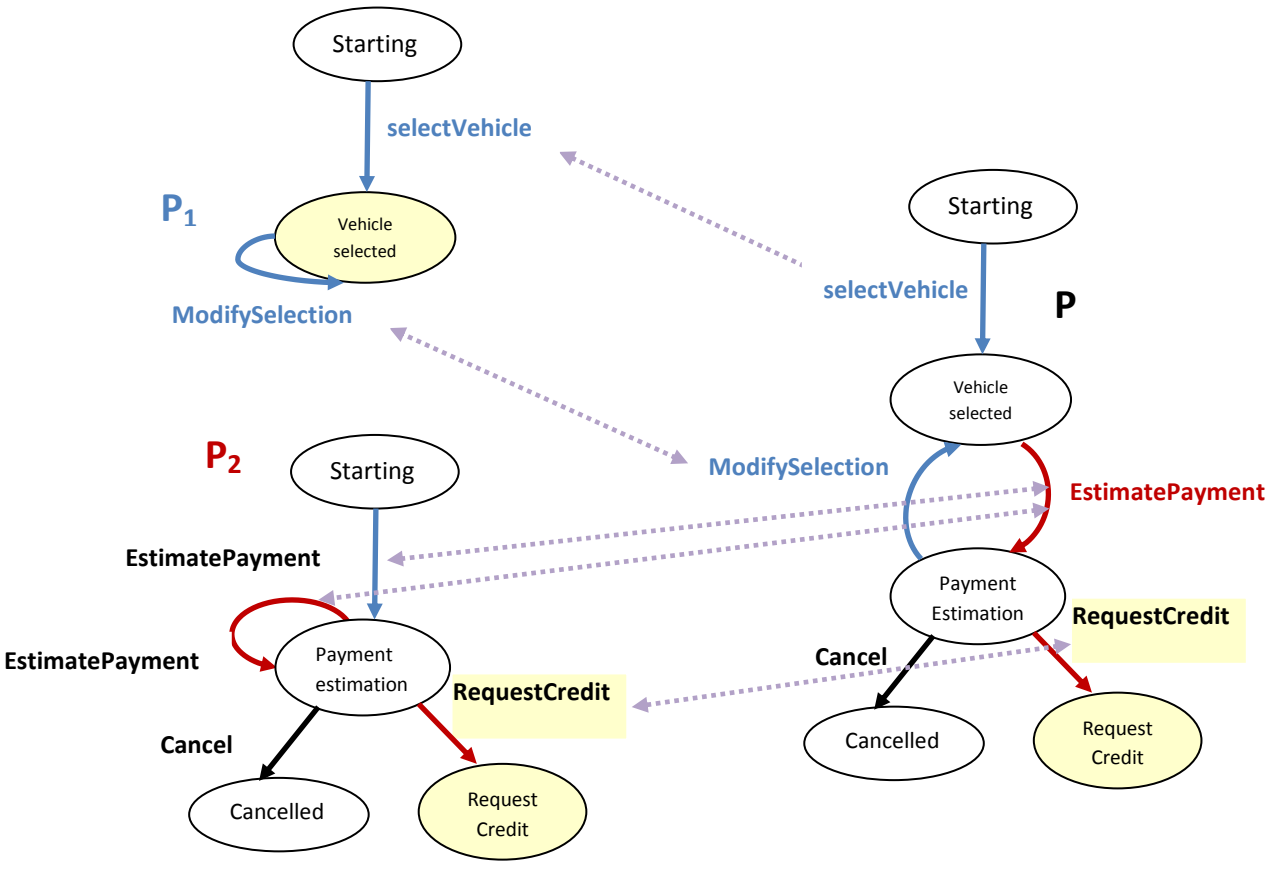


Figure IV.3. Relation de simulation entre P et (p1,p2)

- L'approche Polymorphique Process Model PPM [Schuster et al, 2000] crée des workflows statiques et dynamiques, les workflows statiques sont générées de façons similaire à Eflow [Casati et al, 2000], par contre les workflows dynamiques, sont générés à base de raisonnement sur les automates (state machines). Dans ce cas chaque service web est représenté avec un automate d'état fini.

b-Modèle Colombo

Il est similaire au modèle romain sauf qu'il suppose que les arcs des automates sont conditionnés par l'état de la base de données du service en question, ceci implique qu'une transition n'est pas toujours franchissable, et qu'elle doit vérifier la condition portant sur la base de données. L'approche [berardi et al, 2005 b] vérifie l'existence d'un contrôleur (médiateur) pour le modèle de colombo en testant la satisfiabilité d'une formule codée en logique propositionnelle dynamique.

c-Modèle Machines de Mealy « Mealy Machines »

Dans [bultan et al, 2003], [Lustig et Vardi, 2009] les auteurs adoptent le modèle « mealy machines », chaque service web est considéré comme un automate asynchrone de type « mealy machine », ces services échangent des messages selon une topologie de communication prédéfinie nommée canaux de communication, (le nombre de messages échangés est fini).Chaque automate possède une file d'attente pouvant contenir des messages

Plus formellement, l'infrastructure de la composition est un triplet (P,C,M) avec :

P :un ensemble fini de pairs

C : un ensemble fini de canaux de communications

M : un ensemble fini de messages

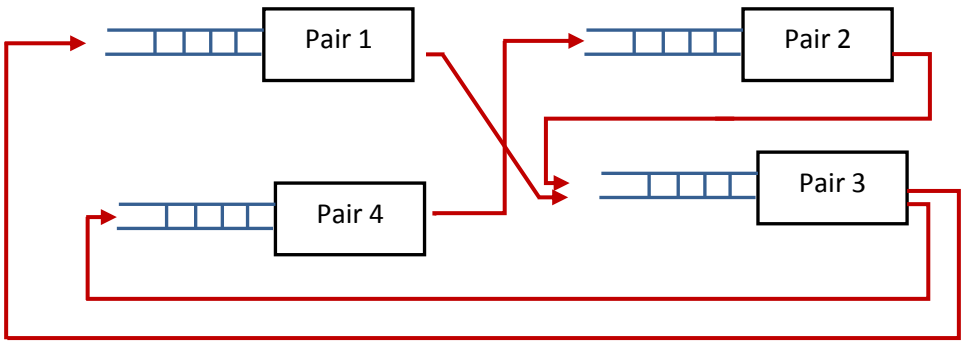


Figure IV.4. Le modèle « mealy machines »

Selon cette approche le problème de composition accepte comme entrée :

Une infrastructure (P,C,M)

Un comportement global désiré (noté L(S)), i.e. un ensemble de séquences de messages que doit produire l'ensemble de pairs,

Le résultat de la synthèse est la spécification (l'implémentation) de « mealy machines » associées aux services de P de telle sorte que le langage produit par la collaboration des pairs C(I) est équivalent au comportement global désiré L(S).

d-Modèle de Machines de Moore

Ce sont des automates qui permettent de produire et de recevoir des messages de manière synchrone (voir la figure IV.5), le symbole !x représente la production d'un message, alors que le symbole ?x représente la réception d'un message.

Deux types compositions peuvent être envisagés en utilisant les machines de moore, [pistore et al, 2005 b] La composition à base de flux de données, et la composition à base de flux de control.

Dans les deux cas nous essayons, de trouver un sous ensemble de machines appartenant à la collection de services, qui produisent de manière collaborative une conversation désirée (une séquence de messages). La conversation est généralement modélisée avec une formule logique.

Dans le cas d'une composition à base de flux de données, les machines sont lancées lors de la réception des événements (messages), et peuvent lancer d'autres machines par la suite, nous notons que ce type de problème est indécidable.

Dans le cas d'une composition à base de flux de contrôle, les machines ne peuvent être lancées que lorsque la tâche ou le service précédent est complètement achevé. Ce type de problème est décidable, et sa complexité est 2exptime complete.

Dans [pistore et al, 2005 a], [pistore et al, 2005 b] les services web sont modélisés comme des automates d'états finis communicants et non déterministes et partiellement observables, i.e. que nous ne pouvons pas connaitre toutes les informations d'un état donné.

Leur proposition consiste à créer un contrôleur qui coordonne les services « composants » de telle sorte que les besoins de l'utilisateur sont satisfaits (en termes de formule logique temporelle).

Les services web composants sont générés à partir des spécifications BPEL ou OWLS.

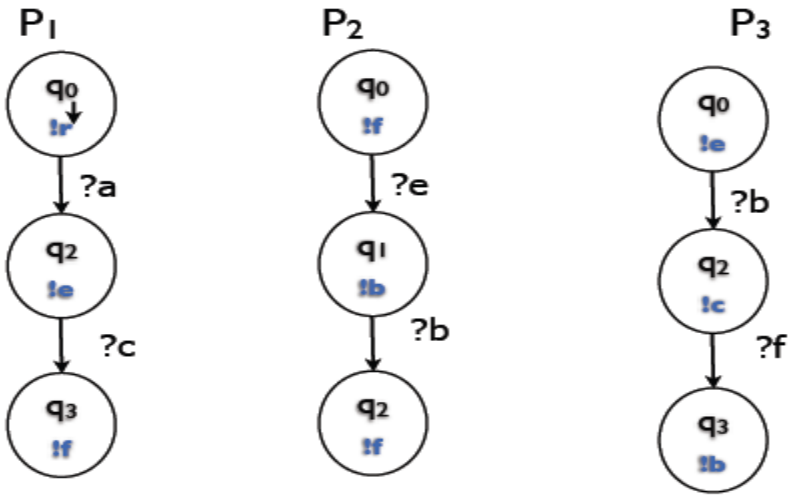


Figure IV.5. Le modèle de machines de moore

Modéles			References
Service Avec etat	Activités abstraites	Automate d'état fini (AEF)	[Berardi 2003, Berardi 2005a, Muscholl 2007, Ragab 2008]
	données	AEF+Données	[Berardi,2005b]
	I/O	Machines de Mealy Machines de Moore	[Bultan et al, 2003, Lustig et Vardi, 2009,Pistore et al, 2005 b]

Table IV.1. Principales méthodes de composition basées sur les automates

- **Approches Orientées Théorie de Graphes**

[Liang et su, 2005] construisent dynamiquement des graphes and/or à partir d'une requête et un ensemble de composants abstraits. Ce travail est répété tant que le développeur n'a pas accepté le résultat. Ensuite ils instancient les nœuds (composants abstraits) avec des services réels, et génèrent une spécification exécutable [Leymann, 2001].

Hashemian and Mavaddat [Hashemian et Mavaddat, 2005] utilisent les automates et la théorie de graphe pour la composition de services avec états, en particulier ils modélisent les services web avec des automates, les entrées et les sorties forment les nœuds et les arcs représentent les invocations d'opérations, dans une première étape les auteurs détectent les services qui peuvent participer dans la composition.

Et dans un second lieu, la dépendance entre les services est établie. Cette approche est aussi étendue pour les services sans états.[Hashemian et Mavaddat, 2006].

[Gekas et Fasli, 2005] créent un annuaire de composition qui contient un hyper graphe (tous les services pouvant être composés), ce dernier est exploité comme espace de recherche afin de dériver un sous graphe qui répond à la requête.

Les auteurs analysent la structure de cet hypergraphe, afin de générer des heuristiques qui peuvent guider le processus de composition.

- **Autres Approches**

- [Hamadi et Benatallah, 2004] adoptent une algèbre de processus basée sur les réseaux de petri pour modéliser le flux de contrôle et la sémantique des compositions de services, le système propose plusieurs structures de contrôle (séquence, choix, boucles…), mais les auteurs ne proposent pas d'approches de composition manuelle ou semi-automatique.
- Les réseaux de Petri sont également utilisés pour représenter formellement les services [Hammadi et Benatallah, 2003]. Par ailleurs, les réseaux de Petri permettent aussi la description des conditions et des effets pour les transitions.

- [Narayanan et McIlraith, 2002] traduisent les processus OWL-S (la partie model), en un réseau Petri en vue de raisonner sur les propriétés de la composition. L'outil proposé vérifie les propriétés de la composition produite, en particulier il vérifie la consistance du graphe d'interaction, l'accessibilité des états, le respect des contraintes de l'utilisateur, l'existence des impasses, etc.
- [Ouyang et al, 2005] traduisent les documents BPEL en réseaux Petri annotés. Cette traduction permet le raisonnement sur la structure du graphe d'interaction, ainsi que ses propriétés. Les auteurs affirment que la traduction du flux BPEL peut mener à des ambigüités.
- Dans la même optique [Yi et Kochut, 2004], proposent la traduction d'une composition BPEL en réseaux Petri. Les auteurs proposent un système qui traduit les

processus BPEL en réseaux Petri, ensuite ils vérifient les propriétés du réseau de manière formelle.

II.3) Approches à Base de Metaheuristiques

Selon [Dreo et al, 2003] une metaheuristique est un algorithme d'optimisation utilisant des heuristiques, réutilisables pour une large gamme de problèmes d'optimisation.

[Bleul et al, 2007] Proposent plusieurs algorithmes de recherche sémantique (aveugles et heuristiques) des compositions de services sans états,

Ces algorithmes vérifient la compatibilité sémantique du flux de données reliant les constituants de la composition. Plus précisément, les algorithmes s'assurent que le concept d'entrée de chaque service subsume le concept de sortie du service qui le précède.

Dans ce qui suit nous citons les algorithmes proposés par les chercheurs :

- La recherche en profondeur d'abord (bornée par une limite de la profondeur)
- La recherche en profondeur itérative (une version qui fait une recherche en avant et une autre qui fait une recherche en arrière)
- La recherche heuristique gourmande (gloutonne) : l'auteur propose une heuristique complexe qui englobe plusieurs critères tels que : le nombre de concepts non satisfaits, le nombre de concepts satisfaits, le nombre de concepts produits en sortie, la taille de la composition.
- L'algorithme génétique : il utilise deux fonctions de mutations (sans croisement) la première permet la suppression d'un élément de la composition, la deuxième permet l'ajout d'un élément au début de la composition

Les expérimentations montrent que la recherche aveugle (telle que la recherche en profondeur itérative) n'est faisable que pour les petites collections de tests et les petites solutions (en termes de taille de composition), elle est incomplète pour les grands espaces de recherche.

L'algorithme génétique et la recherche gourmande trouvent toujours des solutions mais la recherche génétique est plus lente par rapport à la recherche gourmande.

[Lecue, 2009] Propose un algorithme génétique pour optimiser une composition issue d'un workflow statique (i.e. le nombre de tâches est fixé).

La fonction fitness prend en compte, la qualité sémantique des liens (reliant les services) la QOS, et une pénalité de nature dynamique (si la composition viole les contraintes globales alors on diminue le score de la performance (ou fitness).

Les résultats montrent que cette approche est très performante en termes de taux d'optimalité et de temps d'exécution, son inconvénient principal est le caractère statique de la requête (i.e. le workflow).

[Bekkouche et al, 2012] proposent un algorithme génétique pour sélectionner des compositions de services, tout en prenant en compte plusieurs aspects (la compatibilité sémantique des entrées-sorties, l'optimisation des paramètres de Qualité de service, la satisfaction des contraintes globales de l'utilisateur), ces aspects sont agrégés en une seule fonction objective.

La compatibilité sémantique est évaluée avec 04 scores discrets : exact, plugin, subsume et fail.

Il est utile de noter que cette approche est dynamique, i.e. que la fonction objective dépend de la requête de l'utilisateur (qui peut changer), l'algorithme génétique adopté emploie à la place du croisement, deux fonctions de mutations, la première permet la suppression des gènes (ou les services), la deuxième permet l'ajout des services.

Les résultats obtenus sont satisfaisables (une fitness proche de 0.7) malgré la petite taille de la collection de test adoptée.

II.4) Hybridation Planification-Metaheuristiques

[POP et al, 2010 a] combinent la planification avec les meta-heuristiques pour la composition de services. Pour cela ils génèrent en premier lieu, un « planning graph » augmenté avec une matrice de liens sémantiques (Enhanced Planning Graph (EPG)) (voir la figure suivante).

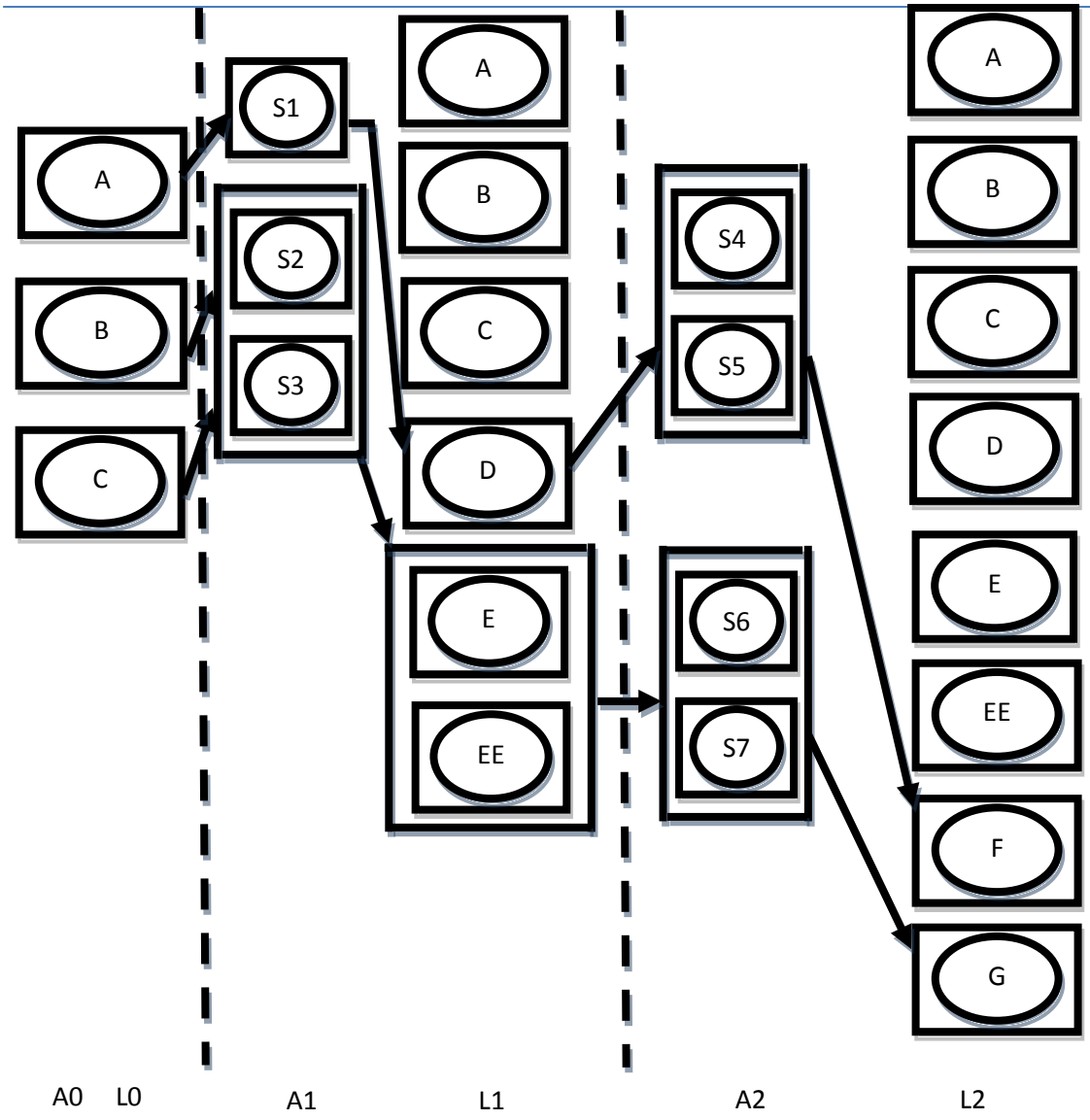


Figure IV.6. Graphe EGP

Nous notons que l'EGP contient des clusters de services similaires, d'un point de vue fonctionnel, (mais différents selon les valeurs de QOS). Par exemple la couche d'actions A1 possède deux services similaires s2 et s3. Le graphe contient aussi des concepts similaires, par exemple les concepts E et EE sont groupés dans un même cluster, parce qu'ils ont le même parent.

L'étape suivante permet l'optimisation de la meta- solution fournie par l'EGP, et pour cela les auteurs proposent l'optimisation par colonies d'abeilles [Teodorovic et Dell'Orco, 2005] pour assurer cette tâche.

La fonction objective des colonies d'abeilles, agrège la qualité de service de la solution ainsi que sa qualité sémantique (l'approche ne gère pas les contraintes globales).

La qualité sémantique de la solution est calculée à l'aide d'une fonction de similarité sémantique [Skoutas et al, 2007] qui fait intervenir les critères de rappel et de précision.

Les résultats obtenus montrent la faisabilité de l'approche (elle peut garantir une solution optimale pour une petite collection de test (110 services formalisés en sawsdl)).

Dans [Pop et al, 2009], [Pop et al, 2010 b] les auteurs proposent toujours, une composition à deux phase, la première phase permet génération de l'EGP (le « planning graph » augmenté par les liens sémantiques) la deuxième phase (celle de l'optimisation) emploie la sélection clonale [Castro et Zuben, 2002] pour le choix des services concrets appartenant aux clusters.

Les résultats obtenus sont très satisfaisants en termes d'optimalité.

[Salomie et al, 2011] Proposent une amélioration du travail présenté en [pop et al, 2010 b], plus précisément ils ajoutent une dimension d'apprentissage par renforcement à la sélection clonale, de même ils proposent des opérateurs évolutionnaires (reproduction et sélection) et un redémarrage aléatoire périodique de l'algorithme.

Ces modifications ont pour objectif d'élaguer l'espace de recherche (en effet l'algorithme explore uniquement 0.001% de l'espace de recherche.

L'approche proposée possède, plusieurs avantages par rapport aux metaheuristiques développées en état de l'art, en premier lieu, l'algorithme construit un workflow (EGP) dynamique (i.e. pour chaque requête), en plus il gère l'aspect Qos et l'aspect sémantique des compositions en même temps, en dernier lieu le processus de remplacement de service n'est pas aléatoire mais il dépend de la qualité sémantique et des valeurs de Qos de la composition. Les auteurs estiment que l'approche permet d'atteindre la solution optimale dans 95% des cas.

[Lecue, 2008] propose dans sa thèse, deux techniques de compositions de services web sans états, (functional level web services), ces deux algorithmes sont suivis par une phase d'optimisation qui trie les compositions selon la qualité des liens sémantiques reliant les composants de la solution. Le module d'optimisation est basé sur un outil de programmation par contraintes.

Le premier algorithme de composition nommé « Ra4c » est un chainage arrière (planification régressive) qui emploie les liens sémantiques, comme un critère de composition. L'algorithme Ra4c calcule une matrice de qualité de liens, entre les constituants de la solution en utilisant des mesures de similarité sémantique (à base de subsomption).

Le deuxième algorithme de composition est une extension du système GOLOG, il permet de prendre en charge deux critères de composition : les liens sémantiques (i.e. les entrées-sorties des services), et les liens causaux (i.e. les pre-conditions et les post-conditions des services)

L'outil proposé est testé sur 03 collections : Télécommunication scenario, E-Tourisme scenario, E-Santé, ces collections ont des tailles différentes.

Les résultats ont montré que le temps d'exécution du deuxième algorithme est plus grand que le premier. En outre, le nombre de résultats du deuxième algorithme, est plus réduit par rapport au premier, ceci confirme la haute précision du second algorithme.

La partie la plus couteuse du deuxième l'algorithme, est celle du chainage arrière, de même, il est utile de noter que, la phase d'optimisation est peu couteuse en termes de temps d'exécution (moins de 1 sec).

III) Conclusion

Nous avons présenté dans ce chapitre un survol sur les méthodes de composition, ces approches sont classées en quatre groupes : les workflows, la planification en I.A, les meta-heuristiques et l'hybridation metaheuristiques-planification. Dans le chapitre suivant nous traitons une classe particulière d'approches de composition, elles concernent l'instanciation des workflows statiques à base de QOS.

Chapitre 5 :

La sélection des services web composés à base de QOS

Sommaire

I) Introduction

La sélection des compositions de services « QoS-aware service composition », est l'une des problématiques les plus importantes de l'architecture orientée service. Pour la présenter on considère l'exemple suivant :

On suppose qu'il y a un utilisateur qui veut planifier un voyage, pour cela il a besoin de consommer 03 types de services, une réservation d'hôtel, une réservation de billet d'avion et une location de voiture, on note aussi qu'on doit sélectionner un seul service (ou entreprise) de chaque catégorie, en utilisant les critères de QOS (réputation, fiabilité, coûts, temps d'exécution…), en plus l'utilisateur exige des contraintes globales[11] sur chaque critère de QOS, par exemple le coût total des 03 services, ne doit pas excéder une certaine limite.

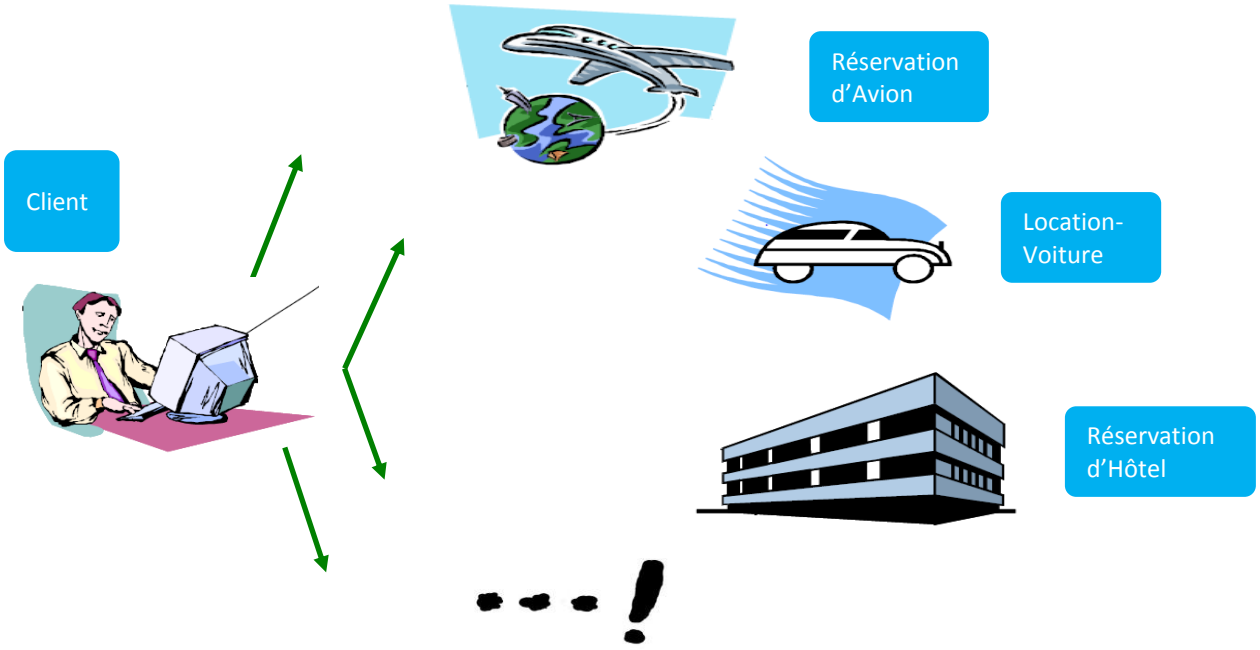


Figure V.1. Scenario de motivation [Yu et Bouguettaya, 2010]

Il est clair que ce problème est NP-Hard, en effet il est équivalent au MMKP(multi-dimensional multiple choice knapsack problem) [Pisinger, 1995].

Les solutions exactes de ce problème ont une complexité exponentielle, et de ce fait nous ne pouvons plus garantir les exigences temps réelles, ou le passage à l'échelle.

[11]une contrainte globale s'applique sur les 03 services sélectionnés)

II) Formalisation du Problème

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

Soient :

- CA ={S1, . . . , Sn} , une composition abstraite qui représente la requête de l'utilisateur, cad les n classes de services à consommer « Si ».
- CONS = {c_1, . . . , c_m}, Un ensemble de contraintes globales définies par l'utilisateur.
- C = {s_1, . . . , s_n} une composition concrète, cad nous remplaçons chaque classe Si par un service concret $s_i \in$ Si

Nous devons rechercher une composition concrète C telle que:

- La ou les fonctions objectives $U_k(C)$ sont optimisées. (U_k représente la fonction objective du K^{eme}attribut de QOS. Si on agrège les m attributs de QOS en une seule valeur alors nous aurons besoin d'une seule fonction objective U.)
- les contraintes globales sont vérifiées, i.e.$Q'_k(C) \geq c_k$, $\forall$ $c_k \in$ CONS

Avec $Q'_k(C)$, est la valeur du K^{eme} critère de QOS de la composition c. Si nous supposons que le nombre de candidats par classe est L, alors le nombre global de compositions possibles est L^n.

L'énumération totale de ces compositions ne peut être faite en temps raisonnable, en plus la présence des contraintes globales, impose un temps exponentiel pour avoir une solution exacte.

III) Etat de l'Art

La sélection de services, a fait l'objet de plusieurs travaux, de façon générale on distingue 03 grandes classes (voir la figure V.2. [Hadjila et al, 2012 d]. [Hadjila et al, 2013 c]) :

La sélection mono objective, La sélection multi objective, La sélection hybride (mono et multi objective)

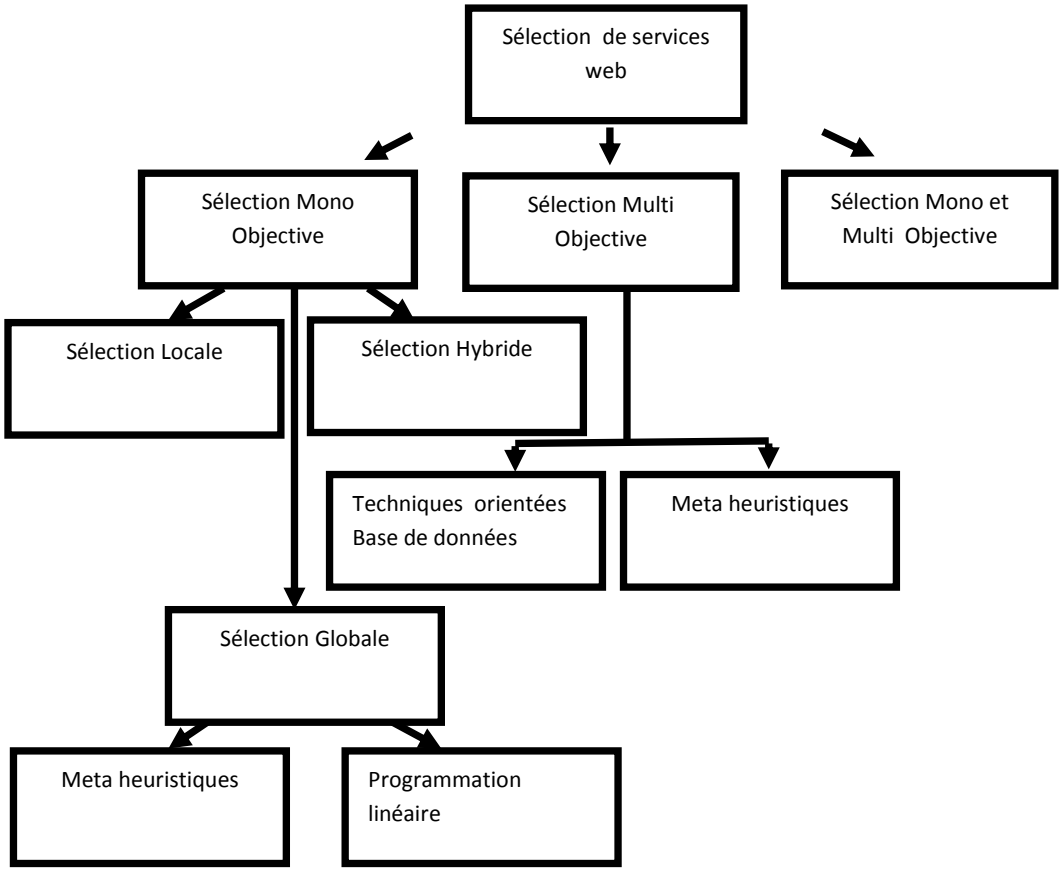


Figure V.2. Les différentes classes de sélection de services.

III.1) Sélection Mono Objective

Cette catégorie suppose que les m valeurs de qualité de service sont agrégées en un seul score, i.e. on considère une seule fonction objective qui associe des poids aux différents attributs de QOS. Elle est divisée en 03 sous classes :

La sélection locale, globale et hybride (globale et locale).

- **Sélection Globale**

Ces approches explorent un espace de recherche dont les nœuds sont des compositions complètes (cad contenant toutes les tâches), on distingue deux sous classes d'optimisation globale : exacte et approximative.

- **Optimisation Exacte**

Elle se base sur la programmation entière ou la programmation par contrainte, ou les énumérations exhaustives, ces méthodes donnent des résultats optimaux mais elles ont un temps d'exécution exponentiel. [Zeng et al, 2003] ; [Zeng et al, 2004] ; [Ardagna et Pernici, 2005] ; [Ardagna et Pernici 2007]; [Kritikos et Plexousakis, 2009]. Les travaux de Zeng et ses collègues [Zeng et al, 2004] utilisent les techniques de programmation entière et mixtes (MixedIntegerProgramming ou MIP) [Nemhauser et Wolsey, 1988] pour trouver la composition optimale des services.

Dans la même optique [Ardagna et Pernici, 2005], [Ardagna et Pernici, 2007] étendent le modèle de programmation linéaire afin d'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).

[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 ayants 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 objective.

[Zhai et al, 2009] proposent un algorithme à base de MIP afin de remplacer des services en pannes, la réparation doit garantir la satisfaction des contraintes globales de la composition.

[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. la gestion de conflits est détaillée dans [Ai et Tang, 2008] et [Gao et al, 2006]. Le travail présenté dans [Gao et al, 2006] modélise les services avec des variables binaires X_{ij} et ajoute des contraintes portant sur les variables incompatibles.

Par exemple $X_{11} + X_{23} \leq 1$ exprime que le service 1 de la classe 1 n'est pas compatible avec le service 2 de la classe 3

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.

Selon [Maros, 2003] la programmation entière est efficace si le nombre de service L est petit, par contre si L dépasse une certaine limite (quelques milliers) alors le temps d'exécution n'est plus raisonnable.

- **Optimisation Approximative**

Cette catégorie consiste à explorer une partie de l'espace de recherche, en adoptant des recherches heuristiques ou des metaheuristiques, les meta- heuristiques [Dreo et al, 2003] sont des algorithmes d'optimisation génériques, qui adoptent une recherche locale ou globale, elles permettent de donner des résultats quasi-optimaux, tout en ayant un temps d'exécution abordable.

Par opposition, les recherches heuristiques ne peuvent être généralisées pour le reste des problèmes.

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[12]). L'auteur propose une heuristique nommée UHE pour sa résolution, UHE utilise une mesure appelée la consommation des ressources agrégés, pour mettre à jour un élément de chaque groupe à chaque tour de sélection. Dans [Akbar et al, 2001] les auteurs modifient l'heuristique en créant M-UHE, ils proposent une étape de prétraitement de 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 la 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 que l'algorithme C-UHE fonctionne

[12] multi-dimensional multiple choice knapsack problem

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 sont pas scalables (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).

[Xia et al, 2011] Proposent une sélection globale de composition de services en adoptant 04 structures de flux de contrôle : parallélisme, séquence, choix conditionnels, et les boucle, la requête est formalisée en BPEL, elle possède 05 critères de QOS. L'algorithme est baptisé qssac, il peut donner un résultat proche de l'optimal (mais les auteurs n'ont pas donné d'expérimentations sur le degré d'optimalité).

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 02 ou K tâches du document BPEL, et les trient à l'aide d'une fonction objective.

Dans [Zhang et Ren, 2011], les auteurs proposent une normalisation des valeurs de QOS légèrement différente par rapport à [Alriafai et al, 2008], [Alriafai et al, 2010], [Alriafai et al, 2012], ils utilisent un algorithme génétique adaptative, i.e. que les taux de mutations et de croisement dépendent de l'affinité des compositions (leur fitness). Plus l'affinité est grande, plus les taux de croisements et de mutation sont faibles, en plus les auteurs introduisent quatre constantes pour contrôler ces taux.

Les expérimentations montrent que le temps d'exécution est moins de 40 sec mais la base considérée est très petite (03 classes et moins de 500 services).

Dans [Canfora et al, 2005] et [Wang et 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 et 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 et Hou, 2008] propose 03 fonctions fitness (une pour chaque critère de QOS).

Les algorithmes génétiques souffrent des problèmes de scalabilité, (le codage des chromosomes est fixe), en plus la possibilité de se stagner dans un optimum local est toujours présente.

Dans [Ming et Zhen, 2007] les auteurs utilisent l'optimisation en essaims particulaires PSO [kennedy et Eberhart, 1995], selon les auteurs, PSO est plus rapide et plus scalable par rapport aux algorithmes génétiques, dans ce travail les auteurs modifient les services concrets des particules en changeant la vélocité avec des opérateurs tels que l'addition, la soustraction, la multiplication.

Mais nous notons aussi que, PSO ne peut pas éviter le problème des optimums locaux. Et de ce fait l'approche de [Liu et al, 2007] propose l'hybridation des PSO avec les algorithmes génétiques afin de marier les opérateurs locaux avec les opérateurs globaux, par rapport à [Ming et Zhen, 2007] les auteurs ajoutent le croisement, en vue d'explorer de nouvelles portions d'espace de recherche.

Dans [Rongxi et al, 2010], les auteurs proposent l'optimisation par colonies de fourmis (ACO) pour choisir une composition quasi-optimale, ils analysent les performances en variant plusieurs paramètres (le taux d'évaporation de la phéromone, la population initiale…)

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.

L'hybridation algorithme génétique -recherche taboue est proposée dans [Bahadori et al, 2009], 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 optimums locaux.

- **Sélection Locale**

Elle consiste à élire un seul service de chaque classe en utilisant une fonction objective (et indépendamment des autres classes), ensuite elle compose les n résultats qui correspondent aux n classes.

Cette approche possède une complexité linéaire O(l) [Alrifai et al, 2012], en plus elle est fortement adaptée aux environnements distribués, en effet la gestion de la QOS (mesures, mise à jours … est faite par des facilitateurs distribués). [Benatallah et al, 2002 b],[Li et al, 2007]. Mais nous notons en contrepartie, que ce type de méthodes ne prend pas en charge les contraintes globales (ex : le coût global de la composition), ce qui la rend obsolète pour les problèmes réels.

[Alrifai et al, 2008] adaptent la sélection locale pour gérer les contraintes globales. Pour cela ils divisent les contraintes globales, en contraintes locales en se basant sur la distribution statistiques des valeurs de QOS. Les auteurs gèrent uniquement la séquence.

Dans [Benouaret et al, 2011a] et [Benouaret et al, 2011b] Les auteurs considèrent la sélection de service web en prenant en compte les préférences sur les sorties de services, pour cela ils appliquent un tri local (pour chaque classe) en adoptant la notion de fuzzy dominance, cette dernière permet la comparaison de 02 services de manière plus efficace que la notion de pareto dominance. Enfin ils retiennent les K premières solutions. Les auteurs ne gèrent ni la qualité de service, ni les contraintes globales.

- **Sélection Hybride**

[Alrifai et Risse, 2009] les auteurs transforment le problème d'optimisation globale (exacte), en un problème d'optimisation approximative. (Moins gourmand en termes de temps).

En effet les méthodes d'optimisation globales et exactes dépendent fortement du nombre de services de la collection de test, puisque nous associons une variable binaire, pour chaque service (n*l variables sachant que n est le nombre de classes, et l est le nombre de services par classe). Ceci explique le caractère exponentiel et la non scalabilité de ces approches. Pour pallier à ce problème, les auteurs proposent deux phases d'optimisation, une phase globale qui considère des intervalles de valeurs de Qos comme variables à la place des services directs, le problème revient à sélectionner des bornes b'^{z}_{jk}, pour chaque critère de Qos et pour chaque classe de telle sorte que la majorité des effectifs de la classe est retenu après optimisation, en plus de la préservation des contraintes globales de l'utilisateur.

Pour cela, les auteurs divisent chaque intervalle de Qos de chaque classe, en d sous intervalles, ensuite ils choisissent un seul sous intervalle pour l'utiliser au niveau de l'étape suivante. Pour sélectionner les intervalles, l'approche adopte le modèle MIP [Nemhauser et Wolsey, 1988] (la programmation en nombre entiers ou Mixed Integer Programming).

Le modèle MIP doit maximiser la fonction suivante (i.e. choisir des intervalles d qui ont une forte chance pour avoir des services optimaux et vérifiant les contraintes globales)

maximiser $\prod^{n}_{j=1} \prod^{m}_{k=1} (P^{z}_{jk})$, $1 \leq z \leq d$ ou la version linéaire donnée ci-dessous

maximiser $\sum^{n}_{j=1} \sum^{m}_{k=1} \sum^{d}_{z=1} (\ln(P^{z}_{jk}) * x^{z}_{jk})$

$\forall$ k : $(\sum^{n}_{j=1} \sum^{d}_{z=1} (q^{z}_{jk} * x^{z}_{jk})) \leq c'_{k}$, avec $1 \leq k \leq m$.Les contraintes globales que doivent vérifier les solutions.

Le sous intervalle retenu, contient la borne recherchée b'^{z}_{jk}. b'^{z}_{jk} est choisi comme étant la valeur de Qos la plus fréquente dans cet intervalle.

La variable P^{z}_{jk} est definie comme suit :

P^{z}_{jk}= (effectif(q^{z}_{jk})/L)∗(Affinité(q^{z}_{jk})/Affinité-Max)

Avec :

effectif(q^{z}_{jk}) : est le nombre d'instances de la classe j et de l'intervalle z associé au critere k, qui verifient la contrainte locale b'^{z}_{jk}.

L :le nombre total de services pour la classe j.

Affinité(q^z_{jk}) : l'affinité (fitness) du meilleur service des services qualifiés dans l'intervalle z.(l'affinité d'un service S est une somme ponderée de ses valeurs de QOS normalisées).

Affinité-Max : est la meileure affinité de la classe j.

Cette formule permet de favoriser le choix d'un intervalle z qui contient une bonne partie de l'effectif de la classe, et un bon service (en termes de QOS) de cet intervalle.

Les constantes c'_k jouent le rôle de bornes supérieures, pour chaque critère (contraintes globales).

Pour garantir la scalabilité de l'approche, il faut que d<=l/m, sinon le nombre de variables sera égal au modèle d'optimisation global.

Une fois les constantes b'^z_{jk} ont été choisies, la deuxième phase extrait le meilleur service de chaque classe (à l'aide de la fonction d'affinité) et assemble ces résultats pour avoir la solution quasi-optimale. (L'optimisation locale).

Les expérimentations utilisent deux collections de test : une collection synthétique (générée de façon aléatoire et selon loi uniforme), elle contient 20000 services web, et collection réelle nommée QWS[13] [Al-Masri et Mahmoud, 2008].

QWS contient 09 attributs de QoS et 2500 services web réels. Les résultats obtenus montrent que l'optimisation hybride est plus efficace en termes de temps par rapport à l'optimisation globale, en plus l'optimisation hybride est capable d'atteindre un taux d'optimalité proche de 96%.

optimalité= affinité(Sol-courante)/affinité(c*) : Cette formule permet de calculer le degré d'optimalité, c'est le rapport entre l'affinité de la solution courante (hybride) et la solution optimale c* (globale). Enfin les différents essais du paramètre d ont conclu que la valeur 10, (d=10) donne la meilleure performance en termes de temps d'exécution.

Dans [Alrifai et al, 2012] les auteurs étendent le travail proposé en [Alrifai et Risse, 2009], pour prendre en compte d'autres flux de contrôle tels que le parallélisme, les boucles et les choix conditionnels. Pour ce faire, les auteurs introduisent des classes abstraites pour

[13] http://www.uoguelph.ca/ qmahmoud/qws/index.html/.

construire un mandataire des tâches de services employant ces nouvelles structures de contrôle.

Les auteurs comparent le temps d'exécution et le degré d'optimalité de l'approche hybride avec l'approche globale à base de MIP de [Liu et al, 2004], [Ardagna et Pernici, 2007] et l'algorithme heuristique WS-HEU de [Yu et al, 2007],

Les auteurs varient le nombre de classes, le nombre de services par classe, et le nombre de contraintes globales, durant ces expérimentations et concluent que l'approche hybride est la plus efficace en termes de temps d'exécution, ces résultats sont valables pour toutes les collections de test (synthétiques et réelles QWS [Al-Masri et Mahmoud, 2008]).

Nous notons que les auteurs ont généré 03 bases synthétiques à l'aide d'un générateur aléatoire la première est positivement corrélé, la deuxième est anti corrélée, et la troisième est indépendante (voir la figure V.3). Chacune d'elles contient 10000 vecteurs de QOS et 09 attributs (Q1…Q9).

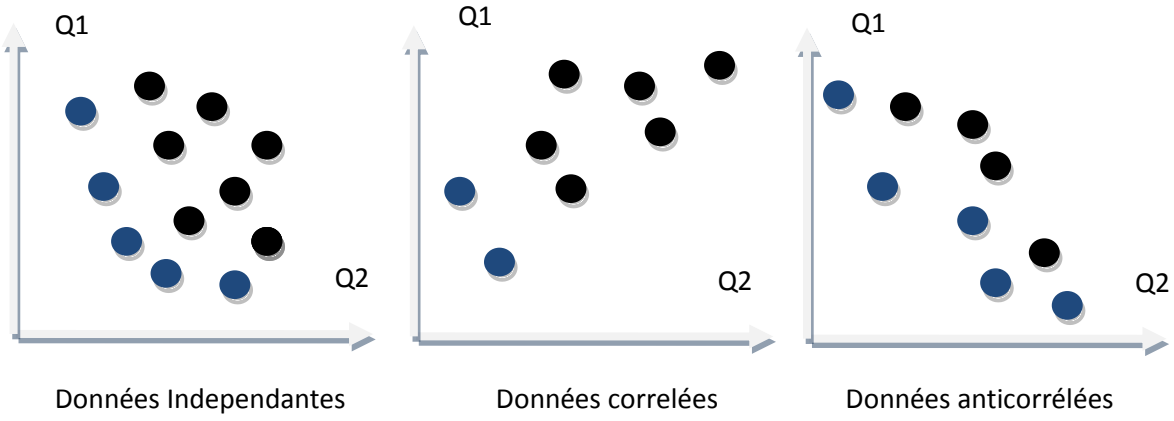


Figure V.3. Les différents types de corpus

En termes d'optimalité l'algorithme WS-HEU est plus performant par rapport à l'approche hybride.

III.2) Sélection Multi-Objectives

Plusieurs travaux, ont été proposés pour la recherche multi objectives des services skylines, la majorité utilise les meta heuristiques telles que l'algorithme NSGAII (Nondominated Sorting Genetic Algorithm II).

NSGA-II [Deb et al, 2002], et SPEAII (Strengh Pareto elitist algorithm II) [Zitizler et Thiele, 1998].

Dans [Claro et al, 2005] les auteurs considèrent 04 fonctions objectives :

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

Deux expérimentations ont été menées pour évaluer l'algorithme NSGAII

Le nombre de service de la base varie entre 30 et 60, et le nombre de tâches ou classes varie entre 03 et 05.

Pour la première expérimentation, la taille de la population des chromosomes varie de 10 à 200 individus et le nombre de générations est compris entre10 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 le deuxième test effectué, 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 05 classes et 50 services le nombre de skylines=161, et le temps d'exécution=25 sec.

Dans [Gonsalves et al, 2009] les auteurs proposent une optimisation multi objectives à base d'essaims particulaires, ils considèrent deux fonctions fitness (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 objectives.

III.3) Sélection Mono et Multi-objective

[Alrifai et al, 2010] proposent des approches combinant la sélection multi-objective (skylines) et mono-objective, En particulier ils proposent 03 algorithmes à base de skylines

Le premier est nommé « exact skylines », il extrait les services skylines de chaque classe et applique une optimisation globale à base de MIP sur ces résultats.

Nous notons 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 fitness).

La figure suivante présente un clustering hiérarchique des skylines d'une classe donnée. Nous montrons juste deux critères le prix (Q1) et le temps d'exécution (Q2). (Pour faciliter l'affichage).

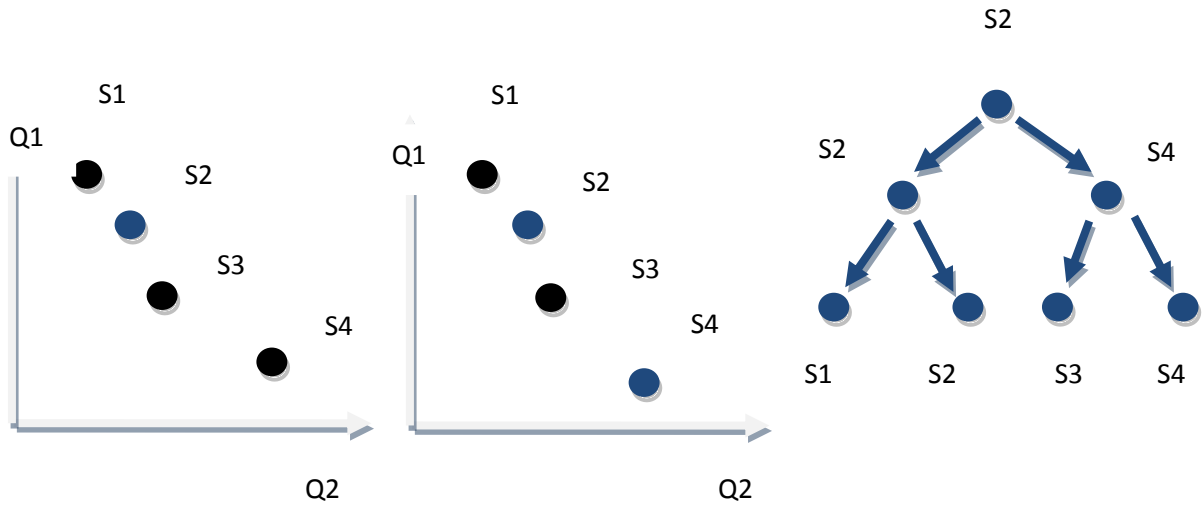


Figure V.4.Custering hiérarchique des skylines

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 [Alrifai et Risse, 2009] pour l'ensemble des skylines extraits de chaque classe.

Les expérimentations utilisent 04 types collections de test, (une collection réelle développée par [Al-Masri et Mahmoud, 2008] et trois synthétiques).

Elles comparent les 03 algorithmes avec l'optimisation globale exacte [Liu et al, 2004]; [Ardagna et Pernici, 2007], et l'approche hybride [Alrifai et Risse, 2009].

En termes de temps d'exécution, l'ordre de performance est le suivant (pour toutes les collections de test):

representative- skylines> l'approche hybride > hybride- skylines >exact- skylines> l'optimisation globale exacte.

La relation X> Y veut dire que l'approche X est meilleure que Y.

En termes d'optimalité, l'ordre de performance est le suivant (pour toutes les collections de test):

representative- skylines> l'approche hybride > hybride- skylines . Le degré d'optimalité est proche de 100% pour représentative- skylines.

IV) 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 étudié la sélection mono objective avec toutes ses variantes, la sélection multi objective, et la sélection hybride.

Dans le chapitre suivant, nous présentons un ensemble d'approches appartenant aux catégories mono- objective et hybride.

Partie II :

Contributions.

Chapitre 6 :

Approches proposées

Sommaire

I) Introduction

Dans ce chapitre nous décrivons les approches proposées pour résoudre les problématiques citées en état de l'art. Ce chapitre est divisé en trois sections, dans la première partie nous présentons formellement le problème de découverte, ainsi que nos deux contributions. Dans la deuxième partie nous présentons le problème de composition à l'aide d'un scenario de motivation, ensuite nous introduisons deux algorithmes permettant la composition dynamique des services web. Dans la troisième section, nous présentons formellement le problème de sélection de services composés, ensuite nous décrivons cinq algorithmes proposés, pour l'optimisation de la sélection. Et enfin nous résumons nos principales contributions, dans la conclusion

II) Découverte Sémantique de Services Web

II.1) Préliminaires

La découverte sémantique des services web, consiste à comparer les descriptions sémantiques des services web avec celles des requêtes, ces descriptions représentent généralement l'aspect fonctionnel.

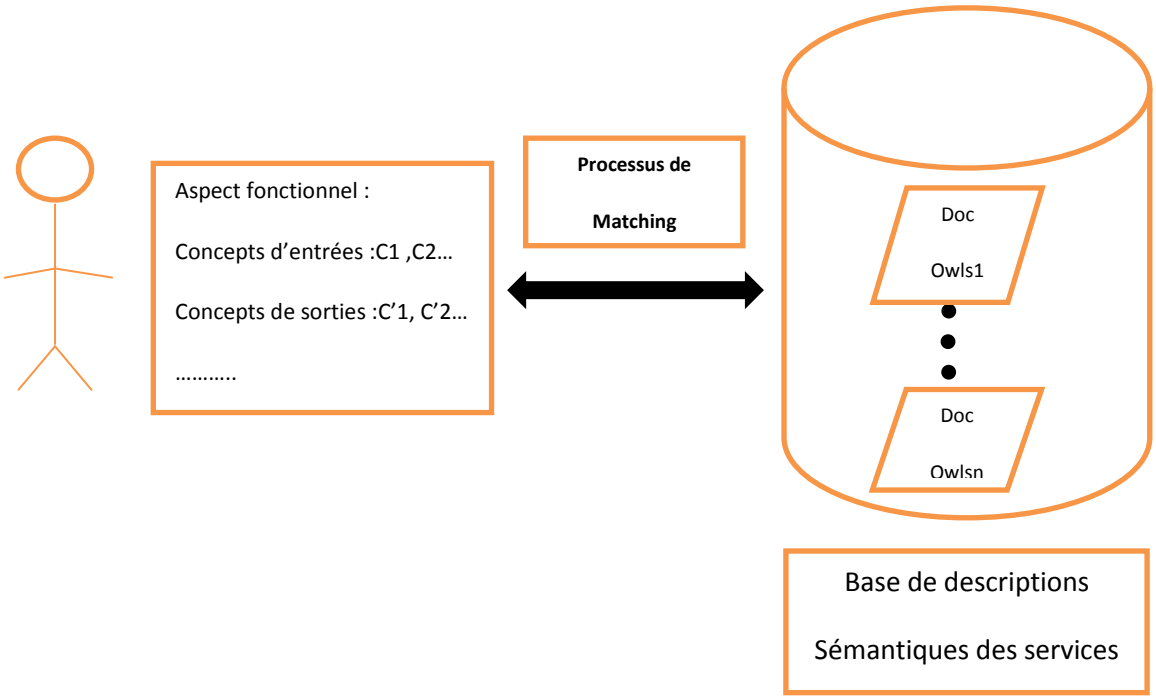


Figure VI.1. Processus de découverte sémantique

II.2) Motivations

La majorité des travaux de l'état de l'art modélisent l'aspect fonctionnel par la signature des opérations i.e. (les entrées et les sorties des services). Comme nous l'avons déjà mentionné dans la partie revue de la littérature, ces solutions sont classées en trois catégories :

- Les approches sémantiques logiques
- Les approches sémantiques non logiques
- Les approches sémantiques hybrides.

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 due aux scores « subsume » ou « subsumed by »
- La présence de faux négatifs due aux insuffisances des scores logiques
- nous ne pouvons pas comparer les services ayant le même score logique. (et de ce fait les scores numériques sont plus adaptatifs et plus flexibles)

Selon [Klush et al, 2009 a] Les approches hybrides ont amélioré les performances approches logiques, mais ils restent toujours au-dessous des performances (rappel et précision) de certaines approches non logiques telles que [Plebani et Pernici, 2009].

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.

Pour pallier aux difficultés citées ci-dessus, nous avons choisi des algorithmes appartenant à la classe non logique, et plus précisément nous avons choisi deux mesures de similarité pour comparer les concepts de la requête et des services.

La première est nommée « cosine » ou cosinus, elle permet de mesurer l'angle séparant les vecteurs décrivant la ta requête et le service.

$$Cosine(A,B) = \langle A, B \rangle / (||A||. \ ||B||)$$

Avec $< A, B>$ représente le produit scalaire des vecteurs A,B

$||A||., || B||$: représentent les amplitudes des concepts

Selon [Cohen et al, 2003] cosine est l'une des quatre meilleures mesures employées dans le domaine de recherche d'information. En plus, sa complexité est polynomiale. La deuxième est nommée SimWP [Wu et Palmer, 1994], elle permet de calculer la proximité sémantique entre deux concepts en utilisant les arcs de l'ontologie, elle est simple à implémenter, et sa complexité est polynomiale.

En effet, cette mesure dépend uniquement des profondeurs des concepts, et puisque la majorité des ontologies ont des profondeurs limitées par rapport au nombre de concepts, alors le temps d'exécution de la mesure est toujours acceptable. (Par exemple le thesaurus wordnet Version 2.1 possède 17 niveaux et plus de 81426 concepts [Strunjas-Yoshikawa et al, 2006]).

SimWP(c1,c2)= (2* profondeur(PPS(c1,c2))/(profondeur(c1)+profondeur(c2))

Ou SimWP(c1,c2) = (2 * N3) /(N1+N2 +2*N3)

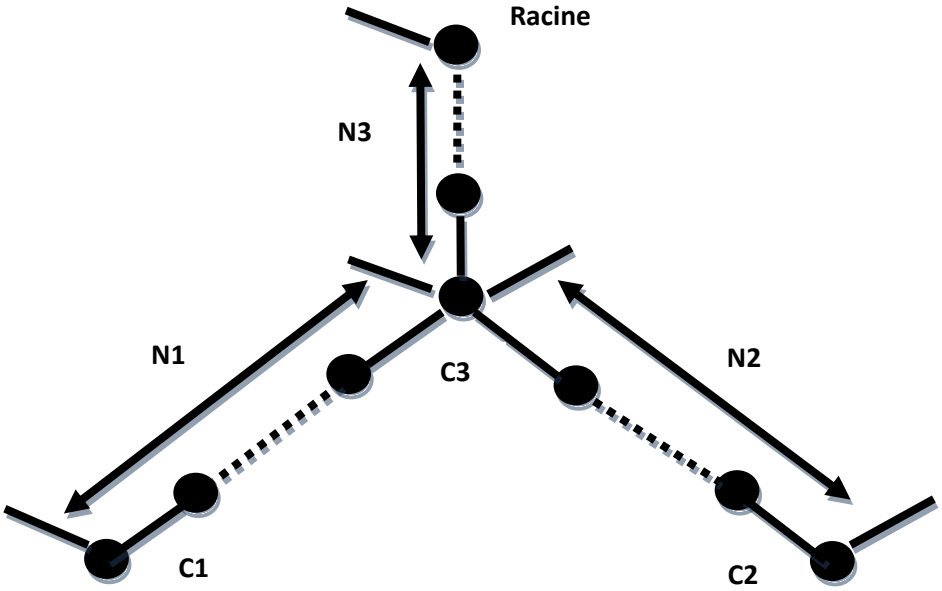


Figure VI.2. Exemple de calcul de la mesure SimWP

Avec :

profondeur (c1) : est le nombre d'arcs qui séparent c1 et la racine de l'ontologie O.

profondeur (c2) : est le nombre d'arcs qui séparent c2 et la racine de l'ontologie O.

PPS (c1,c2) :est le plus petit subsumant de c1,c2, i.e. c'est l'ascendant commun de c1 et c2 ayant la plus grande profondeur par rapport à la racine.

II.3) Première Proposition

Dans cette approche [Hadjila, Chikh, 2013 a], nous avons modélisé les services et les requêtes par deux vecteurs de fréquence de concepts Vi et Vo.

Vi représente l'indexation conceptuelle des entrées de la requête R ou du service S, par exemple si S possède les entrées C1 et C2, et si C1 possède C4 , C5, C0 comme concepts subsumants, et C2 possède C5 et C0 comme concepts subsumants, alors Vi={ C0(2), C1(1), C2(1), C4(1),C5(2)}.

Vi contient les fréquences de chaque concept qui appartient à la requête, ou qui figure dans les ascendants des concepts de la requête.

De façon similaire, Vo représente l'indexation conceptuelle des sorties de la requête R ou du service S.

Par exemple, si S possède les sorties C7 et C8, et si C7 possède C9, C0 comme concepts subsumants, et C8 possède C0 comme concepts subsumants, alors Vo={ C0(2), C7(1), C8(1), C9(1)}.

Dans la deuxième phase (recherche), nous avons employé la mesure Cosine pour comparer les vecteurs Vi et Vo de chaque paire <requête :R, service : S>.

Un service S est retenu, s'il dépasse le seuil θ fixé par l'utilisateur.

Le pseudo code de notre première approche est donné ci-dessous :

Algorithme1 : Découverte sémantique de services à base d'indexation conceptuelle

Entrées : une requête R, une base de services B formatée en OWLS

Sorties : un ensemble de services retenus A

La phase d'indexation

1- Extraire les entrées et les sorties de chaque service S∈ Base B (à partir des documents OWLS)
   a. Construire le vecteur Si contenant la fréquence des concepts d'entrées ainsi que leur subsumants. Par exemple si les entrées sont {c1,c2}, et c1 possède {c3 ,c4} comme subsumants, et c2 possède {c5,c4} comme subsumants,

alors Si= {c1(1), c2(1), c3(1), c4(2), c5(1)}(nous notons que C4 a deux occurrences).

b. Nous faisons la même chose pour les sorties (la construction de So).

2- Nous faisons le même traitement (étape a & b) pour la requête (nous calculons Ri et Ro).

La phase de matching

1- A= ∅ // on initialise l'ensemble des résultats à vide.

2- Pour chaque service S de la base, nous comparons les 02 vecteurs (d'entrées-sorties) de S avec la requête à l'aide de la mesure cosine: cosine $(R_x,S_x) = < R_x, S_x>/(||R_x||. || S_x||)$ avec $< R_x, S_x>$ est le produit scalaire de R_x et S_x (x dénote les entrées ou les sorties. cette étape est réalisée comme suit :

- Score1 = cosine (Ri, Si).
- Score2 = cosine(Ro, So)
- Score = (Score 1+Score 2)/2

3- nous retenons les services S dont le score est plus grand que le seuil θ (le seuil θ est choisi par l'utilisateur, $\theta \in [0,1]$). i.e. $A= A \cup \{S\}$

4- nous trions les résultats par ordre décroissant des scores.

Cet algorithme est évalué dans le chapitre suivant à l'aide des critères de rappel et de précisions.

Exemple

Considérons le service S, et la requête R suivants:

Si={ C0(2), C1(1), C2(1), C4(1),C5(2)}

So={ C0(2), C7(1), C8(1), C9(1)}.

Ri= { C0(2), C2(1),C5(1)}

Ro= { C0(2), C7(1), C9(1), C10(1)}

La similarité est calculée comme suit :

Score(R,S)= (Score1(Ri,Si)+Score2(Ro,So)) /2.

$$= ((2*2+1*1+1*2)/(\sqrt{2^2+1^2+1^2+1^2+2^2}*\sqrt{2^2+1^2+1^2}\ ) +$$

$$(2*2+1*1+1*1)/(\sqrt{2^2+1^2+1^2+1^2}*\sqrt{2^2+1^2+1^2+1^2}))/2$$

$$= (7/8{,}12 +6/7)/2= 0{,}859$$

L'avantage principal de cet algorithme, est sa simplicité et sa capacité à représenter la sémantique des services de manière fidèle. En plus la partie matching possède une complexité polynomiale. Mais son inconvénient majeur, est le caractère exponentiel de l'indexation conceptuelle, en effet elle est de l'ordre de $O(m^l)$ avec l : la profondeur maximale de l'ontologie de domaine, et m est le nombre maximal de parents directs d'un concept donné.

Nous notons que les services de la base peuvent être indexés de manière off-line, (avant l'exécution) et de ce fait, nous indexons uniquement la requête pendant l'exécution. Ceci permet de réduire considérablement le temps d'exécution.

II.4) Deuxième Proposition

Dans cette approche [Hadjila et Chikh, 2010 a], nous avons modélisé les services et les requêtes, par deux vecteurs de concepts notés Vi et Vo. Vi représente les concepts d'entrées de la requête R ou le service S, Vo représente les concepts de sorties de la requête R ou le service S.

Le pseudo code de cette approche est donné ci-dessous :

Algorithme 2 : Découverte sémantique des services à base de mesure SimWP

Entrées : une requête R, une base de services B formatée en OWLS

Sorties : un ensemble de services retenus A

1- A= ∅
2- Pour chaque service S de la base B
    - construire 02 vecteurs de concepts Si et So à partir des documents OWLS (Si contient les concepts d'entrées et So contient les concepts de sorties)
3- faire la même chose pour la requête, et par conséquent nous aurons les vecteurs Ri et Ro

4- calculer la similarité des entrées :
   - (affectation-entrées,sim(Ri,Si)) = optimisation-matching(Ri,Si)
5- calculer la similarité des sorties
   - (affectation-sorties,sim(Ro,So)) = optimisation-matching(So,Ro)
6- calculer la similarité globale:
   - sim (R,S)= (sim(Ri,Si)+ sim(Ro,So))/2
7- nous retenons le service S, si sa similarité globale est supérieure ou égale au seuil θ. (le seuil θ est choisi par l'utilisateur, $\theta \in [0,1]$),i.e. $A= A \cup \{S\}$
8- nous trions les résultats par ordre décroissant des scores.

Dans ce qui suit, nous décrirons notre algorithme qui permet l'optimisation des matching. En effet la comparaison de deux ensembles de concepts X et Y, (Ri et Si ou Ro et So) nécessite un processus d'optimisation.

Pour cela, nous avons proposé un algorithme [Hadjila et al, 2011 a], [Hadjila et chikh, 2010 a] pour affecter au mieux, les concepts de la requête avec les concepts du service.

Algorithme 3 : Optimisation-matchings

Entrées : un ensemble de concept X, un ensemble de concept Y

Sorties : affectations /*l'ensemble des concepts appariés*/, score-final

1- construire la matrice des similarités partielles T, sachant que:
   - T possède |X| lignes
   - T possède |Y| colonne
   - L'élément T(i,j)=SimWp(Ci,Cj), avec $Ci \in X$ et $Cj \in Y$
2- Classer les scores partiels (Ci,Cj,T(i,j)), par ordre décroissant et les mettre dans un vecteur T' (la première contient le score le plus fort).
3- affectations=∅, K=1, N=0 /*N représente nombre de concepts appariés*/
4- **Tant Que** ((K<= | T' |) **et** (N< |Y|)) // | T' |=|X|*|Y|

Faire

- (Ci,Cj,score)=T'(N)
- **Si (**Ci ne figure pas dans affectations **et** Cj ne figure pas dans affectations) **Alors**
  - affectations= affectations ∪{ (Ci,Cj,score)}
  - Incrémenter N

  FinSi
- K=K+1

Fin-faire

5- Score-final=$(\sum_{i}^{|affectations|} extraire - score(affectations[i]))/|Y|$

6- Retourner (affectations, Score-final)

Pour expliquer le principe de fonctionnement de ces deux algorithmes (2 & 3), nous considérons l'exemple suivant :

S=<entrées={ C6}, Sorties={C1,C2,C3}>

R=<entrées={ C6}, Sorties={C4,C5}>

Pour le cas d'entrées :sim(Ri,Si)= SimWp(C6,C6)=1.

Pour le cas de sorties, nous devons construire la matrice des similarités partielles suivante :

Concepts de Y / Concepts de X	C4	C5
C1	0.75	0.9
C2	0.5	0.2
C3	0.6	0.3

Table VI.1. Exemple de Matrice des similarités partielles T

Apres le tri, nous retenons les triplets (C1,C5, 0.9) et (C3,C4,0.6). l'élément (C1,C4, 0.75) n'est pas retenu parce que C1 est déjà affecté à C5 (i.e. il figure dans la liste des affectations).

Pour l'exemple précédent le score final est :

- Score-final= (0.9+0.6)/2=0.75
- Affectations={(C1,C5, 0.9) , (C3,C4,0.6)}

Le score global entre la requete et le service est:

sim (R,S)= (sim(Ri,Si)+ sim(Ro,So))/2=(1+0,75)/2=0,875

Remarques

- Pour les entrées, la matrice T possède K colonnes (qui représentent les K concepts d'entrées du service), et K' lignes (qui représentent les K' concepts d'entrées de la requête). les éléments de cette matrice représentent les scores de similarité SimWp(Ci,Cj).
- Pour les sorties, la matrice T possède d colonnes (qui représentent les d concepts de sorties de la requête), et d' lignes (qui représentent les d' concepts de sorties du service) , les éléments de cette matrice représentent les scores de similarité SimWp(Ci,Cj).

Il est utile de noter que, l'introduction de l'algorithme d'optimisation (algorithme 3), permet la minimisation des faux négatifs, et par conséquent l'amélioration des rappels. Ceci sera confirmé dans le chapitre d'expérimentation.

Nous notons aussi que, la mesure SimWp peut favoriser parfois les concepts voisins (ayant un même parent), par rapport à des concepts appartenant à la même ligne d'hiérarchie (un concept et ses grands-parents.

Cette limite peut être contournée en pénalisant le score des concepts voisins. (par exemple, en multipliant le score avec une constante $\in]0,1[$). Mais ce remède alourdit encore le processus de comparaison.

III) Composition Sémantique de Services

III.1) Introduction

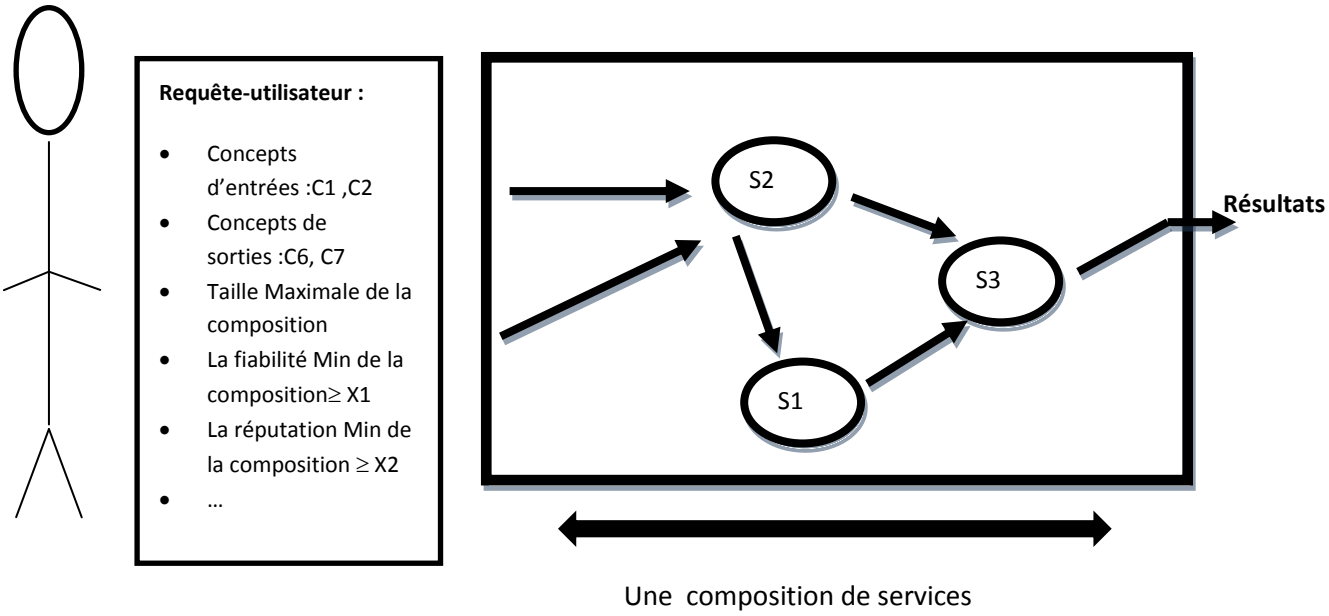
La composition de services web est le processus qui consiste à combiner plusieurs services pour satisfaire un besoin complexe de l'utilisateur [Rao, 2004]. Dans cette thèse nous nous intéressons uniquement à la composition de services web sans états, i.e. des services qui ont une interaction simple (requête -réponse, une seule réponse, une seule requête), et qui n'ont pas besoin de mémoire pour mettre à jours les invocations des utilisateurs.

Nous considérons aussi, que ces services sont de type « information gathering services », i.e. que leur aspect fonctionnel est modélisé avec un ensemble de concepts d'entrées et de sorties. Les compositions générées contiennent uniquement la séquence. L'ajout de la sémantique, assure de la correction du flux de données entre les services constituants la composition, mais il ajoute une charge de temps additionnelle (qui peut être exponentielle).

L'exemple suivant (Figure VI.3) montre un scenario qui nécessite la composition de plusieurs services pour satisfaire le besoin de l'utilisateur. (Exprimé en termes fonctionnels, et non fonctionnels). Plus précisément, nous cherchons une combinaison de services, qui :

- accepte un ensemble de concepts d'entrées (par exemple C1,C2).
- fournit un ensemble de concepts de sorties (par exemple C6,C7).
- optimise un ensemble de critères de QOS (tels que la réputation, le coût.).
- satisfait un ensemble de contraintes globales (par exemple FiabiltéMin ≥ X1, veut dire que la fiabilité minimale de la composition recherchée est supérieure à X1)

Il clair que, tous les services individuels (S1,S2 ou S3) ne peuvent assurer les entrées et les sorties de la requête, mais une composition de ces 03 services sans états, peut satisfaire les besoins fonctionnels de l'utilisateur.



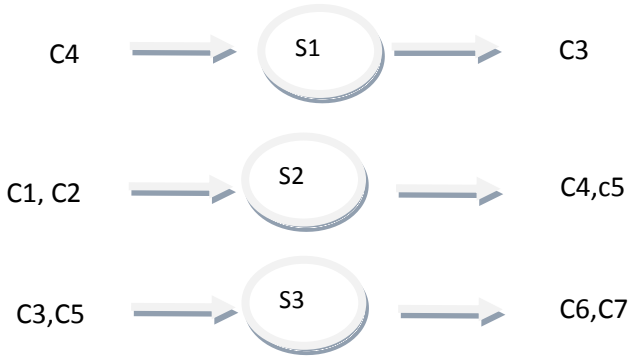


Figure VI.3. Exemple de composition de services

Dans les sections suivantes, nous proposons deux approches pour créer de telles compositions.

III.2) Motivations

La plupart des travaux de composition de services sans états, se basent sur la planification en intelligence artificielle, ou l'hybridation de la planification avec certaines meta-heuristiques. L'inconvénient principal de ces méthodes est la complexité exponentielle de la planification (qui est un problème P sapce complet), et par conséquent elles ne peuvent passer à l'échelle. En outre, certaines approches gèrent uniquement l'aspect sémantique [Bleul et al, 2007] sans prendre en compte le QOS, ou l'inverse [Alrifai et al, 2010]. Pour pallier à ces

problèmes, nous proposons deux versions de meta-heuristiques qui permettent l'optimisation de l'aspect sémantique et QOS, en même temps. Ces deux meta-heuristiques se basent sur les essaims particulaires PSO [Eberhart et Kennedy, 1995]. PSO possède une complexité polynomiale et de ce fait, nous aurons des temps d'exécutions raisonnables. La composition de service est modélisée sous forme d'un problème d'optimisation combinatoire, avec contraintes.

Les apports de nos deux Meta heuristiques sont résumés comme suit :

- La fonction objective tient compte de la sémantique et la QOS des compositions.
- L'algorithme ne dépend pas d'une requête (ou workflow) fixée, en effet les compositions peuvent avoir des tailles dynamiques.
- L'exploration de l'espace de recherche utilise plusieurs types d'opérateurs, tels que : le mouvement social, le mouvement aléatoire, la cohésion, l'alignement.

III.3) Première Approche de Composition (PSO)

Nous considérons la composition sémantique des services sans états comme un processus d'exploration d'un graphe de solutions. Ce processus se base sur une version discrétisée, de l'optimisation à base d'essaims particulaire PSO (particle swarm optimization).

L'approche proposée possède une architecture complètement maillée, i.e. que toute particule peut communiquer avec tout l'essaim, et partager la meilleure position. Nous avons discrétisé l'algorithme continu de [Eberhart et Kennedy, 1995], en adoptant deux types de mouvements :

- un mouvement aléatoire qui change un service de la composition par un autre (appartenant à la base), ou par -1 si nous voulons supprimer cette composante.
- un mouvement social qui remplace un service de la composition par un autre service appartenant à la meilleure composition de l'essaim (i.e. la connaissance sociale partagée).

Chaque solution/composition (ou nœud du graphe) est modélisée, comme un vecteur de services ayant une certaine taille notée Taille-c.

Chaque élément de ce vecteur représente un identifiant de service ∈base-services ou -1, si cette composante est absente. Taille-c est introduit par l'utilisateur, il représente la taille maximale de la composition.

Ce paramètre, permet aussi l'élagage de l'arbre de recherche. L'utilisateur spécifie aussi un ensemble valeurs numériques qui représentent les contraintes globales.

III.3.1) Formalisation

Notre objectif, est d'explorer le graphe des solutions (ou compositions) présenté précédemment, plus précisément nous cherchons une composition c qui maximise la fonction suivante :

$F2(c,requete)=F1(c,requete)+P(c,requete)$. ..... (1)

Avec F1(c) : est la sous fonction objective qui mesure l'aspect sémantique et QOS de la solution c (voir la suite).

Une composition c est dite optimale ssi

- C est faisable i.e. elle ne viole pas de contraintes globales i.e. P(c,requete)=0.
- Elle possède la valeur maximale pour la fonction F1

P(c,requete) : est la fonction de pénalité, elle donne un score négatif lorsqu'une ou plusieurs contraintes globales sont violées, sinon elle est nulle.

$P(c,requete)= -\sum^{R}_{k=1} D^2_k(c)$ ..... (2)

Avec:

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

Cons(k) : représente la K^{eme} contrainte globale (i.e. la borne minimale que doit satisfaire le K^{eme} critère de QOS de la composition).

$Q_k'(c)$: représente la valeur du K^{eme} critère de QOS de la composition c

$F1(c,requete)=W_1*U1(c)+W_2*U_2(c,requete)+W_3*U_3*(c,requete)+ W_4*U_4(c)$ ..... (3)

Avec $\sum W_i = 1$.(dans cette thèse, nous avons pris des poids équitables pour chaque fonction, i.e. que $W_i = 0.25$)

La sous fonction objective U1 réalise la somme pondérée des différentes valeurs de QOS. Elle donne des scores normalisés dans l'intervalle [0,1], plus le score est proche de 1, plus la composition est bonne, et vice versa. U1 est formalisée comme suit :

$$U_1(c) = \sum_{k=1}^{R} w'_k * \frac{\left(Q'_k(c) - Q\min'(k) \right)}{Q\max'(k) - Q\min'(k)} \quad \text{.....(4)}$$

W_k':représente la priorité associée à l'attribut K, en plus : $w'_k \in R+$ et $\sum_{k=1}^{R} w'_k = 1$

Avec $Q_k'(c)$: représente la valeur agrégée du K^{eme} critère de QOS, associée à la composition c.

Pour homogénéiser l'ensemble des critères, nous avons multiplié les critères négatifs (i.e. nécessitant une minimisation) par -1. De cette façon tous les critères doivent être maximisés. Les poids W_k' sont équitables, i.e. $W_k'=1/R$.

Les fonctions d'agrégation [Alrifai et al, 2012] des R critères de QOS sont données dans la table VI.2. [Alrifai et al, 2012] et [Liu et al, 2004] adoptent les critères de QOS suivants :

Temps de réponse : c'est la période qui s'écoule entre l'envoi de la requête et la réception des résultats.

Coût : c'est le prix fixé par le fournisseur du service.

Disponibilité : c'est la probabilité d'accessibilité du service. (Le pourcentage de temps au cours duquel le service est accessible).

Fiabilité : le nombre d'invocations réussies sur le nombre total d'invocations.

Réputation : elle est mesurée avec les feedbacks des utilisateurs.

Attribut de QOS	Fonction d'agrégation (n est la taille de la composition)			
	Séquence	Boucle	Parallèle	Choix conditionnel
Temps d'exécution	$Q1'(C) = \sum_{j=1}^{n} Q1(sj)$	$Q1'(C) = \sum_{j=1}^{n} Q1(s)$	$Q6'(C) = MAX_{j=1}^{n} Q6(sj)$	$Q1'(C) = MAX_{j=1}^{n} Q1(sj)$
Réputation	$Q2'(C) = 1/n * \sum_{j=1}^{n} Q2(sj)$	Q2(s)	$Q2'(C) = 1/n * \sum_{j=1}^{n} Q2(sj)$	$Q2'(C) = MIN_{j=1}^{n} Q2(sj)$
Cout	$Q3'(C) = = \sum_{j=1}^{n} Q3(sj)$	$Q3'(C) = = \sum_{j=1}^{n} Q3(s)$	$Q3'(C) = = \sum_{j=1}^{n} Q3(s_j)$	$Q3'(C) = MAX_{j=1}^{n} Q3(sj)$
Fiabilité	$Q4'(C) = \Pi_{j=1}^{n} Q4(sj)$	$Q4'(C) = \Pi_{j=1}^{n} Q4(s)$	$Q5'(C) = \Pi_{j=1}^{n} Q5(sj)$	$Q4'(C) = MIN_{j=1}^{n} Q4(sj)$
Disponibilité	$Q5'(C) = \Pi_{j=1}^{n} Q5(sj)$	$Q5'(C) = \Pi_{j=1}^{n} Q5(s)$	$Q5'(C) = \Pi_{j=1}^{n} Q5(sj)$	$Q5'(C) = MIN_{j=1}^{n} Q5(sj)$

Table VI.2. Les fonctions d'agrégation de valeurs de QOS.

Qmin'(k) : représente la valeur minimale du K^{eme} critère de QOS, pour toutes les compositions possibles (de taille « taille-c »).

Qmax'(k) : représente la valeur maximale du K^{eme} critère de QOS, pour toutes les compositions possibles (de taille « taille-c »).

Qmin'(k)= taille-c*Qmin(k) ..... (5)

Qmax'(k)= taille-c*Qmax(k) ..... (6)

Qmin(k :) représente la valeur minimale du K^{eme} critère de QOS pour tous les services du corpus.

Qmax(k) : représente la valeur maximale du K^{eme} critère de QOS pour tous les services du corpus.

$$Qmin(k) = Min(Q_k(s_i)) \quad / \forall s_i \in corpus \quad \text{..... (7)}$$

$$Qmax(k) = Max(Q_k(s_i)) \quad / \forall s_i \in corpus \quad \text{..... (8)}$$

La fonction U2 permet de calculer la proximité sémantique entre les entrées de la requête et les entrées de la composition c. elle est spécifiée comme suit :

U2(c,requete)=

1 : si toutes les entrées de la composition sont équivalentes à celles de la requête : $\forall$ Ci $\in$ entrées(c) $\exists$ Cj $\in$ entrées(requete) tel que: Cj $\equiv$ Ci.(Ce score masque, le score suivant ie 0.75)

0.75 : si toutes les entrées de la composition sont équivalentes ou plus générales que celles de la requête : $\forall$ Ci $\in$ entrées(c) $\exists$ Cj $\in$ entrées(requete) tel que: Cj $\equiv$ Ci ou Cj $\subseteq$ Ci

-100 : s'il y a au moins une entrée de la composition qui ne subsume pas les concepts de la requête : $\exists$ Ci $\in$ entrées(c), et il n y a pas de Cj $\in$ entrées(requete) tel que, Cj $\equiv$ Ci ou Cj $\subseteq$ Ci (nous pénalisons la solution qui viole la compatibilité sémantique avec la requête)

La fonction U3 permet de calculer la proximité sémantique entre les sorties de la requête et les sorties de la composition c. elle est spécifiée comme suit :

U3(c,requete)=

1 : si toutes les sorties de la requête sont équivalentes à celles de la composition : $\forall$ Ci $\in$ sorties(requete) , $\exists$ Cj $\in$ sorties(c) tel que: Cj $\equiv$ Ci.(Ce score masque, le score suivant ie 0.75)

0.75 : si toutes les sorties de la requête sont équivalentes ou plus générales que celles de la composition : $\forall$ Ci $\in$ sorties(requete), $\exists$ Cj $\in$ sorties(c) tel que: Cj $\equiv$ Ci ou Cj $\subseteq$ Ci

-100 : s'il y a au moins une sortie de la requête qui ne subsume pas les sorties de la composition : $\exists$ Ci $\in$ sorties(requete) , et il n y a pas de Cj $\in$sorties(c) tel que : Cj $\equiv$ Ci ou Cj $\subseteq$ Ci (nous pénalisons la solution qui viole la compatibilité sémantique avec la requête)

La fonction U4 permet de mesurer la compatibilité sémantique entre les constituants de la solution c.

$U_4(c)=(1/(n-1))*\sum_{i=1}^{n-1}$ proximité (OComp(c,i+1),IComp(c,i+1))

proximité (C1,C2)= 1 si C1$\equiv$ C2
0.75 si C2 $\supseteq$ C1
-100 sinon // nous pénalisons la solution qui
Viole la compatibilité sémantique du flux de données interne.

OComp(c,k) dénote les sorties de tous les composants précédant le service k de c

IComp(c,k) dénote les entrées du K^{eme} composant de c.

Remarque

Pour étendre l'application de la fonction de similarité "proximité" sur deux ensembles de concepts X et Y (au lieu de deux concepts), nous utilisons l'algorithme 3 de ce chapitre, en remplaçant la mesure SimWp(Ci,Cj), par "proximité(Ci,Cj), en d'autres termes :

Proximité-Etendue(X,Y)= optimisation-matchings(X,Y)

III.3.2) Algorithme de Composition

Algorithme 4: Composition sémantique de services web à base d'essaims particulaires

- **Entrées :**
  - ✓ requête:
    - Un ensemble de concepts d'entrée de la requête :C1,…,Cn
    - Un ensemble de concepts de sortie de la requête :C'1,…,C'm
    - La taille Maximale de la composition :taille-c // taille des vecteur $X_i(t)$
    - Les contraintes globales:BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5
  - ✓ Le corpus des services:corpus
  - ✓ L'ontologie de domaine:O
- **Sorties:** une composition maximisant F2 (notée c)

1-Initialiser les individus (particules) de l'essaim P(t), avec des positions aléatoires $X_i(t)$ (t = 0).

2. évaluer la performance F2($X_i(t)$,requete) de chaque position $X_i(t)$.

3. Comparer la performance de chaque individu avec la meilleure position individuelle:

si F2($X_i(t)$,requete) > $pbest_i$ **Alors**

- ✓ $pbest_i$ = F2($X_i(t)$,requete) // MAJ de la meilleure performance individuelle
- ✓ $Xpbest_i = X_i(t)$ // MAJ de la meilleure position individuelle

4. Comparer la performance de chaque individu avec la meilleure position de l'essaim

si $pbest_i$ > gbest **Alors**

- ✓ gbest = $pbest_i$ // MAJ de la meilleure performance de l'essaim
- ✓ $Xgbest = Xpbest_i$ // MAJ de la meilleure position de l'essaim

5. Changer la position de chaque individu:

- ✓ Choisir une composante h1 de façon aléatoire de l'ensemble {1..taille-c}
- ✓ Choisir une composante h2 de façon aléatoire de l'ensemble {1..taille-c}

- ✓ $X_i(t)[h1] = Xgbest[h1]$ // mouvement social
- ✓ $X_i(t)[h2]$ = un service aléatoire ∈ corpus, ou -1// mouvement aléatoire

6. passer à l'itération suivante: t = t + l

7. **Aller à** 2, et **répéter jusqu'à** la convergence ou t=MaxIteration.

8. Retourner (Xgbest, gbest)

L'étape 1, initialise les constituants de la composition (solution) avec des valeurs aléatoires, si la valeur d'un constituant est égale -1 alors cette composante n'existe pas dans la solution, et le flux de données des constituants qui la précède, est véhiculé vers les composantes consécutives.

L'étape primordiale de cet algorithme est l'étape 5, elle consiste à réaliser deux mouvements:

Le mouvement social tend à se rapprocher de la meilleure position de l'essaim.

Le mouvement aléatoire remplace la composante h2 de la solution courante, par un autre service de la base, ou par -1 si nous voulons la supprimer.

La convergence veut dire que les performances moyennes, (au sens de F2) des essaims P(t) et P(t+1) sont presque identiques.

III.4) Deuxième Approche de Composition (BPSO)

Nous gardons, dans cette version [Hadjila et al, 2012 b] la même formalisation de la section III.3.1. Mais nous introduisons deux nouveaux mouvements, par rapport à la l'algorithme précédent.

Ces nouveaux mouvements, sont inspirés des comportements des boids de reynolds [Reynolds, 1987]. Le premier est nommé cohésion et le deuxième est appelé alignement.

Nous confirmons, dans le chapitre suivant, que l'hybridation des essaims particulaires avec les boids de reynolds (Boid Particle Swarm Optimization) permet l'amélioration des performances de l'optimisation.

Algorithme 5: Composition de services web à base d'essaims particulaires hybrides

- **Entrées :**
  - ✓ requête:
    - Un ensemble de concepts d'entrée de la requête :C1,…,Cn
    - Un ensemble de concepts de sortie de la requête :C'1,…,C'n
    - La taille Maximale de la composition :taille-c // taille des vecteur $X_i(t)$
    - Les contraintes globales:BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5
  - ✓ Le corpus des services:corpus
  - ✓ L'ontologie de domaine:O
- **Sorties:** une composition maximisant F2 (notée c)

1-Initialiser les individus (particules) de l'essaim P(t), avec des positions aléatoires $X_i(t)$ (t = 0).

2. évaluer la performance $F2(X_i(t),requete)$ de chaque position $X_i(t)$.

3. Comparer la performance de chaque individu avec la meilleure position individuelle:

si $F2(X_i(t),requete) > pbest_i$ **alors**

- ✓ $pbest_i = F2(X_i(t),requete)$ // MAJ de la meilleure performance individuelle
- ✓ $Xpbest_i = X_i(t)$ // MAJ de la meilleure position individuelle

4. Comparer la performance de chaque individu avec la meilleure position de l'essaim

si $pbest_i > gbest$ **alors**

- ✓ $gbest = pbest_i$ //MAJ de la meilleure performance de l'essaim
- ✓ $Xgbest = Xpbest_i$ //MAJ de la meilleure position de l'essaim

5. calcul de la cohésion de l'essaim

Pour k=1 jusqu'à taille-c

$Coh[k]= extraire\text{-}service\text{-}fréquent\ (X_i(t)[k])_{i=1..|P(t)|}$

6.calcul de l'alignement de l'essaim

Pour k=1 jusqu'à taille-c

$Align[k]= extraire\text{-}service\text{-}fréquent\ (Xpbest_i\ [k])_{i=1..|P(t)|}$

7. Changer la position de chaque individu:

- ✓ Choisir une composante h1 de façon aléatoire de l'ensemble {1..taille-c}
- ✓ Choisir une composante h2 de façon aléatoire de l'ensemble {1..taille-c}
- ✓ Choisir une composante h3 de façon aléatoire de l'ensemble {1..taille-c}
- ✓ Choisir une composante h4 de façon aléatoire de l'ensemble {1..taille-c}
- ✓ $X_i(t)[h1] = Xgbest[h1]$ // mouvement social
- ✓ $X_i(t)[h2]$ = un service aléatoire ∈ corpus, ou -1// mouvement aléatoire

- ✓ $X_i(t)[h3] = coh[h3]$ //cohesion
- ✓ $X_i(t)[h4] = align[h4]$ // alignement

8. passer à l'itération suivante: t = t + l

9. **Aller à** 2, et **Répéter Jusqu'à** la convergence ou t=MaxIteration.

10. Retourner (Xgbest, gbest)

Les étapes 1,2,3,4 sont similaires à l'algorithme 4.

L'étape 5 permet le calcul des valeurs les plus fréquentes des positions courantes.

L'étape 6 permet le calcul des valeurs les plus fréquentes des meilleures positions individuelles.

Le changement majeur de cette version réside dans l'étape 7, nous avons ajouté deux mouvements :

Le premier est appelé cohésion: il permet de se rapprocher du service, ayant la frequence la plus elevée (au niveau des particules courantes).

Le deuxième est appelé alignement : il permet de se rapprocher du service, ayant la frequence la plus elevée (au niveau des meilleures particules).

Le chapitre suivant, permet de comparer ces deux versions à l'aide du taux d'optimalité et du temps d'exécution.

IV) Sélection de Services Web Composés à Base de QOS

IV.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 QOS, et en satisfaisant des contraintes globales. [Zeng et al, 2003].Comme nous le montrons dans la figure VI.4, 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 et Dimopoulos, 2005] , et de ce fait il n' y a pas de solution polynomiale qui permet d'acquérir l'optimum global.

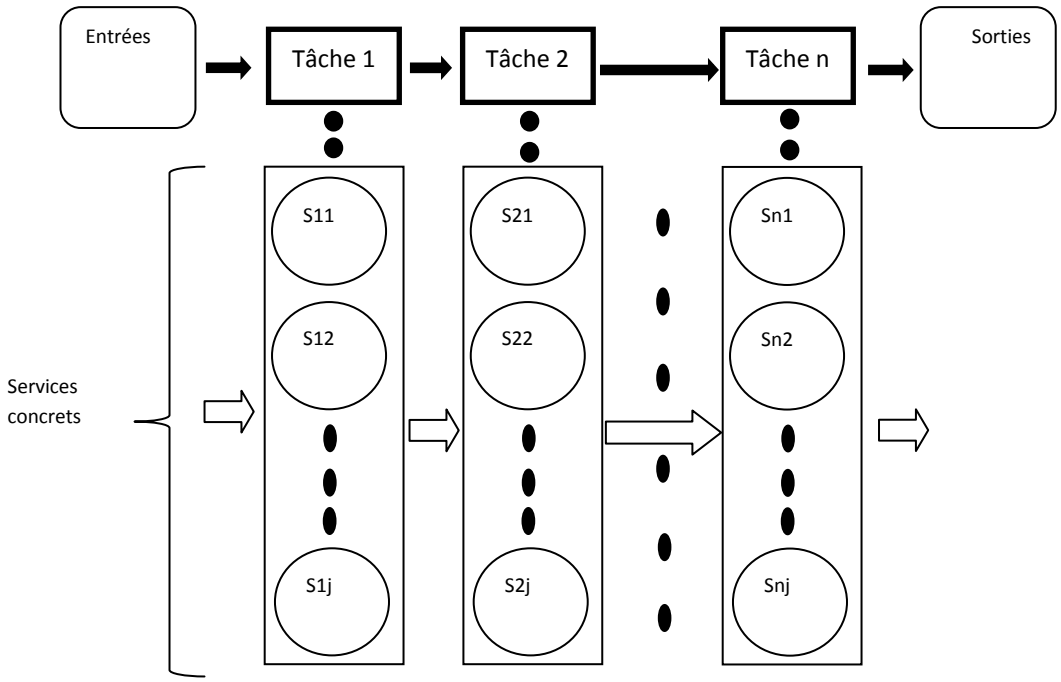


Figure VI.4. Le workflow abstrait

IV.2) Formalisation du Problème

Pour formaliser le problème de sélection de services à base QOS, nous devons concevoir une fonction objective qui tient compte des critères de QOS négatifs et positifs, ainsi que les contraintes globales.

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 :

Les valeurs minimales et maximales du K^{eme} critère de QOS d'une composition c sont :

$Qmin'(k) = \sum^{n}_{j=1} Qmin(j,k)$ ..... (9)

$Qmax'(k) = \sum^{n}_{j=1} Qmax(j,k)$ ..... (10)

$Qmin(j, k) = \min_{\forall sji \in Sj} Qk(sji)$ ..... (11)

$Qmax(j, k) = \max_{\forall sji \in Sj} Qk(sji)$ ..... (12)

Avec :

Qmin(j, k) est la valeur minimale du k^{eme} critère (e.g. le coût minimal) appartenant à la classe abstraite j.

Qmax(j, k) est la valeur maximale du k^{eme} critère (e.g. 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:

$U(s) = \Sigma^{R}_{k=1} w_k *(Q_k(s) - Qmin(j, k))/(Qmax(j, k) - Qmin(j, k))$ ..... (13)

$U'(c) = \sum^{R}_{k=1} w_k *((Q'_k(c) - Qmin'(k))/(Qmax'(k) - Qmin'(k))$ ..... (14)

Avec $w_k \in R+$ ($\sum^{R}_{k=1}(w_k) = 1$) représentent les poids (importance) 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^{eme} critère de QOS de la composition c (voir la table VI.2 pour plus d'informations.)

$Q_k(s)$: représente la valeur du K^{eme} critère de QOS d'un service concret s.

La fonction U est utilisée uniquement dans la cinquième approche proposée, alors que U' est utilisée dans toutes les approches proposées.

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' ∈ Sj tel que:

1. U'(c) est maximisée

2. Q' k(c) ≥ Cons(k), ∀ Cons(k)∈ CONS

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

Pour intégrer les contraintes globales dans la fonction objective, nous introduisons une fonction de pénalité notée P(c).

L'objectif de P(c), est de 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], [Adeli et Cheng, 1994], (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).

$P(c)= -\sum_{k=1}^{R} (D_k)^2(c)$ (pareille à la formule 2)

Avec

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

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) ..... (15)

Notre but, est de chercher la solution x qui maximise la fonction objective f (nommée aussi fitness). Pour cela nous avons proposé Cinq algorithmes, chacun d'eux est spécifié dans les sections suivantes.

IV.3) Motivations

La sélection des services composés à base de QOS est reconnue comme étant NP-hard, pour cela, nous devons proposer des techniques efficaces (en termes de temps), et qui peuvent atteindre des solutions quasi-optimales.

Pour ce faire, nous avons proposé quatre meta-heuristiques qui ont prouvé leur faisabilité et leur efficacité dans la pratique [Brownlee, 2011].

- La première est basée sur les algorithmes génétiques [Hadjila et al, 2011 b].

- La deuxième s'articule sur l'algorithme à base d'abeilles artificielles [Hadjila et al, 2013 b]
- La troisième s'articule sur l'optimisation à base d'essaims particulaires. [Hadjila et al, 2012 a]
- La quatrième est basée sur la sélection clonale [Hadjila et al, 2012 c]
- La cinquième approche est une combinaison de trois étapes (algorithmes) :

La première étape, extrait les services skylines de chaque classe abstraite en utilisant l'algorithme « Sort First Skyline » [Chomicki et al, 2003].

Les skylines [Alrifai et al, 2010] sont définis comme suit (nous supposons que tous les critères doivent être maximisés) :

Definition1 (skylines)

(Services Skyline) l'ensemble de skylines d'une classe S, dénoté par SL_S, est l'ensemble de services $\in$ S qui ne sont pas dominés par d'autres, i.e., $SL_S = \{x \in S / \nexists y \in S : y \succ x\}$.

Definition2 (dominance)

considérons deux services x, y $\in$ S, caractérisés par un ensemble de R valeurs de QoS . x domine y, (nous notons $x \succ y$) ssi

- x est meilleur ou égal à y dans tous les paramètres de QOS (les R valeurs).
- x est au moins meilleur par rapport à y dans un paramètre de QOS

i.e. :$\forall$ k $\in$ paramètres de QOS : $Qk(x) \geq Qk(y)$ et $\exists$ k $\in$ paramètres de QOS : $Qk(x) > Qk(y)$. [Alrifai et al, 2010]

La deuxième étape, permet le clustering hiérarchique ascendant à base ward [Ward, 1963] des éléments skylines de chaque classe abstraite.

L'avantage de cette méthode par rapport à celle utilisée dans [Alrifai et al, 2010], est qu'elle donne un résultat plus stable (la méthode K-means est toujours dépendante de la phase d'initialisation aléatoire, et de ce fait elle n'est pas stable).

Chaque cluster, possède un représentant (une instance de service) qui maximise la fonction U (présentée dans la formule 13).

Malgré sa gourmande consommation de temps, le clustering hiérarchique ascendant est toujours souhaitable parce qu'il s'exécute de façon hors-ligne (off- line).

La troisième étape, permet de vérifier si un ensemble de représentants (affiliés aux n classes abstraites) satisfont les R contraintes globales, si oui l'ensemble noté x est stocké comme une solution quasi-optimale, en plus l'optimum global est mis à jours si x est meilleur par rapport à l'ancien optimum au sens de U'.

Le parcours des représentants d'un même niveau (ou partition) se fait de manière systématique (exhaustif). Les avantages de cette approche sont résumés comme suit :

- Le calcul des skylines permet un premier l'élagage de l'arbre de recherche, (en effet le nombre de skylines est toujours réduit par rapport au nombre de services de la base)
- Le temps d'exécution associé au calcul des skylines n'est pas important parce que cette étape peut se faire en off-line (avant la présentation de la requête).
- Même chose concernant le clustering hiérarchique des skyline, il peut se faire en off-line.
- Il est prouvé dans [Alrifai et al, 2010] que la solution optimale (si elle existe) est composée forcement des éléments skylines des N classes abstraites.
- La recherche systématique des N hiérarchies permet un deuxième élagage de l'arbre de recherche, en effet l'obtention de la meilleure solution du niveau k, arrête la recherche (nous n'avons pas besoin de consulter les niveaux restants).
- L'obtention de l'optimum global (s'il existe) est toujours garantie.

IV.4) Algorithmes Proposés

IV.4.1) Algorithme Génétique

Le premier algorithme proposé se base sur les algorithmes génétiques [Holland, 1962], sa phase de reproduction est réalisée avec le modèle de tournoi binaire.

Le pseudo code est donné comme suit :

Algorithme 6 : Sélection de services en utilisant les algorithmes génétiques

Entrées :

- la requête R= <n=taille-comp, BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5> avec :
  - ✓ taille-comp : représente le nombre de tâches abstraites requises par l'utilisateur.// c'est aussi la taille de la position du chromosome i : $X_i(t)$.
  - ✓ BorneQOS1… BorneQOS5: représentent les exigences globales de l'utilisateur imposées sur les valeurs de QOS agrégées.
- Une base de services B segmentée en n classes (chaque service est caractérisé par R valeurs de QOS).

Sorties :

- Le meilleur chromosome: MC // (un vecteur maximisant la fonction f)

1. Initialiser la population de chromosomes, P, tel que la position xi(t) de chaque individu $p_i \in P$ est aléatoire (avec t = 0), nombre-chromosome=|P|.

2**. Tant Que** (t<=Tmax) **ou** (⌉convergence)

Faire

a. Evaluer la performance $f(x_i(t))$ de chaque chromosome $\in P$.

b. **P'**= reproduction(P)// selon le modèle de tournoi binaire

c. P' =croisement (P', taux de croisement)// variation de la population

d. P=mutation (p', taux de mutation)// évitement de minima locaux

e. MC= meilleur (P,MC) // mise à jour du meilleur chromosome MC

Fin_Faire

3. retourner **MC**

La condition « convergence » signifie que la performance moyenne (au sens de f) de deux populations consécutives P_t et P_{t+1} est presque la même.

Dans le chapitre suivant, nous évaluons cet algorithme en variant quelques paramètres tels que le nombre de chromosomes.

IV.4.2) Algorithme à Base d'Abeilles

Le deuxième algorithme proposé, se base sur le comportement d'abeilles lors de la récolte du nectar. Les abeilles constituent une colonie d'insectes auto-organisée, elles utilisent plusieurs sortes de moyens pour communiquer et partager l'information.

La recherche de l'optimum est basée sur la collaboration entre les abeilles butineuses, et les abeilles recrutées, ces groupes utilisent les relations de voisinages et les connaissances sociales (la dance) pour explorer les meilleures régions.

L'algorithme à base d'abeilles artificielles [Pham et al, 2006] est adapté à notre problème comme suit :

Algorithme 7 : Sélection de services en utilisant l'algorithme à base d'abeilles

Entrées :

- la requête R= <n=taille-comp, BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5> avec :
  - ✓ taille-comp : représente le nombre de tâches abstraites requises par l'utilisateur.// c'est aussi la taille de la position d'abeille i : $X_i(t)$.
  - ✓ BorneQOS1… BorneQOS5: représentent les exigences globales de l'utilisateur imposées sur les valeurs de QOS agrégées.
- Une base de services B segmentée en n classes (chaque service est caractérisé par R valeurs de QOS).

Sorties :

- La meilleure abeille : meilleure_abeille // (un vecteur maximisant la fonction f)

1. Initialiser la population d'abeilles, P, tel que la position xi(t) de chaque abeille $p_i \in P$ est aléatoire (avec t = 0), nombre-abeille=|P|.
2. **Tant Que** (t<=Tmax)

   Faire

   a. Evaluer la performance $f(x_i(t))$ de chaque abeille.

   b. **meilleure_abeille**= meilleure_solution(P)

   c. prochaine_generation= ∅

   d. meilleurs_sites=selection _meilleurs_sites(P,nombre_sites)

   e. **Pour** chaque $site_i \in$ meilleurs_sites **Faire**

Si i < nombre_elite_sites **Alors**

voisinage=creer_abeilles_voisines($site_i$, taille_patch,e_abeille)

prochaine_generation= prochaine_generation ∪ meilleur(voisinage)

Sinon

voisinage =creer_abeilles_voisines ($site_i$, taille_patch,o_abeille)

prochaine_generation= prochaine_generation∪ meilleur(voisinage)

Finsi

f. nombre_abeilles_restantes ← (nombre_abeilles- nombre_sites);

g. **Pour** j =1 to nombre_abeilles_restantes **Faire**

prochaine_generation= prochaine_generation ∪ Creer_abeille_aleatoire()

Fin_Faire

h. P ← prochaine_generation

i. passer à l'itération suivante: t = t + l

Fin_Tant Que

3. retourner **meilleure_abeille**

La boucle principale « tant que » contient plusieurs étapes:

Dans les phases **a** et **b,** nous trions la population selon la fitness $f(x_i)$, et nous mémorisons la meilleure abeille notée « **meilleure_abeille** »

La phase **c** initialise la nouvelle génération avec ∅.

Dans la phase **d** nous gardons les abeilles ayant les plus grands scores de fitness.

Dans la phase **e** nous décomposons l'ensemble précèdent en deux sous-ensembles: **élites_sites** et **autres**, sachant que **élites_sites** contient les abeilles ayant les meilleurs scores, et autres contient les abeilles ayant des scores moyens.

Pour chaque élément d'**élites_ sites**, nous générons **e_abeilles** voisins et nous insérons le meilleur d'entre eux dans la prochaine génération. (Les voisins sont générés par permutation des composantes).

Pour chaque élément de « **autres** », nous générons **o_abeilles** voisins et nous insérons le meilleur d'entre eux dans la prochaine génération. (Les voisins sont générés par permutation des composantes).

Dans les phases **f** et **g**, nous générons les abeilles restantes de façon aléatoire i.e. (**nombre_abeilles _restantes**=|**P**|-|**meilleurs_sites**|)

La phase **h** met à jours la population

Dans la phase **I** nous incrémentons le nombre d'itération et nous réexécutions la boucle jusqu'à la dernière itération.

Le choix des valeurs associées aux variables **nombre_sites, nombre_elites_sites, e_abeilles, o_abeilles, nombre_abeilles, taille_patch** est empirique (voir la partie expérimentation du chapitre suivant).

Par exemple **taille_patch** détermine le nombre maximal de voisins d'une abeille, plus il est grand plus les chances d'obtention d'une solution satisfaisante sont bonnes.

nombre_sites : représente le nombre d'abeilles qui méritent d'être examinées, afin de rechercher des optimums locaux.

nombre_elites_sites : représente une portion réduite d'abeilles qui ont une fitness élevée, et qui méritent d'être examinée attentivement (leur voisinage doit être parcouru intensivement). Et par conséquent nous extrairons **e_abeilles** (au lieu de **o_abeilles**) de ce voisinage, ensuite nous sélectionnons la meilleure solution pour qu'elle soit intégrée dans la nouvelle population. Nous notons que **o_abeilles < e_abeilles.**

Le même traitement est effectué pour l'ensemble « **autres** », sauf que nous extrairons **o_abeilles** du voisinage d'une abeille appartenant à l'ensemble **autres** :

Autres : est un ensemble d'abeilles ayant une fitness un peu réduite par rapport à ceux de **elites_sites**.

nombre_abeilles : représente la taille de l'essaim, plus il est grand, plus le temps d'exécution est élevé, et plus les chances d'obtention d'une solution satisfaisante sont bonnes.

IV.4.3) Optimisation à Base d'Essaims Particulaires

La troisième contribution emploie les essaims particulaires [Eberhart et Kennedy, 1995], le pseudo code de la version discrétisée est donné comme suit :

Algorithme 8: Sélection de services web à base d'essaims particulaires

- **Entrées :**
  - ✓ la requête R= <n=taille-comp, BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5> avec :
    - taille-comp : représente le nombre de tâches abstraites requises par l'utilisateur.// c'est aussi la taille de la position de la particule i : $X_i(t)$.
    - BorneQOS1… BorneQOS5: représentent les exigences globales de l'utilisateur imposées sur les valeurs de QOS agrégées.
  - ✓ Une base de services B segmentée en n classes (chaque service est caractérisé par R valeurs de QOS).
- **Sorties:** une composition maximisant f (notée c)

1-Initialiser les individus (particules) de l'essaim P(t), avec des positions aléatoires $X_i(t)$ (t = 0).

2. évaluer la performance $f(X_i(t))$ de chaque position $X_i(t)$.

3. Comparer la performance de chaque individu avec la meilleure position individuelle:

Si $F2(X_i(t),requete) > pbest_i$ **Alors**

- ✓ $pbest_i = f(X_i(t))$// MAJ de la meilleure performance individuelle
- ✓ $Xpbest_i = X_i(t)$ //MAJ de la meilleure position individuelle

Fsi

4. Comparer la performance de chaque individu avec la meilleure position de l'essaim

Si $pbest_i > gbest$ **Alors**

- ✓ $gbest = pbest_i$ //MAJ de la meilleure performance de l'essaim
- ✓ $Xgbest = Xpbest_i$ //MAJ de la meilleure position de l'essaim

Fsi

5. Changer la position de chaque individu:

- ✓ Choisir une classe abstraite h1 de façon aléatoire de l'ensemble {1..taille-comp}
- ✓ Choisir une classe abstraite h2 de façon aléatoire de l'ensemble {1..taille-comp}
- ✓ $X_i(t)[h1] = Xgbest[h1]$ // mouvement social
- ✓ $X_i(t)[h2]$ = un service aléatoire $\in$ classe abstraite h2 // mouvement aléatoire

6. passer à l'itération suivante: $t = t + 1$

7. **Aller** à 2, et **répéter jusqu'à** la convergence ou t=MaxIteration.

8. Retourner (Xgbest, gbest)

Cet algorithme est déjà expliqué dans la section de composition.

IV.4.4) Sélection Clonale

La quatrième contribution emploie la sélection clonale [De Castro, Von Zuben, 2000], le pseudo code est adapté à notre problème comme suit :

Algorithme 9: Sélection de services web à base de sélection clonale

- **Entrées :**
  - ✓ la requête R= <n=taille-comp, BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5> avec :
    - taille-comp : représente le nombre de tâches abstraites requises par l'utilisateur.// c'est aussi la taille de la position de la cellule i : $X_i(t)$.
    - BorneQOS1… BorneQOS5: représentent les exigences globales de l'utilisateur imposées sur les valeurs de QOS agrégées.
  - ✓ Une base de services B segmentée en n classes (chaque service est caractérisé par R valeurs de QOS).
- **Sorties:** une composition maximisant f (notée c)

1. Initialiser les cellules B notées « a=(Ins1,Ins2….Ins_n) » de la population avec des valeurs aléatoires.// Insk ∈ {1,2,….nombre_instances}
2. Initialiser le compteur d'itérations à 1 (t=1)
3. **Tant que** (t<=T_Max) **Faire**
   a. pour chaque celluleB (notée a) ∈ Population : évaluer son affinité f(a) (ou sa performance).
   b. Calculer l'ensemble "clones-set" pour chaque a, selon le clonning_rate // duplication des cellules selon le taux de clonage.
   c. faire l'hypermutation pour chaque cellule B de clones-set et calculer sa nouvelle affinité f(à) // variation des cellules clonées
   d. Union=Population ∪ Clones_set.
   e. 5. Trier Union selon l'ordre décroissant des affinités.

f. 6. Mettre à jour la population, en retenant les M premières cellules B de « union ». (M est la taille de Population)// les meilleures cellules B sont gardées.

g. créer L cellules B de façon aléatoire (L=M*insertion_rate). // génération aléatoire de cellules B.

h. remplacer les L dernières cellules B de Population (en termes d'affinité) par les cellules de l'étape g. // remplacement des mauvaises cellules B par des cellules aléatoires

i. t=t+1

j. Fin Tant-Que

4. Retourner la meilleure cellule B de la population

Le principe fondamental de cet algorithme est de créer plusieurs copies de chaque cellule B (ou composition), ensuite chacune d'elle va subir une hypermutation, i.e. un changement de valeur, l'hypermutation est inversement proportionnelle avec l'affinité (ou la fitness) de la composition. Enfin une petite partie de la population est remplacée par des cellules aléatoires

Cet algorithme possède plusieurs paramètres de contrôle, tels que nombre de cellules B, taux de clonage (clonning_rate), taux d'insertion aléatoire.

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

IV.4.5) Hybridation Clustering Hiérarchique-Recherche Systématique

Comme nous l'avons mentionné dans la section de motivation, cette approche est constituée de trois étapes :

- Extraction des skylines à base de l'algorithme SFS, pour chaque classe abstraite.
- Le clustering hiérarchique ascendant des skylines, de chaque classe.

- Et enfin le parcours systématique des différentes hiérarchies.

L'algorithme suivant « Sort First Skyline » [Chomicki et al, 2003], permet l'extraction des skylines à partir d'une collection de services, chacun d'eux est représenté par un vecteur de R valeurs de QOS.

Algorithme 10 : Extraction des skyline à base de SFS

Entrées

- Une classe de services B (chaque service est caractérisé par R valeurs de QOS).

Sorties

- Un ensemble de skylines Sk (qui ont les hautes valeurs dans la majorité des dimensions)

1. trier les services de la base B, par ordre décroissant selon une fonction objective monotone « f » (f doit agréger les R valeurs de QOS, par exemple en calculant la moyenne de R valeurs de QOS).

2. mettre le résultat de tri dans une liste L, t=0, Sk=∅ ;
3. **Tant Que** (t<|L|)
   Faire
   a. extraire le service L(t)
   b. **si** ∃ e ∈ Sk tel que e › L(t) **Alors** négliger L(t) et passer au suivant (i.e. t=t+1) .
      Sinon
         si ∃ e ∈ Sk tel que L(t) › e **Alors** remplacer e par L(t) et passer au suivant
         Sinon ajouter L(t) à Sk et passer au suivant (i.e. t=t+1)
   F**in_Faire**
4. retourner Sk

En triant les services durant l'étape 1, nous accélérons l'extraction des skylines, car les éléments des premiers rangs ont de fortes chances pour être des skylines. (Puisque la valeur de f est élevée).

L'étape **2.b alors** : signifie que L(t) est déjà dominé par des éléments de Sk.

L'étape **2.b sinon** : signifie que L(t) est un skyline

Dans la suite, nous présentons l'algorithme de clustering hiérarchique ascendant (à base de Ward).il est appliqué sur toutes les classes abstraites Sj.

Algorithme 11 : Clustering hiérarchique ascendant (à base de ward)

Entrées

- Un ensemble de services B (chaque service est caractérisé par R valeurs de QOS).

Sorties

- Une hiérarchie de clusters notée HC

1. Calculer la matrice de similarité entre les services de B (en utilisant la distance de ward)
2. Affecter chaque service Si à un cluster Ci, et initialiser HC avec ces clusters (le représentant de chaque cluster Ci est le service Si)
3. **Répéter**
4. fusionner les deux clusters les plus proches et mettre à jours le représentant du nouveau cluster
5. Mise à jour de la matrice de similarité et de HC
6. **Jusqu'à** l'obtention d'un seul cluster
7. Retourner HC

L'algorithme combine à chaque fois deux clusters (ou deux services) qui présentent, la plus petite distance de ward dans la matrice de similarité (voir la figure VI.5).

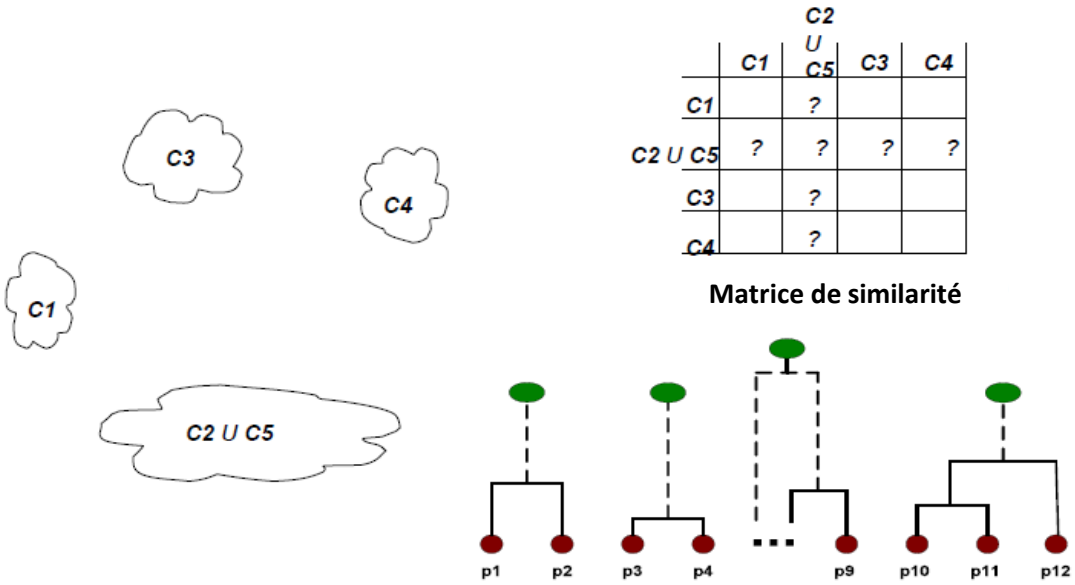


Figure VI.5. Principe de clustering hiérarchique à base de ward

La formule de Ward est une fonction calculant une distance équilibrée, entre deux ensembles est données (voir la figure suivante), elle permet de minimiser l'inertie intraclasses.

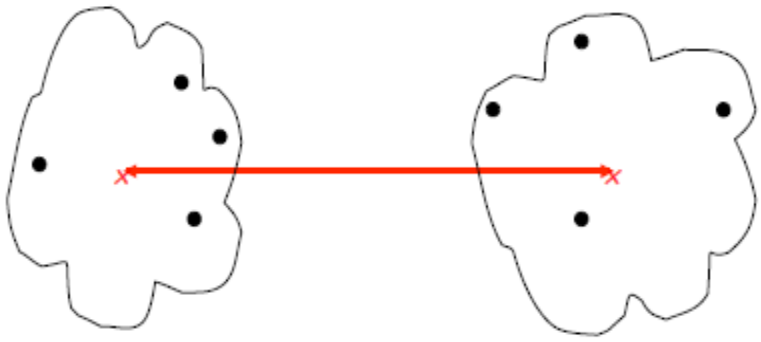


Figure VI.6. La Distance de Ward

Formellement la distance est donnée comme suit :

Distance Ward (A,B)=((|A|*|B|)/(|A|+|B|)) * || gA – gB||²

Avec gA, gB : sont les centres de gravité des ensembles A, B.

Le représentant d'un cluster Ci est un service S qui maximise la fonction U (voir la formule 13). i.e. representant(Ci)= ArgMax U(S_m), avec $S_m \in$ Ci.

La figure VI.7 montre un exemple d'hiérarchie de skylines (associée par exemple à la quatrième classe. Chaque nœud (ou cluster) représente un ensemble des services skylines similaires.

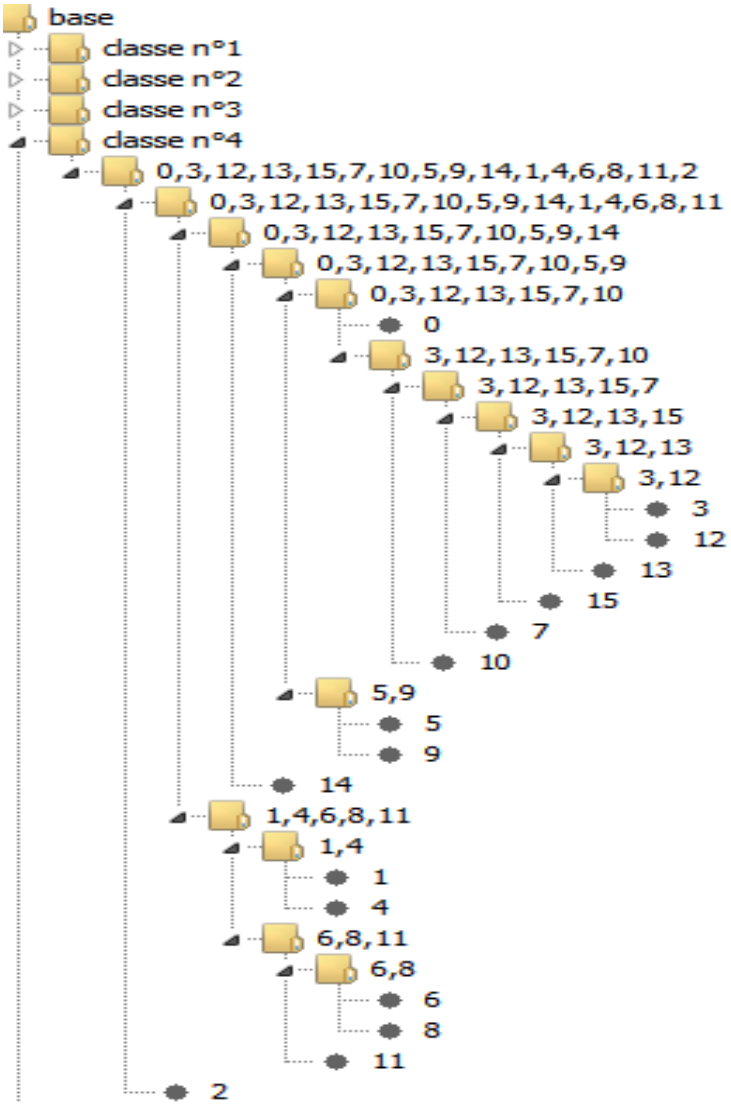


Figure VI.7. Un exemple d'hiérarchie de services

La troisième et la dernière étape de cette approche, permet de parcourir les N hiérarchies de clusters obtenus lors de l'étape précédente. Pour ce faire, nous appliquons une recherche systématique pour tous les représentants du niveau courant.

Notre algorithme parcourt les N hiérarchies niveau par niveau, (jusqu'au dernier, s'il n'y a pas de solutions rencontrées auparavant). La présence d'une solution satisfaisant les contraintes globales dans le niveau K annule la recherche dans les niveaux ultérieurs.

On se contente de parcourir uniquement ce niveau de manière exhaustive. La solution finale représente l'optimum global.

Si nous n'aurons pas de solution satisfaisante (faisable) après le parcours du dernier niveau, alors nous confirmons l'absence de solutions pour ce problème.

Algorithme 12 : Recherche systématique

Entrées

- N hiérarchies de clusters de services (chaque cluster est caractérisé par un représentant).
- Dernier_niveau=$Max_{i\in\{1..N\}}$ (profondeur(HC_i))// avec HC_i est la hiérarchie de clusters associée à la i^{eme} classe abstraite.

Sorties

- L'optimum global : og (s'il existe), sinon « échec»

1.og=echec // nous initialisons l'optimum global à « échec »

2. initialiser le niveau courant : niveau_courant =1 (la racine de chaque arbre)

3. pour j=1 jusqu'à N

Lj=representants_classe(j,niveau_courant)

4. **Tant que** ((niveau_courant≠ dernier_niveau+1) et (og=echec))

Faire

a- composition=recherche_exhaustive (L_1,…L_N)

b- **Si** composition = échec **Alors //** les contraintes globales ne sont pas vérifiées.

1- niveau_courant=niveau_courant+1

2- pour j=1 jusqu'à N : Lj=representants_classe(j,niveau_courant)

Sinon

og= composition

fin_Faire

5. Retourner og

L'étape 1 initialise l'optimum global.

L'étape 2 initialise le niveau courant (en considérant les racines des N arbres).

L'étape 3 extrait les représentants du niveau 1 (des N hiérarchies). Pour ce niveau nous avons un seul représentant par hiérarchie (ou classe abstraite), c'est le service qui maximise la fonction U (formule 13). Pour les niveaux consécutifs nous pouvons avoir jusqu'à 2^{x-1} représentants, sachant que x est le numéro de niveau.

Le niveau K de la hiérarchie j peut avoir jusqu'à 2^{k-1} clusters, chacun d'eux possède un seul service représentant. (Celui qui maximise la fonction U).

L'étape 4 applique une recherche exhaustive sur les représentants des N classes. Plus précisément, l'étape 4-a parcourt toutes les solutions possibles à ce niveau en vérifiant les contraintes globales, et en mettant à jours la solution finale dans le cas où la composition courante est meilleure en termes de fonction U'(formule 14).

L'étape 4 fournit soit échec (si les contraintes globales sont violées) ou la solution optimale. Dans le cas d'échec nous parcourons le niveau suivant.

L'étape 5 donne la solution trouvée.

V) Conclusion

Nous avons présenté dans ce chapitre les solutions proposées pour le problème de découverte, de composition, et de sélection de services.

Nous avons proposé deux algorithmes pour la recherche sémantique des services, le premier est basé sur l'indexation conceptuelle et la mesure cosinus, le deuxième combine la mesure de similarité de [Wu et Palmer, 1994] avec un algorithme d'optimisation des matchings.

Pour le cas de composition, nous avons proposé deux meta-heuristiques basées sur les essaims particulaires, ces approches modélisent la composition sous forme d'un problème d'optimisation avec contraintes.

Nous avons présenté aussi cinq algorithmes pour le problème de sélection services. La première est basée sur les algorithmes génétiques, la deuxième est inspirée du comportement de colonies d'abeilles.

La troisième est basée sur les essaims particulaires, la quatrième adopte la sélection clonale, et la dernière hybride le clustering hiérarchique des skylines avec la recherche systématique.

Chapitre 7 :

Implémentation et expérimentation

Sommaire

I) Introduction

Dans ce chapitre nous décrivons un prototype implémentant l'environnement de recherche, de composition et de sélection de services web. Nous décrivons en premier lieu, l'architecture générale du système proposé (i.e. les différents composants 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.

II) Présentation de l'Environnement de Développement

II.1) Préambule

Nous avons développé un prototype nommé, il est implémenté avec java (JDK 1.7.0) et netbeans 7.0, dans ce qui suit, nous décrivons ses principales fonctionnalités.

II.2) Fonctionnalités du Système

Le system EDCSS (Environnement de Découverte, de Composition et de Sélection de Services) est présenté dans la figure VII.1, il est composé des modules suivants :

II.2.1) 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.

II.2.2) Module de Génération des Compositions de Services (Conversions en Format BPEL)

Ce module permet la génération des orchestrations BPEL, à partir d'une représentation interne de la composition de services (concrètement un vecteur d'instances de services).

II.2.3) 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 propose deux algorithmes : le premier est basé sur la représentation vectorielle et la mesure « cosine », le deuxième adopte la matrice des liens sémantiques et la mesure de « Wu&Palmer ».

II.2.4) Module de Composition de Services

Ce composant peut être sollicité si le module de découverte échoue (i.e. qu' il n' y a pas de service individuel qui satisfait la requête). Nous proposons deux algorithmes pour composer séquentiellement les services sans états : le premier est basé sur les essaims particulaires, et le deuxième est une hybridation des essaims avec les boids de reynolds.

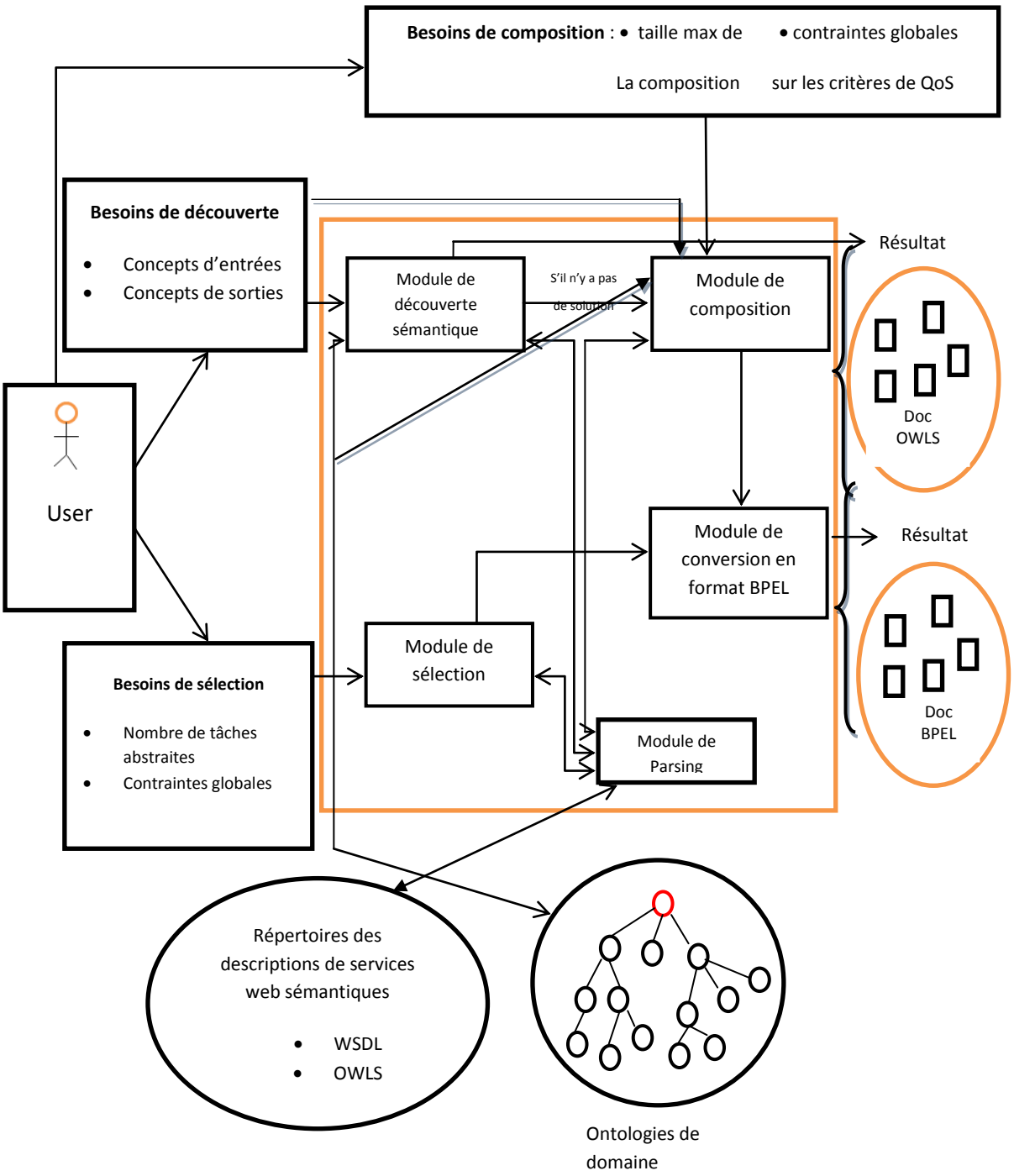


Figure VII.1. L'architecture du système proposé

II.2.5) Module de Sélection de Services

Ce module effectue une recherche afin d'instancier les tâches abstraites avec des services concrets. Plus précisément, il optimise une fonction objective, 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. Plusieurs algorithmes sont offerts pour réaliser cet objectif (les algorithmes génétiques, les essaims particulaires, la sélection clonale, l'algorithme à base d'abeille, l'hybridation clustering-recherche systématique).

Notre système utilise plusieurs sortes d'entrées (base de services, ontologies, requête-utilisateur), elles sont décrites comme suit :

II.2.6) 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).

Les documents WSDL de la base de composition, possèdent des annotations supplémentaires, en effet chaque paramètre d'entées-sorties, est annoté avec un concept de l'ontologie de domaine, le document WSDL contient aussi un élément décrivant les 05 critères de QOS.

II.2.7) Collection d'Ontologies

Les raisonnements sémantiques effectués lors de la découverte ou la composition, emploient des ontologies offertes par les collections de tests. L'objectif principal de ces ontologies, est de nous permettre la vérification des relations de subsomption entre concepts.

II.2.8) Requête d'Utilisateur

Nous distinguons trois 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 composition, la requête R =< entrées-ens, sorties-ens, taille-max, BorneQOS1, BorneQOS2, Borne QOS3, BorneQOS4, BorneQOS5>
  - ✓ entrées-ens, sorties-ens : représentent les concepts d'entrées et de sorties de la composition recherchée.
  - ✓ taille-max : représente le nombre maximal des services constituants la composition. Les valeurs de cette variable, sont comprises entre 1 et le nombre total des services du corpus.
  - ✓ BorneQOS1… BorneQOS5: représentent les exigences globales de l'utilisateur imposées sur les valeurs de QOS agrégées. Ils représentent les bornes minimales ou maximales que doit satisfaire la composition pour chaque critère de QOS.
- 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.

III) Corpus Utilisés

Nous décrivons dans ce qui suit, les différentes bases employées pour évaluer les approches proposées. Nous avons 03 bases, chacune d'elles est associée à une problématique donnée (la découverte, la composition ou la sélection).

III.1) Corpus de Découverte

Nous avons utilisé un extrait du corpus open source OWLS-TC version 2.2.1[14] 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 07 classes : le domaine d'éducation, domaine médical, le domaine de nourriture, le domaine militaire, le domaine de voyages (tourisme), le domaine de communication, et le domaine d'économie.

[14] projects.semwebcentral.org/projects/owls-tc/

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 reparties sur les 07 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 relevant ou non par rapport à une requête donnée. En d'autre termes, chaque service web possède une étiquette binaire (relevant 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

III.2) Corpus de Composition

Le web service challenge[15] est un corpus synthétique destiné pour l'évaluation des approches de composition de services sans états. La version de l'année 2008 WSC contient 05

[15] http://www.ws-challenge.org/

sous corpus qui ont des tailles différentes (nombre de services, nombre de concepts d'ontologie), chacun d'eux dispose d'une seule requête utilisateur qui comporte un ensemble de concepts d'entrées et de sorties, nous avons augmenté cette requête par un ensemble de bornes (05 nombre réels) sur les 05 critères de QOS adoptés. Ces bornes représentent les contraintes globales de l'utilisateur.

Chaque service web ou requête possède une interface WSDL dont les paramètres d'entrées/sorties sont annotés avec des ontologies OWL. Le port-type contient une seule opération. Nous avons augmenté chaque service (document WSDL) par un vecteur de 05 valeurs de QOS. Ceci permet l'optimisation des compositions. Les intervalles de ces paramètres de QOS sont spécifiés dans la table VII.1.

Le sous corpus utilisé est généré aléatoirement, il contient 158 services, une seule requête, et une seule ontologie. Cette dernière possède 500 concepts, la requête accepte plusieurs solutions. L'espace de recherche exploré est très large, en effet nous avons 158!+158*157*..*2+....+158*157+158 Solutions candidates. La solution optimale de la base notée c*, est une composition qui satisfait les entrées et les sorties de la requête avec le score exact, i.e. qu'il y a une équivalence entre les entrées (respectivement les sorties) de c* et la requête, en plus elle maximise les critères de QOS positifs, et minimise les critères de QOS négatifs.

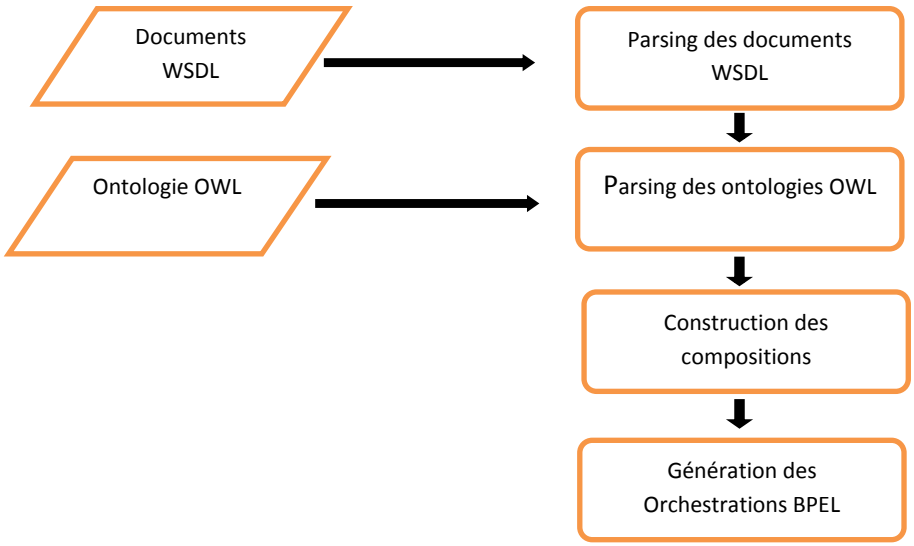


Figure VII.2. Une vue générale du défi de composition de services (WSC08)

Pour plus d'information sur la structure de la requête et les services, veillez consulter l'url http://www.ws-challenge.org/.

III.1) 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 constitué de :

- 05 nombres qui représentent les contraintes globales de l'utilisateur.
- un entier noté n qui indique le nombre de tâches abstraites (voir la figure VI.4).

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

Coût (sol)<= borne1

Temps-exécution(sol)<= borne2

Réputation(sol)>= borne3

Disponibilité(sol)>= borne4

Fiabilité(sol)>= borne5

Chaque tâche abstraite, 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 100 et 1000 services.

A Chaque instance nous associons un vecteur de 05 paramètres de QOS (le temps d'exécution, le coût, la fiabilité, la disponibilité, la réputation).

Les intervalles de valeurs de chaque paramètre sont inspirés de [Yu et Bouguetaya, 2010] (voir la table VII.1).

Pour des raisons techniques, nous avons utilisé les valeurs log(fiabilité) et log (disponibilité) au niveau des approches de composition et de sélection, au lieu des valeurs originales de ces critères. Ceci permet l'uniformisation des fonctions d'agrégations, i.e. que tous les critères de QOS vont utiliser la somme comme fonction d'agrégation, au lieu du produit. (Notons que log (a.b)=log a+ log b).

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

Table VII.1. Les intervalles des paramètres de QOS de la base de sélection [Yu et Bouguetaya, 2010]

IV) Expérimentation

Dans cette section, nous décrivons les expériences permettant l'analyse des performances des approches proposées.

Ces expérimentations sont menées sur une plateforme, ayant processeur Corei3 avec 04 GO de RAM et sous le système d'exploitation Windows7. Le système est développé avec java (jdk 1.7.0) et sous l'environnement Netbeans 7.0.

Pour le cas de la découverte, nous étudions les taux de rappels et de précisions pour les deux approches (l'algorithme à base de mesure « cosine » et l'algorithme à base de mesure « Wu&Palmer »), nous mesurons aussi les temps d'exécutions associés.

Pour le cas de la composition, nous étudions les taux d'optimalité et les temps d'exécutions pour les deux approches proposées (Swarm Particle Optimization et Boid swarm particle Optimization).

Pour le cas de sélection, nous évaluons les taux d'optimalité et le temps d'exécution pour les différentes approches proposées (les essaims particulaires (ou PSO), les algorithmes génétiques, la sélection clonale, l'algorithme à base d'abeilles, l'hybridation clustering-recherche systématique). 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 (pour le cas de découverte, composition ou sélection).

Le taux d'optimalité : il 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*).

IV.1) Performances des Approches de Découverte

Contexte

La table IIV.2 montre les requêtes utilisées, pour évaluer les approches (la première est basée sur la mesure cosine et la deuxième est basée sur la mesure Wu&Palmer).

Name	Inputs	Outputs
Car_price_service (R1)	Car	Price
Grocerysore_food_serveice (R2)	Grocerystore	Food

Table IIV.2. Les requêtes de la découverte

Résultats

Les tables IIV. 3 et IIV.4 représentent un échantillon des résultats délivrés par la première approche (basée sur « cosine »). Les colonnes nommées score1, score2, score représentent les valeurs calculées par l'algorithme 1 du chapitre 6.

L'avant dernière colonne, représente la décision de l'algorithme 1, en utilisant un seuil =0.8, et la dernière représente l'avis de l'expert humain.

Le nom du service	Les entrées du service	score1	Les sortie du service	Score2	Score	La décision du système	La décision de l'expert humain
3wheeledcar_price	3wheeledcar	0.953	price	1.0	0.976	Accepté	Accepté
1personbicyclecar_price	_4WheeledCar, _1personbicycle	0.809	price	1.0	0.904	Accepté	Rejeté
car_price	Car	1.0	price	1.0	1.0	Accepté	Accepté
3wheeledcaryear – Recommendedprice	3WheeledCar, Year	0.725	Recommended Price	0.5	0.612	Rejeté	Accepté
3WheeledAudiCarprice	-	0.0	price	1.0	0.5	Rejeté	Accepté
food Exportservice	-	0.0	food	0.0	0.0	Rejeté	Rejeté
drugstore tea	DrugStore	0.463	Tea	0.0	0.231	Rejeté	Rejeté
grocerystore_ butterquantity	GroceryStore	0.463	Butter, Quantity	0.0	0.231	Rejeté	Rejeté
grocerystore_ food_service	GroceryStore	0.463	food	0.0	0.231	Rejeté	Rejeté
retailstore_food quality_service	RetailStore	0.467	Food, Quality	0.0	0.233	Rejeté	Rejeté

Table IIV.3.Quelques résultats associés à la requête R1 (l'approche basée sur Cosine)

Le nom du service	Les entrées du service	score1	Les sortie du service	Score2	Score	La décision du système	La décision de l'expert humain
3wheeledcar_price	3wheeledcar	0.442	Price	0.0	0.221	Rejeté	Rejeté
1personbicyclecar_price	_4WheeledCar, _1personbicycle	0.625	Price	0.0	0.312	Rejeté	Rejeté
car_price	Car	0.463	Price	0.0	0.231	Rejeté	Rejeté
3wheeledcaryear – Recommendedprice	3WheeledCar, Year	0.336	RecommendedPrice	0.0	0.168	Rejeté	Rejeté
3WheeledAudiCarprice_	-	0.0	Price	0.0	0.0	Rejeté	Rejeté
food Exportservice	-	0.0	Food	1.0	0.5	Rejeté	Rejeté
drugstore tea	DrugStore	0.985	Tea	0.932	0.958	Accepté	Accepté
grocerystore_ butterquantity	GroceryStore	1.0	Butter, Quantity	0.745	0.872	Accepté	Accepté
grocerystore_ food_service	GroceryStore	1.0	Food	1.0	1.0	Accepté	Accepté
retailstore_food quality_service	RetailStore	0.992	Food, Quality	0.912	0.952	Accepté	Accepté

Table IIV.4.Quelques résultats associés à la requête R2 (l'approche basée sur Cosine)

Dans ce qui suit nous étudions l'influence du seuil de rétention des résultats sur les taux de rappel et de précision associés aux requêtes de la table IIV.2.

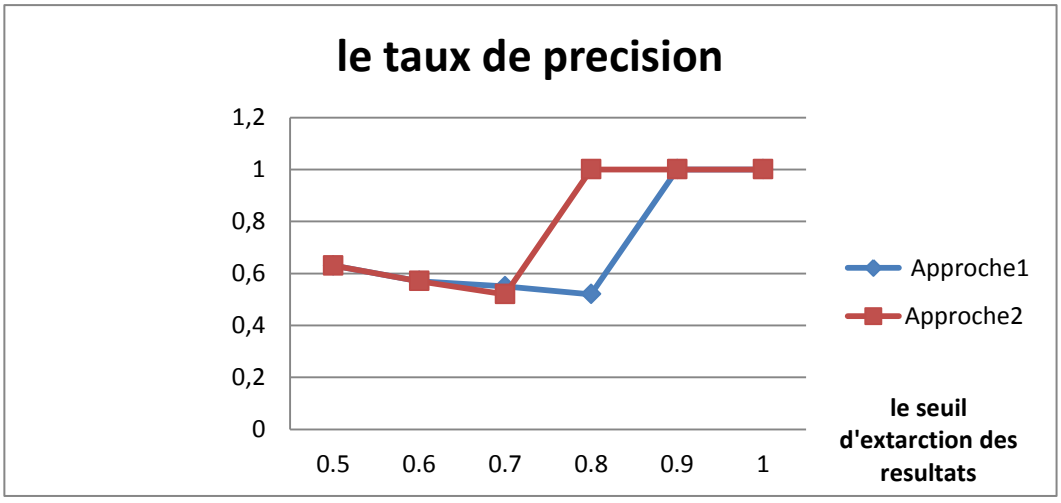


Figure VII.3. La précision associée à la requête R1

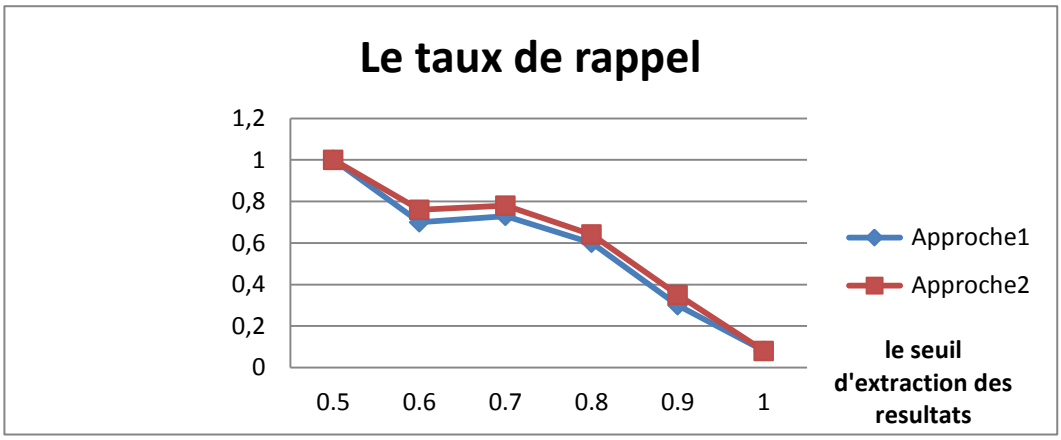


Figure VII.4. Le rappel associé à la requête R1

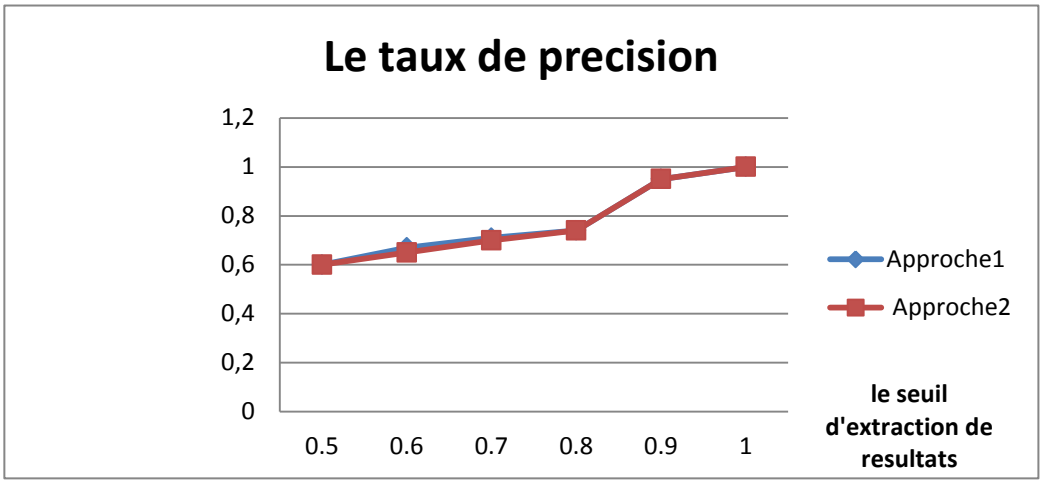


Figure VII.5. La précision associée à la requête R2

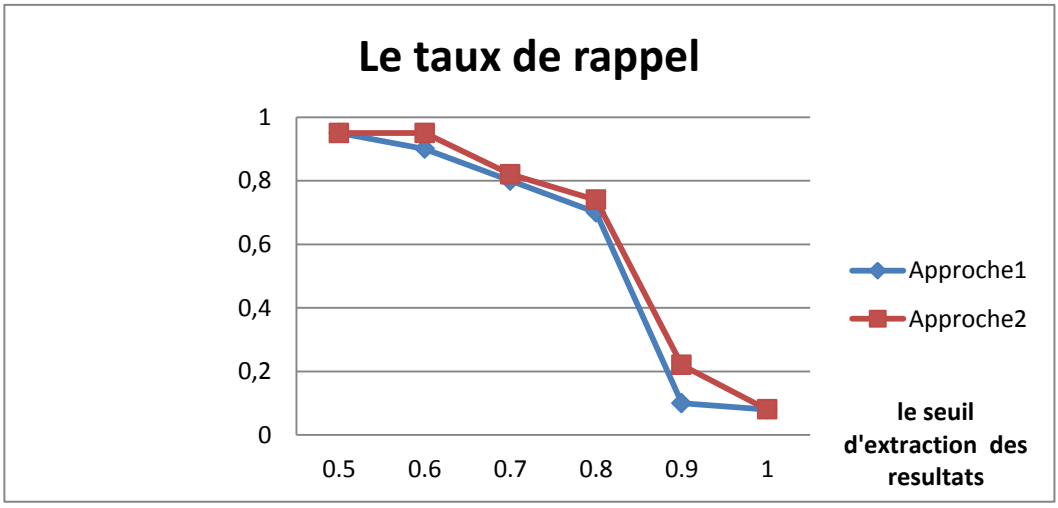


Figure VII.6. Le rappel associé à la requête R2

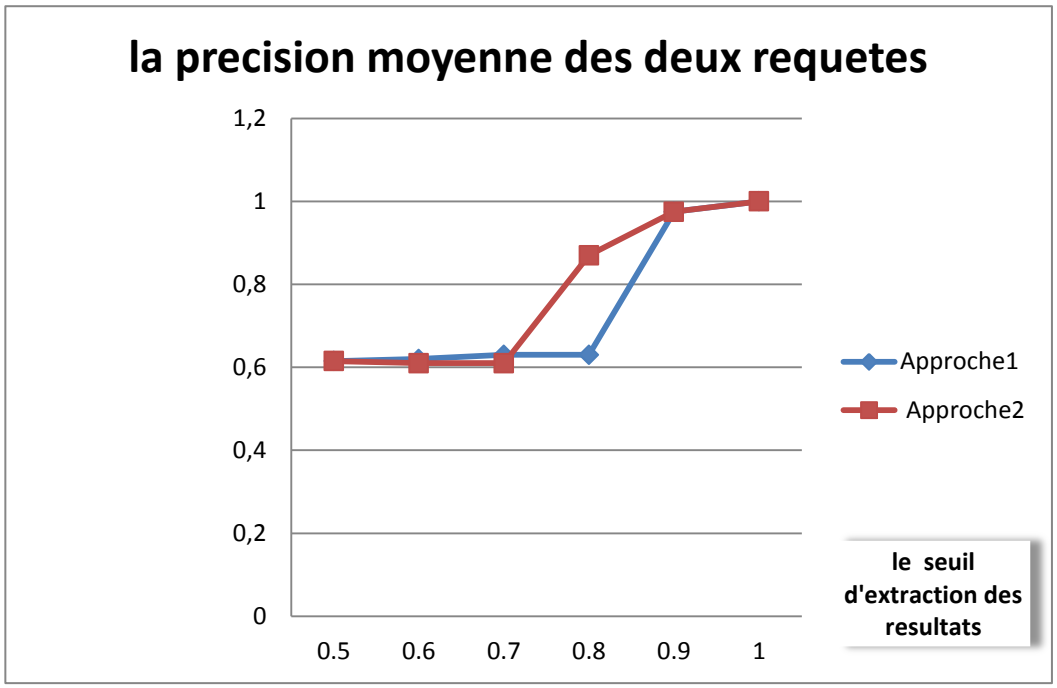


Figure VII.7. La précision moyenne

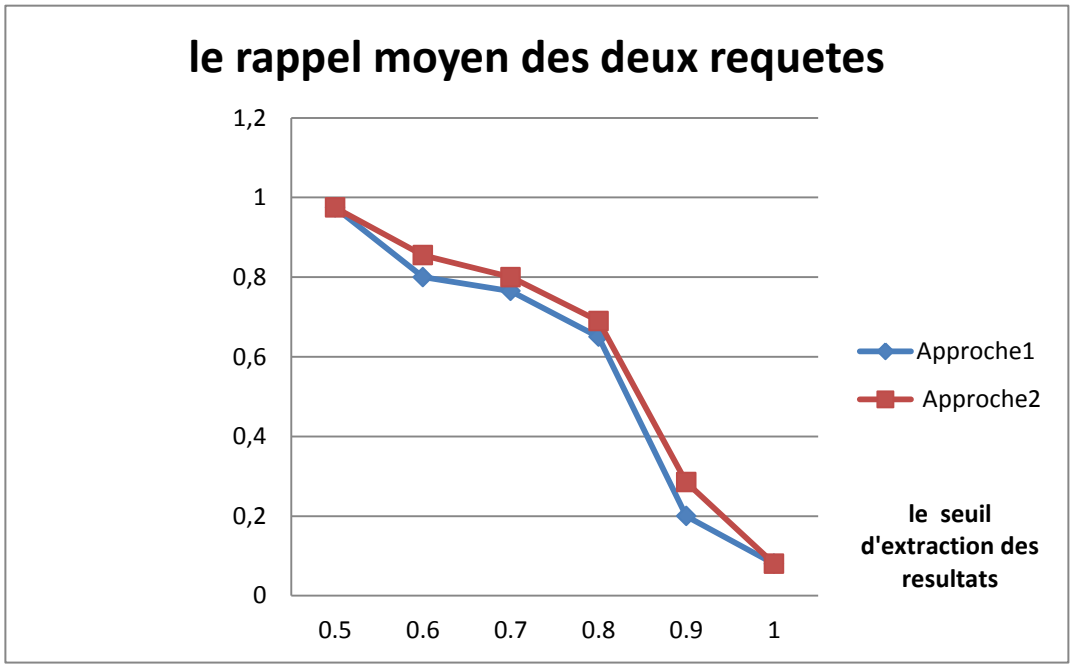


Figure VII.8. Le rappel moyen

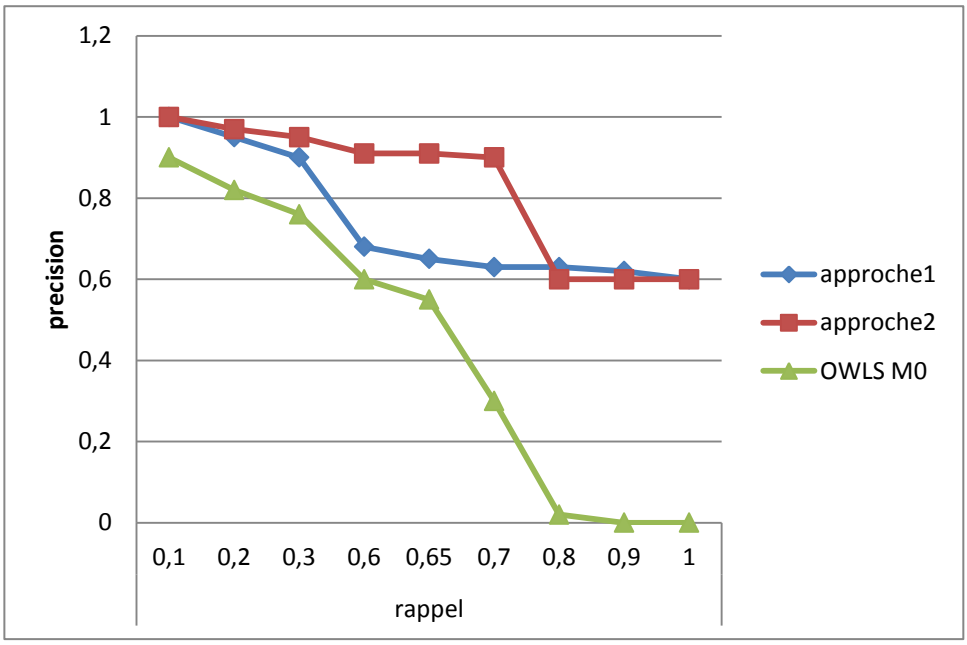


Figure VII.9. Comparaison rappel-précision: approche1 vs approche2 vs OWLS M0

Discussion

Nous remarquons que les deux approches ont des performances similaires en termes de précision, par contre nous remarquons une légère supériorité de l'approche 2 par rapport à la première en termes de rappel.

Nous notons aussi que les approches (la mesure cosine et Wu & Palmer) donnent des scores très rapprochés, si les services (et la requête) ont un seul concept comme entrée ou sortie. Et de ce fait, nous avons eu des performances presque identiques surtout pour la précision.

Par contre, si nous avons plusieurs concepts dans les entrées/sorties, nous constatons que l'approche 2 est plus performante que l'approche1, ceci est dû à l'application de l'algorithme d'optimisation (algorithme 3 chapitre 6). Ce dernier affecte les éléments de la requête aux éléments des services de manière quasi-optimale.

Par conséquent, les scores de matchings seront un peu élevés. (Et donc, le nombre de faux négatifs sera réduit).

Par opposition, l'approche1 n'a pas d'algorithme d'optimisation des matchings, elle fusionne tous les concepts d'entrées (respectivement de sorties) dans un seul vecteur et

applique la mesure cosine entre la requête et le service, ceci induit des scores assez faibles à cause du caractère creux de ces vecteurs, et par conséquent nous aurons des rappels réduits.

Pour élucider, ces remarques nous examinons l'exemple suivant :

Soit S1=<entrées={retailstore},sorties={butter,quantity}>, l'expert humain considère S1 comme une réponse correcte par rapport à la requête R1=<{grocerystore},{food}>.

Pour la mesure W&Palmer notée WP

SimWP(grocerystore,retailstore)=0.93

SimWP(food,butter)=0.92

SimWP(food,quantity)=0,4

L'algorithme d'optimisation (algorithme 3) retient la paire (food,butter) au lieu de (food,quantity).

Donc SimWP(entréeR1, entréesS1)=0.92 et la similarité finale entre S1 et R1 est :(0.93+0.92)/2=0.925

Pour la mesure cosine

Cosine(grocerystore,retailstore)=0.93

Cosine (food,{butter,quantity})=0.7

La similarité finale entre S1 et R1est : (0.93+0.7)/2=0.82

Si nous considérons un seuil=0.9 alors S1 est n'est pas retenu par l'approche1, alors qu'il est retenu par l'approche 2. Ceci induit une augmentation du rappel de la deuxième approche par rapport à la première.

La Figure VII.9 compare les approches OWLSM0 [Klusch et al, 2006], approche1 et approche2, en utilisant les critères de rappel et de précision. La figure confirme la supériorité de l'approche2 par rapport à l'approche1 et OWLSM0. En effet le matching logique (test de subsomption) est trop rigide.

Par exemple, il ne tolère pas les concepts voisins lors de la comparaison des sorties, et ceci peut provoquer des faux négatifs.

De même, le matching logique peut tolérer des faux positifs au niveau du score « subsume » ou « subsumed by », surtout si la chaine de subsomption, reliant les deux concepts à comparer est longue.

Le principal avantage de nos deux approches par rapport OWLSM0, est la présence des scores numériques qui quantifient la longueur des chaines de subsomption. Ces valeurs diminuent le risque d'accepter des faux positifs, ou d'oublier des faux négatifs.

IV.2) Performances des Approches de Composition

Contexte

Nous cherchons des compositions notées c, qui ont des entrées et des sorties, spécifiées dans la requête du client. Cette dernière contient aussi des contraintes globales sur 05 critères de QOS, et une taille maximale de la composition.

Les intervalles des valeurs des critères de QOS sont montrés dans la table VII.1. Dans ce qui suit nous donnons un exemple de la requête-utilisateur.

- Entrées :

{ concept-name="**con2056039516**" , concept-name="**con369185398**" , concept name="**con605130906**", concept-name="**con1935404941**" , concept-name="**con1064121911**" }.

- Sorties :{ concept-name="**con510302591**"}
- Besoin en QOS { T.E(sol)<=1000, Coût(sol)<=100, log(Fiabilité(sol))>= -2, log(Disponibilité(sol))>=-2, Réputation(sol)>=2}.
- Taille-max=10.

Nous montrons par la suite, les taux d'optimalité et les temps d'exécutions associés à plusieurs simulations, et à plusieurs tailles de composition « Taille-max ». Nous avons pris des poids équitables pour chaque critère de QOS.

Résultats

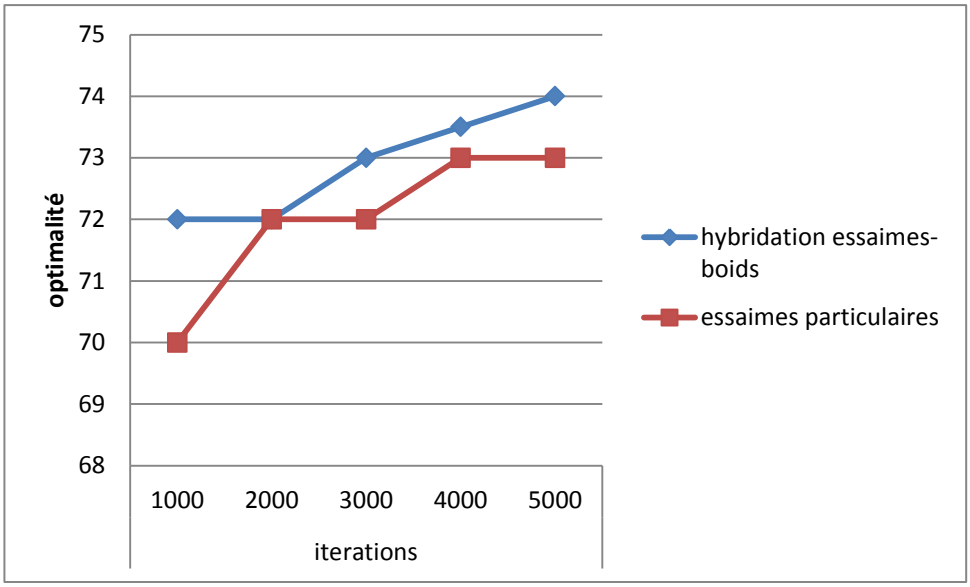


Figure VII.10. Le taux d'optimalité obtenu pour une taille de composition inférieure ou égale à 10.

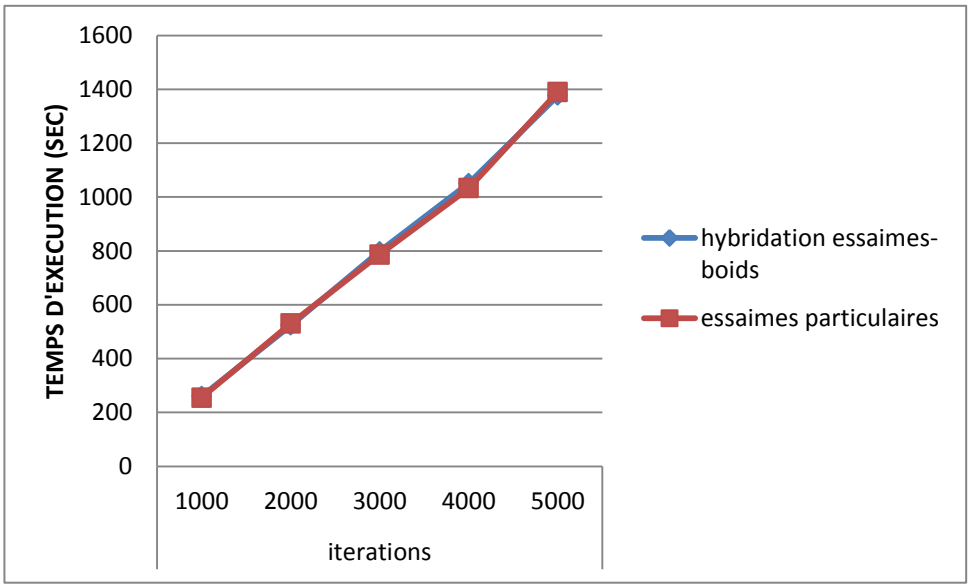


Figure VII.11. Le temps d'exécution obtenu pour une taille de composition inférieure ou égale à 10.

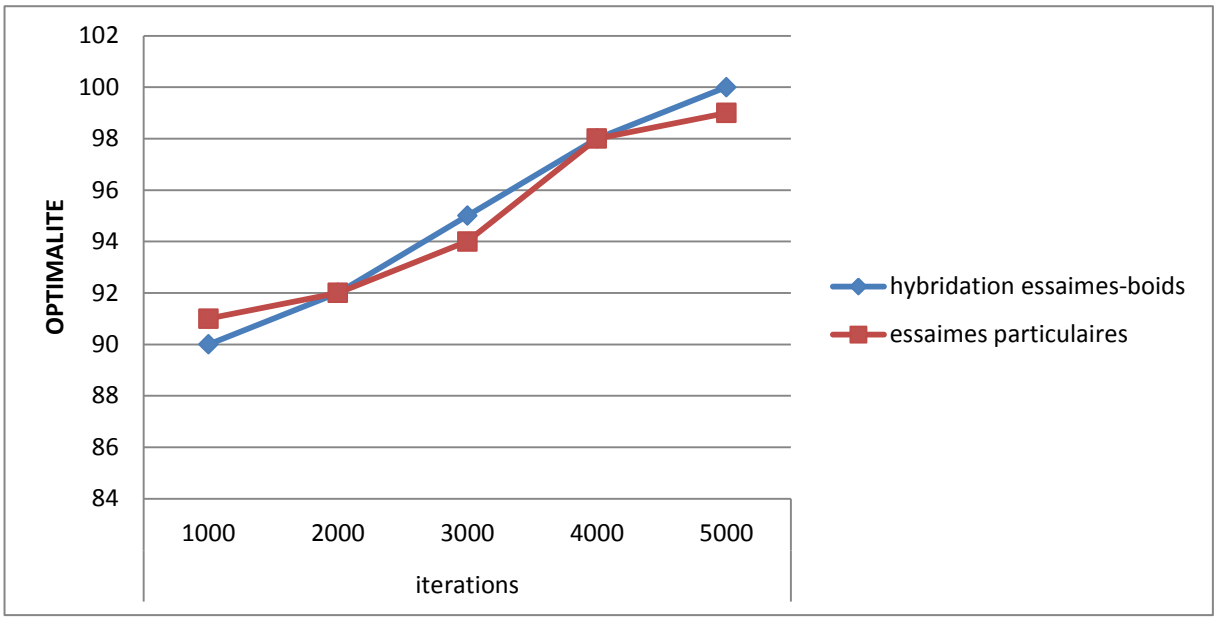


Figure VII.12. Le taux d'optimalité obtenu pour une taille de composition inférieure ou égale à 5.

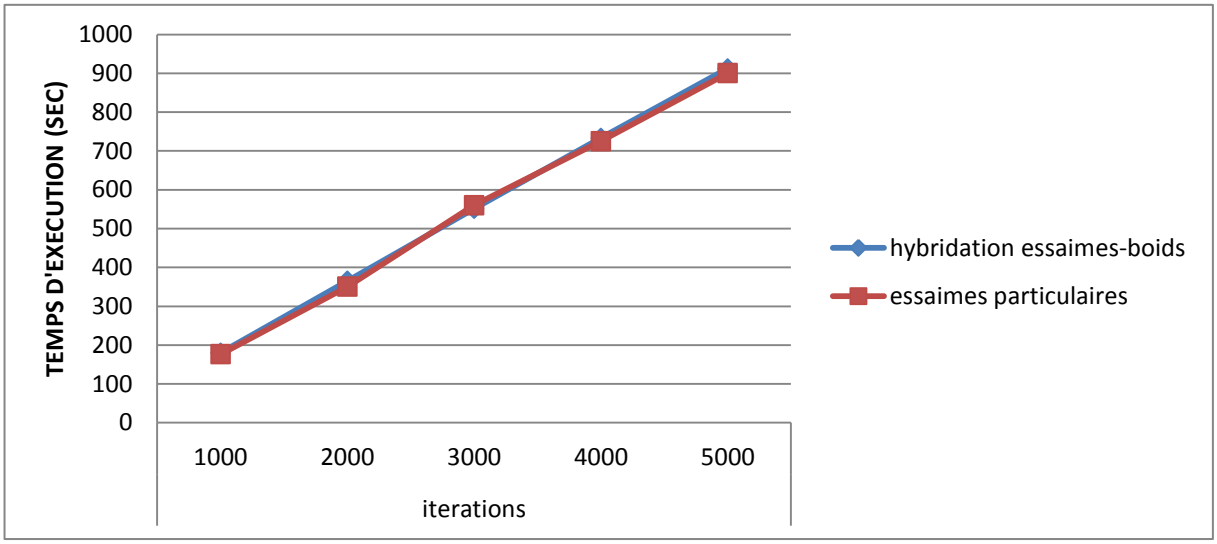


Figure VII.13. Le temps d'exécution obtenu pour une taille de composition inférieure ou égale à 5.

Les deux algorithmes de composition utilisent la même configuration i.e. 100 particules avec un voisinage complètement maillé. D'après les figures VII.10, VII.12, nous remarquons que l'approche hybride essaims-boids de reynolds est plus performante que l'approche classique des essaims particulaires, (un gain de 1% d'optimalité). Ceci est dû à l'introduction des nouveaux opérateurs de mouvements (cohésion, alignements).

De même, le temps d'exécution de la version hybride n'est pas changé (brutalement) par rapport à la version classique (voir les figures VII.11, VII.13). En effet Les deux algorithmes ont une complexité polynomiale.

Ces expérimentations confirment l'importance des mouvements des boids (cohésion, alignements), par rapport aux mouvements standards des essaims particulaires (i.e. la contribution individuelle, la contribution sociale).

La comparaison des figures VII.10, VII.12, montre que l'optimum global de la composition peut être obtenu par l'approche hybride « boids-essaims particulaires », si l'espace de recherche est réduit (la taille Max de la composition=5). Par contre si l'espace de recherche est grand (la taille Max de la composition=10) alors l'optimum n'est plus détecté par les deux approches, en effet l'ensemble des solutions est vaste (158*157*...*149+…+158*157+158) et par conséquent l'optimum ne peut être obtenu en un temps limité (moins de 5000 itérations).

IV.3) Performances des Approches de Sélection

Contexte

Nous cherchons des compositions c qui satisfont les contraintes globales(QOS) indiquées dans la requête suivante, l'utilisateur introduit aussi le nombre de classes (tâches) abstraites du workflow recherché. Nous avons pris aussi, des poids équitables pour chaque critère de QOS.

- Besoin en QOS {T.E(c) <=1000, coût(c)<=100, log(fiabilité(c))>=-2, log(disponibilité(c))>= -2, réputation(c)>=2}
- Le nombre de tâches abstraites :Taille-c=10.

Résultats

Le diagramme suivant, montre les taux d'optimalités obtenus par un ensemble d'approches de sélection et pour un nombre fixé de classes abstraites (taille-c=10). Nous avons réalisé plusieurs simulations en changeant les paramètres de chaque algorithme de sélection. Dans ce qui suit nous donnons les meilleures configurations de chacun d'eux :

Pour l'algorithme génétique la meilleure configuration est:

- Nombre de chromosomes=140.
- Mutation=10%.
- croisement=80%.

De même les meilleurs paramètres d'algorithme d'abeilles sont:

- nombre d'abeilles=140, nombre_sites=70%*nombre d'abeilles.
- Nombre d'abeilles recrutées :e_abeille=7, o_abeilles=2.
- nombre_elite_sites=20%* Nombre_sites, taille_patch=10.

Pour la sélection clonale, la meilleure configuration est :

- nombre de cellule B=20.
- taux de clonage= 10%.(i.e chaque cellule possède 02 clones puisque 2=10%*20).
- taux d'insertion aléatoire =5%. (une seule abeille est insérée dans la population).

Pour l'optimisation à base d'essaims particulaires, la meilleure la configuration est :

- nombre de particules=140.
- voisinage complètement maillé (l'essaim entier)

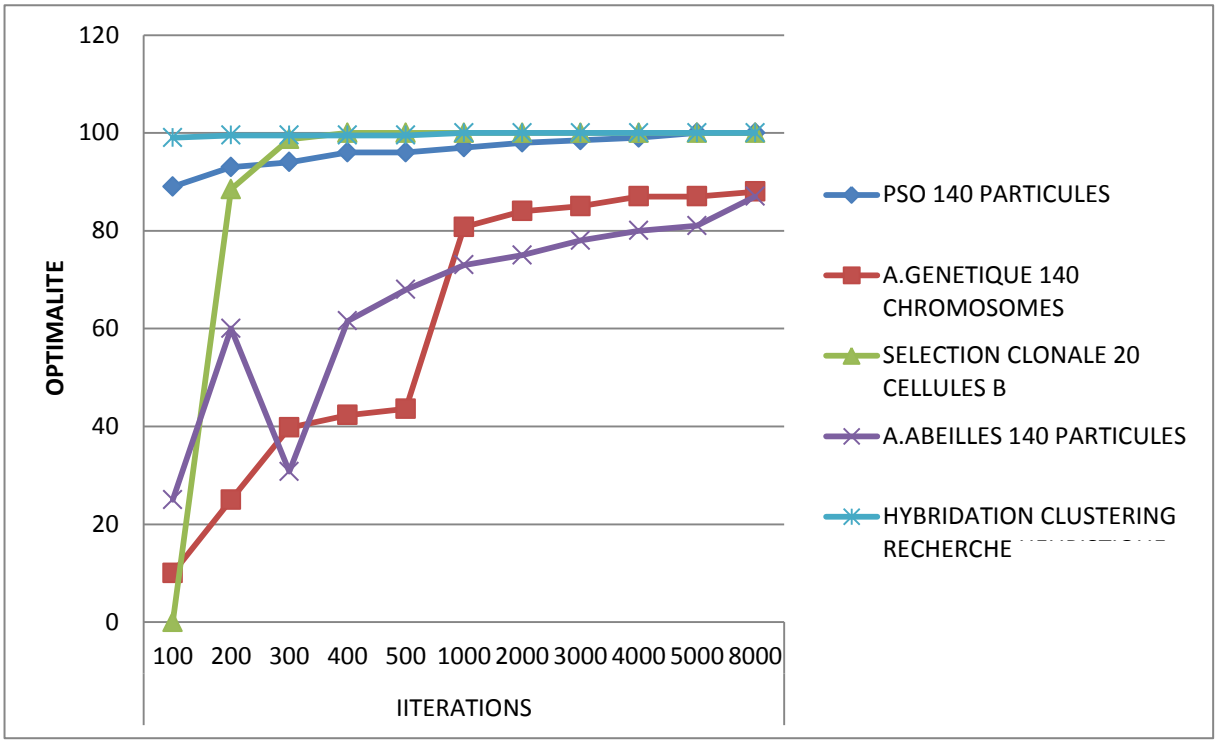


Figure VII.14. Le taux d'optimalité obtenu pour les différentes approches proposées.

Le diagramme suivant, montre le temps d'exécution des mêmes approches citées précédemment (et pour un nombre fixé de classes abstraites (taille-c=10)).

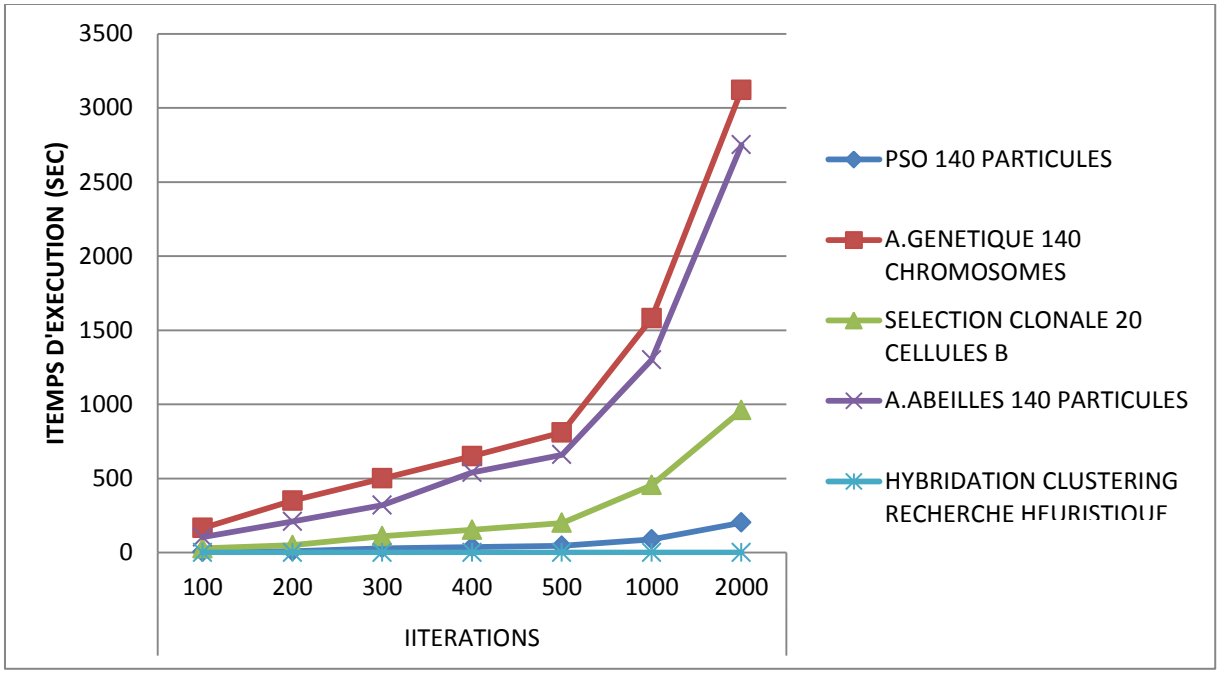


Figure VII.15. Le temps d'exécution associé aux approches de sélections proposées

Nous remarquons que, seules les approches PSO, Sélection Clonale, Hybridation Clustering-recherche systématique peuvent atteindre l'optimum global (Figure VII.14).

Nous constatons que l'Hybridation Clustering-recherche systématique, donne l'optimum en un temps réduit (0,1 sec) (Figure VII.15), ceci est dû

- à l'organisation hiérarchique des représentants de services
- le parcours intelligent des différentes hiérarchies qui élague une bonne partie de l'espace de recherche.

D'autre part, la sélection clonale possède le deuxième meilleur temps d'exécution (153 sec i.e. environ de 400 itérations), (Figure VII.15). Ces résultats confirment l'importance de l'operateur d'hypermutation des clones, dans la résolution des problèmes NP-HARD. L'approche PSO, peut aussi atteindre l'optimum global après 500 sec i.e. plus de 5000 itérations (Figures VII.15, VII.14). Cette simulation adopte un seul groupe contenant 140 particules.

En effet, l'augmentation du nombre de groupes, en gardant le même nombre particules de l'essaim, n'améliore pas les performances.

En dernier lieu, nous constatons que les approches de sélection à base d'algorithme génétique et d'abeilles ont de faibles performances (temps d'exécution et optimalité). Et par conséquent elles ne peuvent être utilisées pour les problèmes à grande échelle.

Ces expérimentations confirment que, la simple recherche locale est mal adaptée aux problèmes d'optimisation combinatoire, à grande échelle.

Les diagrammes suivants, montrent les taux d'optimalités et les temps d'exécutions obtenus lorsqu'on varie le nombre d'instances par classe abstraite.

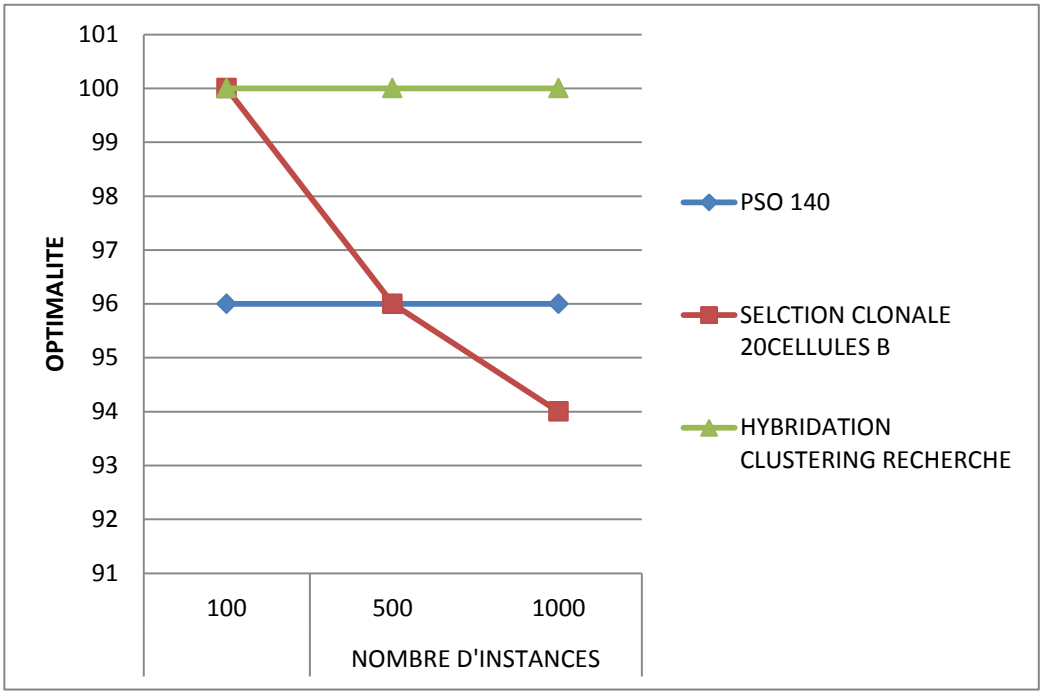


Figure VII.16. Le taux d'optimalité vs le nombre d'instances

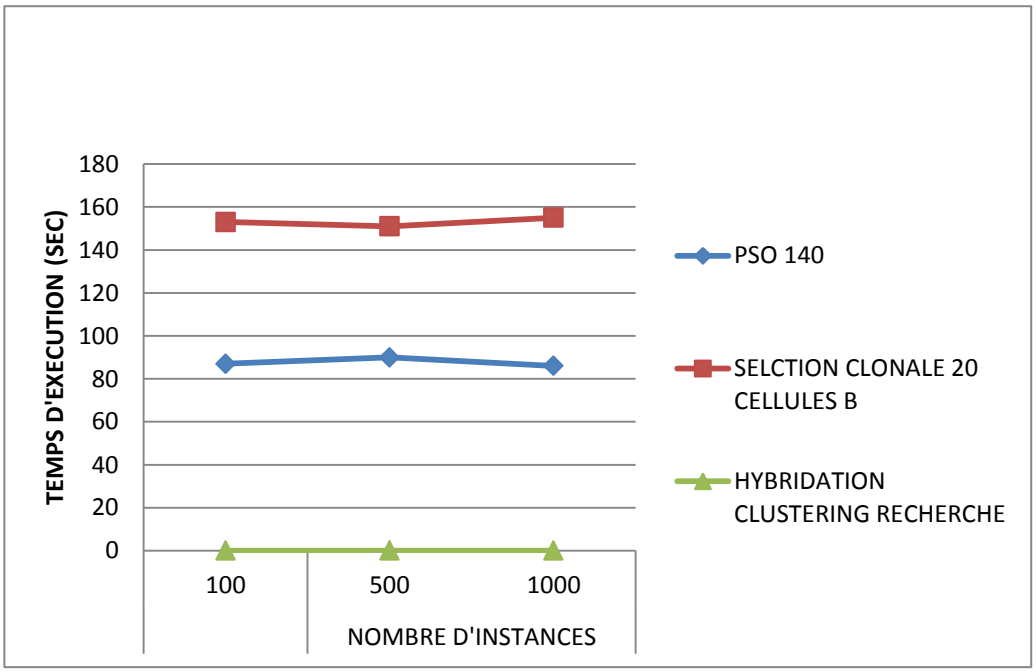


Figure VII.17. Le temps d'exécution vs le nombre d'instances

Les expérimentations (figures VII.16 et VII.17) montrent que l'hybridation clustering-recherche systématique est mieux adaptée aux conditions d'échelles (l'augmentation du nombre d'instances par classe).

En effet, la charge du temps d'exécution ajoutée est toujours dans délais imposés par l'utilisateur (pour 1000 instances, le temps d'exécution toujours égal à 0,1 sec).

Ceci est justifié comme suit :

- En premier lieu le nombre d'éléments skylines est toujours réduit, même si le nombre d'instances est grand, et donc l'espace de recherche n'est pas brusquement changé.
- Le parcours intelligent des niveaux, permet l'élagage de l'arbre de recherche.

Par contre, les autres approches sont mal adaptées au passage à l'échelle, en effet le temps d'exécution n'est plus temps réel, et les taux d'optimalité s'affaiblissent au fur et à mesure, surtout pour la sélection clonale.

Le diagramme suivant (figure VII.18) montre l'influence de la taille de population (des algorithmes génétiques et PSO) sur le taux d'optimalité.

Nous avons pris 1000 itérations pour évaluer toutes ces simulations.

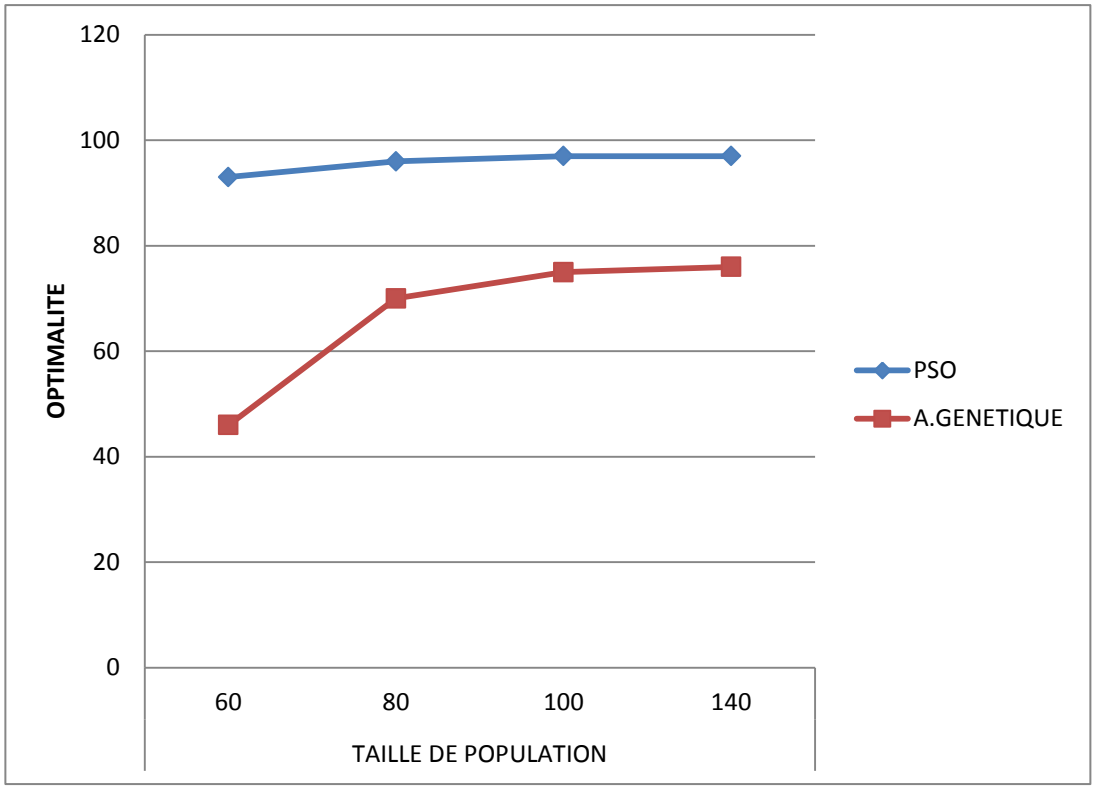


Figure VII.18. Le taux d'optimalité vs la taille de la population

Nous pouvons conclure que, le nombre idéal d'individus pour la sélection d'un workflow à 10 tâches est compris entre 100 et 140, au-delà de 140 nous remarquons une stagnation des résultats d'optimalité (A.G et PSO).

V) Conclusion

Dans ce chapitre, nous avons présenté notre prototype dénommé EDCSS, ce dernier permet la résolution de 03 types de problématiques : la découverte, la composition et la sélection.

Les expérimentations menées en découverte, confirment que les approches non logiques (à base de mesures de similarité) sont efficaces, en termes et de rappel et de précision. Nous remarquons une légère supériorité de la deuxième proposition par rapport à la première, surtout en termes de rappel.

Les expérimentations menées en composition, ont démontré l'habilitation des essaims particulaires dans la recherche des compositions quasi-optimales (en termes de sémantique et

de QOS). En effet, l'approche hybridant les boids de reynolds et les essaims particulaires peut atteindre une optimalité moyenne égale à 74%.

Les expérimentations menées en sélection, confirment la capacité des approches PSO (essaims particulaires), sélection clonale, et hybridation clustering-recherche systématique, à obtenir des solutions quasi-optimales.

Nous notons que, la dernière approche atteint toujours l'optimum global pour une composition ayant 10 classes abstraites.

Chapitre 8 :

Conclusion et perspectives

Plusieurs types de paradigmes de développement d'applications distribuées ont été proposés, au cours de ces dernières années. Le paradigme SOA est l'un des modèles les plus convenables au développement d'applications distribuées sur le web. Avec l'immense prolifération des services web sur le réseau internet, les utilisateurs auront toujours besoin de techniques efficaces pour satisfaire les exigences de découverte, de composition et de sélection de services. Les travaux présentés dans cette thèse s'inscrivent dans ce cadre-là.

Synthèse

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

Nous avons présenté aussi, trois problématiques importantes dans le domaine des services web sémantiques : la découverte de services, la composition de services, et la sélection de services.

Nous avons présenté un état de l'art détaillé de chaque problématique, les travaux présentés sont structurés en hiérarchies, qui mettent en évidence les points forts et les points faibles de chaque catégorie.

La deuxième partie est consacrée aux contributions, nous avons décrit plusieurs propositions pour chaque problème.

Concernant la découverte de services, nous avons proposé deux algorithmes basés sur les mesures de similarité, le premier [Hadjila et Chikh, 2013 a] utilise la mesure cosinus « ou cosine » pour comparer les représentations conceptuelles des services avec le besoin de l'utilisateur, le deuxième [Hadjila et Chikh, 2010 a] utilise la mesure de [Wu et Palmer, 1994] et une procédure d'optimisation de matchings pour assurer le même objectif. Les résultats montrent des performances similaires en termes de précision, et une légère supériorité du deuxième algorithme en termes de rappel.

La partie composition est modélisée comme un problème d'optimisation combinatoire avec contraintes. L'idée principale est de représenter les séquences de services comme des vecteurs d'entiers, et de parcourir l'espace des compositions avec des metaheuristiques dont le but est de maximiser une fonction objective. Cette dernière combine les aspects sémantiques et les aspects QOS.

Pour cela, nous avons proposé deux meta-heuristiques : les essaims particulaires et l'hybridation boids-essaims particulaires. Ces algorithmes utilisent des populations de particules qui partagent connaissances sociales afin de guider la recherche. Le deuxième algorithme emploie des mécanismes supplémentaires tels que la cohésion et l'alignement. Ces operateurs permettent une meilleure exploration de l'espace de recherche. Les expérimentations ont montré la supériorité de la deuxième approche par rapport à la première [Hadjila et al, 2012 b].

Pour le cas de la sélection, nous avons proposé plusieurs algorithmes d'optimisation (mono et multi objectifs), en effet la sélection peut être vue, comme un cas particulier de la composition, puisque nous considérons que l'utilisateur possède un ensemble prédéfini de tâches abstraites à instancier.

Nous avons implémenté cinq algorithmes d'optimisation : les algorithmes génétiques, l'algorithme à base d'abeilles, les essaims particulaires PSO, la sélection clonale, et l'hybridation clustering hiérarchique-recherche systématique. [Hadjila et al, 2011 b], [Hadjila et al, 2012 a], [Hadjila et al, 2012 c], [Hadjila et al, 2013 b].

Nous notons que l'optimum global est obtenu par les approches : essaims particulaires PSO, sélection clonale, et l'hybridation clustering hiérarchique-recherche systématique. En plus nous constatons que la dernière approche est la plus efficace en termes de temps d'exécution.

Perspectives

Plusieurs améliorations et extensions peuvent être envisagées pour enrichir les approches de découverte, de sélection et de composition.

La majorité des travaux portant sur la découverte négligent l'aspect contexte des utilisateurs, la prise en charge de ces aspects pendant la recherche constitue un besoin majeur pour les industriels. Le terme contexte inclut, toute entité qui participe dans l'interaction utilisateur-environnement, et généralement il couvre le profil de l'utilisateur, son emplacement géographique, la période de temps courante (par exemple la saison), son

historique…., ces données peuvent être utilisées intensivement pour personnaliser la recherche du service web adéquat.

La définition de méthodes efficaces pour la recherche sémantique multicritères, constitue un autre défi pour le problème de découverte. Par exemple, l'adoption du jugement majoritaire, les skylines probabilistes, la paréo-dominance et la fuzzy-dominance peuvent donner des résultats satisfaisants.

L'aspect apprentissage au niveau des algorithmes de matching, peut aussi améliorer les performances des algorithmes individuels, pour cela les méthodes ensemblistes (boosting, adaboost, random-forest…) peuvent être employés.

Pour le cas de la composition et la sélection de services, il est souhaitable de prendre en compte des aspects transactionnels, afin de garantir des exécutions correctes et fiables.

De même, la prise en compte des aspects de sécurité et de vie privée, lors de la composition et la sélection de services web peut être envisagée.

La prise en compte des aspects de préférences, est un facteur très important pour la sélection de compositions, en effet l'enrichissement des requêtes et des descriptions de services avec les préférences floues, pour les données fonctionnelles et non fonctionnelles, permet une meilleure personnalisation des réponses.

Les approches de compositions proposées dans cette thèse, gèrent uniquement la séquence, et par conséquent, il est souhaitable d'étendre les structures de contrôles pour prendre en compte les choix conditionnels, les boucles et le parallélisme.

Liste des publications

Plusieurs articles de journaux et de conférences ont été publiés durant cette thèse. La liste est donnée ci-dessous :

- **Journaux internationaux avec comité de lecture et indexés par Thomson**

  - ✓ Hadjila Fethallah Chikh, Mohammed Amine. Automated Retrieval of Semantic Web Services: a Matching Based on Conceptual Indexation International arab journal of information technologies IAJIT Volume 10-No.1 January 2013 **(Impact Factor= 0,39).**

- **Journaux internationaux / nationaux avec comité de lecture**

  - ✓ Hadjila Fethallah Chikh Mohammed Amine Merzoug Mohammed Optimizing Semantic Web Service Composition by Using Boid Particle Optimization International Journal of Applied Information Systems (IJAIS) Volume 3– No.6, july 2012.

  - ✓ Hadjila Fethallah Chikh Mohammed Amine Recherche des Services Web en Utilisant le Contenu d'OWLS revue RIST Volume 18 - N° 1 janvier 2010.

- **Conférences internationales avec comité de lecture**

  - ✓ Hadjila fethallah, Chikh Amine, Merzoug Mohammed QoS-Aware Web Service Selection Based On Bees Algorithm 10eme colloque sur l'optimisation et les systemes d'informations COSI '13 Alger Algerie 2013.

  - ✓ Hadjila Fethallah, Chikh Amine, Merzoug mohammed, Kameche Zeyneb QoS-aware Service Selection Based on swarm particle optimization In Proceedings of IEEE ICITES'12 Sousse Tunisia 2012

  - ✓ Hadjila Fethallah, Chikh Amine, Merzoug mohammed, QoS-aware Service Selection Based on Clonal Selection In Proceedings of ICACIS '12 Batna. Algerie. 2012.

  - ✓ Hadjila Fethallah, Chikh Amine, Belabed Amine Semantic Web Service Composition: a Similarity Measure Based Approach Algorithm In Proceedings of ICIST'11 Tebessa Algeria 2011.

- ✓ Hadjila fethallah, Chikh Amine, Dali Yahiya mohammed QoS-aware Service Selection Based on Genetic Algorithm In Proceedings of CIIA'11 Saida. Algeria. 2011.
- ✓ Hadjila Fethallah, Chikh Amine, Automated discovery of web services: an interface matching approach based on similarity measure In Proceedings of ISWSA'10 Aman. Jordanie. 2010.

- **Conférences nationales avec comité de lecture**

- ✓ Hadjila fethallah, Chikh Amine, Merzoug Mohammed QoS-Aware Web Service Selection Based On Memetic Algorithm CNEDI'13 Skikda algerie 2013.
- ✓ Hadjila Fethallah, Chikh Amine, Merzoug Mohammed QoS-Aware Web Service Selection Based On Tabu search JEESI '12 Alger. Algerie. 2012.

Annexe A : Exemple d'un document WSDL annoté

Dans ce qui suit, nous donnons un exemple d'un document WSDL de la base de composition[16]. Ce document est annoté avec la sémantique et la QOS, il représente les différentes facettes d'un service web.

<?xml version="1.0" encoding="UTF-8"?>

0:<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
1:xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
2:xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
3:xmlns:xs="http://www.w3.org/2001/XMLSchema"
4:xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
5:xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"

6:xmlns:service="http://www.ws-challenge.org/WSC08Services/"
7:targetNamespace="http://www.ws-challenge.org/WSC08Services/">

<!-- partie interface wsdl standard-->

8:<service name="Task0Service">

9 : <port binding="service:Task0SOAP" name="Task0Port">

10: <soap:address location="http://www.unknownexamplehost.ukn/" />

11: </port>

12 : </service>

13 : <binding name="Task0SOAP" type="service:Task0PortType">

14: <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />

15: <operation name="Task0Operation">

[16] http://www.ws-challenge.org/

```
16 :      <soap:operation soapAction="http://www.ws-challenge.org/Task0" />
17:      <input>
18 :        <soap:body use="literal" />
19:      </input>
20 :     <output>
21 :        <soap:body use="literal" />
22:      </output>
23 :    </operation>
24 :  </binding>
25 :  <portType name="Task0PortType">
26:   <operation name="Task0Operation">
27 :     <input message="service:Task0RequestMessage" />
28:      <output message="service:Task0ResponseMessage" />
29 :    </operation>
30:  </portType>
31 :  <message name="Task0RequestMessage">
32    <part element="service:1485922554" name="1485922554Part" />
33:    <part element="service:727230669" name="727230669Part" />
34 :    <part element="service:497402324" name="497402324Part" />
35 :    <part element="service:2027942102" name="2027942102Part" />
36 :    <part element="service:391399439" name="391399439Part" />
37:   <part element="service:1989403934" name="1989403934Part" />
```

```
38 :  </message>
39 :  <message name="Task0ResponseMessage">
40 :    <part element="service:ComplexElement0" name="ComplexElement0Part" />
41 :    <part element="service:749934143" name="749934143Part" />
42:   <part element="service:970756628" name="970756628Part" />
43 :    <part element="service:1731066005" name="1731066005Part" />
44 :  </message>
45 :  <types>
46:   <xs:schema targetNamespace="http://www.ws-challenge.org/WSC08Services/">
47:     <xs:element name="1485922554" type="xs:string" />
48:     <xs:element name="727230669" type="xs:string" />
49:     <xs:element name="497402324" type="xs:string" />
50:     <xs:element name="2027942102" type="xs:string" />
51:     <xs:element name="391399439" type="xs:string" />
52:     <xs:element name="1989403934" type="xs:string" />
53:     <xs:element name="ComplexElement0">
54 :       <xs:complexType>
55 :         <xs:sequence>
56 :           <xs:element name="1577607530" type="xs:string" />
57 :         </xs:sequence>
58 :       </xs:complexType>
59 :     </xs:element>
```

60: <xs:element name="749934143" type="xs:string" />

61: <xs:element name="970756628" type="xs:string" />

62: <xs:element name="1731066005" type="xs:string" />

63: </xs:schema>

64 : </types>

< !--partie annotation sémantique des paramètres d'entrées sorties-->

65 : <mece:semExtension xmlns:mece="http://www.vs.uni-kassel.de/mece">

66 : <mece:semMessageExt id="Task0RequestMessage">

67: <mece:semExt id="1485922554">

68: <mece:ontologyRef>http://www.ws-
69:challenge.org/wsc08.owl#inst1867161353</mece:ontologyRef>

70: </mece:semExt>

71 : <mece:semExt id="727230669">

72: **<mece:ontologyRef>http://www.ws-**
73:challenge.org/wsc08.owl#inst104320940</mece:ontologyRef>

74: </mece:semExt>

75 : <mece:semExt id="497402324">

76: <mece:ontologyRef>http://www.ws-
77:challenge.org/wsc08.owl#inst727162353</mece:ontologyRef>

78: </mece:semExt>

79 : <mece:semExt id="2027942102">

80: <mece:ontologyRef>http://www.ws-
81:challenge.org/wsc08.owl#inst730643408</mece:ontologyRef>

82: </mece:semExt>

83 : <mece:semExt id="391399439">

84: <mece:ontologyRef>http://www.ws-
85:challenge.org/wsc08.owl#inst280461322</mece:ontologyRef>

86: </mece:semExt>

87 : <mece:semExt id="1989403934">

88: **<mece:ontologyRef>http://www.ws-**
89:challenge.org/wsc08.owl#inst1039300291</mece:ontologyRef>

90: </mece:semExt>

91 : </mece:semMessageExt>

92 : <mece:semMessageExt id="Task0ResponseMessage">

93: <mece:semExt id="749934143">

94 : <mece:ontologyRef>http://www.ws-
95:challenge.org/wsc08.owl#inst1875968180</mece:ontologyRef>

96: </mece:semExt>

97 : <mece:semExt id="970756628">

98: **<mece:ontologyRef>http://www.ws-**
99:challenge.org/wsc08.owl#inst349893555</mece:ontologyRef>

100: </mece:semExt>

101: <mece:semExt id="1731066005">

102: <mece:ontologyRef>http://www.ws-
103:challenge.org/wsc08.owl#inst795570225</mece:ontologyRef>

104: </mece:semExt>

105 : <mece:semExt id="1577607530">

106: <mece:ontologyRef>http://www.ws-

107:challenge.org/wsc08.owl#inst1849957915</mece:ontologyRef>

108: </mece:semExt>

109 : </mece:semMessageExt>

110 : </mece:semExtension>

< !--partie annotation avec les attributs de QOS -->

111:<ServiceQosDefinition name="ServiceQosDefinitionTask0">

112:<Operation name=" Task0Operation">

< !—le temps d'execution du service-->

113:<SLAParameter name="**SLAParameterExecutiontimeTask0**" type="long"
114:unit="milliseconds"> <Metric>AverageExecutionTimeMetric</Metric>

…

115: <Validity><Start>2008-01-30T14:00:00</Start><End>2008-05-

116: 29T17:29:00</End></Validity>

117:<Expression> <Predicate

118: xsi:type="Greater"><SLAParameter>SLAParameterExecutiontimeTask0

119: </SLAParameter>**<Value>20</Value>**</Predicate>

120:</Expression>

121:</SLAParameter>

< !—le cout du service-->

122:<SLAParameter name=**"SLAParameterCostTask0"** type="long" unit="$">
123:<Metric>CostMetric</Metric>

…

124: <Validity><Start>2008-01-30T14:00:00</Start><End>2008-05-

125: 29T17:29:00</End></Validity>

126:<Expression> <Predicate

127: xsi:type="Greater"><SLAParameter>SLAParameterCostTask0

128: </SLAParameter>**<Value>2,5</Value>**</Predicate>

129:</Expression>

130:</SLAParameter>

< !—le fiabilité du service-->

131:<SLAParameter name=**"SLAParameterReliabilityTask0"** type="long" unit="">
132:<Metric>ReliabilityMetric</Metric>

…

133: <Validity><Start>2008-01-30T14:00:00</Start><End>2008-05-

134: 29T17:29:00</End></Validity>

135:<Expression> <Predicate

136: xsi:type="Greater"><SLAParameter>SLAParameterReliabilityTask0

137: </SLAParameter>**<Value>0.90</Value>**</Predicate>

138:</Expression>

139 :</SLAParameter>

< !—la disponibilité du service-->

140:<SLAParameter name="**SLAParameterAvailabilityTask0"** type="long" unit="">
141:<Metric>AvailabilityMetric</Metric>

…

142: <Validity><Start>2008-01-30T14:00:00</Start><End>2008-05-

143: 29T17:29:00</End></Validity>

144:<Expression> <Predicate

145: xsi:type="Greater"><SLAParameter>SLAParameterAvailabilityTask0

146: </SLAParameter>**<Value>0.80</Value>**</Predicate>

147:</Expression>

148:</SLAParameter>

< !—la reputation du service-->

149:<SLAParameter name="**SLAParameterReputationTask0"** type="long"
150:unit="milliseconds"> <Metric>ReputationMetric</Metric>

…

151: <Validity><Start>2008-01-30T14:00:00</Start><End>2008-05-

152: 29T17:29:00</End></Validity>

153:<Expression> <Predicate

154: xsi:type="Greater"><SLAParameter>SLAParameterReputationTask0

155: </SLAParameter>**<Value>3</Value>**</Predicate>

156:</Expression>

157:</SLAParameter>

158:</Operation>

159:</ServiceQosDefinition>

160 :</definitions>

Les lignes 0..7 : introduisent les différents espaces de noms requis dans le document WSDL.

Les lignes 8..12 : introduisent un seul service constitué d'un binding et d'un point d'accès (i.e. l'adresse http://www.ws-challenge.org/Task0")

Les lignes 13..24 : introduisent un binding de type RPC, il est associé au PortType "Task0PortType"

Les lignes 25..30 : présentent un porttype constitué d'une seule opération de type requête réponse

Les lignes 31..44 : présentent le message d'entrée du service ainsi que le message de sortie, chacun d'eux est constitué de plusieurs parties.

Les lignes 45..64 : représentent les types (complexes et simples) associées aux parties introduites précédemment.

Les lignes 65..110 : introduisent les annotations sémantiques des paramètres d'entrées sorties, par exemple la partie dont l'id= 1485922554 est associée avec l'instance http://www.ws-69:challenge.org/wsc08.owl#inst1867161353 de l'ontologie de domaine.

Les lignes 11..159 : introduisent la qualité de service des opérations, dans cet exemple nous présentons cinq attributs de QOS : le temps d'exécution, le coût, la fiabilité, la disponibilité, et la réputation. Par exemple dans la partie 113..120, nous introduisons la métrique temps d'exécution, sa valeur (20 msec), et la période durant laquelle elle est valable.

La même chose est faite pour le reste des critères de QOS.

Annexe B : Exemple d'une composition BPEL

Nous présentons dans ce qui suit, une partie d'une orchestration BPEL. Elle est constituée de plusieurs sortes d'activités : des séquences, des choix conditionnels, et des exécutions parallèles.

1:<**bpel:process** name="WSC"

2:targetNamespace="http://www.wsc-hallenge.org/WSC08Composition/">

3:<**bpel:sequence** name="main">

4:<**bpel:receive** name="receiveQuery" portType="solutionProcess" variable="query"/>

5:<**bpel:sequence**>

6:<**bpel:switch** name="Alternative-Services">

7:<**bpel:case** name="Execute-serv1056747493Service">

8:<bpel:sequence>

9:<bpel:invoke name="service:serv1056747493Service"
10:portType="service:serv1056747493PortType"
11:operation="service:serv1056747493Operation"/>

12:</bpel:sequence>

13:</**bpel:case**>

14:<**bpel:case** name="Execute-serv364063707Service">

15:<bpel:sequence>

16:<bpel:invoke name="service:serv364063707Service"
17:portType="service:serv364063707PortType"
18:operation="service:serv364063707Operation"/>

19:</bpel:sequence>

20:</**bpel:case**>

21:</**bpel:switch**>

22:<**bpel:flow**>

23:<**bpel:invoke** name="service:serv1195611959Service"
24:portType="service:serv1195611959PortType"
25:operation="service:serv1195611959Operation"/>

26:<**bpel:invoke** name="service:serv3195611968Service"
27:portType="service:serv3195611968PortType"
28:operation="service:serv3195611968Operation"/>

29:<**bpel:invoke** name="service:serv2096592482Service"
30:portType="service:serv2096592482PortType"
31:operation="service:serv2096592482Operation"/>

32:</**bpel:flow**>

33:<**bpel:invoke** name="service:serv850089338Service"
34:portType="service:serv850089338PortType"
35:operation="service:serv850089338Operation"/>

36:</**bpel:sequence**>

37:</**bpel:sequence**>

38:</**bpel:process**>

Les lignes 1..2: introduisent le nom du processus BPEL, ainsi que l'espace de nom.

Les lignes 3..5: introduisent les deux grandes opérations de ce processus, la première permet la réception des entrées à partir de la requête d'utilisateur, la deuxième est une opération complexe, elle est décrite par la suite.

Les lignes 5..36: introduisent le flux de contrôle et le flux de données de cette composition. On distingue trois grandes activités, la première est un choix conditionnel « switch »entre deux opérations simples, la deuxième est une exécution parallèle « flow » de trois opérations simples, la troisième est une invocation « invoke » d'un service atomique.

Les lignes 6..21 : présentent une activité complexe qui consiste à faire une invocation alternative entre deux services serv1056747493Service et serv364063707 Service.

Les lignes 22..32 : présentent une activité complexe qui consiste à faire une invocation parallèle de trois services serv1195611959Service, serv3195611968Service et serv2096592482Service.

Les lignes 33..35 : permettent l'invocation du service atomique serv850089338Service et retourne ce résultat comme sortie finale.

Références bibliographiques

[Abi Lahoud, 2010] E. Abi Lahoud. Composition dynamique de services : application à la conception et au développement de systèmes d'information dans un environnement distribué. Thèse de doctorat en Informatique. Université de Bourgogne. 2010.

[Abramsky, 1994] S. Abramsky. Proof as processes. In Theorical Computer Science, volume 135, pages 5–9, July 1994.

[Adeli et Cheng, 1994] J. Adeli, H. and Cheng, N.T. Augmented lagrangian genetic algorithm for structural optimization, Journal of Aerospace Engineering, 7, 104-118, 1994.

[Aggarwal et al, 2004] R. Aggarwal, K. Verma, J. Miller, W.Milnor, Constraint Driven Web Service Composition in METEOR-S in the proceesings of the IEEE international conference on services computing 5SCC) . Proceedings of the IEEE Computer Society pp.23-30,Shanghai China, 2004.

[Ai et Tang, 2008] L. Ai, M. and Tang, A Penatly-based Genetic Algorithm for QoS-Aware Web Service Composition with Inter-Service-Dependencies and Conflicts“. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation (CIMCA’08), Proceedings of the IEEE Computer Society, Wien, Austria, December 10-12, 2008, pp. 738-743.

[Akbar et al, 2001] M-M. Akbar, E-G. MANNING, G-C. SHOJA, AND S. KHAN. Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In Proceedings of the International Conference on Computational Science. Part II. Springer, Berlin, 659–668. 2001.

[Akbar et al, 2006] M-M. Akbar, M-S.Rahman, M.Kaykobad, E-G.Manning, AND G-C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. Comput. Oper. Res. 33, 5,1259–1273.2006.

[Akkiraju et al, 2005] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth, and K. Verma. Web service semantics - WSDL-S, November 2005.

[Algergawy et al, 2010] A.Algergawy , E.Schallehn, G.Saake. “Combining Schema and Level-Based Matching for Web Service Discovery,”.

ICWE 2010, LNCS Springer, Heidelberg. vol 6189, pp. 114–128, 2010.

[Al-masri et Mahmoud, 2008] E. Al-masri, and Q.H.Mahmoud. Investigating web services on the world wide web. In Proceeding of the 17th International Conference on World Wide Web (WWW'08). ACM, New York, 795–804.2008.

[Almulla et al, 2011] M. Almulla, K. Almatori, H. Yahyaoui A QoS-based Fuzzy Model for Ranking Real WorldWeb Services.In Proceedings of the IEEE International Conference on Web Services. 2011.

[Alrifai et al, 2008] M. Alrifai, T Risse, P. Dolog, W.Nejdl. Scalable approach for qos-based web service selection in proceeding of Service-Oriented Computing–ICSOC Workshop. 2008.

[Alrifai et Risse, 2009] M.Alrifai, And T.Risse. 2009. Combining global optimization with local selection for efficient qos-aware service composition. In Proceedings of the 18th International Conference on World Wide Web (WWW'09). ACM, New York, 881–890.

[Alrifai et al, 2010] M. Alrifai , D. Skoutas, T. Risse Selecting Skyline Services for QoS-based Web Service Composition ACM WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.

[Alrifai et al, 2012] M. Alrifai, T. Risse, and W. Nejdl, A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints. ACM Transactions on the Web, Vol. 6, No. 2, Article 7, 2012.

[Andrews et al, 2003] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S.Weerawarana. Business process execution language for web services (version 1.1). Technical report, In http://www.ibm.com/developerworks/library/ws-bpel/, May 2003.

[Ankolekar et al, 2002] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Mcdermott, D. Martin, S. A. Mcilraith, M. Paolucci, T. R.Payne, and K. Sycara. Daml-s : Web service description for the semantic web. Daml s Coalition. pages 348-363, 2002.

[Ardagna,et Pernici, 2005] D. Ardagna, and B. Pernici. Global and local QoS constraints guarantee in web service selection. In Proceedings of the IEEE

International Conference on Web Services. IEEE, Los Alamitos, CA, 805–806.2005.

[Ardagna et Pernici, 2007] D. Ardagna, and B. Pernici. Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. 33, 6, 369–384.2007.

[Aslam et Montague, 2001] J. A. Aslam and M. H. Montague. Models for metasearch. In SIGIR, pages 275–284, 2001.

[Bahadori et al, 2009] S.Bahadori, S. Kafi, K. Zamani far, M. R. Khayyambashi, Optimal web service composition using hybrid GA- TABU search, Journal of Theoretical and Applied Information Technology, Vol. 9, No. 1, pp. 10-15, 2009.

[Bekkouche et al, 2012] A.Bekkouche, S.M. Benslimane, F.Hadjila. QoS-Aware Semantic Web service Composition based on Genetic Algorithm. In Proceedings of ICDSD'12 Oran. 2012.

[Bellin et Scott, 1994] G. Bellin and P.J. Scott. On the pi-calculus and linear logic. In Theorical Computer Science, Proceedings MFPSS, volume 135, pages 11–65, July 1994.

[Benatallah et al, 2002 a] B.Benatallah, M.Dumas, M.-C. Fauvet, F.A.Rabhi, Q.Z. Sheng. Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges, Vol.3, N03, pp.9-16. 2002.

[Benatallah et al, 2002 b] B. Benatallah, Q. Z. Sheng, A. H. H. Ngu, and M. Dumas. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In 18th International Conference on Data Engineering, pages 297–308. IEEE Computer Society, 2002.

[Benatallah et al, 2003] B. Benatallah, Q. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. IEEE Internet Computing, 7(1):40–48, 2003.

[Benatallah et al, 2005 a] B.Benatallah, R.Dijkman, M.Dumas, Z.Maamar. Service Composition: Concepts, Techniques, Tools and Trends. In: Stojanovic Z., Dahanayake A., Eds. Service- Oriented Software Engineering: Challenges and Practices. Idea Group Inc (IGI), 2005, pp.48-66.

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

[Benouaret et al, 2011a] K. Benouaret , D. Benslimane, A. Hadjali , M. Barhamgi FuDoCS: A Web Service Composition System Based on Fuzzy Dominance for Preference Query Answering Proceedings of the VLDB Endowment, Vol. 4, No. 12. 2011.

[Benouaret et al, 2011b] K. Benouaret, D. Benslimane, and A. Hadjali. Top-k service compositions: A fuzzy set-based approach. In ACM SAC, pages 1033-1038, TaiChung, Taiwan. 2011.

[Berardi et al, 2003] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In 1st Int. Conf. on Service Oriented Computing (ICSOC), pages 43-58, 2003.

[Berardi et al, 2005 a] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic service composition based on behavioral descriptions. Int. J. Cooperative Inf. Syst., 14(4):333-376, 2005.

[Berardi et al, 2005 b] D. Berardi, D. Calvanese, D. G. Giuseppe, R. Hull, and M. Mecella. Automatic Composition of Transition-based Semantic Web Services with Messaging. In 31st International Conference on Very Large Databases, pages 613–624, 2005.

[Berardi et al, 2006] D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella. Composing web services with nondeterministic behavior. In Proceedings of the International Conference on Web Services (IEEE ICWS 06). pages 909–912, Chicago, Illinois, USA, 2006.

[Bleul et al, 2007] S. Bleul, T.Weise, and K. Geihs. Making a fast semantic service composition system faster. In Proceedings of IEEE Joint Conference (CEC/EEE 2007) on E-Commerce Technology (9th CEC'07) and Enterprise Computing, E-Commerce and E-Services (4th EEE'07), 2007, pages 517–520. 2nd place in 2007 WSC. Online available at http://www.it-weise.de/documents/files/BWG2007WSC.pdf

[Bieberstein et al, 2005] N.Bieberstein, S.Bose, M.Fiammante, K.Jones, and R.Shah. Service- Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap, IBM Press, October 19, 2005.

[Booth et al, 2004] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C.Ferris, et D. Orchard. Web services architecture, W3C Working Group Note 11. In http://www.w3.org/TR/ws-arch/, 2004.

[Box et al, 2004] D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, M. Hadley, C. Kaler, D. Langworthy, F. Leymann, B. Lovering, S.Lucco, S.Millet, N.Mukhi, M.Nottingham, D. Orchard, J.Shewchuk, E.Sindambiwe, T.Storey, S. Weerawarana, S. Winkler. Web Services Addressing, W3C Member Submission. 10 August. 2004.

[Brownlee, 2011] J. Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes Lulu Enterprises, First Edition, 2011.

[Bruijn et al, 2005] D.Bruijn, H. Lausen, A. Polleres, D. Fensel: WSML - a Language Framework for Semantic Web Service. W3C Workshop on Rule Languages for Interoperability, Washington USA, 27–28 April 2005. http://dip.semanticweb.org/WSML-aLanguageFrameworkforSemanticWebServices.htm

[Bultan et al, 2003] T. Bultan, X. Fu, R. Hull, and J.Su. Conversation specification: a new approach to design and analysis of e-service composition. In WWW '03: Proceedings of the 12th international conference on World Wide Web, pages 403-410, New York, NY, USA, 2003. ACM Press.

[Cabrera et al, 2005] L. F. Cabrera, G. Copeland, M.Freingold, R-W.Freund, T. Freund et J.Johnson. Web Services Coordination, Version 1.0. In http://www.ibm.com/developerworks/library/ws-coor/, 2005.

[Cámara et al, 2006] J. Cámara, C. Canal, J. Cubo, and A. Vallecillo. Formalizing WSBPEL business processes using process algebra. Electronic Notes in Theoretical Computer Science, 154 :159 – 173, 2006..

[Canfora et al, 2005] G. Canfora, M. Di Renta, R. Esposito, M. L. Villani, "An Approach for QoS aware Service Composition based on Genetic Algorithms", GECCO'05, Washington, USA, 2005.

[Carey, 2008] M. J. Carey. SOA what ? IEEE Computer, 41(3) :92-94, 2008.

[Casati et al, 2000] F. Casati, S. Ilnicki, L-J. Jin, V. Krishnamoorthy, and M-C Shan. Adaptive and dynamic service composition in efow. In CAiSE '00 : Proceedings of the 12th International Conference

on Advanced Information Systems Engineering, pages 13-31, London, UK, Springer-Verlag. 2000.

[Chinnici et al, 2007] R.Chinnici, J-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0, World Wide Web Consortium, http://www.w3.org/TR/wsdl2.0. June 2007.

[Chomicki et al, 2003] J.Chomicki, P.Godfrey, J.Gryz, D.Liang. Skyline with Pre-sorting. IEEE International Conference on Data Engineering (ICDE), 717-719, Bangalore, India, March 5-8. 2003.

[Christensen et al, 2004] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. World Wide Web Consortium, http://www.w3.org/TR/wsdl/, 2001

[Clement et al, 2004] A. Clement, C. Hately, V. Riegen, and T. Rogers. Uddi version 3.0.2. http://uddi.org/pubs/uddi v3.htm, 2004.

[Cohen et al, 2003] W.Cohen, P .Ravikumar, and S.Fienberg. A comparison of string distance metrics for name-matching tasks. In Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03). 2003. DBLP at http://dblp.uni-trier.de

[Curbera et al, 2002] F.Curbera, F.Duftler, R. Khalaf, W.Nagy, N. Mukhi, and S.Weerawarana .Unraveling . the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 6(2). (2002).

[Crasso et al, 2008] M.Crasso, A.Zunino, and M. Campo. “Easy web service discovery: A query-by-example approach.” Science of Computer Programming, 71 (2008), pp. 144-164, Elsevier, 2008.

[Deb et al, 2002] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. Journal IEEE Transaction on Evolutionary Computation, 6 :182–197, 2002.

[De Bruijn et al, 2005] J. De Bruijn, D. Fensel, U. Keller, M. Kifer, H. Lausen, R.Krummenacher, A. Polleres, and L. Predoiu. Web service modeling language (wsml) - w3c member submission, June 2005

[De Castro et V. Zuben, 2000] L. N. de Castro and F. J. Von Zuben. The clonal selection algorithm with engineering applications. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO

'00), Workshop on Artificial Immune Systems and Their Applications, pages 36–37, 2000

[De Castro et V. Zuben, 2002] L. De Castro, F. von Zuben, Learning and Optimization using the Clonal Selection Principle, IEEE Transactions on Evolutionary Computation, Volume 6, Number 3, pp. 239-251, 2002.

[Dodani, 2004] M-H. Dodani. From objects to services. A journey in search of component reuse nirvana. Journal of Object Technology, 2004

[Dong et al, 2004] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In VLDB, pages 372–383, 2004.

[Downing et, 1998] T. B. Downing and R. M. I. Java. Remote method invocation. Foster City, Calif.: IDG Books Worldwide, 1998.

[Dreo et al, 2003] J.Dreo, A.Petrowski, P. Siarry et E. Taillard. Metaheuristiques pour l'optimisation difficile. EYRROLES. 2003.

[Erl, 2004] T. Erl. Service-Oriented Architecture: A Field Guide to Integrating XML and Web. Services. Prentice Hall. 2004.

[Eberhart et Kennedy, 1995] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science, pages 39–43, 1995.

[Eijk et Diaz, 1989] P. V. Eijk and M. Diaz, editors. Formal Description Technique Lotos: Results of the Esprit Sedos Project. Elsevier Science Inc., 1989.

[Emmerich, 2002] W. Emmerich. Distributed component technologies and their software engineering implications. In ICSE '02 : Proceedings of the 24th International Conference on Software Engineering, pages 537-546, New York, NY, USA, 2002. ACM.

[Farrell et Lausen, 2007] J. Farrell and H. Lausen. Semantic annotations for wsdl and xml schema. Technical report, W3C Candidate Recommendation, January 2007. http://www.w3.org/TR/2007/CR-sawsdl-20070126/.

[Farah et Vanderpooten, 2007] M. Farah and D. Vanderpooten. An Outranking Approach for Rank Aggregation in Information Retrieval. In SIGIR, pages 591–598, 2007.

[Fellbaum, 1998] C. Fellbaum. WordNet: An Electronic Lexical Database. The MIT press, 1998.

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

[Ferrara, 2004] A. Ferrara. Web Services: A Process Algebra Approach. In 2nd International Conference on Service Oriented Computing, pages 242–251. ACM Press, 2004.

[Fox et Shaw, 1993] E. A. Fox and J. A. Shaw. Combination of Multiple Searches. In 2nd TREC, NIST, pages 243–252, 1993.rv W3C. Web services architecture requirements. online:http://www.w3.org/TR/wsa-reqs, October 2002.

[Fowler, 2003] M. Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2003.

[Galves de olivera et parente de olivera , 2010] frederico Galves de olivera jr josé M parente de olivera an infrastructure for evolving dynamic web service composition international journal of intelligent computing research volume 1 issue 2 june 2010.

[Gannod et al, 2006] G. C. Gannod, J. Timm, and R. J. Brodie. Facilitating the Specification of Semantic Web Services Using Model-Driven Development. International Journal of Web Services Research, 3(3):61–81, 2006.

[Gannod et al, 2007] G. C. Gannod, J. Timm. Specifying Semantic Web Service Compositions using UML and OCL. In 5th International Conference on Web Services. IEEE press, 2007.

[Gao et al, 2006] A. Gao, D. Yang, S. Tang, and M. Zhang QoS-Driven Web Service Composition with Inter Service Conflicts APWeb 2006, LNCS 3841, pp. 121–132, 2006.

[Gardarin, 1993] G. Gardarin. Maîtriser les Bases de Données. Modèles et Langages. Edition Eyrolles, Paris, France, 1993.

[Garofalakis et al, 2004] J.Garofalakis, Y. Panagis, E. Sakkopoulos and A.Tsakalidis. Web service discovery mechanisms: looking for a needle in a haystack in International Workshop on Web Engineering. 2004.

[Gekas et fasli, 2005] J. Gekas and M. Fasli. Automatic Web Service Composition Based on Graph Network Analysis Metrics. In International Conference on Cooperative Information Systems, pages 1571–1587. LNCS 3761, Springer-Verlag, 2005.

[Ghallab et al, 1998] M. Ghallab, C. Knoblock Isi, S. Penberthy, D. E. Smith, Y Sun, and D.Weld. Pddl - the planning domain definition language. Technical report, Ecole Nationale Superieure D'ingenieur des Constructions Aeronautiques, 1998

[Gonsalves et al, 2009] T. Gonsalves, K. Yamagishi and K. Itoh Service cost and waiting time – a multi-objective optimization scenario. In Proceeding of IEEE. 2009.

[Gronmo et Solheim, 2004] R. Grønmo and I. Solheim. Towards Modeling Web Service Composition in UML. In 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure, pages 72–86, 2004.

[Gudin et al, 2003] M. Gudgin, M. Hadley, N. Mendelsohn, J.J. Moreau, and H.F. Nielsen. Simple object access protocol (soap) 1.2. World Wide Web Consortium, http ://www.w3.org/TR/soap, 2003.

[Haarslev et Möller, 2003] V.Haarslev, R.Möller. Racer: An OWL Reasoning Agent for the Semantic Web. In Proceedin of International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with 2003 IEEE/WIC International Conference. on Web Intelligence pp. 91-95 Vol(13). 2003

[Hadjila et Chikh, 2013 a] F.Hadjila, M.A.Chikh. Automated Retrieval of Semantic Web Services: a Matching Based on Conceptual Indexation International arab journal of information technologies IAJIT Volume 10-No.1 January .2013

[Hadjila et al, 2013 b] F.Hadjila, M.A Chikh, M. Merzoug. QoS-Aware Web Service Selection Based On Bees Algorithm 10eme colloque sur l'optimisation et les systemes d'informations COSI '13. Alger Algerie 2013.

[Hadjila et Chikh, 2013 c] F.Hadjila, M.A Chikh. QoS-aware Service Selection Based on Mimetic Algorithm. In Proceedings of CNEDI '12 Skikda. Algerie. 2013.

[Hadjila et al, 2012 a] F.Hadjila, M.A Chikh, M. Merzoug, Z. Kameche QoS-aware Service Selection Based on swarm particle optimization In Proceedings of IEEE ICITES'12 Sousse Tunisia 2012

[Hadjila et al, 2012 b] F.Hadjila, M.A Chikh, M. Merzoug. Optimizing Semantic Web Service Composition by Using Boid Particle Optimization. International Journal of Applied Information Systems (IJAIS) Vol 3– No.6, july 2012.

[Hadjila et al, 2012 c] F.Hadjila, M.A Chikh, M. Merzoug. QoS-aware Service Selection Based on Clonal Selection. In Proceedings of ICACIS ’12 Batna. Algerie. 2012.

[Hadjila et Chikh, 2012 d] F.Hadjila, M.A Chikh. QoS-aware Service Selection Based on Tabu Serach. In Proceedings of JEESI ’12. Alger. Algerie. 2012.

[Hadjila et al, 2011 a] F.Hadjila, M.A Chikh, A.Belabed. Semantic Web Service Composition: a Similarity Measure Based Approach Algorithm. In Proceedings of ICIST’11 Tebessa Algeria 2011.

[Hadjila et al, 2011 b] F.Hadjila, M.A.Chikh, M.Dali Yahiya. QoS-aware Service Selection Based on Genetic Algorithm. In Proceedings of CIIA’11 Saida. Algeria. 2011.

[Hadjila et Chikh, 2010 a] F.Hadjila, A.Chikh. Automated discovery of web services: an interface matching approach based on similarity measure. In Proceedings of ACM ISWSA’10 Aman. Jordanie. 2010.

[Hadjila et Chikh, 2010 b] F.Hadjila et M.A.Chikh. Recherche des Services Web en Utilisant le Contenu d’OWLS. Revue. RIST. Volume 18 - N° 1 janvier. 2010.

[Hamadi et Benatallah, 2003] R. Hamadi and B. Benatallah. A Petri Net-based Model for Web Service Composition. In 14th Australasian Database Conference, pages 191–200. Australian Computer Society. Inc., 2003.

[Hashemian et Mavaddat, 2005] S. V. Hashemian and F. Mavaddat. A Graph-Based Approach to Web Services Composition. In IEEE/IPSJ International Symposium on Applications and the Internet, pages 183–189. IEEE Computer Society, 2005.

[Hashemian et Mavaddat, 2006]S. V. Hashemian and F. Mavaddat. A Graph-Based Framework for Composition of Stateless Web Services. In 4th IEEE European Conference on Web Services, pages 75– 86. IEEE Computer Society, 2006.

[Holland, 1962] J. H. Holland. Information processing in adaptive systems. In Processing of Information in the Nervous System, pages 330–338, 1962.

[Horstmann et Kirtland, 1997] M. Horstmann and M. Kirtland. Dcom architecture. Microsoft Corporation, July, 23, 1997.

[Huang et al, 2006] C.Huang, C.Lo, K.Chao and M.Younas. "Reaching consensus: A moderated fuzzy web services discovery method." Journal of Information and Software Technology, 48. pp. 410-423. 2006.

[Hwang et & Yoon, 1981] C.L.Hwang, & , K.P.Yoon. Multiple attribute decision making: Methods and applications. New York: Springer-Verlag. 1981.

[Kaufer et Klusch, 2006] F. Kaufer, M. Klusch: WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker. Proceedings of the 4th IEEE European Conference on Web Services (ECOWS 2006), IEEE CS Press, Zurich, Switzerland, 2006.

[Kavantzas et al, 2005] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0. World Wide Web Consortium, http://www.w3.org/TR/ws-cdl-10, 2005.

[Keller et al, 2004] U. Keller, R. Lara, A.Polleres, I.Toma, M. Kifer, and D. Fensel. Wsmo web service discovery, November 2004.

[Keller et al, 2005] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In Proceedings of the 2nd European Semantic Web Symposium (ESWS2005), Heraklion, Crete, June. 2005.

[Khan, 1998] M-S. Khan. Quality adaptation in a multisession multimedia system: Model, algorithms, and architecture. Ph.D. dissertation. 1998.

[Kiefer et Bernstein, 2008] C.Kiefer, and A. Bernstein, The Creation and Evaluation of iSPARQL Strategies for Matchmaking, in The Semantic Web: Research and Applications. 2008. p. 463-477.

[Kifer, et al, 2004] M.Kifer, R. Lara, A. Polleres, C. Zhao,U.Keller, H. Lausen and D. Fensel. A logical framework for web service discovery. in ISWC 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications. CEUR Workshop Proceedings, Japan. 2004.

[Kleppe et al, 2003] A. Kleppe, J. Warmer, and W. Bast. AMDA Explained: The Model Driven Architecture– Practice and Promise. Addison-Wesley Professional, 2003.

[Klein et Konig-Ries,2004] M. Klein and B. K¨onig-Ries. Coupled signature and specification matching for automatic service binding. In Proceedings of European Conference on Web Services (ECOWS 2004), LNCS. 3250. Springer, page 183, Erfurt, Germany, September. 2004.

[Klusch et al, 2006] M.Klusch, B.Fries & K.Sycara. Automated Semantic Web Service Discovery with OWLS-MX. Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan, ACM Press. 2006.

[Klusch et Kapahnke, 2008] M. Klusch, & P.Kapahnke, Semantic Web Service Selection with SAWSDL-MX. CEUR Proceedings of 2nd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2), Karlsruhe, Germany, CEUR 416. 2008.

[Klusch et al, 2009 a] M.Klusch, B.Fries, & K. Sycara OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services. Web Semantics, 7(2), Elsevier. 2009.

[Klusch et al, 2009 b] M.Klusch, P.Kapahnke, & I.Zinnikus. SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. Proceedings of IEEE 7th International Conference on Web Services (ICWS), Los Angeles, USA, IEEE CS Press. 2009.

[Klusch et Kapahnke, 2009 c] M.Klusch, & P. Kapahnke. OWLS-MX3: An Adaptive Hybrid Semantic Service Matchmaker for OWL-S. Proceedings of 3rd International Workshop on Semantic Matchmaking and Resource Retrieval (SMR2) at ISWC, Washington, USA. 2009.

[Kourtesis et Paraskakis, 2008] D.Kourtesis and I.Paraskakis Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. Proceedings of 5th European Semantic Web Conference ESWC 2008, LNCS 5021, pp. 614 – 628, 2008.

[Kritikos et Plexousakis, 2009] K.Kritikos, K. AND D.Plexousakis. Mixed-integer programming for QoS-based web service matchmaking. IEEE Trans. Services Comput. 2, 2, 122–139. 2009.

[Lämmermann, 2002] S. Lämmermann. Runtime service composition via logic-based program synthesis. PhD thesis, Royal Institute of Technology,. Department of Microelectronics and information Technology, Royal Institute of Technology, Stockholm, Sweden. June 2002.

[Lecue, 2008] F.Lecue. Composition de Services Web: Une Approche basée liens Sémantiques. Thèse de doctorat en Informatique. L'Ecole Nationale Supérieure des Mines de Saint-Etienne. octobre 2008.

[Lecue, 2009] F. Lecue. Optimizing QoS-Aware Semantic Web Service Composition. In Proceedings of the eighth International Semantic Web Conference (ISWC 2009), pages 375-391, October 25-29, Westfields Conference Center (near Washington, DC). LNCS 4273 Springer 2009,

[Leymann, 2001] F. Leymann. Web Services Flow Language, Version 1.0. In http://www.ebpml.org/wsfl.htm, 2001.

[Li et Horrocks, 2003] L. Li, and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology", Proceedings of the Twelfth International Conference on the World Wide Web (WWW 2003), ACM Press, May 2003, pp. 331 – 339.

[Li et al, 2004] Y. Li, F. Zou, Z. Wu, and F. Ma. PWSD: A scalable Web Service Discovery architecture based on peer-to-peer overlay network, in Advanced Web Technologies and Applications. pages 291–300. Springer Berlin / Heidelberg, 2004.

[Li et al, 2006] K .Li, K.Verma, R. Mulye, R. Rabbani, J.A. Miller & A.P.Sheth. Designing Semantic Web Processes: The WSDL-S Approach. Semantic Web Services, Processes and Applications. J. Cardoso, A. Sheth (eds.), Springer. 2006.

[Li et al, 2007] F.Li, F.Yang, K.shuang, and S.Su. Q-peer: A decentralized QoS registry architecture for web services.In Proceedings of the International Conference on Services Computing.145–156. 2007.

[Liang et al, 2004] Q.,Liang, L-N Chakarapani, S.Su, R-N Chikkamagalur, H. Lam. A Semi-Automatic Approach to Composite Web Services Discovery, Description and Invocation. International Journal of Web Services Research,, 2004. 1(4): p. 64-89.

[Liang et al, 2005] Q.,Liang, S. Su. AND/OR Graph and Search Algorithm for Discovering Composite Web Services. International Journal of Web Services Research, 4(2):48–67, 2005.

[Liu et al, 2004] Y. Liu, A. H. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In Proceedings of the International World Wide Web Conference, pages 66–73, 2004.

[Liu et al, 2007] J.Liu, J. Li, K. Liu,W. Wei, "A Hybrid Genetic and Particle Swarm Algorithm for Service Composition",In Proceedings of the Sixth International Conference on Advanced Language Processing and Web Information Technology, pp.564-567. 2007.

[Lo et al, 2010] C-C.Lo, D-Y. Chen, C-F. Tsai, K-M. Chao. Service selection based on fuzzy TOPSIS method. In Proceedings of IEEE 24th International Conference on Advanced Information Networking and Applications Workshops. pp.367-372. 2010.

[Lucchi et Mazzara, 2007] R. Lucchi and M. Mazzara. A pi-calculus based semantics for ws-bpel. The Journal of Logic and Algebraic Programming, 70 :96 – 118, 2007.

[Lustig et Vardi, 2009] Y. Lustig, M-Y. Vardi: Synthesis from Component Libraries. In Proceedings of FOSSACS. 2009.

[Mandel et McIlraith, 2003] D.J. Mandell et S.A. McIlraith. Automating web service discovery, customization, and semantic translation with a semantic discovery service. In Proceedings of the Twelfth International World Wide Web Conference Work-shop on E-Services and the Semantic Web (ESSW'03), 2003.

[Martin et al, 2004] D. Martin, M. Paolucci, S. Mcilraith, M. Burstein, D. Mcdermott, D. Mcguinness, B. Parsia, T. Payne, M. Sabou, M.Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services : The owl-s approach. pages 26-42. Springer, 2004.

[McDermot, 2002] D. McDermott. Estimated-regression planning for interactions with web services. In Proceedings of the 6th International Conference on IA Planning and Scheduling, Toulouse, France,. AAAI Press. 2002.

[McIlraith et Son, 2002] S-A. McIlraith and T-C. Son. Adapting golog for composition of semantic web services. In Dieter Fensel, Fausto Giunchiglia,

Deborah McGuinness, and Mary-Anne Williams, editors, Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), pages 482-496, San Francisco, CA, April 22-25 Morgan Kaufmann Publishers. 2002.

[Medjahed et al, 2003] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. The VLDB Journal, 12 :333–351, 2003.

[Miller et al, 2004] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. Wsdl-s : Adding semantics to wsdl, 2004.

[Milner ,1982] R. Milner. A Calculus of Communicating Systems. Springer-Verlag New York, Inc., 1982.

[Ming, Zhen-wu, 2007] C. Ming, W. Zhen-wu, "An Approach for Web Services Composition Based on QoS and Discrete Particle Swarm Optimization", Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 37-41, 2007.

[Mohebbi et al, 2010] K. Mohebbi, S. Ibrahim, M. Khezrian, K. Munusamy, and S-G-H Tabatabaei. A Comparative Evaluation of Semantic Web service Discovery Approaches iiWAS2010 8-10 November, 2010, Paris, France.

[Montague et Aslam, 2002] M. H. Montague and J. A. Aslam. Condorcet Fusion for Improved Retrieval. In ACM CIKM, pages 538–548, 2002.

[Muscholl et Walukiewicz 2007] A. Muscholl, I. Walukiewicz: A lower bound on web services composition. Proceedings FOSSACS, LNCS, Springer, Volume 4423, page 274--287 - 2007.

[Nadalin et al, 2004] A. Nadalin, C. Kaler, M. Goodner M.Gudgin, A.Barbir, N-H.Granqvist. Web Services Security: SOAP Message Security 1.1. In http://www.oasis-open.org/committees/wss/, 2004.

[Narayanan et McIlraith, 2002] S. Narayanan and S-A. McIlraith. Simulation, verification and automated composition of web services. In WWW '02 : Proceedings of the 11th international conference on World Wide Web, pages 77-88, New York, NY, USA, 2002. ACM Press.

[Nau et al, 2003] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN Planning System. Journal of Artificial Intelligence Research, 20:379– 404, 2003.

[Nemhauser et Wolsey, 1988] G. L. Nemhauser and L. A. Wolsey. Integer and Combinatorial Optimization. Wiley-Interscience, New York, NY, USA, 1988.

[OASIS, 2007 a] OASIS SOA Reference Model Technical Committee, Last accessed: 2nd June, 2007. URL http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

[OASIS, 2007 b] OASIS. Web Services Business Process Execution Language. Specification available at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf, april 2007

[OMG, 2008] Object Management Group. Common object request broker architecture (corba/iiop). http://www.omg.org/spec/CORBA/3.1, January 2008. Une citation à la page 15.

[OMG, 1992] Object Management Group. Common object request broker architecture (corba/iiop). http://www.omg.org/spec/CORBA/3.1, January 2008.

[OMG, 2007] OMG Object Construct Language, Version 2.0, 2007. URL ttp://www.omg.org/docs/ptc/05-06-06.pdf.

[Orriens et al, 2003] B. Orriens, J. Yang, and M. P. Papazoglou. Model Driven Service Composition. In 1st International Conference on Service Oriented Computing, pages 75–90. LNCS 2910, Springer-Verlag, 2003.

[OSGI alliance, 2005] OSGi Alliance. OSGi Service Platform Service Compendium. http://www.osgi.org, 2005.

[Ouyang et al, 2005] C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hofstede, and H. M. W. Verbeek. Formal semantics and analysis of control ow in WSBPEL (revised version). BPM Center Report BPM-05-15, BPMcenter.org, 2005.

[Paolucci et al, 2002] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, “Semantic Matching of Web Services Capabilities” In First International Semantic Web Conference (ISWC), 2002, pp.333–347

[Papazoglou et Dubray, 2004] M-P. Papazoglou and J-j. Dubray. A survey of web service technologies. Technical report, 2004.

[Parra-Hernandez et Dimopoulos, 2005] R. Parra-Hernandez and N. J. Dimopoulos. A new heuristic for solving the multichoice multidimensional knapsack problem. In Proceedings of IEEE Trans. on Systems, Man, and Cybernetics, Part A, 35(5):708–717, 2005.

[Pathak, 2007] J. Pathak. Interactive and verifiable web services composition, specification reformulation and substitution. Phd Thesis in Computer Sciences. Iowa State University 2007.

[Patil et al, 2004] A-A. Patil, S.A. Oundhakar, A.P. Sheth, and K. Verma. Meteor-s web service annotation framework. In WWW, pages 553-562, 2004.

[Peltz, 2003] C. Peltz. Web services orchestration and choreography. In Proceedings of IEEE Computer, 36(10) :46-52, 2003.

[Pham et al, 2006] D. T. Pham, Ghanbarzadeh A., Koc E., Otri S., Rahim S., and M.Zaidi. The bees algorithm - a novel tool for complex optimisation problems. In Proceedings of IPROMS 2006 Conference, pages 454–461, 2006

[Pisinger, 1995] D. Pisinger. Algorithms for Knapsack Problems. PhD thesis, University of Copenhagen, Dept. of Computer Science, February 1995.

[Pistore et al, 2005 a] M.Pistore, P. Roberti, and P.Traverso. Process-level composition of executable web services: "on-the-y" versus "once-for-all" composition. In ESWC, pages 62-77, 2005.

[Pistore et al, 2005 b] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated Composition of Web Services by Planning at the Knowledge Level. In 19th Internatinal Joint Conferences on Artificial Intelligence, pages 1252–1259, 2005.

[Pistore et al, 2005 c] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi. Automated synthesis of composite bpel4ws web services. In Proceedings of the IEEE International Conference on Web Services (ICWS 05), pages 293–301, Orlando, FL, USA, 2005. IEEE.

[Plebani et Pernici, 2009] P. Plebani & B.Pernici, URBE: Web Service Retrieval Based on Similarity Evaluation. IEEE Transactions on Knowledge and Data Engineering, 21(11). (2009).

[Ponnekanti et Fox, 2002] S-R. Ponnekanti and A.Fox. SWORD : A developer toolkit for web service composition. In Proceedings of the 11th International World Wide Web Conference, Honolulu, HI, USA, 2002.

[Pop et al, 2010 a] C.B.Pop, V.R Chifu, I.Salomie, M.Dinsoreanu, M.Fodor, I.Condor. A Bee-inspired Approach for Selecting the Optimal Service Composition Solution 10th International Conference on Development And Application Systems, Suceava, Romania, May 27-29, 2010

[Pop et al, 2009] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, I. Vartic, M. Vlad, "Immune-inspired Web Service Composition Framework", Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC2009), September 26-29, Timisoara Romania, ISBN 978-0-7695-3964-5, pp. 376-383, 2009.

[Pop et al, 2010 b] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, Immune-Inspired Method for Selecting the Optimal Solution in Web Service Composition, LNCS, Volume 6162/2010, pp. 1-17, 2010.

[Ragab et al, 2008] R. Ragab, L.Nourine, and F.Toumani. Protocol-based web service composition. In ICSOC, pages 38–53, 2008.

[Rao et al, 2003] J.Rao, P. Küngas, et M.Matskin. Application of linear logic to web service composition. In Proceedings of 1st International Conference on Web Services, Las Vegas, USA, june 2003.

[Rao, 2004] J.Rao, Semantic Web Service Composition via Logic-based Program. Phd Thesis. In Computer Sciences. Norwegian University of Science and Technology.Trondheim, Norway. Synthesis.2004.

[Rao et al, 2004] J.Rao, P.Küngas, and M.Matskin. Logic-based web services composition : from service description to process model. In Proceedings of the 2004 IEEE. International Conference on Web Services, San Diego, USA, July 2004.

[Rao et su, 2004] J .Rao and X .Su. A survey of automated web service composition methods. In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, San Diego, California, USA, July 2004.

[Rao et al, 2006] J.Rao, P.Küngasa, and M.Matskinb. Composition of semantic web services using linear logic theorem proving. Information Systems, 31 :340– 360, 2006.

[Rey et al, 2003] C.Rey, F.Toumani M.S.Hacid, and A.Leger. An algorithm and a prototype for the dynamic discovery of e-services. Technical report, Technical Report RR05-03, LIMOS, Clemont-Ferrand, France, 2003.

[Reynolds, 1987] C.W.Reynolds. ‟Flocks, Herds, and Schools: A Distributed Behavioral Model‟, Computer Graphics, vol.21, no.4, pp.25–34. 1987.

[Roman et al, 2005] D.Roman, U.Keller, H.Lausen, J.Bruijn, R.Lara, Michael Stollberg, A.Polleres, C.Feier, C.Bussler, et D.Fensel. Web service modeling ontology. Appl. Ontol., 1(1) :77-106, 2005.

[Rongxi et al, 2010] W.Rongxi, M.Li, et C.Yanping, The Application of Ant Colony Algorithm in Web Service Selection, International Conference on Computational Intelligence and Software Engineering, pp. 1-4, 2010.

[Roy, 1968] B.Roy. Classement et choix en presence de pint d vue multiples. (la methode ELECTRE). Revue d'informatique et de recherche operationnelle. RIRO.8 :57-75.1968.

[Russel et Norvig, 2009] S.Russel et P.Norvig Artificial Intelligence A Modern Approach. Third Edition. Prentice Hall. 2009.

[Salomi et al, 2011] I.Salomie, M. Vlad, V-R. Chifu, C- B. Pop. Hybrid Immune-inspired Method for Selecting the Optimal or a Near-Optimal Service Composition. In the Proceedings of the IEEE. Federated Conference on Computer Science and Information Systems pp. 997–1003. 2011.

[Salaun et al, 2004] G. Salaun, L. Bordeaux, and M. Schaerf. Describing and Reasoning on Web Services using Process Algebra. In 2nd IEEE International Conference on Web Services, pages 43–50. IEEE Computer Society, 2004.

[Schulte et al, 2010] S. Schulte, U. Lampe, J. Eckert, and R. Steinmetz. "log4sws.kom": Self-adapting semantic web service discovery for "sawsdl". In IEEE Congress on Services, Miami, FL, USA, July 5-10, pages 511-518. IEEE Computer Society, 2010.

[Schumacher et al, 2008] M.Schumacher, H.Helin, & H.Schuldt. CASCOM – Intelligent Service Coordination in the Semantic Web. Birkhäuser Verlag, Springer. Eds. 2008.

[Schuster et al, 2000] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multienterprise processes. In Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

[Skoutas et al, 2007] D.Skoutas, A. Simitsis, and T.Sellis 'A ranking mechanism for semantic web service discovery', In Proceedings of the IEEE Congress on Services, Salt Lake City, UT, pp. 41-48. 2007.

[Skoutas et al, 2009] D. Skoutas, D. Sacharidis, A. Simitsis, V. Kantere T. Sellis Top-k Dominant Web Services Under Multi-Criteria Matching EDBT 2009, March 24–26, Saint Petersburg, Russia. 2009.

[Sirin et al, 2003] E. Sirin, B. Parsia, D. Wu, J. Hendler. Semi-automatic composition of web services using semantic descriptions. In Proceedings of Web Services Modeling, Architecture and Infrastructure Workshop. 17-24. 2003.

[Sirin et al, 2004] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau . HTN planning for web service composition using shop2. Web Semantics Journal, pages 377-396, 2004.

[Srinivasan et al, 2006] N.Srinivasan, M. Paolucci, and K. Sycara. Semantic Web Service Discovery in the OWL-S IDE. in System Sciences, 2006. Proceedings of the HICSS '06. Hawaii. USA. 2006.

[Stroulia et Wang, 2004] E. Stroulia, & Y. Wang. Structural and Semantic Matching for Assessing Web-Service Similarity. Cooperative Information Systems, 14(4), World Scientific. 2004.

[Strunjas-Yoshikawa et al, 2006] S.Strunjaˇs-Yoshikawa, F.S. Annexstein, and K. A. Berman. Compact Encodings for All Local Path Information in Web Taxonomies with Application to WordNet. 32nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM06, Czech Republic. pp. 511–520. 2006.

[Sycara et al, 2002] K.Sycara, M.Klusch, S.Widoff, and J. Lu, 2002. Larks: Dynamic matchmaking among heterogeneous software agents

in cyberspace. Autonomous Agents and Multi-Agent Systems, 5(2), Kluwer .

[Syeda-Mahmood et al, 2005] T.Syeda-Mahmood, G.Shah, R.Akkiraju, A.A.Ivan, R.Goodwin, Searching service repositories by combinig semantic and ontological matching, in Proceedings of the IEEE International conference on web services, Orlando 2005.

[Teodorovic et Dell'Orco, 2005] D.Teodorovic, M. Dell'Orco, Bee Colony Optimization – A Cooperative Learning Approach to Complex Transportation Problems, Advanced OR and AI Methods in Transportation, pp. 51—60, 2005.

[Tim et Gannod, 2005] J. Timm and G. Gannod. A Model-Driven Approach for Specifying Semantic Web Services. In 3rd International Conference on Web Services, pages 313–320. IEEE press, 2005.

[Tim et Gannod, 2007] J. Timm and G. Gannod. Specifying Semantic Web Service Compositions using UML and OCL. In 5th International Conference on Web Services, IEEE press, 2007.

[Toma et al, 2005] I. Toma, D. Fensel, M. Moran, K. Iqbal, T. Strang, and D. Roman. An Evaluation of Discovery approaches in Grid and Web services Environments. In The 2nd International Conference on Grid Service Engineering and Management, Erfurt, Germany, September 2005.

[Van Rijsbergen , 1979] C.Van Rijsbergen. Information Retrieval. Butterworths, London. 1979.

[Verma et al, 2006] K.Li, K.Verma, R Mulye, R.Rabbani, J. A. Miller & A.P.Shet. Designing Semantic Web Processes: The WSDL-S Approach. Semantic Web Services, Processes and Applications. J. Cardoso, A. Sheth (eds.), Springer.2006.

[Waldinger, 2001] R. J. Waldinger. Web agents cooperating deductively. In FAABS ’00 : Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers, pp. 250–262, London, UK, Springer-Verlag. 2001.

[Wang et al, 06] P. Wang, K-M. Chao, C-C. Lo, C-L. Huang, Y. Li. A Fuzzy Model for Selection of QoS-Aware Web Services In Proceedings of IEEE International Conference on e-Business Engineering (ICEBE'06). 2006.

[Wang et Hou, 2008] J. Wang, Y. Hou, "Optimal Web Service Selection based on Multi-Objective Genetic Algorithm", Proceedings of the ISCID 2008, pp. 553-556, Wuhan, 2008.

[Wang et Stroulia, 2003] Y.Wang, and E. Stroulia "Flexible Interface Matching for Web- Service Discovery." Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE'03), IEEE 2003.

[Ward, 1963] J.H Ward. Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58, PP. 236-244. 1963

[Wu et al, 2003] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. automatic web services composition using SHOP2. In Workshop on Planning for Web Services, Trento, Italy, June 2003.

[Wu et al, 2011] B. Wu, S. Deng, Y. Li, j. Wu, J. Yin. An automatic service planner based on heuristic state space serach. In Proceedings of IEEE international conference on web services . 2011.

[Wu et Palmer, 1994] Z. Wu & M. Palmer Verb Semantics and Lexical Selection, Proceedings of the 32nd Annual Meetings of the Associations for Computational Linguistics, pp. 133-138 .1994.

[W3C, 2002 a] W3C. Web service choreography interface (WSCI) 1.0, 2002.

[W3C, 2002 b] W3C. Web services conversation language (WSCL) 1.0, 2002.

[W3C, 2004 a] W3C. OWL Ontology Web Language,2004. http://www.w3.org/TR/owl-ref/, http://www.w3.org/TR/owl-features/.

[W3C, 2004 b] W3C. Web Services Architecture. W3C Working Group Note 2004; disponible à: http://www.w3.org/TR/ws-arch/.

[Xiong et Fan, 2007] P.Xiong, Y. Fan, QoS-aware Web Service Selection by a Synthetic Weight. In Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, pp. 632-637, 2007.

[Xu et al, 2007] Z.Xu, P.Martin, W. Powley, and F. Zulkernine. "Reputation-Enhanced QoS-based Web Services Discovery." Proceedings of the 2007 IEEE International Conference on Web Services (ICWS 2007). 2007.

[Xu et al, 2011] L. Xu, L. Shi, R.Wang, B. Jennings. A multiple criteria service composition selection algorithm supporting time-sensitive rules.Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network management Dublin Irland 2011.

[Xia et al, 2011] Y. Xia, P. Chen, L. Bao, M. Wang, J. Yang; "A QoS-Aware Web Service Selection Algorithm Based on Clustering", 2011 IEEE International Conference on Web Services (ICWS), pp. 428 – 435,2011.

[Xia et al, 2008] Y. Xia, J. Chen, and X. Meng, On the dynamic ant colony algorithm optimization based on multipheromones, IEEE/ ACIS International Conference on Computer and Information Science, pp. 630-635, 2008.

[Xiong et Fan, 2007] P.Xiong, Y.Fan, QoS-aware Web Service Selection by a Synthetic Weight. In Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, 632-637, 2007.

[Yang et al, 2010] Z. Yang, C. Shang, Q. Liu, and C. Zhao, A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm, Journal of Computational Information Systems, Vo1.6, No.8, pp.2617-2622, 2010

[Ye et al, 2009] G.Ye, J.Yue, S.Cheng, and W. Chanle. "A QoSaware Model for Web Services Discovery." Proceedings of the 2009 First International Workshop on Education Technology and Computer Science, IEEE. 2009.

[Yeniay, 2005] O.Yeniay. penalty function methods for constrained optimization with genetic algorithms journal of Mathematical and Computational Applications, Vol. 10, No. 1, pp. 45-56, 2005

[Yi et Kochut, 2004] X. Yi and K. Kochut. A CP-nets-based design and veri_cation framework for web services composition. In ICWS, pages 756-760. IEEE Computer Society, 2004.

[Yu et Bouguettaya, 2010] Q. Yu, A. Bouguettaya. Foundations for Efficient Web Service Selection. Springer Science+Business Media, 2010.

[Yu et al, 2007] T.Yu, Y.Zhang, AND K-J.Lin.Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Transactions on the Web 1. 2007.

[Yun et al, 2008] B. Yun, J. Yan, and M. Liu, "Behavior-based Web Services Matchmaking." Proceedings of the 2008 IFIP International Conference on Network and Parallel Computing, pp. 483-487, IEEE. 2008.

[Zeng et al, 2003] L.Zeng, B.Benatallah, M.Dumas, J.Kalagnanam, and Q-Z.Sheng. Quality driven web services composition. In Proceedings of the 12th International Conference on the World Wide Web (WWW '03). ACM, New York, 411–421. 2003.

[Zeng et al, 2004] L.Zeng, B.Benatallah, A-H-H.Ngu, M.Dumas, J.Kalagnanam, and Q-Z.Sheng Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30, 5, 311–327.2004.

[Zhai et al, 2011] Y.Zhai, J.Zhang, AND K-J.Lin. Soa middleware support for service process reconfiguration with endto- end QoS constraints. In Proceedings of the IEEE International Conference on Web Services (ICWS'09).IEEE, Los Alamitos, CA, 815–822. 2009.

[Zhang et Ren, 2011] Y. Zhang and M. Ren, Web Service Selection Based on Utility of Weighted Qos Attributes WISM 2011, CCIS 238, pp. 417–425, 2011.

[Zhipeng et al, 2009] G.Zhipeng, C. Jian, Q. Xuesong, M. Luoming, QoE/ QoS driven simulated annealing-based genetic algorithm for web services selection, T he Journal of China Universities of Posts and Telecommunications, Vol. 16, Issue. 08, pp. 102- 107, 2009.

[Zinnikus et al, 2006] I.Zinnikus, H.-J. Rupp & K. Fischer, Detecting Similarities between Web Service Interfaces: The WSDL Analyzer. Proceedings of 2nd International Workshop on Web Services and Interoperability (WSI 2006). 2006.

[Zitizler et Thiele, 1998] E. Zitizler and L. Thiele. Multiobjective optimisation using evolutionary algorithms - a comparative case study. In Parallel Problem Solving from Nature V, pages 292–301, Germany, 1998