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

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

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

Par
Mme Amina BEKKOUCHE épouse MERZOUG

Vers une composition automatique des services web

Thèse soutenue publiquement le 03 Juillet 2018 devant le jury :

- **Pr. CHIKH Amine**	Université de Tlemcen	Président
- **Pr. CHIKH Azeddine**	Université de Tlemcen	Examinateur
- **Dr. AMAR BENSABER Djamel**	ESI de Sidi Bel Abbès	Examinateur
- **Dr. KESKES Nabil**	ESI de Sidi Bel Abbès	Examinateur
- **Pr. BENSLIMANE Sidi Mohammed**	ESI de Sidi Bel Abbès	Directeur
- **Pr. HUCHARD Marianne**	Université de Montpellier	Co-Directrice
- **Dr. HADJILA Fethallah**	Université de Tlemcen	Invité

Année universitaire 2017-2018

Remerciements

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

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

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

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

J'exprime ma profonde reconnaissance à Monsieur CHIKH Azeddine, Professeur à l'université de Tlemcen, Monsieur AMAR BENSABER Djamel, Maitre de conférence à l'Ecole Supérieure en Informatique de Sidi Bel Abbès, et Monsieur KESKES Nabil, Maitre de conférence à l'Ecole Supérieure en Informatique de Sidi Bel Abbès, pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant de l'examiner et d'en être rapporteurs.

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

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

Je remercie chaleureusement ma famille pour son soutien, son écoute et ses encouragements tout au long de cette thèse. Mes parents, mes sœurs, mon frère et mon époux.

Je dédie cet humble travail
A mes très chers parents, que Dieu tout puissant les protège
A mon époux et mes enfants : Meryem et Abderrahmane
A mon frère, mes sœurs et leurs familles
A ma belle famille
A mes collègues

Résumé

La méthode de conception des architectures orientées services (SOA) est basée sur des standards et permet de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Dans la pratique, il n'est pas toujours facile de trouver des services correspondant aux requêtes des utilisateurs. Par conséquent, la composition des services satisfaisant la requête est un besoin grandissant de nos jours. La composition de services implique la capacité de sélectionner, de coordonner, d'interagir, et de faire interopérer des services existants. Elle constitue une tâche complexe. Cette complexité est due principalement au grand nombre de services disponibles et à leur hétérogénéité puisqu'ils sont créés par des organisations différentes. Cette complexité est renforcée quand il s'agit de composer automatiquement des services à la demande pour répondre à des exigences qui ne sont pas réalisées par les services existants.

Dans cette perspective, différentes approches ont été développées pour résoudre le problème de la composition de services web. Cependant, la plupart des processus de composition ne considèrent pas à la fois les propriétés fonctionnelles (c.-à-d. le flux de données échangé entre les services web en terme d'entrées et de sorties), les propriétés non-fonctionnelles (les paramètres de QoS des services web tels que le temps de réponse, le coût, la réputation, etc.) et les contraintes non-fonctionnelles de l'utilisateur (contraintes imposées sur les paramètres de QoS).

Dans cette thèse, nous proposons une nouvelle approche de composition basée sur un graphe de planification amélioré et un algorithme Harmony Search (ou la recherche par harmonie) pour sélectionner la composition optimale ou quasi-optimale. Ainsi, notre approche combine : (1) un mécanisme de matching sémantique pour calculer la similarité sémantique entre les différents services constituant le service composite recherché et (2) une stratégie d'optimisation multi-critères de QoS utilisant l'algorithme Harmony Search satisfaisant les contraintes non-fonctionnelles de l'utilisateur. Pour améliorer le processus de sélection, nous avons comparé la version originale de l'algorithme Harmony Search avec ses deux variantes récemment développées : *Improved Harmony Search (IHS)* et *Global Best Harmony Search (GHS)*. Nous menons une série d'expérimentations pour évaluer la performance de notre approche de composition en utilisant la collection de données de la compétition sur le WSC *Web Service Challenge 2009*. Les résultats obtenus montrent que : 1) notre approche est efficace pour trouver la composition optimale ou quasi optimale dans divers scénarios ; et 2) les deux variantes IHS et GHS ont apporté des améliorations en termes d'optimalité et de temps d'exécution.

Mots clés : Composition des services web sémantiques, Matching sémantique, Graphe de planification, Algorithme Harmony Search, Qualité de service (QoS).

Abstract

The design approach of service oriented architectures (SOA) is based on standards which gives the possibility of creating an integrated IT infrastructure capable of rapidly responding to new user needs. Actually, it is not always easy to find services that meet user requests. Therefore, the service composition satisfying the user intention is a growing need. The composition of services implies the ability to select, coordinate, interact, and to interoperate existing services. The composition is considered as a complex task. This complexity is mainly due to the large number of available services and their heterogeneity as they are created by different organizations. This complexity is increased when services must be automatically composed to meet requirements which are not satisfied by existing services.

In this perspective, different approaches have been developed to solve the problem of the composition of web services. However, most composition processes do not consider both the functional properties (i.e the data flow exchanged between web services in terms of inputs and outputs), the non-functional properties (QoS parameters of web services such as response time, cost, reputation, etc.) and non-functional constraints of the user (imposed constraints on QoS parameters).

In this thesis, we propose a new composition approach based on an improved planning graph and a Harmony Search algorithm to select the optimum or near-optimal composition. Thus, our approach combines: (1) a semantic matching mechanism to compute the semantic similarity between the different services constituting the desired composite service and (2) a QoS multi-criteria optimization strategy using the Harmony Search algorithm satisfying the non-functional constraints of the user. To improve the selection process, we compared the original version of the Harmony Search algorithm with its two recently developed variants: *Improved Harmony Search (IHS)* and *Global Best Harmony Search (GHS)*. We are conducting a series of experiments to evaluate the performance of our composition approach using the WSC *Web Service Challenge 2009* competition data collection. The results obtained show that: 1) our approach is effective in finding optimal or near optimal composition in various scenarios; and 2) both IHS and GHS variants provided improvements in terms of optimality and execution time.

Keywords: Semantic Web Service Composition, Semantic Matching, Planning Graph, Harmony Search Algorithm, Quality of Service (QoS)

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et la diffusion de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre les applications. Un de ces paradigmes qui a pris de l'ampleur au cours de ces dernières années est l'architecture orientée services (SOA) [Oppong and Khaddaj, 2010]. La méthode de conception des architectures SOA est basée sur un ensemble de standards permettant de créer une infrastructure informatique capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Les services web constituent un moyen de plus en plus normalisé, étendu et puissant pour mettre en œuvre une architecture SOA.

La réutilisation et l'interopérabilité sont au cœur du paradigme des services web. Cette technologie permet une réutilisation et une interopérabilité transparente des composants web qui facilite le développement et l'intégration rapide des applications. Néanmoins, les technologies de services web comme WSDL [Chinnici et al., 2007] et UDDI [Bellwood et al., 2002] n'ont pas de sémantique nécessaire pour les machines pour qu'elles puissent utiliser d'une façon autonome cette technologie. Les approches du web sémantique visent à procurer cette sémantique. L'objectif est d'apporter une structure au web existant à travers des annotations sémantiques en utilisant des langages comme RDF [Klyne et al., 2004] et OWL [McGuinness et al., 2004]. Ainsi, il sera possible aux machines et aux utilisateurs de collaborer d'une manière plus efficace. Plusieurs langages de services web sémantiques sont développés comme OWLS [Martin et al., 2007] et WSMO [Fensel and Bussler, 2002] de telle sorte que des agents logiciels peuvent utiliser les annotations sémantiques pour faire face de manière plus efficace à la découverte, la composition, l'invocation et l'exécution de services web.

L'utilisation des services web nécessite une étape préliminaire de découverte, qui consiste à rechercher et localiser des services web correspondant aux besoins du client qui sont exprimés au moyen d'une requête définissant les attentes fonctionnelles et non-fonctionnelles (qualité de service : QoS). Plusieurs critères de qualités de services

peuvent être pris en considération comme le temps de réponse (response time), le débit (throughput), le coût (cost), la réputation (reputation), la fiabilité (reliability), le taux d'exécution réussie (successful execution rate) et la disponibilité (availability) [Zeng et al., 2004].

Cependant, les services web individuels restant limités par leurs capacités, l'entreprise est amenée à composer un ensemble de services afin de créer des services plus complexes. On parle ainsi de la composition des services web [Gustavo et al., 2004]. Cette composition se présente comme un paradigme fondamental de la technologie des services web. Elle permet de résoudre des problèmes complexes en combinant des services de base disponibles pour satisfaire un but initial. Ce paradigme demeure l'un des axes de recherche les plus actifs des services web, durant ces dix dernières années, vu la complexité du processus de composition et l'évolution rapide des normes et des standards de cette technologie. Cette complexité est due principalement à l'hétérogénéité des services web puisqu'ils sont créés par des organisations différentes [Bucchiarone and Gnesi, 2006].

La composition qui vise à satisfaire les objectifs fonctionnels et à optimiser la QoS, est connue sous le nom de la composition de service sensible à la QoS "*QoS-aware service composition*" [Yan et al., 2012].

Du fait du nombre important des services web disponibles sur le web, il est difficile, voire impossible, de les composer de façon manuelle, c'est la raison pour laquelle la composition doit être automatique, cela est connu sous le nom "*Automated Web Service Composition*". En fait, la composition automatique permet une adaptation aux exigences des utilisateurs. Elle vise à tirer le meilleur parti des propriétés intrinsèques de la plate-forme d'exécution des services : décentralisation et modularité [Pellier and Fiorino, 2009].

C'est dans ce contexte que s'inscrit notre travail. Nous proposons une approche de composition automatique des services web basée sur une technique de planification de l'IA et une stratégie d'optimisation multi-critères. L'objectif est de trouver une composition proche de l'optimum en un temps raisonnable tout en satisfaisant les exigences fonctionnelles et non-fonctionnelles de l'utilisateur.

1.2 Problématique et motivations

Etant donné un ensemble de besoins fonctionnels et non-fonctionnels de l'utilisateur, le défi est comment construire un *service composite* d'une manière automatique tel que la valeur globale de QoS soit optimale ?. Ce problème est connu sous le nom "*QoS-aware automatic web service composition*" [Bartalos and Bieliková, 2009], [Salomie et al., 2010].

Différentes approches ont été développées pour résoudre la problématique de la composition de services web. Elles peuvent généralement être classées dans l'une des classes suivantes [Deng et al., 2013], [Rodriguez-Mier et al., 2016], [Jiang et al., 2010] : les approches de composition automatique des services web (automated web service composition approaches), les approches de composition des services web sensible à la QoS (QoS-aware service composition approaches) et les approches hybrides (QoS-aware automatic web service composition approaches).

Notre étude bibliographique a révélé que les approches de la première classe sont basées sur les techniques de planification de l'intelligence artificielle (IA). Ces techniques sont souvent considérées comme les plus efficaces et même prédominantes pour

résoudre automatiquement des problèmes de compositions [Yan et al., 2012]. Cependant, la plupart de ces approches cherchent à trouver des plans de composition valides sans considération de la qualité de service (QoS) globale de la solution recherchée. Par conséquent, ces méthodes seraient incapables de trouver des compositions optimales en satisfaisant les contraintes de l'utilisateur en terme de QoS.

Quant aux approches de composition des services web sensible à la QoS, ce sont des approches qui peuvent aboutir à des solutions optimales (ou quasi-optimales) et de bonne qualité en un temps raisonnable. Cependant, la plupart des travaux proposés modélisent la composition comme un workflow abstrait c'est-à-dire supposent l'existence préalable de plans de composition de services abstraits, et chaque service abstrait sera par la suite être lié à un service concret pris parmi les services concrets qui sont fonctionnellement équivalents.

Pour les approches hybrides, elles combinent les techniques de planification et les techniques d'optimisation afin de trouver la composition recherchée d'une manière automatique. Les solutions trouvées prennent en considération non seulement l'aspect fonctionnel des services web mais aussi l'aspect non fonctionnel regroupant un ensemble de paramètres de qualité de service (temps de réponse, fiabilité, etc). Cependant, la plupart d'entre elles ne prennent pas en compte les contraintes non-fonctionnelles de l'utilisateur.

Dans cette thèse, nous proposons une approche de composition automatique des services qui considère à la fois les propriétés fonctionnelles (c.-à-d. le flux de données échangé entre les services web en terme d'entrées et de sorties), les propriétés non-fonctionnelles (les paramètres de QoS des services web tels que le temps de réponse, le coût, la réputation, etc.) et les contraintes non-fonctionnelles de l'utilisateur (contraintes imposés sur les paramètres de QoS). Pour cela, nous avons adopté une technique de planification qui est le graphe planification amélioré EPG (Enhanced Planning Graph) [Pop et al., 2009]. Ce graphe permet la définition d'un espace de recherche efficace, dont l'exploration permet de connaître l'existence de la solution et même son extraction. Toutefois, l'algorithme de l'extraction de la solution peut être en pratique très coûteux en terme de temps d'exécution car le problème sous-jacent est NP-complet [Yan et al., 2012]. Pour cette raison, des techniques d'optimisation sont souvent utilisées.

1.3 Contributions

Ce travail a pour but de favoriser l'intégration des préférences des utilisateurs en termes de QoS dans le processus de la composition de services web. Notre contribution pour résoudre le problème "QoS-aware automatic service composition" est basée sur un graphe de planification amélioré EPG [Pop et al., 2009] et un algorithme Harmony Search (ou la recherche par harmonie) [Geem et al., 2001] pour sélectionner la composition optimale ou quasi-optimale [Bekkouche et al., 2017]. L'utilisateur peut spécifier plusieurs contraintes non-fonctionnelles. Ainsi, notre méthode est capable de trouver un service composite qui optimise la valeur de QoS globale, tout en satisfaisant les contraintes spécifiées. Nous résumons ci-dessous nos principales contributions :

1. *L'élaboration des états de l'art des approches de composition :* nous avons effectué des états de l'art détaillés relativement aux trois classes d'approches de composition des services à savoir, les approches de composition automatique des services web, les approches de composition des services web sensible à la

QoS et les approches hybrides (qui combinent les techniques de planification et les techniques d'optimisation). Une analyse comparative des différentes contributions selon des critères d'évaluation que nous avons définis nous a permis d'une part de dégager les atouts et les limites de chaque approche, et d'autre part de mieux positionner notre approche pour voir comment il serait possible d'améliorer ce domaine de recherche.

2. *Mécanisme de Matching Sémantique* : le "matching" ou "appariement" est le moyen d'identifier un degré de similitude entre les concepts sémantiques (d'entrée et/ou de sortie) [He and Chang, 2003]. Il constitue une étape cruciale dans n'importe quel processus de composition. Nous proposons un nouveau mécanisme de matching en nous basant sur les types de matching définis dans [Klusch and Kapahnke, 2008]. Et cela dans le but de calculer les scores de similarité sémantique entre les services constituant le service composite recherché sur les différents niveaux du graphe de planification.
3. *Algorithme de sélection de la composition optimale (ou quasi-optimale)* : beaucoup de travaux ont été proposés pour trouver la composition optimale ou quasi-optimale dans le contexte de la composition automatique [Chifu et al., 2010], [Pop et al., 2009], [Pop et al., 2011b], [Chifu et al., 2015] et [Salomie et al., 2014]. Nous proposons un *algorithme Harmony Search* qui à notre connaissance n'a pas été utilisé pour résoudre le problème "QoS-aware automatic service composition". Ainsi le processus de sélection est formulé comme un problème d'optimisation mono-objectif où une fonction objectif est proposée. Cette fonction objectif évalue l'ensemble des solutions candidates de composition en prenant compte non seulement les contraintes non fonctionnelles de l'utilisateur mais aussi la qualité sémantique des différents services constituant la solution recherchée.
4. *Comparaison de l'algorithme de sélection avec d'autres méta-heuristiques* : nous comparons notre algorithme de sélection basé sur *l'algorithme Harmony Search* avec ses deux variantes développées postérieurement : *Improved Harmony Search (IHS)* [Mahdavi et al., 2007] et *Global Best Harmony Search (GHS)* [Omran and Mahdavi, 2008] dans le but d'améliorer le processus de sélection en termes de temps d'exécution de l'extraction de la solution d'une part et le taux d'optimalité d'une autre part.
5. *Validation et expérimentations* : nous présentons l'architecture de l'environnement de composition proposée en détaillant les différents modules le constituant. Nous proposons aussi des expérimentations pour valider notre contribution.

1.4 Organisation du document

Cette thèse est constituée de deux parties principales : la première est intitulée «État de l'art», elle introduit le contexte dans lequel se place cette thèse et montre un état de l'art de notre problématique. La seconde partie du document est intitulée « Contributions », elle présente l'approche proposée pour résoudre notre problématique et présente aussi le système de composition des services web proposé ainsi que l'ensemble des expérimentations effectuées.

1. Partie I : État de l'art. Cette partie inclut quatre chapitres :

- **Chapitre 2 : Composition des services web.**
  Ce chapitre introduit les principes de la technologie des services web en définissant le paradigme SOA, ainsi que les protocoles, les langages et les modèles qui sont en relation avec le concept des services web traditionnels et sémantiques. Par la suite nous abordons le problème de la composition des services ainsi que tous les concepts liés.
- **Chapitre 3 : Composition automatique des services web** Ce chapitre est consacré à une étude bibliographique des différentes recherches dans le domaine de la composition automatique des services web à savoir, le calcul de situation, la planification hiérarchique (HTN), le système PDDL, la planification basée sur les règles, la preuve par théorème automatique, le planificateur MBP, la planification à base de graphe, et la planification distribuée. Une étude comparative ainsi qu'une synthèse sont présentées à la fin du chapitre, dans laquelle nous identifions notamment les limitations de chaque approche étudiée.
- **Chapitre 4 : Composition des services web sensible à la QoS**
  Ce chapitre est consacré à l'analyse et à la définition de la problématique de sélection des services web dans une composition à base de paramètres de QoS. Une classification des différentes approches existantes relatives à l'optimisation du problème de sélection est présentée. Cette classification regroupe 4 catégories qui sont : les approches de résolution exactes, heuristiques, méta-heuristiques et les approches basées sur la dominance au sens de Pareto. Nous effectuons un survol des approches proposées dans la littérature. Ensuite, nous comparons les approches étudiées en termes de prise en compte de certaines exigences telles que l'algorithme de sélection utilisé, le type de la stratégie adoptée, la scalabilité, etc. Enfin, nous terminerons le chapitre par une synthèse
- **Chapitre 5 : Composition des services web fondée sur les techniques de planification et les techniques d'optimisation** Dans ce chapitre, on va s'intéresser au problème "QoS-aware automatic service composition". Ensuite l'essentiel des travaux qui traitent cette problématique est présenté. Par la suite une comparaison des différents travaux en termes de la prise en compte de certaines exigences des environnements orienté service est effectuée.

2. Partie II : Contributions. Cette partie regroupe deux chapitres :
   - **Chapitre 6 : Approche de composition automatique des services web basée sur l'algorithme Harmony Search et les contraintes non-fonctionnelles de l'utilisateur.**
     Ce chapitre présente notre approche pour résoudre le problème "QoS-aware automatic service composition" en satisfaisant les contraintes non-fonctionnelles (QoS) de l'utilisateur. Nous commençons par présenter notre formulation des différentes notions de base du problème étudié, telles que les concepts de service web sémantique, la QoS, les contraintes globales de QoS, la requête et le problème de la composition automatique du service sensible au QoS. Ensuite nous présentons en détails notre approche de composition automatique qui est basée sur une structure de graphe de planification amélioré et un algorithme de recherche d'harmonie en décri-

vant le mécanisme de matching sémantique proposé ainsi que l'algorithme de sélection.

- **Chapitre 7 : Implémentation et expérimentations.**
  Ce chapitre est consacré à l'expérimentation et à l'évaluation de notre système. Nous y présentons les résultats d'une expérimentation visant à valider l'approche adoptée et à tester les algorithmes conçus.
  Nous terminons cette thèse avec un chapitre conclusion et perspectives.
- **Chapitre 8 : Conclusion et Perspectives.**
  Ce chapitre présente une synthèse des principales idées de nos propositions. A l'occasion, nous reprenons certaines réflexions dans le but de mettre en avant les principales contributions et d'identifier les questions ouvertes et les perspectives de ce travail.

Chapitre 2

Composition des services web

Sommaire

2.1 Introduction

Dans ce chapitre, nous présentons, dans la première partie, le paradigme émergeant de l'Informatique Orientée Services (SOC – Service Oriented Computing) qui utilise les concepts de service et d'Architecture Orientée Service (SOA – Service Oriented Architecture) pour la construction rapide et à faible coût de nouvelles applications logicielles. Cette construction est rendue possible grâce au mécanisme de composition de services qui constitue actuellement un des challenges les plus importants de l'informatique orientée services. Lorsque les services fournis ne peuvent pas satisfaire individuellement la demande du client, il est alors possible, grâce à la composition de services, de les combiner ensemble afin de satisfaire cette demande.

Nous distinguons deux niveaux d'abstraction des services. Les services de base qui fournissent des fonctionnalités élémentaires, et les services composites qui agrègent un ensemble de fonctionnalités dans des applications de haut niveau proches des besoins de l'utilisateur. Les services composites sont construits par le processus de composition de services qui permet la liaison entre les besoins de l'utilisateur et les services de base disponibles. Les développeurs peuvent ainsi résoudre des problèmes complexes en combinant les services de base disponibles et les ordonner pour mieux satisfaire les exigences de leurs problèmes. De plus, le processus de composition de services accélère le développement d'applications, la réutilisation de services, et la réalisation de services plus complexes [Milanovic and Malek, 2004].

Dans ce chapitre, nous introduisons dans un premier temps la notion de services et nous présentons les concepts fondamentaux de l'architecture orientée services. Nous exposons dans un deuxième temps une classification des différentes catégories de composition de services rencontrées dans la littérature.

2.2 Concepts de base du paradigme orienté service

2.2.1 Évolution vers les approches orientées services

En génie logiciel, un certain nombre d'approches ont été proposées pour le développement de logiciels. Chaque nouveau paradigme proposé essaie de résoudre les limitations des paradigmes précédents en essayant de réutiliser certains principes en ajoutant de nouveaux concepts. C'est ainsi que les approches orientées objets [Taylor, 1997], orientées composants [Szyperski et al., 1999] et dernièrement les approches orientées services sont apparues.

Pour dissimuler la complexité d'intégration d'applications autonomes et hétérogènes, trois architectures à base de composants ont été proposées [Vinoski, 2004], [Schantz and Schmidt, 2001] : EJB (Enterprise Java Beans), CORBA (Common Object Request Broker Architecture), et .NET. Cependant, le couplage entre les objets est relativement fort dans ces technologies à base de composants distribués. Par exemple, la norme CORBA permettant la manipulation des objets à distance avec n'importe quel langage, nécessite une connaissance de la structure des objets par les applications clientes et par les fournisseurs. Ceci implique que les différents partenaires doivent définir au préalable des règles de transformation objets/ messages. En outre, on ne peut assembler que des objets CORBA (ou COM) entre eux, puisque chaque architecture propose sa propre infrastructure. Par conséquent, la mise en œuvre de ces architectures dans le cadre d'une infrastructure ouverte telle qu'internet soulève des difficultés. Pour palier les limites de toutes ces architectures, les efforts de conception ont donné lieu au concept d'architecture orientée service (Service Oriented Architecture ou SOA en abrégé) [Papazoglou et al., 2008]. Ainsi, les approches orientées services ont vu le jour.

Le paradigme orienté services (Service-Oriented Computing) est un paradigme interdisciplinaire destiné pour les applications distribuées. Son apparition a introduit une nouvelle manière de concevoir, d'intégrer, de déployer et d'utiliser les logiciels. Les approches orientées services [Huhns and Singh, 2005] mettent en avant l'idée de composer des services indépendants pour réaliser des applications logicielles agiles faiblement couplées. Ces approches exploitent le concept de service comme élément fondamental autour duquel les applications complexes sont développées. L'idée de

base est d'encapsuler les fonctionnalités offertes par les organisations sous forme de services. Par conséquent, l'intégration et la gestion des applications à base de services se focalisent au niveau des fonctionnalités sans tenir compte des technologies utilisées et en cachant les détails de l'implémentation.

Les approches à base de services offrent la possibilité de réaliser une interopérabilité à grande échelle en garantissant la souplesse nécessaire afin de s'adapter à l'évolution des technologies et des besoins des utilisateurs (individus ou entreprises). Un des principaux bénéfices des approches à base de services est le couplage faible entre le fournisseur et le consommateur de service d'une part, et entre les différents services réunis dans une même application d'une autre part. le consommateur d'un service n'a pas besoin d'avoir connaissance des détails techniques tels que la technologie d'implémentation et la plateforme d'exécution d'un service [Dustdar and Papazoglou, 2008]. Ce faible couplage permet le développement d'applications de façon parallèle et indépendante, ce qui entraîne une réduction des coûts de développement et d'intégration des applications [Fki, 2015].

2.2.2 Les services

Le service représente la brique de base dans une architecture orientée services. Il est assez difficile de donner une définition précise au service car il est utilisé dans plusieurs domaines. Il peut être défini tout simplement comme suit : un service est une action réalisée par une entité au profit d'une autre. De manière intuitive, la notion de service correspond à une abstraction des fonctionnalités d'une entité. Cette entité peut être simple comme une donnée ou un objet sur le web. Elle peut aussi être complexe comme un système d'information adaptable d'une entreprise. Avant l'utilisation d'un service, il est nécessaire de connaître les fonctionnalités qu'il accomplit et même ses caractéristiques non-fonctionnelles comme sa performance ou sa disponibilité. Plus particulièrement, un service logiciel (qui représente une entité logicielle mise à la disposition d'autres applications) doit être décrit et publié. Ses capacités sont ensuite découvertes par les utilisateurs éventuels qui peuvent par la suite exécuter le service pour obtenir la fonctionnalité demandée.

Nous étudions maintenant certaines définitions précises qui ont été données pour le service. Selon [Dustdar and Papazoglou, 2008], *« Services are self-contained processes deployed over standard middleware platforms, e.g., J2EE, or .NET that can be described, published, located (discovered), and invoked over a network... Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled. »*. Cette définition présente le service comme un processus autonome qui est décrit, publié, découvert et invoqué via le réseau. La conception du service est indépendante de sa technologie d'implémentation et sa plate-forme d'exécution. En plus, le service ne connaît pas le contexte dans lequel il va être utilisé. Cette indépendance caractérise fortement les services et elle facilite le faible couplage. Ceci permet d'utiliser des services réalisés avec des technologies d'implémentation différentes et de les déployer sur des plates-formes hétérogènes.

OASIS, un consortium mondial travaillant sur le développement, la convergence et l'adoption de standards pour les applications informatiques, propose la définition suivante : *« A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description... A service is ac-*

cessed by means of a service interface where the interface comprises the specifics of how to access the underlying capabilities. » [MacKenzie et al., 2006]. Cette définition présente un service comme un mécanisme permettant l'accès à une ou plusieurs fonctionnalités et dont l'accès est offert grâce à une interface définie. Cette interface décrit les fonctionnalités offertes par le service et un ensemble de contraintes et de politiques d'accès aux fonctionnalités offertes.

[Han et al., 2009] considère un service comme une abstraction du niveau métier implémentant les processus métiers. Une définition du concept de service est énoncée par [Cauvet and Guzelian, 2008], qui considèrent un service comme une unité réutilisable encapsulant un ou plusieurs fragments d'un processus métier et visant à satisfaire des buts métiers. A travers les différentes définitions, nous ressortons une idée principale, à savoir qu'un service permet d'exposer une ou plusieurs fonctionnalités décrites par une description de service. Cette description est utilisée par les clients du service pour rechercher, sélectionner et invoquer le service adéquat.

Le service est l'élément de base de l'architecture orientée service introduite ci-après.

2.2.3 Architecture orientée services (SOA)

SOA est un paradigme largement adopté fournissant un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services interopérables et indépendants. Généralement, le paradigme de l'architecture orientée service est basé sur un ensemble de principes qu'il faut respecter [Huhns and Singh, 2005], [Erl, 2008] :

- **Couplage faible** Le couplage est le niveau d'interaction entre deux ou plusieurs composants logiciels, deux composants sont couplés s'ils échangent de l'information. Ce couplage est fort s'ils échangent beaucoup d'information et il est faible dans le cas contraire. Dans une architecture SOA, le couplage entre les applications clientes et les services doit être faible. Vu que la communication avec les services web est réalisée via des messages décrits par le standard XML caractérisé par sa généricité et son haut niveau d'abstraction, les services Web permettent la coopération d'applications tout en garantissant un faible taux de couplage. Par conséquent, il est possible de modifier un service sans briser sa compatibilité avec les autres services composant l'application.
- **Interopérabilité** L'interopérabilité permet à des applications écrites dans des langages de programmation différents et s'exécutant sur des plateformes différentes de communiquer entre elles. En manipulant différents standards que ce soit XML ou les protocoles d'Internet, les services web garantissent un haut niveau d'interopérabilité des applications et ceci indépendamment des plateformes sur lesquelles elles sont déployées et des langages de programmation dans lesquels elles sont écrites. Ainsi, en s'appuyant sur un format d'échange de messages standard et sur l'ubiquité de l'infrastructure d'Internet, l'interopérabilité est donc une caractéristique intrinsèque aux services Web.
- **Reutilisabilité** L'avantage de la réutilisation est qu'elle permet de réduire les coûts de développement en réutilisant des composants déjà existants. Dans le cas de l'approche service web, l'objectif de la séparation des opérations en services autonomes est en effet pour promouvoir leur réutilisation. Ainsi, lorsqu'un client définit ses exigences, il est généralement possible de réutiliser des services

déjà existants pour satisfaire une partie des exigences. Ceci facilite la maintenance de l'application et permet un gain de temps considérable.

- **Découverte** C'est une étape importante qui permet la réutilisation des services. En effet, il faudra être en mesure de trouver un service afin de pouvoir en faire usage. L'approche services web tend à diminuer autant que possible l'intervention humaine en vue de permettre une découverte des services automatique. En effet, pour réaliser son application, un développeur peut simplement interroger un moteur de recherche de services afin de trouver le service adéquat.
- **Composition** Des collections de services peuvent être coordonnées et assemblées afin de former une composition de services. Cette possibilité de construire de nouveaux systèmes à partir de services existants constitue un des avantages de SOA. La composition permet la réutilisation des services web, après la découverte, il faut être en mesure de composer ces derniers en exploitant les technologies offertes par Internet et en utilisant un ensemble de standards pour la composition.

L'architecture orientée services repose sur un modèle qui décrit un ensemble d'interactions entre trois types d'acteurs : fournisseur de service, client de service, registre de services [O'sullivan et al., 2002]. Les interactions entre ces trois acteurs peuvent être résumées en trois primitives de communication : la publication, la découverte, et l'invocation. Les services sont créés est déployés par les fournisseurs. Les fournisseurs de service enregistrent la description de leurs services dans le registre (i.e. ; étape 1 : la publication). Les consommateurs de service envoient des requêtes au registre pour découvrir un service qui répond à leurs besoins (i.e. ; étape 2 : la découverte). Une fois le service choisi, le client de service peut se connecter au fournisseur et utiliser le service à travers sa description (i.e. ; étape 3 : l'invocation). Ce modèle d'interaction est illustré dans la Figure 2.1.

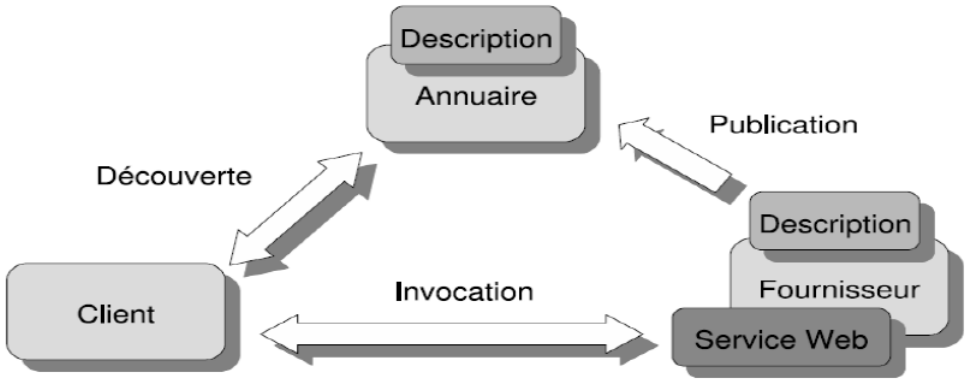

Figure *2.1:* *Interactions au sein d'une architecture orientée service*

L'objectif de l'architecture orientée services est de fournir un accès simple et rapide aux fonctionnalités fournies par des tiers distribués et hétérogènes. En ce sens, il faut mettre en place un environnement qui décrit les modalités d'interaction et d'utilisation des services. Cet environnement inclue un ensemble de standards, protocoles, et mécanismes qui peuvent être regroupés en deux catégories : mécanismes fonctionnels et mécanismes non-fonctionnels. Les mécanismes fonctionnels décrivent les principales tâches liées à l'utilisation des services à savoir, la publication, la découverte, la composition, et l'invocation. Les mécanismes non-fonctionnels prennent

en charge les besoins optionnels notamment la sécurité, les transactions, et la qualité de service.

2.3 Services web

Les services web sont certainement la technologie la plus populaire dans le monde industriel et académique pour migrer vers l'architectures à services. Les principaux mécanismes de fonctionnalités requises pour la réalisation d'une SOA sont assurés par une pile de standard de protocoles construite autour de la définition des services web. En fait, il existe deux types de services web, les services web SOAP, et les services web REST. Ces deux types de services web proposent différents standards et langages afin de mettre en place les principes de la SOA.

2.3.1 Service web SOAP

Les services web SOAP sont la première technologie qui a été proposée dans le but d'implémenter l'architecture SOA. Cette réalisation de la SOA est assurée par une panoplie de standards qui sont tous basés sur le modèle XML. XML est un format universel compréhensible par toutes les plateformes de programmation ; ce qui respecte les principes de la SOA, tels que : le couplage faible, et l'interopérabilité. Le W3C (World Wide Web Consortium) définit les services web SOAP comme étant :

Un service web est une entité logicielle capable d'être interopérable avec d'autres systèmes distribués. Il a une interface décrite dans une description facilement exploitable, appelée WSDL. Les autres systèmes interagissent avec le service web en se basant sur des messages SOAP, généralement transmis via le protocole HTTP avec une sérialisation XML.

La Figure 2.2 présente l'utilisation des technologies de services web par les acteurs fournisseur et demandeur :

- Le fournisseur de service publie ses web services sur l'annuaire en utilisant UDDI.
- Le demandeur recherche un web service avec les caractéristiques X, Z et Y.
- L'annuaire trouve un service avec les caractéristiques X, Z et Y, et envoie les informations sur le fournisseur du service et sur le service.
- Le demandeur demande le contrat du service du fournisseur.
- Le fournisseur envoie le contrat du service (WSDL).
- Le demandeur de service appelle le web service selon son contrat (appel en SOAP).
- Le web service retourne le résultat de l'appel (SOAP).

L'originalité de l'infrastructure des services web consiste à les mettre en place en se basant exclusivement sur les protocoles d'Internet tels que http et les formats standards d'échange tels que XML. L'infrastructure des services web s'est concrétisée autour de trois spécifications considérées comme des standards, à savoir SOAP, UDDI et WSDL [Chinnici et al., 2007]. Dans la suite de cette partie, nous allons présenter UDDI, WSDL et SOAP qui constituent les langages de base des services web.

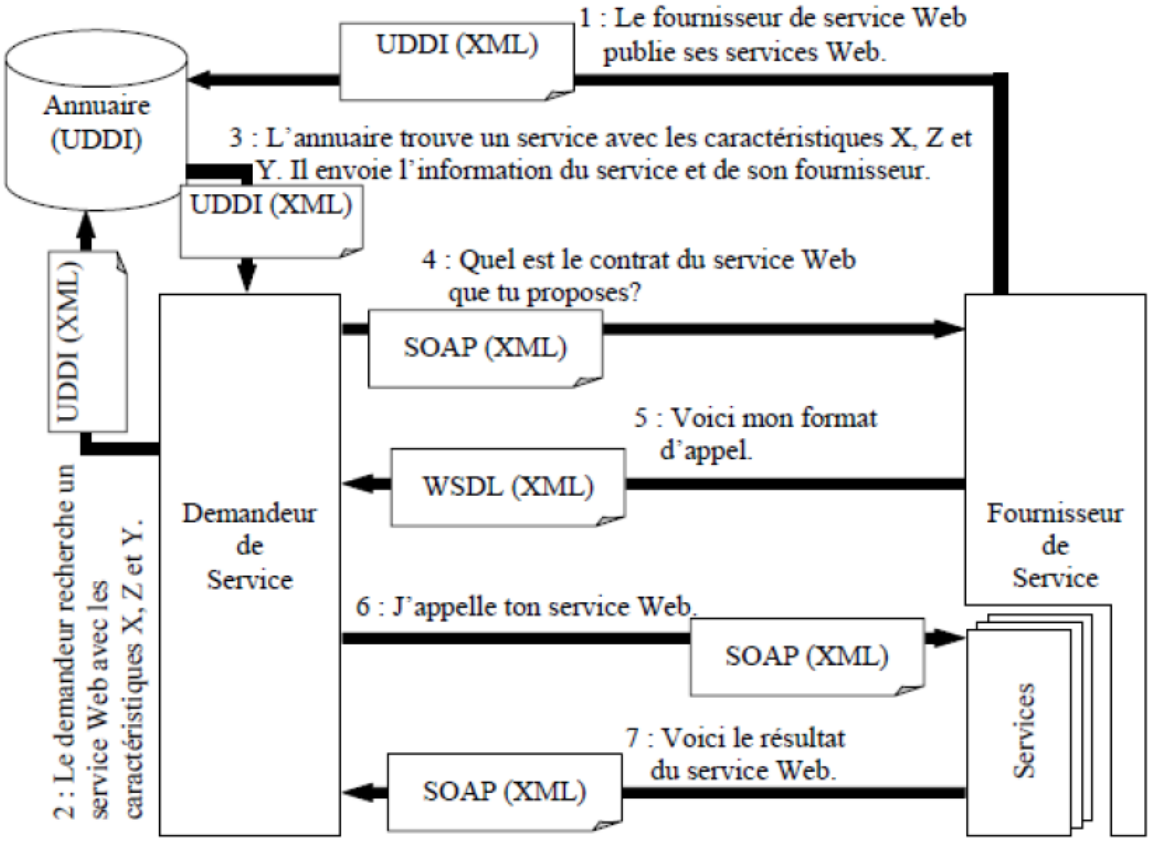

Figure ***2.2:*** *L'utilisation des technologies des services web par les acteurs fournisseur et demandeur*

2.3.1.1 Découverte UDDI

UDDI [Bellwood et al., 2002](Universal Description, Discovery and Integration) est une spécification technique sponsorisée par OASIS, et il est le résultat d'un accord interindustriel entre plusieurs organisations telles que Microsoft, Ariba, IBM, etc. Il offre un mécanisme de registre distribué de services permettant leur publication et leur découverte. Le registre se compose de trois parties : les pages blanches, pages jaunes et pages vertes. Les données stockées dans l'UDDI sont structurées en XML :

- **Pages blanches :** Comprennent des descriptions générales sur les fournisseurs de services. Nous y retrouvons donc des informations comme le nom de l'entreprise, ses coordonnées, la description de l'entreprise mais également l'ensemble de ses identifiants.
- **Pages jaunes :** Comportent des descriptions détaillées sur les fournisseurs de services catalogués dans les pages blanches de façon à classer les entreprises et les services par secteurs d'activités. De plus, elles recensent les services Web de chacune des entreprises sous le standard WSDL.
- **Pages vertes :** Fournissent des informations techniques précises sur les services fournis. Typiquement, on indiquera dans ces informations les adresses Web des services et les moyens d'y accéder. Ces informations incluent la description du service, du processus de son utilisation et des protocoles utilisées pour son invocation.

2.3.1.2 Description WSDL

WSDL (Web Service Description Language) est un langage de description de services Web basé sur XML proposé par W3C [Chinnici et al., 2007]. Le W3C a défini notamment les catégories d'informations à prendre en compte dans la description d'un service web. Les éléments décrits dans WSDL sont principalement les opérations proposées par le service Web, les données et messages échangés lors de l'appel d'une opération, le protocole de communication et les ports d'accès au service. Le standard WSDL offre une description sur deux niveaux, abstrait et concret (voir figure 2.3). Le niveau abstrait est utilisé principalement lors du processus de sélection tandis que le niveau concret est plutôt utilisé lors de l'invocation des opérations du service web.

La partie abstraite : Décrit les messages et les opérations disponibles via les éléments suivants :

- **Les types de données :** Informe sur les structures de données des paramètres utilisés par le service. Il est optionnel et un seul est autorisé. La balise utilisée est <types>).
- **Les messages :** Encapsule les données véhiculées pour l'opération invoquée. Pour chaque opération du service, il existe deux éléments le premier correspond à la requête tandis que le second correspond à la réponse. La balise est <message> et plusieurs messages sont autorisés.
- **Les opérations :** Composé d'une ou plusieurs opérations décrites par des éléments <operation>. Chaque opération possède un nom, 0 ou plusieurs messages en entrée et 0 ou plusieurs messages de sortie ou d'erreur.<portType> (plusieurs autorisés) .

La partie concrète : Décrit le protocole et le type d'encodage à utiliser pour les messages nécessaires pour invoquer un service Web particulier. Les principales informations décrites au niveau concret sont le protocole de communication et le service.

- **Le protocole de communication :** définit les protocoles de communication utilisés pour l'invocation du service web. Cette définition permet d'établir le lien, d'une part, entre le document et les messages SOAP et d'autre part, entre les messages SOAP et les opérations invoquées. Plusieurs protocoles de communication sont autorisés et la balise utilisée est <binding>.
- **Service :** L'accès au service est défini par une collection de ports d'accès regroupés dans un élément XML noté "service". Ces ports représentent les adresses URI (Uniform Resource Identifier) du service. Ainsi, un même service web peut être accessible depuis plusieurs ports.

2.3.1.3 Transport SOAP

SOAP (Simple Object Access Protocol) [Mitra et al., 2003] est un protocole pour l'échange d'informations structurées avec les services web, recommandé par le W3C. Il utilise principalement les protocoles HTTP (Hyper-Text Tranfer Protocol) et SMTP (Simple Mail Transfer Protocol) pour le transport de messages. SOAP se base sur le standard XML pour encoder les données. Par conséquent, il profite des avantages de généricité, d'abstraction et de portabilité qu'offre ce standard pour la normalisation et la structuration des données. SOAP forme la couche inférieure de la pile des protocoles

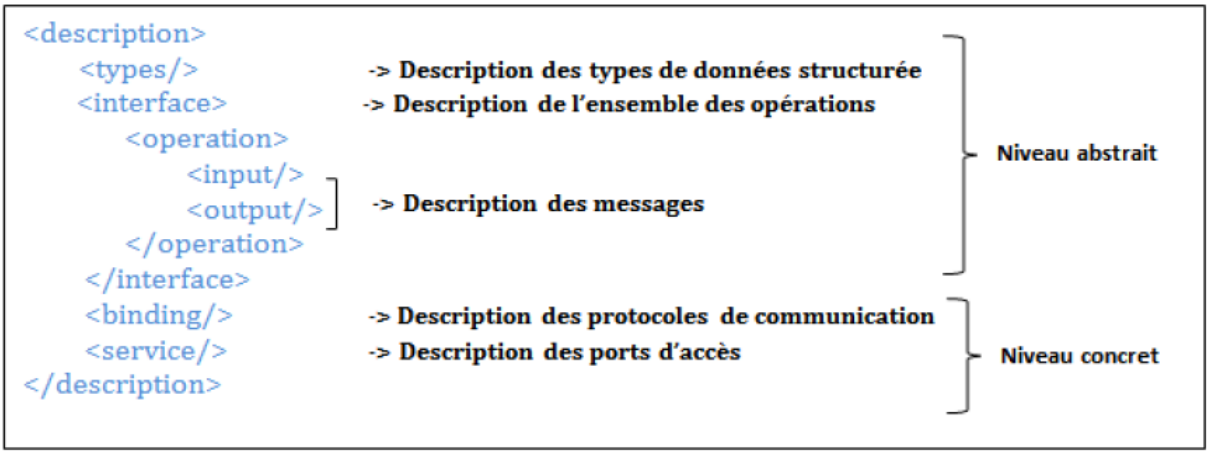

Figure ***2.3:*** *Structure générale d'un document WSDL 2.0.*

des services Web, fournissant un framework pour l'échange de messages sur lequel les services Web peuvent se baser.

Un message SOAP est structuré en trois parties : un entête et un corps à l'intérieur d'une enveloppe (voir Figure 2.4). L'entête est facultatif et il apporte des données supplémentaires au message SOAP. Le corps renferme, du côté client, l'opération du service invoquée ainsi que des valeurs des paramètres nécessaires cette invocation, et du côté service, le résultat de l'exécution de l'opération invoquée.

Figure ***2.4:*** *Structure d'un message SOAP*

2.3.2 Service web REST

REST a été proposé par Roy Thomas Fielding dans sa thèse *Architectural Styles and the Design of Network-based Software Architectures* [Fielding and Taylor, 2000]. REST est l'acronyme de *Representational State Transfer*. C'est un style d'architecture qui se base sur un ensemble de contraintes qui permettent, lorsqu'elles sont appliquées aux composants d'une architecture, d'optimiser certains critères propres au cahier des charges du système à concevoir. Cela signifie que REST définit un ensemble de contraintes sur les systèmes hypermedia distribués comme le World Wide Web afin d'optimiser les qualités désirées comme la séparation des tâches, la simplicité, la généricité ou les performances réseaux. REST fournit un véritable modèle pour décrire le fonctionnement que devrait avoir le web aujourd'hui. Bien que moins formel, ce modèle peut être comparé aux paradigmes objet ou entité-association.

L'architecture REST suit les principes de base du modèle client/serveur. L'information de base, dans une architecture REST, est appelée ressource. Toute information qui peut être nommée est une ressource : un article d'un journal, une photo, un service ou n'importe quel concept. Chaque ressource est un composant distribué qui est géré par une norme, et une interface commune rendant possible la manipulation des ressources. En plus de la ressource, l'architecture REST est basée sur les éléments suivants :

- *Identifiant de la ressource* : Une ressource est identifiée par un identificateur de ressource. Il permet aux composants de l'architecture d'identifier les ressources qu'ils manipulent. Sur le web ces identificateurs sont les URI *(Uniform Resource Identifier).*
- *Représentation de la ressource* : Les composants de l'architecture manipulent ces ressources en transférant des représentations de ces ressources. Sur le web, on trouve aujourd'hui le plus souvent des représentations au format HTML, JSON ou XML.
- *Opération sur la ressource* : Les ressources sont manipulées par le transfert des représentations à travers une interface uniforme adressée par l'identifiant de ressource.

2.4 Service web sémantique

Dans cette section nous allons définir les services web sémantiques, puis étudier les langages permettant la description de ce type de services web. L'objectif principal du web sémantique est de définir et de lier les ressources du web afin de simplifier leur utilisation, leur découverte, leur intégration et leur réutilisation dans le plus grand nombre d'applications [Shadbolt et al., 2006].

Le web sémantique doit fournir l'accès à ces ressources par l'intermédiaire de descriptions sémantiques exploitables et compréhensibles par des machines. Cette description repose sur des ontologies. Selon [Gruber, 1995], une ontologie est une spécification explicite d'une conceptualisation. Une conceptualisation est un modèle abstrait qui représente la manière dont les personnes conçoivent les choses réelles dans le monde et une spécification explicite signifie que les concepts et les relations d'un modèle abstrait reçoivent des noms et des définitions explicites. Donc, les services web sémantiques sont la combinaison de deux technologies : des services web et du web sémantique [McIlraith and Martin, 2003].

2.4.1 Langages de description de services web sémantiques

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

2.4.1.1 OWL-S

OWL-S (Ontology Web Language for Service) [Martin et al., 2007] est une ontologie web [Horrocks et al., 2003] pour les services web. Il a été développé en vue de soutenir la découverte automatique, la sélection et la composition de services Web. OWL-S permet de décrire les services web de façon non-ambiguë et interprétable par des programmes [Claro et al., 2005].

L'ontologie OWL-S structure notamment la description d'un service web en trois composants : le profil du service (ServiceProfile), son modèle de processus (ServiceModel) et ses liaisons (ServiceGrounding). La Figure 2.5 résume la structure de l'ontologie supérieure d'OWL-S.

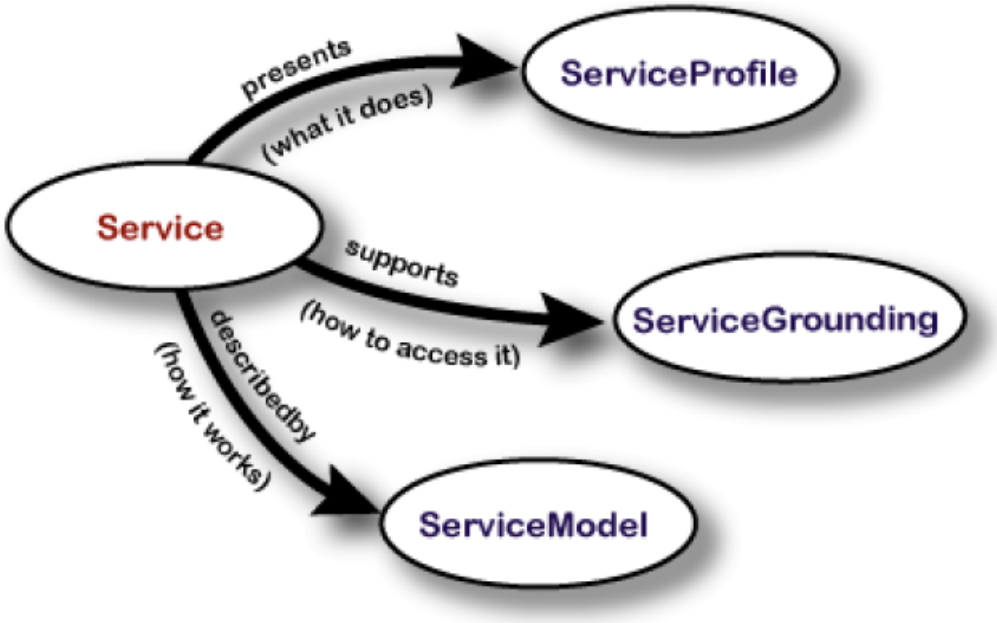

Figure *2.5:* *Structure de l'ontologie de services OWL-S*

- **ServiceProfile** Le profil du service indique ce que le service fait. Cette section est utilisée à la fois par les fournisseurs pour publier leurs services et par les clients pour spécifier leurs besoins pour déterminer si le service répond aux besoins nécessaires. ServiceProfile est utilisé pour publier le service en décrivant ses propriétés fonctionnelles (par exemple, entrée, sortie, pré- condition, et les effets) et les propriétés non-fonctionnelles (par exemple, la confiance de service, la fiabilité, l'objet, le coût, etc.).
- **ServiceModel** Le modèle de processus décrit comment le service fonctionne et comment l'utiliser. Il est utilisé pour décrire essentiellement le fonctionnement (modèle) d'un service composite. Il décrit comment demander le service et, en outre, il explique ce qui se passe lorsque le service est exécuté. OWL-S modélise les services en tant que processus défini par ses entrées/sorties.
- **Service Grounding** Explique comment interagir avec le service. Souvent, le service grounding spécifie un protocole de communication, des détails spécifiques au service ou la façon de contacter le service. Ce type d'informations est particulièrement utile pour l'invocation automatique de services.

OWL-S définit trois types de processus : les processus atomiques (AtomicProcess), simples (SimpleProcess) et composites (CompositeProcess). Chaque processus possède des entrées, des sorties, des pré-conditions et des effets. Les processus atomiques n'ont pas de sous-processus et peuvent être exécutés en une seule étape. Les processus simples fournissent une vue abstraite d'un processus existant. Cependant, à la différence des processus atomiques, un processus simple n'est pas associé à un grounding. Un processus composite est composé d'autres processus via les structures de contrôle telles que Sequence, Split, Split-Join, Any-Order, Choice, If-Then-Else, Repeat-While, Repeat-Until, et Iterate.

2.4.1.2 WSMO

WSMO (Web Service Modeling Ontology) est une ontologie qui décrit les différents aspects de la composition dynamique de services Web, y compris la découverte dynamique, la sélection, la médiation et l'invocation. Il est basé sur WSMF (Web Service Modelling Framework) [Fensel and Bussler, 2002] qui précise les principaux éléments de description sémantique des services web. Par ailleurs, pour modéliser un service web, WSMO utilise le langage WSML (Web Service Modeling Language) [Roman et al., 2005]. WSMO propose quatre éléments clés pour modéliser les différents aspects des services web sémantiques : les ontologies, les médiateurs, les objectifs ou goals, et les services (voir Figure 2.6).

- **WSMO Ontologies** Les ontologies fournissent une terminologie pour décrire sémantiquement des éléments appartenant à des domaines spécifiques en fournissant les concepts et les relations entre eux. Elles regroupent un ensemble d'attributs à savoir : concepts, relations, fonctions, instances et axiomes. Afin de décrire les propriétés sémantiques des relations et des concepts, une ontologie WSMO fournit aussi un ensemble d'axiomes, qui sont exprimés dans un langage logique.
- **WSMO Mediators** WSMO utilise un ensemble de médiateurs pour résoudre les incompatibilités (structurelles, sémantiques ou conceptuelles) détectées au niveau données ou processus afin de connecter les ressources WSMO hétérogènes.
- **WSMO goals** WSMO décrit les buts qu'un utilisateur de service web cherche à satisfaire à travers le goal. L'utilisateur définit ses critères de recherche en décrivant l'interface et les fonctionnalités attendues à travers la description du but. Un médiateur spécifique est utilisé pour connecter le but demandé avec le service web correspondant.
- **WSMO Web services** Différents aspects sont considérés dans la description d'un service web WSMO, à savoir l'aspect fonctionnel et l'aspect comportemental. Pour ce faire, WSMO utilise deux points de vue différents : la capacité (Capability) et l'interface (Interface). La capacité renseigne sur les propriétés fonctionnelles du service en décrivant les variables partagées (SharedVariables) entre les pré-conditions (Preconditions), post-conditions (Postconditions), assomptions (Assumptions) et effets (Effects).

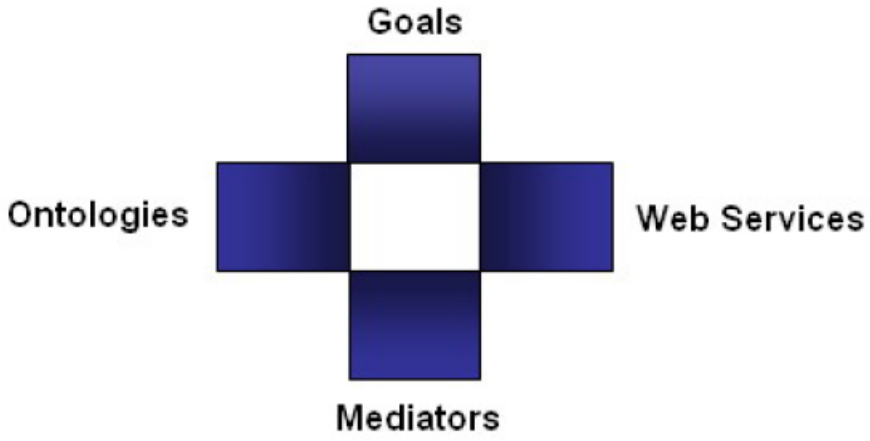

Figure *2.6:* *Structure de WSMO*

2.5 Processus de composition des services web

Si l'objectif du concepteur d'une application n'est pas atteint par l'invocation d'un simple service élémentaire, alors le concepteur doit combiner les fonctionnalités d'un ensemble de services. Ce processus est appelé composition de services [Benatallah et al., 2005]. Dans la littérature, plusieurs définitions ont été attribuées à la composition de services.

Nous retenons certaines d'entres elles qui nous paraissent les plus pertinentes. La composition de services a été définie par [Casati and Shan, 2002] comme étant la capacité d'offrir des services à valeur ajoutée en combinant des services existants probablement offerts par différentes organisations. Elle a été définie également par [Kellert and Toumani, 2003] comme étant une technique permettant d'assembler des services pour réaliser un objectif particulier, par l'intermédiaire de primitives de contrôles (boucles, tests, traitement d'exception, etc.) et d'échanges (envoi et réception de messages). Autrement dit, la composition de services spécifie quels services ont besoin d'être invoqués, dans quel ordre, quelles sont les données à échanger, et comment traiter les situations d'exceptions. La composition de services peut être vue comme un mécanisme qui permet l'intégration des services dans une application [Benatallah et al., 2005]. La définition la plus générale et la plus référencée dans la littérature est celle énoncée par [Benatallah et al., 2005]. Les auteurs considèrent la composition des services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services. Nous constatons que les différentes définitions s'accordent sur le fait que la composition de services vise la création de nouveaux services, offrant de nouvelles fonctionnalités, à partir des services existants. Le nouveau service résultat d'une composition de services est appelé *service composite*. Son exécution nécessite alors l'invocation de plusieurs autres services afin de faire appel à leurs fonctionnalités. Les services invoqués lors d'une composition de services sont appelés *services composants*. La Figure 2.7 présente le principe de la composition de services ; à partir d'un ensemble de services disponibles dans un registre, nous pouvons construire un service composite [Chollet, 2009].

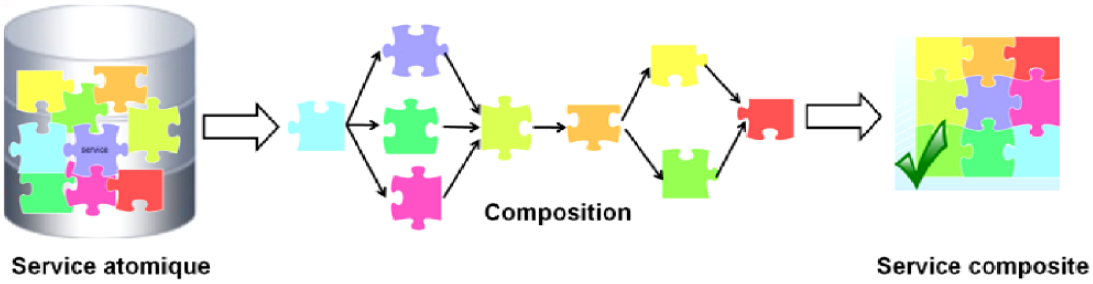

Figure *2.7:* *Processus de composition de services*

2.5.1 Cycle de vie de la composition de services

Le processus de composition de services permet le passage incrémental d'une spécification abstraite vers une composition concrète, c.à.d. une composition prête à être exécutée. Le cycle de vie de ce processus inclut quatre phases : la phase de *définition*, la phase de *découverte et de sélection*, la phase de *déploiement et d'exécution*, et la phase de *monitoring* [Sheng et al., 2014] (Figure 2.8)

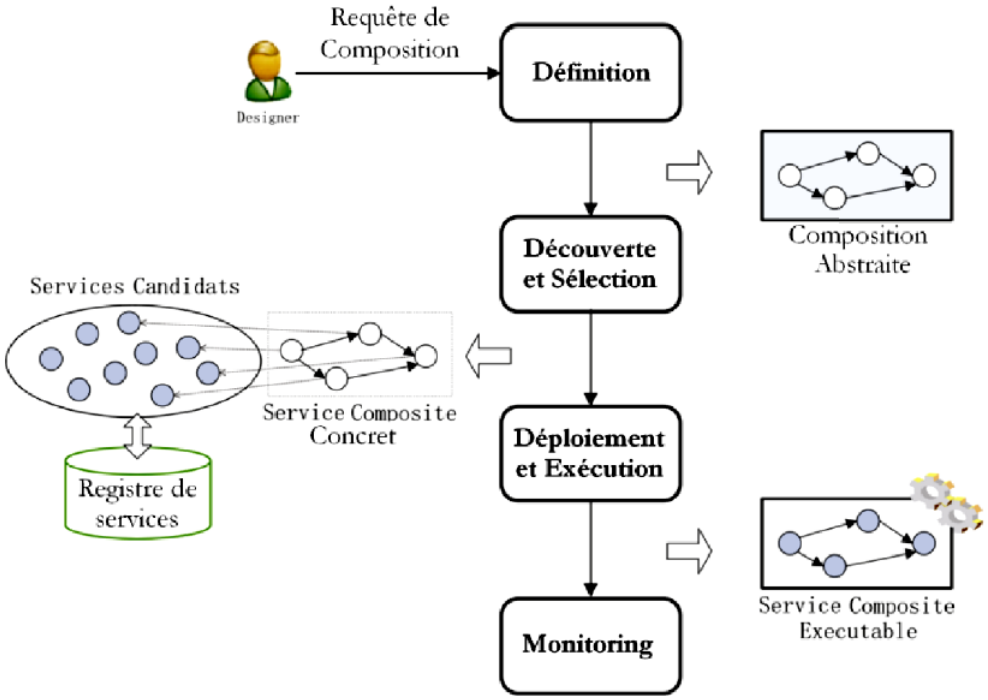

Figure *2.8:* *Le cycle de vie d'une composition de services*

- **La phase de définition :** cette phase consiste à identifier les fonctionnalités attendues du service résultant de la composition de services. Ces fonctionnalités peuvent être spécifiées sous la forme d'une description abstraite, c'est-à-dire d'une composition abstraite de services qui exprime les besoins fonctionnels de l'utilisateur. Cette description définit, d'une part comment les services abstraits

interagissent les uns avec les autres, et d'autre part, les exigences de QoS du service composite.

- **La phase de découverte et de sélection :** cette phase consiste à identifier les services nécessaires à la composition afin de répondre aux besoins fonctionnels identifiés au préalable. Une recherche dans le registre de services permet la découverte des services répondant aux fonctionnalités de chaque service abstrait de la composition. Cette recherche se base généralement sur la description (syntaxique ou sémantique) des services disponibles. Le résultat de la phase de découverte consiste en général en plusieurs services candidats possédant des fonctionnalités similaires, mais des propriétés non fonctionnelles (liées notamment à la QoS) différentes. La phase de sélection permet, quant à elle, selon les propriétés non-fonctionnelles, de déterminer, dans l'ensemble des services candidats précédemment identifiés, les services les plus appropriés pour la composition.
- **La phase de déploiement et d'exécution :** lors de cette phase, les services candidats sélectionnés précédemment sont déployés sur des plateformes permettant leur instanciation et invocation par les utilisateurs finaux. Le service composite instancié est exécuté par le moteur d'exécution qui est aussi responsable de l'invocation des services composants. L'exécution d'un service composite peut être vue comme une séquence d'échange de messages entre les services composants. Cet échange constitue le transfert des données de sortie d'un service composant aux services composants qui le suivent immédiatement dans la description de service composite.
- **La phase de monitoring :** l'exécution d'une instance de service composite est contrôlée par le processus de monitoring permettant d'obtenir une vue claire sur la façon dont le service composite et ses services composants évoluent au sein de l'environnement d'exécution. Le monitoring de la composition de services inclut plusieurs activités distinctes, notamment : (1) la journalisation et la visualisation des détails d'exécution du service composite et des instances de services composants, (2) l'obtention des statistiques de QoS à partir de l'analyse des données d'exécution du service composite et de celles des services composants, et (3) la vérification des exigences fonctionnelles, ainsi que l'évaluation des propriétés non-fonctionnelles du service composite.

2.5.2 Catégories de méthodes de composition

Les services qui possèdent diverses fonctionnalités peuvent être composés, donnant ainsi différentes façons d'établir de nouveaux services à valeur ajoutée. Les services composites obtenus peuvent à leur tour être invoqués par d'autres services, formant ainsi des modèles arbitrairement complexes de calculs distribués. Les approches proposées pour la composition de services peuvent être catégorisées suivant plusieurs aspects notamment le type de contrôle des services, le degré d'automatisation, et la gestion des services [Khanouche et al., 2016], [Yachir, 2014] (Figure2.9).

2.5.2.1 Type de contrôle

Selon le type de contrôle d'exécution des services, une composition de services peut être vue comme une orchestration ou comme une chorégraphie de services. Ces

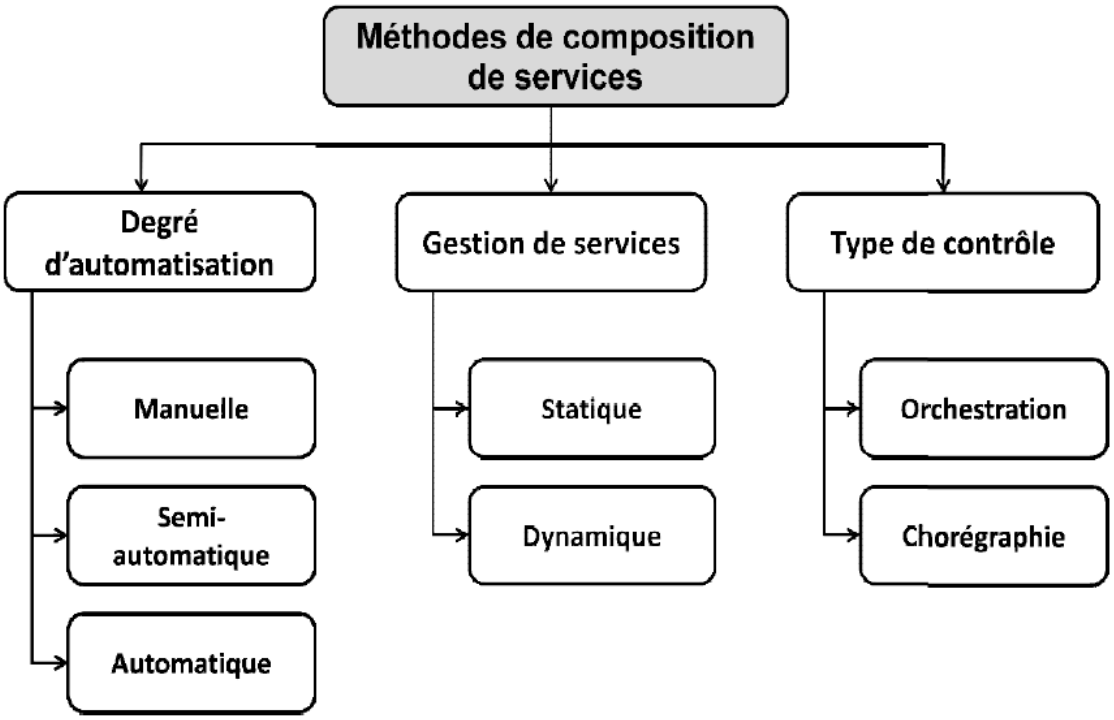

Figure *2.9:* *Classification des méthodes de composition de services*

deux points de vue de la composition sont utilisés actuellement dans la technologie des services Web [Chollet, 2009].

L'orchestration : une composition de type orchestration de services définit un schéma de services abstraits qu'une entité centrale instancie et exécute en fonction des services concrets disponibles [Zribi, 2014]. L'orchestration est définie dans [Barros et al., 2005], [Benatallah et al., 2005] comme un ensemble de processus exécutés dans un ordre prédéfini et ayant un objectif précis. L'invocation des services dans une orchestration est centralisée par un moteur d'exécution, appelé *orchestrateur* qui gère l'enchaînement des services à travers une logique de contrôle. La conception d'une orchestration de services nécessite alors de décrire les interactions entre le moteur d'exécution et les services selon un canevas prédéfini. Ces interactions correspondent aux appels, effectués par le moteur d'exécution, d'action(s) proposée(s) par les services composants [Juric et al., 2006].

La figure 2.10 illustre un exemple d'orchestration de quatre services. La requête de l'utilisateur (humain ou application) est d'abord transmise au moteur d'exécution (orchestrateur). Ce dernier invoque ensuite les services (ici service 1, service 2, service 3 et service 4) selon l'ordre d'exécution préalablement établi.

Il existe plusieurs langages standards d'orchestration de services web BPEL (Business Process Execution Language) est reconnu comme étant le langage le plus utilisé [Standard, 2007].

La chorégraphie : la chorégraphie est aussi appelée composition dynamique dans la mesure où l'exécution des services n'est pas régie de manière statique à l'instar d'une orchestration [Peltz, 2003]. Une composition de type chorégraphie n'est pas connue, ni décrite à l'avance, mais elle est construite progressivement. A chaque pas, un service choisit le service qui lui succède et implémente ainsi une partie de

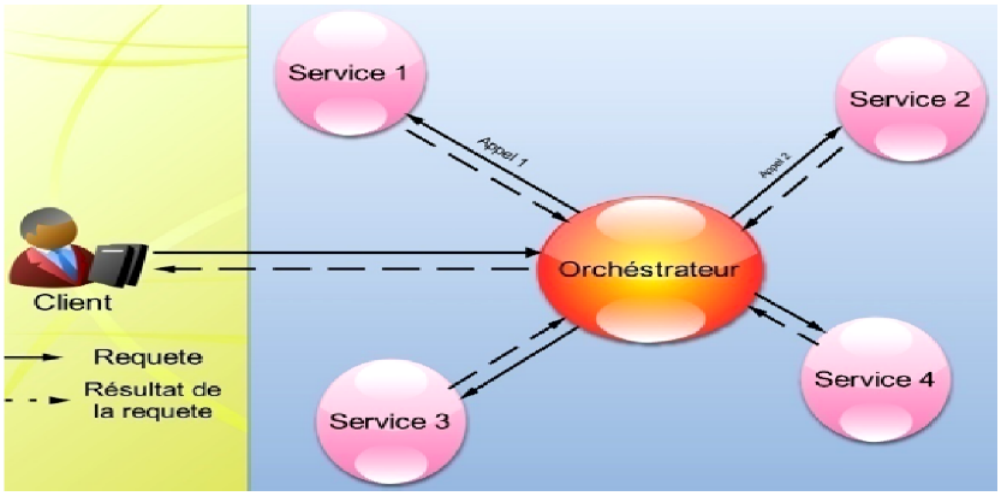

Figure *2.10:* *Vue d'ensemble de l'orchestration de services*

la chorégraphie. D'après [Barros et al., 2005], contrairement à l'orchestration qui dépend d'un orchestrateur central, la chorégraphie permet de décrire la composition comme une alliance de services collaborant pour la réalisation d'un but commun. Pour concevoir une chorégraphie, les interactions entre les différents services faisant partie de la composition doivent être décrites. La coordination et le contrôle sont assurés par chacun des services intervenant dans la composition rendant ainsi l'exécution du processus distribuée.

La figure 2.11 illustre un exemple de composition de quatre services (service1, service 2, service 3 et service 4) de type chorégraphie. L'utilisateur (humain ou application) envoie une requête qui est d'abord transmise au premier service (service 1). Ce dernier découvre ensuite le service lui succédant. Une fois le service 2 découvert, les deux services (service 1 et service 2) vérifient, par échange de message, la faisabilité de leur communication dans le le cadre de la requête. Si les échanges sont concluants, le résultat de l'exécution du service 1 est transmis au service 2. Ce processus se poursuit jusqu'à ce que le service 4 termine son exécution complétant ainsi le service requis à l'utilisateur.

Il est à noter que l'organisme de standardisation W3C a recommandé WS-CDL pour la description des interactions entre les services dans une chorégraphie [Ross-Talbot and Fletcher, 2006], [Kavantzas et al., 2005],

2.5.2.2 Degré d'automatisation

Selon le degré de participation de l'utilisateur dans la définition du schéma de composition, trois types de composition peuvent être distingués : *manuelle, semi-automatique ou automatique.*

La composition manuelle : dans ce type de composition, un expert s'occupe de la spécification des connexions entre les services à composer et se charge de la sélection des services les mieux adaptés à chaque fonctionnalité. Les limites de la composition manuelle résident, premièrement, dans la découverte et la sélection de services qui posent des problèmes de passage à l'échelle dans le cas d'un environnement à grand nombre de services. Deuxièmement, ce type de composition nécessite un savoir faire de l'utilisateur qui doit s'appuyer sur son expertise du domaine. En effet, l'utilisateur

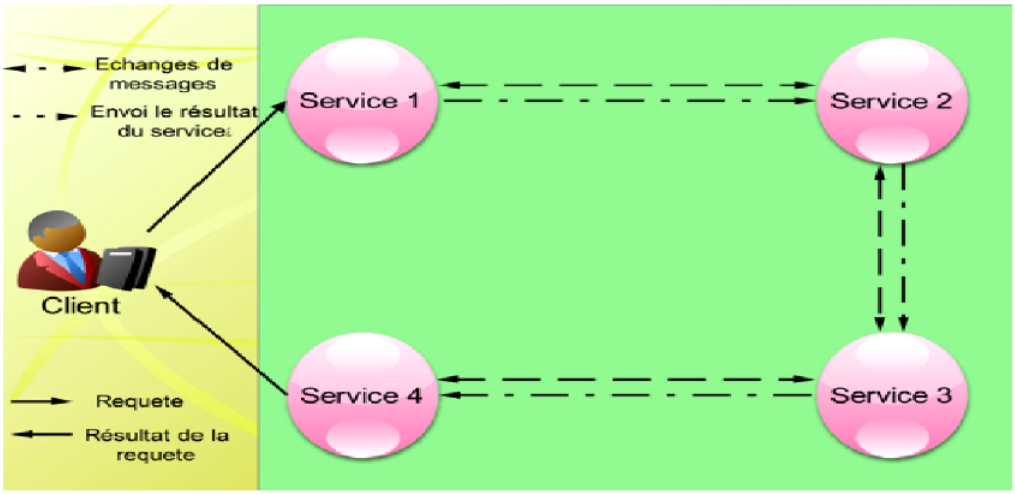

Figure *2.11:* *Vue d'ensemble de la chorégraphie de services.*

se charge d'abord de définir ses besoins en termes de composition de services à travers une requête. Ensuite, il génère à la main via un éditeur de texte, et sans l'aide d'outils dédiés, le schéma de composition en spécifiant les flux de données et de contrôle. Enfin, le schéma de composition généré est transmis à un moteur en vue de son exécution.

La composition semi-automatique : les méthodes semi-automatiques constituent des outils permettant d'assister l'utilisateur durant le processus de composition. En effet, ces méthodes offrent un certain nombre d'opérateurs graphiques pour aider l'utilisateur à construire le service composite sous la forme d'un workFlow. Ensuite, ces méthodes proposent à l'utilisateur des suggestions quant à la découverte et la sélection des services à composer. Une fois le service composite défini, ces méthodes instancient le workFlow correspondant et le soumettent à un moteur d'exécution. Bien que ces méthodes permettent de résoudre certains problèmes liés aux méthodes de composition manuelle, leur utilisation reste toutefois inadaptée aux environnements de services à large l'échelle. Par ailleurs, le rôle de l'utilisateur reste central dans la définition du schéma de composition et dans le choix définitif des services.

La composition automatique : dans ce type de composition, l'intervention de l'utilisateur n'est requise que pour la spécification des besoins sous la forme d'une requête de services. Par la suite, le processus de composition est entièrement pris en charge et est réalisé automatiquement de manière tout à fait transparente à l'utilisateur. En fonction des besoins de l'utilisateur, les services sont sélectionnés et composés à la volée en tenant compte des préférences et du contexte de l'utilisateur. Cette composition peut s'effectuer avant ou pendant l'exécution des services. Les méthodes de composition automatique sont souhaitables dans des environnements volatiles, tels que les environnements intelligents ambiants ou les environnements dans un contexte de mobilité où la disponibilité des services est dynamique et les besoins des utilisateurs sont variables et personnalisées.

2.5.2.3 Gestion de services

Le passage d'une spécification de composition abstraite à une application exécutable est mis en œuvre en effectuant une sélection de services.

Composition statique : une composition est dite statique lorsqu'elle est réalisée

au moment de la conception de l'application. Les services à composer sont sélectionnés et reliés au préalable d'une manière figée, c'est-à-dire, qu'ils ne peuvent pas être changés en fonction du contexte de l'utilisateur ; la gestion du flot de données est effectuée a priori [Dustdar and Schreiner, 2005]. La composition statique, appelée aussi composition proactive ou offline, engendre des applications peu flexibles qui parfois ne permettent pas de répondre aux exigences des utilisateurs. Ce type de composition n'est souhaitable que lorsque l'environnement et les services restent statiques ou sont peu évolutifs.

La composition dynamique : selon Osman et al., une composition de services est dite dynamique si les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur [Osman et al., 2005]. Dans ce type de composition qu'on appelle aussi composition réactive ou on-line, la sélection de services s'effectue « à la volée », suite à l'arrivée d'une requête ; les services sélectionnés sont ensuite reliés pour obtenir un service composite. Autrement dit, la sélection des services à composer et l'établissement des liaisons entre ces services ne sont pas prédéfinis à l'avance, mais dépendent des services disponibles au moment de l'exécution. Ceci permet d'élaborer différents scénarios de composition qui offrent les mêmes fonctionnalités, mais qui tiennent compte du contexte dynamique de l'utilisateur. La composition dynamique est donc bien adaptée pour répondre aux exigences des environnements dynamiques où la disponibilité des services peut varier dans le temps et où les besoins des utilisateurs sont variables et personnalisées. Cependant, sa mise en œuvre reste difficile en raison des changements fréquents de contextes des utilisateurs et des services dans de tels environnements.

2.6 Conclusion

La technologie des services web permet à des applications de dialoguer à distance via Internet, indépendamment des plates-formes et des langages sur lesquels elles reposent, en s'appuyant sur des protocoles standards. Dans ce chapitre, nous avons mis en relief les différents langages proposés par le consortium W3C permettant aux fournisseurs de décrire leurs services web. Cette étape de description est le premier processus indispensable dans le cycle de vie d'une application basée sur une Architecture Orientée Service (SOA). Ensuite, nous avons introduit la notion sémantique dans les services web. Les services web, de plus en plus utilisés dans le monde industriel, posent de nouveaux problèmes. Par exemple comment composer ces services ? Nous avons défini par la suite la composition de services web et cité les différentes catégories de méthodes de composition.

Le chapitre suivant introduira un état de l'art détaillé des approches de composition automatique des services web.

Chapitre 3

Composition automatique des services web

Sommaire

3.1 Introduction

L'un des verrous les plus importants du développement des architectures orientées services (SOA) est la création manuelle de services composites [Papazoglou et al., 2008]. Cette composition par un expert nécessite la mise au point de *middlewares* permettant de sélectionner les services répondant à ses exigences fonctionnels et non fonctionnels (qualité de service, confiance, etc.) ; d'organiser les services sélectionnés, les flux de données et de contrôle ; d'exécuter les services et surveiller leurs aléas d'exécution [Pellier and Fiorino, 2009].

Le processus de composition était un processus complexe qui prenait beaucoup de temps, et qui comportait des tâches de programmation de bas niveau répétitives. En outre, au fur et à mesure de la prolifération du nombre de services web disponibles, la découverte des services web adéquats et leur combinaison afin d'atteindre un but déterminé sont devenues des tâches très difficiles à gérer, d'où la nécessité d'automatiser le processus de composition.

Dans la littérature, de nombreux travaux ont porté sur l'automatisation de la composition des services ([Singh and Huhns, 2006] ; [Milanovic and Malek, 2004], [Bourdon, 2007]). Cela est justifié par l'évolution constante de l'offre de services en ligne ainsi que de leurs propriétés non fonctionnelles, ce qui rend une description

experte de la composition difficile à maintenir. La composition automatique permet aussi une adaptation aux exigences des utilisateurs. En d'autres termes, elle vise à tirer le meilleur parti des propriétés intrinsèques de la plate-forme d'exécution des services : décentralisation et modularité [Pellier and Fiorino, 2009].

La plupart des approches de composition automatique sont basées sur les techniques de planification de l'intelligence artificielle (IA). Ainsi, le problème de composition est défini comme un problème de planification où les services sont modélisés comme des actions et la composition comme un plan de connections des services web. Dans un premier temps, le planificateur construit le plan solution en explorant les services web disponibles. Ensuite, il développe les différents états (resp. plans) intermédiaires à partir de l'état initial (resp. d'un plan vide), en appliquant (resp. ajoutant) les services, jusqu'à atteindre l'état but (resp. plan solution). Enfin, il retourne soit un plan qui est un ensemble de services permettant d'atteindre le but, soit la situation d'échec [Ghallab et al., 2004].

D'une manière plus formelle, un problème de planification peut être défini comme un cinq-uplet $(S, A, R, s0, G)$, avec :

- S l'ensemble des états possibles du système considéré,
- $s0$ l'état initial du système
- G l'état final que l'on souhaite obtenir (l'état but). $s0$ et G appartiennent à S.
- A l'ensemble des actions possibles qui permettent le changement d'états.
- R l'ensemble des transitions d'état possibles ; R est donc inclus dans $(S * A * S)$ et représente les pré-conditions et post-conditions de chaque action, les post-conditions étant 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, l'état initial et l'état final sont directement associés à une requête d'un utilisateur. Les actions de A peuvent être effectivement considérées comme des services web [Rao and Su, 2004].

L'étude bibliographique montre que les différentes recherches dans ce domaine appartiennent à l'une des classes suivantes : le calcul de situation [McIlraith and Son, 2002], la planification hiérarchique (HTN) [Wu et al., 2006], le système PDDL [McDermott et al., 1998], la planification basée sur les règles [Medjahed et al., 2003], la preuve par théorème automatique [Waldinger, 2000], le planificateur MBP [Pistore and Traverso, 2001], la planification à base de graphe [Ghallab et al., 2004], et la planification distribuée.

Dans ce qui suit, nous présentons en détail les travaux proposés pour chacune des classes. Ensuite, nous discutons les différents travaux présentés pour nous permettre de mieux positionner notre approche de composition par rapport aux approches existantes.

3.1.1 Calcul de situation

Le calcul de situation [McIlraith and Son, 2002] est un langage logique du premier ordre qui utilise les notions d'actions, de fluents et de situations pour effectuer des raisonnements sur un domaine dynamique (en évolution). Dans ce type de raisonnement, chaque situation S est la résultante d'une séquence d'actions appliquée à

une situation initiale $S0$. Un domaine dynamique peut être l'objet d'une succession de situations quand il subit un ensemble d'actions diverses. Chaque situation décrit l'historique des actions appliquées à ce domaine sans renseigner sur son état suite à l'exécution de ces actions. Ce dernier est en fait décrit par des fluents qui sont des valeurs de fonctions et de relations appliquées à chacune des situations du domaine.

Golog est un langage de programmation logique de haut niveau conçu à partir du langage formel de calcul de situation en vue de représenter des actions primitives ou complexes dans un domaine dynamique. Ce langage propose un ensemble de constructions extra-logiques utilisées pour définir des actions primitives qui peuvent ensuite être réutilisées pour constituer des actions complexes. Des travaux de la littérature [McIlraith and Son, 2002] proposent une approche de composition dynamique des services web fondée sur le langage *Golog*. Dans cette approche, la découverte, la sélection et la composition automatiques des services web sont effectuées par des agents logiciels intelligents. Les auteurs considèrent un service web comme étant une action qui peut être primitive ou complexe. Les actions primitives ont pour rôle de changer l'état d'un élément, ou encore de changer la connaissance de l'agent. Les actions complexes, quant à elles, sont des compositions d'actions primitives. La base de connaissances de l'agent est en fait constituée de pré-conditions et effets des services web exprimées au moyen du langage de calcul de situation. L'agent utilise un langage procédural comprenant des constructions telles que si-alors-sinon, quand-fait, etc., combiné à des concepts relatifs aux services web pour représenter un service composite qu'il génère en répondant à une requête initiale. Cette requête, y compris les contraintes associées, est également exprimée sous forme de procédure générique en utilisant le calcul de situation. L'avantage majeur de l'utilisation des techniques de calcul de situation et plus spécialement du langage *Golog* est l'expressivité naturelle fournie par ce langage.

Quelques années plus tard, [Phan and Hattori, 2006] conçoivent un système *conGolog* (extension du *Golog*) qui permet de mapper les spécifications OWL-S au langage de calcul de situation. Aussi, [Sohrabi et al., 2009] ont proposé de tenir compte des préférences utilisateur dans la planification à base de *Golog*. Les préférences sont exprimées à l'aide d'un langage du premier ordre défini par les auteurs qui utilise une version modifiée de *Golog*. Les résultats d'évaluation montrent l'efficacité de l'introduction préférences dans la recherche des compositions optimales.

3.1.2 Réseau hiérarchique des tâches (HTN)

Hierarchical Task Network (HTN) est une approche qui crée des plans par décomposition de tâches, en plus des éléments standards de la planification (c'est-à-dire 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).

Dans [Wu et al., 2006], [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 ([Ankolekar et al., 2002]). SHOP2 (Simple Hierarchical Ordered Planner) est un planificateur de type réseau de tâches hiérarchiques (Hierarchical Task Network). La différence entre SHOP2 et les autres planificateurs consiste à ce que SHOP2 planifie les tâches dans l'ordre dans lequel elles seront ensuite exécutées.

Ceci permet de connaitre l'état courant du monde dans chaque étape du processus de planification.

Selon [Wu et al., 2006], 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) et de plus il est plus efficace que le système Golog [McIlraith and Son, 2002].

[Sirin et al., 2003] présentent une méthode semi-automatique pour la composition en utilisant le langage DAML-S. La méthode proposée utilise les attributs fonctionnels et des attributs non-fonctionnels définis dans le *ServiceProfile* pour proposer à l'utilisateur les services qui semblent les plus appropriés pour répondre à sa requête. Pour ce faire, les attributs fonctionnels qui sont les paramètres d'entrées et de sorties du service sont représentés par des classes OWL et sont filtrés par un moteur d'inférence basé sur OWL et Prolog. Le moteur d'inférence peut ordonner les services web filtrés en fonction de l'ordre d'éloignement des concepts. L'éloignement est un paramètre qui dénote l'importance des différences entre les classes OWL. Pour affiner le résultat, si plus d'une correspondance est trouvée, le système filtre le résultat en fonction des contraintes fournies par l'utilisateur sur les attributs non-fonctionnels. Les attributs non-fonctionnels sont les attributs utiles qui ne sont pas fournis par les attributs fonctionnels, par exemple, la localisation de la mesure lorsque le service ne la fournit pas lui-même. Seuls les services web qui passent le filtre de contraintes sont présentés à l'utilisateur. La composition semi-automatique est attrayante car elle permet de surmonter les difficultés liées, tout d'abord, à la capture des besoins de l'utilisateur, puis à la composition automatique qui, le plus souvent, ne permet pas de garantir l'exactitude de la composition. L'utilisateur étant proactif, il peut s'assurer que la composition est bien celle qu'il souhaitait. De plus, la méthode proposée est simple et montre que la génération de services web composés peut être effectuée en combinant les fonctionnalités de la machine et les compétences de l'homme.

CASCOM [1] (Context-aware business Application Service CO-ordination in Mobile computing environment) [Helin et al., 2005] était un projet européen (2004-2007) dont l'objectif principal était de mettre en oeuvre, valider et tester une valeur ajoutée pour une infrastructure d'appui du web sémantique utilisant des services à travers des réseaux fixes et mobiles. Le planificateur OWLS-Xplan [Klusch et al., 2005] (du type HTN et Fast Forward Chaining) prend en entrée un ensemble de services OWL-S, une description de domaine de planification fondée sur des ontologies OWL et une requête de planification (but à satisfaire) et renvoie une composition sous forme d'une séquence de services satisfaisant ce but. Un post-traitement permet d'enrichir cette séquence de services par d'autres structures de contrôle (par exemple, split + join). Les outils développés dans ce projet permettent de sélectionner, composer et exécuter les services. L'architecture de CASCOM est du type multi-agents car un agent PA (Personal Agent) envoie une requête à l'agent SCPA (Service Composition Planner Agent) qui obtient les descriptions des services nécessaires à la composition via l'agent SDA (Service Discovery Agent). Le plan de composition obtenu de manière centralisée par l'agent SCPA est transmis à l'agent SEA (Service Execution Agent) qui se charge de l'invocation des services et renvoie les résultats vers l'agent PA.

Qiu et al. [Qiu et al., 2006], [Qiu et al., 2007] se sont basés sur les diagrammes d'états (statecharts) et les graphes acycliques orientés, ou DAG (Directed Acyclic Graph) afin de représenter les plans de composition de services qui sont construits

1. http ://www.ist-cascom.org/

par l'approche de planification HTN. Dans leur travail, les auteurs proposent une extension d'OWL-S, nommée OWL-SC (OWL-S Context) permettant d'intégrer le contexte à la description des services Web. Les auteurs distinguent trois catégories de contexte : le contexte de l'utilisateur (U-Context), le contexte du service web (W-Context) et le contexte de l'environnement (E-Context). Chaque catégorie de contexte est représentée par une ontologie et est intégrée aux ontologies existantes d'OWL-S.

Les auteurs dans [Tang et al., 2013] proposent un framework pour une composition de services web dynamique basé HTN. Ils développent un algorithme de matching de services en deux étapes qui permet de réduire le temps du processus de composition.

Le travail présenté dans [Na-Lumpoon et al., 2014] propose un framework qui fournit une composition automatique de services basée sur le calcul facile (Fluent Calculus). Cette étape, qui utilise la technique de l'inférence de la planification IA en association avec la méthode de programmation par contraintes de flux, génère un plan de composition qui contient des actions en séquence ou en parallèle.

3.1.3 Planning Domain Definition Language (PDDL)

Le *Planning Domain Definition Language* (PDDL) [McDermott 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.

3.1.4 Planification basée sur les règles

Dans [Medjahed et al., 2003], une technique de *rule-based planning* est utilisée pour engendrer des services composites à partir de descriptions déclaratives de haut niveau. Cette méthode utilise des règles de composabilité pour déterminer dans quelle mesure deux services sont composables. L'approche proposée se déroule en quatre phases : une phase de spécification de haut niveau de la composition désirée en utilisant le langage CSSL (Composite Service Specification Langage). La phase de correspondance utilise des règles de composabilité pour générer des plans conformes aux spécifications du service demandeur. Dans la phase de sélection, si plus d'un plan est généré, la sélection est effectuée par rapport à des paramètres de qualité de la composition. Dans la phase de génération, une description détaillée du service composite est automatiquement générée et présentée au demandeur. La principale contribution de cette approche est la notion de règles de composabilité. Les règles de composabilité considèrent les propriétés syntaxiques et sémantiques des services web. Les règles syntaxiques incluent des règles pour les types d'opérations possibles et pour les liaisons protocolaires entre les services (les bindings). Les règles sémantiques incluent des règles concernant la compatibilité des messages échangés, la compatibilité

des domaines sémantiques des services, mais également des règles de qualité de la composition.

Les auteurs définissent les règles de composabilité syntaxique et les règles de composabilité sémantique comme suit :

Les règles syntaxiques incluent :

- La vérification de la compatibilité des modes de fonctionnement, 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 5 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 la pertinence).

SWORD [Ponnekanti and Fox, 2002] est un système d'élaboration de services web composés à base de règles. Contrairement à [Medjahed et al., 2003], SWORD utilise le modèle entité-relation (ER) [Gardarin et al., 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.

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, c'est-à-dire une même entrée peut provoquer 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.

Mokhtar et al. [Mokhtar et al., 2005] ont proposé un système de composition de services sensible au contexte qui étend OWL-S pour intégrer des règles telles que « la distance à l'utilisateur < 300m » ou « mémoire restante > 128Ko » et composer des applications basées sur ces règles intégrées. Bien que cette approche permet au concepteur de spécifier explicitement comment composer des applications pour un

contexte donné, elle ne peut pas s'adapter à des utilisateurs différents parce que (1) des règles prédéfinies ne peuvent généralement pas être modifiées une fois qu'elles sont déployées, (2) il est difficile de définir une règle générique qui est applicable à tous les utilisateurs, et (3) la préférence de certains utilisateurs peut être trop complexe à définir comme étant un ensemble de règles.

3.1.5 Preuve par théorème automatique (Automatic Theorem Proving)

[Waldinger, 2000] 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] applique une synthèse structurelle de programme (SSP, Structural Synthesis of Programme) pour une composition automatique. 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 définissent les services web ou la requête. Ces interfaces sont composées de variables typées, de constantes, de liaisons entre les variables et constantes (binding) d'un axiome. L'axiome inclut seulement les propriétés structurales, i.e. les informations des entrées/sorties, en liant les entrées avec les sorties. Il étend son langage en y ajoutant des "variables de contrôle" qui sont des variables sans valeur mais qui peuvent servir à la définition des pré-/post-conditions. Les interfaces des services web sont vues comme des axiomes et l'interface de la requête est vue comme un théorème à prouver. SSP déduit des interfaces prédéfinies une "preuve" de l'interface qui représente la requête, et déduit de la preuve la composition des services Web. Lämmermann assimile les services web composés à des programmes et utilise le fait que les programmes sont des preuves.

3.1.6 Planificateur MBP (Model Based Planner)

Les techniques de planification *model checking* sont utilisées dans [Pistore et al., 2005] et [Pistore and Traverso, 2001] pour composer automatiquement les web services et synthétiser leur surveillance. Ces techniques sont destinées à être utilisées dans les domaines de planification non déterministe avec une observabilité partielle et des buts prolongés (le but implique souvent des conditions complexes sur le plan résultat et non seulement sur son état final).

L'idée principale derrière le paradigme de la planification en *model checking* est que le planificateur doit résoudre des modèles théoriques où les propriétés du domaine de planification sont formalisées comme des formules temporelles. La génération de plan se fait en explorant l'espace d'états du modèle. A chaque étape, les plans sont générés par la vérification de la véracité de certaines formules appropriées dans le modèle.

Les plans résultants de l'utilisation des techniques *model checking* sont des plans complexes qui peuvent coder des comportements séquentiels, conditionnels et itératifs. Ils sont assez expressifs pour traiter l'observabilité partielle et les buts prolongés.

Le langage utilisé pour la spécification des buts est *EaGLe* [Dal Lago et al., 2002]. Il permet d'exprimer des genres différents des buts qui sont appropriés pour des domaines non déterministes. Le planificateur utilisé pour la composition est le planificateur *MBP*(Model Based Planner) [Bertoli et al., 2001]. Ce dernier implémente les techniques de *model checking* pour planifier avec des buts EaGLe dans des domaines non déterministes. Il peut produire automatiquement des plans pour résoudre les différents problèmes de planification, comme la planification sous l'observabilité totale, partielle ou nulle.

3.1.7 Planification à base de graphe

Le graphe de planification [Ghallab et al., 2004] est un graphe acyclique dirigé qui contient plusieurs niveaux, dans lequel les arcs relient seulement deux niveaux successifs. Il contient des tuples (A_i, L_i) d'un ensemble d'actions et littéraux. Par exemple dans le niveau 1, l'ensemble des actions A_1 contient les actions dont les pré-conditions sont des noeuds dans L_0 et l'ensemble des littéraux L_1 contient l'ensemble des littéraux L_0 et les effets des actions A_1. Dans le problème de composition, chaque action représente un service web, les pré-conditions d'une action sont les concepts d'entrée d'un service web et représentent les littéraux dans le graphe et les effets d'une action sont les concepts de sortie et représentent aussi les littéraux dans le modèle. L'ensemble des actions A_i est un ensemble de services web dont les paramètres d'entrée sont des littéraux dans A_{i-1}. L'ensemble L_i contient les littéraux des L_{i-1} et les paramètres de sortie des A_i. De cette façon, le graphe de planification est étendu d'un niveau à un autre, jusqu'à ce que tous les concepts de sortie d'une requête donnée soient contenus dans l'ensemble des littéraux L_n du dernier niveau n.

Les auteurs dans [Hatzi et al., 2012] présentent une approche intégrée pour la composition automatique des services web sémantiques en utilisant les techniques de planification. L'avantage de cette approche est que le processus de composition, ainsi que la découverte des services atomiques qui participent à la composition, sont sensiblement facilités par l'incorporation de l'information sémantique. Les descriptions des services en OWL-S sont transformées en un problème de planification en utilisant le langage PDDL. La mise en œuvre a été réalisée par le développement et l'intégration de deux frameworks, à savoir PORSCE II et VLEPPO. PORSCE II est responsable du processus de transformation, de la sémantisation des services et de la production des services composites. VLEPPO est un planificateur qui acquière automatiquement les solutions en invoquant des planificateurs externes. Une étude de cas est également présentée pour démontrer la faisabilité et la performance de l'approche proposée.

Dans [Wang and Huang, 2014], les auteurs proposent une méthode heuristique de composition automatique des service web en se basant sur un modèle de graphe de planification. Le processus de composition est divisé en deux modules : la construction du graphe de planification et la recherche d'un schéma de composition. En effet, ils présentent un algorithme pour la construction du graphe et la recherche des solutions possibles basée sur une heuristique "Graph Based Heuristic Web Service Composition *(GBHWSC)* algorithm". Enfin, les expériences vérifient l'efficacité de la méthode proposée.

[Zheng and Yan, 2008] ont fourni quatre stratégies pour supprimer les services web redondants pendant le processus de génération de graphe de planification. Ils ont tenté de trouver une solution dans les plus brefs délais. Cependant, sans l'élagage vers l'arrière (backward pruning), les solutions peuvent contenir des services web

redondants. En outre, ils n'ont pas considéré les paramètres de QoS et la dimension semantique.

[Li et al., 2010] se concentrent sur le problème d'appariement sémantique dans les algorithmes de composition basée sur la planification du graphe. Dans la recherche en arrière, ils ont utilisé la similarité des concepts et un seuil prédéfini pour calculer les degrés de matching du service, qui ont été ajoutés dans le graphe de planification comme des poids des services. En outre, ils ont proposé un mécanisme pour supprimer les services produisant le même paramètre mais avec des poids inférieurs pour garder le graphe de planification aussi simple que possible. Malheureusement, le seuil n'est pas si facile à définir, ce qui limite l'application de cette méthode sur des services réels.

Une autre approche centralisée pour la composition de services web est développée dans la thèse de Freddy Lécué [Lécué, 2008b]. Dans cette approche, les services sont vus comme des fonctions ayant des paramètres i) d'entrée, et de sortie sémantiquement annotés par des concepts d'une ontologie de domaine et ii) des préconditions et effets conditionnels sur le monde. La composition des services web est alors considérée comme une composition des liens sémantiques où les lois de cause à effet ont aussi un rôle prépondérant. Dans cette approche, une matrice de liens sémantiques (Causal Link Matrix - CLM) est utilisée pour représenter les liens sémantiques entre les différents services comme des correspondances output-input ou les lois entre effets et préconditions de services ont aussi un role prepondérant. Ensuite, à partir de la requête définie par un état initial et un but, un algorithme récursif de regression (Ra_4C) est utilisé pour extraire un plan qui répond à la requête.

Dans [Shiaa et al., 2008] les auteurs présentent une approche automatique de composition de services avec un mécanisme de matching sémantique. Etant donné une requête de l'utilisateur (objectifs, entrées et sorties), un ensemble de services est découvert à partir d'un annuaire de services, en appliquant un matching sémantique entre les propriétés du service et la requête de composition. Ensuite, un graphe est créé dynamiquement en se connectant sémantiquement des nœuds similaires (services uniques) à chacun d'eux. Une fois que le graphe est créé, une recherche sur celui-ci est réalisée en construisant des structures d'arbres à partir de nœuds but aux nœuds d'entrée. Cependant, il n'y a pas de résultats expérimentaux pour valider ce modèle.

Une approche de composition de services à base de graphes est également proposée dans [Rodriguez-Mier et al., 2016]. En effet, la correspondance sémantique entre les paramètres d'entrée/sortie des services est utilisée pour construire un DAG (Directed Acyclic graph) représentant la composition de services. Les techniques d'élagage avec retour arrière et la dominance en termes d'entrée/sortie sont ensuite utilisées pour supprimer les services inutiles à la composition, et ainsi réduire la taille du DAG. Un algorithme de recherche est enfin utilisé pour trouver le meilleur service composite en termes de longueur du chemin d'exécution et de nombre de services.

Les auteurs dans [Yan et al., 2008] présentent un algorithme de composition automatique en utilisant le graphe AND/OR. Dans cette proposition, un graphe AND / OR est créé selon une requête de l'utilisateur, reliant les services par leurs entrées/ sorties. Ensuite, une recherche sur le graphe est effectuée en utilisant l'algorithme de recherche AO^*. Bien que cette proposition montre une bonne performance sur de grande annuaires, les auteurs n'ont pas implémenté des techniques d'optimisation pour améliorer la scalabilité de l'algorithme.

Une autre approche intéressante basée sur le graphe a également été présentée dans [Kona et al., 2008]. Dans cet article, les auteurs présentent un framework efficace

pour la composition qui prend en charge la découverte des services web sémantiques. La composition est générée par l'exécution d'une recherche en avant des opérateurs afin de trouver une composition faisable. Les auteurs ont également évalué le système avec le dataset "Web service challenge 2006" et ont présenté une expérimentation détaillée. Leurs résultats expérimentaux démontrent la bonne performance de ce système, cependant ils n'ont pas analysé les optimisations des services afin de supprimer les informations redondantes.

Dans [Lamine et al., 2017], les auteurs proposent une technique de graphe de planification pour la composition de services web en prenant en considération l'information contextuelle des services web. Ils utilisent d'abord une ontologie basée sur un modèle de contexte pour l'enrichissement des descriptions de services web avec des informations sur le contexte. Ensuite, ils transforment le problème de composition en un problème de planification à base de graphe pour construire un ensemble de meilleurs services composites basés sur le contexte de l'utilisateur. La construction du graphe de planification est basée sur un processus de découverte sémantique et contextuel des services web. Cela permet, pour chaque étape de la construction, d'ajouter les services web les plus adaptés en termes de compatibilité sémantique entre les paramètres des services, et leur similarité contextuelle avec le contexte de l'utilisateur. Dans l'étape de la recherche en arrière, les scores de similarité sémantique et contextuelle sont utilisés pour trouver la liste des services web composites. Enfin, dans l'étape de classement, un score est calculé pour chaque solution candidate et un ensemble de solutions classées sont retournées à l'utilisateur.

Dans [Arch-int et al., 2017], les auteurs proposent également un système de composition de services web sémantiques basé sur un graphe de planification composé de deux sous-systèmes : le temps de gestion et le temps d'exécution. Le sous-système de gestion de temps est responsable de la préparation du graphe de dépendances dans lequel un graphe de dépendance des services associés est généré automatiquement selon les règles de correspondance sémantique proposées. Le sous-système d'exécution est chargé de découvrir les services web potentiels et la composition de services web non redondants pour la requête de utilisateur. L'approche proposée a été appliquée à l'intégration des données médicales dans différentes organisations de santé et a été évaluée selon deux aspects : la mesure du temps d'exécution et la mesure de l'exactitude.

Dans [Omrana, 2014] le but du travail est d'aboutir à des plans de composition efficaces et d'optimiser le temps de composition dynamique. Sa recherche propose une approche de composabilité offline. Cette approche consiste à identifier l'ensemble de services composables à différents niveaux, en amont du processus de construction des plans de composition devant être effectué de façon dynamique. Elle définit d'abord un modèle de description de services web multi-aspects qui s'aligne avec les spécifications W3C, à savoir, WSDL 2.0, SAWSDL et WS-Policy 1.5. Ce modèle intègre les propriétés descriptives prévues par ces trois standards et les enrichit par de nouvelles propriétés, dans le but de capturer le maximum d'informations sur un service web, tout en restant conforme aux standards. Sur la base de ce modèle descriptif, l'approche proposée identifie les propriétés descriptives impliquées dans la composabilité offline des services et définit six règles multi-aspects qui exploitent ces informations pour traiter les aspects de composabilité de deux opérations de services web : fonctionnel, non-fonctionnel, contextuel, orienté données et technique. Elle définit aussi une démarche globale de vérification automatique des aspects de composabilité offline de deux opérations.

3.1.8 Planification distribuée

En plus des méthodes centralisées de planification, on trouve dans la littérature d'autres méthodes dites "distribuées". Une méthode de planification distribuée impose un schéma décentralisé dans lequel tous les services participant à une composition de services doivent collaborer afin de réaliser cette composition. Et donc la tâche de réalisation d'un service composite est répartie entre tous les services concernés. Selon [El Falou, 2010] la planification distribuée peut être considérée simplement comme une spécialisation de la résolution distribuée de problèmes, où le problème est résolu par un plan distribué. Cependant, une ambiguité existe puisqu'on ne sait pas exactement ce qui est distribué. En effet, il est possible que le plan soit distribué entre plusieurs systèmes d'éxecution. Ou alors, le processus de planification est distribué, qu'il soit ou non le plan résultat peut l'être.

Dans [El Falou, 2010], le cœur du travail est de fournir une architecture multi-agents de composition de services fondée sur les techniques de planification distribuée. Dans son architecture, chaque agent raisonne sur ses services afin de déterminer le meilleur plan local pouvant satisfaire en partie l'utilisateur. Ensuite, les agents se coordonnent et fusionnent leurs plans locaux afin d'atteindre un but commun prédéfini par l'utilisateur, créant ainsi un plan global représentant une composition possible de leurs services. Une autre contribution de son travail consiste à intégrer au processus de planification, un processus de diagnostic actif et de réparation. Le but est de détecter les fautes pendant l'exécution du service composite, de calculer l'ensemble des états fautifs possibles et de réutiliser le processus de planification pour raffiner les états et réparer les fautes.

Les auteurs dans [Pellier and Fiorino, 2009] proposent une architecture originale de composition automatique de services web par des techniques de planification. Son originalité repose sur la conception d'un modèle de planification entièrement distribué -aussi bien au niveau du contrôle qu'au niveau des données- dans lequel les agents raisonnent conjointement sur leurs services respectifs pour atteindre un but commun prédéfini par l'utilisateur, créant ainsi un plan global représentant une composition possible de leurs services.

[Charif and Sabouret, 2007] proposent une approche dynamique pour la composition de services. L'approche étudiée s'appuie sur la collaboration décentralisée entre une collection de services, dont le but est d'atteindre un objectif donné (chorégraphie des services). En faite, modéliser des capacités de collaboration au niveau des services nécessite qu'ils soient dotés de propriétés d'autonomie, d'adaptabilité et d'interactivité. Ces propriétés, qui font défaut aux services web, ont par ailleurs été largement étudiées dans les communautés agent et multi-agent. L'approche proposée se base sur un modèle de coordination multi-agents, pour la chorégraphie de services Web, dans lequel les capacités de collaboration des services sont modélisées à l'aide d'agents introspectifs. Ces agents sont capables de raisonner sur leurs propres actions (ou sur les services qu'ils offrent), de décomposer dynamiquement une tâche en fonction de leurs compétences, et de se coordonner avec d'autres agents pour pallier leurs limites et couvrir les besoins à satisfaire. L'approche se déroule en quatre étapes :

- un utilisateur humain formule ses besoins en services à travers une interface graphique.
- un module de découverte de services extrait ensuite des mots clé de la requête de l'utilisateur pour découvrir des services candidats à la composition. Ceux-ci sont recherchés à partir d'un registre en ligne de services web.

- les services candidats tentent alors de satisfaire cette requête en s'envoyant des messages. C'est la phase de chorégraphie de services. Leurs interactions sont régies par un protocole de coordination.
- enfin, parmi les services ayant pu répondre à la requête de l'utilisateur, un sous-ensemble répondant aux contraintes globales énoncées par celui-ci est sélectionné.

Dans [Gharzouli, 2011], l'objectif principal est de fournir un scénario purement décentralisé qui supporte la composition automatique des web services sémantiques distribués sur un réseau P2P. A partir d'un réseau purement non structuré, il cherche à composer un web service qui répond à une requête particulière. Ce service peut être composé de plusieurs services appartenant à différents pairs afin de créer un espace collaboratif entre les différents pairs participant à la réalisation d'un but commun. Il a développé un algorithme épidémique qui permet de rechercher des services distribués sur tous les pairs du réseau afin de composer de nouveaux services personnalisés. Aussi, dans cette solution, il a utilisé une table –dite table de composition- afin de préserver la trace du chemin de composition pour une éventuelle future réutilisation. Cette table est considérée comme une mémoire cache utilisée pour accélérer la recherche des web services déjà composés.

Un modèle de composition de services orienté but dans le contexte d'environnements ubiquitaires mobiles est proposé dans [Chen et al., 2016]. Ce modèle consiste en un algorithme de planification décentralisé basé sur le chaînage arrière pour la découverte de services. Une architecture adaptative est également introduite pour permettre aux chemins d'exécution des services composites de s'adapter dynamiquement aux changements fréquents de l'environnement ubiquitaire. Cela permet de réduire les défaillances d'exécution de services, et de diminuer les efforts de réexécution.

3.2 Etude Comparative

Dans cette section, nous présentons une étude comparative entre les différentes approches de composition automatique étudiées. Nous commençons par identifier des critères d'évaluation que nous allons utiliser pour effectuer cette comparaison. Ensuite nous dressons un tableau récapitulatif qui va nous permettre d'analyser les différents travaux et notamment de mieux positionner notre approche de composition.

3.2.1 Critères de comparaison

Beaucoup d'états de l'art sur la composition des services web (automatique ou non) ont été proposés par divers chercheurs, [Srivastava and Koehler, 2003], [Milanovic and Malek, 2004], [Rao and Su, 2004], [Agarwal et al., 2008], [Ulrich et al., 2005], [Marconi and Pistore, 2009], [Pessini, 2014] durant cette dernière décennie. La plupart de ces états de l'art ne précisent pas clairement quelles exigences doivent être satisfaites par une approche permettant de résoudre le problème de la composition automatique avec succès. Les critères d'évaluation que nous avons identifiés sont :

- **Le type de planification** : ce critère correspond au type de planification que l'approche a adoptée (voir section précédente).
- **Nature de l'approche (centralisée ou distribuée)** : selon que la responsabilité de la coordination de l'exécution de la composition de services web soit

confiée à une seule entité ou bien partagée entre les différents fournisseurs des services, une composition est dite centralisée ou distribuée.

- **Dynamicité** : contrairement à la composition statique où le nombre de services fournis est limité et les services à composer sont spécifiés au préalable, la composition dynamique, elle, est initiée par une requête de l'utilisateur. Elle permet de découvrir, sélectionner et combiner dynamiquement les services à partir des annuaires de services web, au moment de l'exécution de la requête, tout en tenant compte des contraintes de l'utilisateur.
- **Sémantisation** : cette caractéristique permet de spécifier si une approche de composition prend ou non en compte la dimension sémantique dans la description des services web et quelle est la technologie utilisée pour ceci.
- **Considération de QoS** : les approches "QoS-aware" prennent en compte non seulement les propriétés fonctionnelles des services, mais aussi les propriétés non fonctionnelles, traitant des aspects de qualité tel que le temps de réponse, le prix, la disponibilité, etc. Considérer les aspects de qualité de service au moment de décider quels services sont à inclure dans un schéma de composition des services est important lorsque les exigences fonctionnelles sont satisfaites par plus d'un service.
- **Considération des pré-/post-conditions** : la sémantique des services web est cruciale afin d'obtenir des solutions précises et correctes pour le problème de la composition automatique. Il y a un consensus sur le fait que la sémantique basée sur les inputs/outputs des services est nécessaire. Cependant, il n'est pas encore décidé que ce niveau de sémantisation est suffisant. Puisque le but de cette dernière est de capturer le comportement des services, des informations plus complexes sont requises. Il semble que pour la plupart des services et beaucoup de scénarios de composition réels, de bons résultats peuvent être obtenus si les pré-/post-conditions sont prises en compte pour mieux exprimer les fonctionnalités des services.
- **Considération du contexte** : la définition la plus largement adoptée dans le domaine de l'informatique du contexte est celle de Dey [Abowd et al., 1999] : « le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes ». Ainsi, les informations contextuelles concernent notamment l'environnement (température, humidité, etc.), l'utilisateur (la température corporelle, le rythme cardiaque, etc.), les appareils (affichage, énergie, etc.) et les applications (type, configuration, etc.). Les données du contexte sont en général perçues dans leurs formes brutes. Des mécanismes de description sémantique du contexte sont alors nécessaires afin de le rendre compréhensible et intelligible par le système. Une approche de composition est adaptable au contexte. Elle est "Context awareness" si elle prend en compte des changements du contexte afin d'adapter les services qu'elle fournit pour répondre, en particulier, aux attentes des utilisateurs.
- **les préférences de l'utilisateur** : Les préférences sont un moyen naturel pour faciliter la définition des contraintes non fonctionnelles dans la requête de l'utilisateur. Elles peuvent être flexibles, d'une part, pour éviter des réponses vides dues aux contraintes très spécifiques de l'utilisateur, et d'autre part, pour

fournir un ensemble adéquat de résultats pertinents, même lorsque l'utilisateur spécifie des contraintes assez générales. Par ailleurs, la théorie des ensembles flous a été utilisée pour modéliser fidèlement le point de vue des utilisateurs dans l'évaluation des préférences [Lemos et al., 2012].

Le tableau 3.1 présente une synthèse des résultats de notre étude comparative des travaux que nous avons présentés. Ce tableau comparatif permet en fait de montrer les caractéristiques de chacun de ces travaux.

Le symbole $\surd$ veut dire que le critère est satisfait par l'approche alors que le symbole $\times$ veut dire l'inverse.

Travaux étudiés	**Le type de planification**	**Nature de l'approche**	**Dynamicité**	**Sémantisation**	**QoS**	**Pré-/post-conditions**	**Contexte**	**les préférences de l'utilisateur**
[McIlraith and Son, 2002]	Calcul de situation	Centralisée	×	DAML-S	×	×	×	×
[Sohrabi et al., 2009]			×	DAML-S	×	×	×	√
[Wu et al., 2006]	HTN	Centralisée	×	DAML-S	×	×	×	×
[Sirin et al., 2004]			×	DAML-S	×	×	×	×
[Sirin et al., 2003]			×	DAML-S	×	×	×	×
[Helin et al., 2005]			×	OWL-S	×	×	×	×
[Qiu et al., 2007]			×	OWL-S	×	×	√	×
[McDermott, 2002]	PDDL	Centralisée	×	×	×	×	×	×
[Medjahed et al., 2003]	RBP	Centralisée	×	OWL-S	√	×	×	×
[Ponnekanti and Fox, 2002]			×	×	×	×	×	×
[Mokhtar et al., 2005]			×	OWL-S	×	×	√	×
[Waldinger, 2000]	Preuve par théorème	Centralisée	×	×	×	×	×	×
[Lämmermann, 2002]			×	×	×	×	×	×
[Pistore et al., 2005]	MBP	Centralisée	×	×	×	×	×	×
[Hatzi et al., 2012]	Planification à base de graphe	Centralisée	×	OWL-S	×	×	×	×
[Wang and Huang, 2014]			×	OWL-S	×	×	×	×
[Zheng and Yan, 2008]			×	×	×	×	×	×
[Li et al., 2010]			×	×	×	×	×	×
[Lécué, 2008b]			×	OWL-S	×	√	×	×
[Shiaa et al., 2008]			×	OWL-S	×	×	×	×
[Yan et al., 2008]			×	×	×	×	×	×

Travaux étudiés	**Le type de planification**	**Nature de l'approche**	**Dynamicité**	**Sémantisation**	**QoS**	**Pré-/post-conditions**	**Contexte**	**les préférences de l'utilisateur**
[Rodriguez-Mier et al., 2016]	Planification à base de graphe	Centralisée	×	OWL-S	×	×	×	×
[Kona et al., 2008]			×	×	×	×	×	×
[Lamine et al., 2017]			×	OWL-S	×	×	√	×
[Omrana, 2014]			√	OWL-S	×	×	×	×
[Arch-int et al., 2017]			×	OWL-S	×	×	×	×
[El Falou, 2010]	Planification distribuée	Distribuée	√	OWL-S	×	×	×	×
[Pellier and Fiorino, 2009]			×	OWL-S	×	×	×	×
[Charif and Sabouret, 2007]			×	OWL-S	×	×	×	×
[Gharzouli, 2011]			×	OWL-S	×	×	×	×
[Chen et al., 2016]			×	OWL-S	×	×	×	×

Tableau 3.1: *Tableau comparatif des approches de composition automatique*

3.3 Discussion

Dans une approche de composition automatique, l'intervention de l'utilisateur est réduite à la spécification de l'état initial et du but à atteindre. Par la suite, le planificateur se charge de trouver un enchaînement approprié de services qui réalise l'objectif spécifié. Les approches de composition par planification semblent être les mieux placées pour répondre à cette exigence. Toutefois, ces approches nécessitent une description standard et intelligible des services afin qu'ils soient interprétables par des machines.

D'après le tableau comparatif 3.1, nous pouvons remarquer que la plupart des travaux menés pour résoudre la problématique de construction automatique des plans de composition des services web utilisent les techniques de planification à base de graphe, cela peut être expliqué par la flexibilité de ces techniques et la possibilité d'utiliser des heuristiques qui permettent d'augmenter la performance des algorithmes de composition proposés.

En examinant le tableau comparatif, nous remarqu'on que la qualité de service(QoS) est ignorée dans les travaux proposées. Le seul travail qui traite cette qualité est celui de [Medjahed et al., 2003], il prend en considération les paramètres non fonctionnels dans la définition des règles de composabilité sémantique. La dynamicité est une autre exigence qui n'a pas été suffisamment explorée dans ces techniques de planification. La grande majorité des travaux visent à produire un schéma de composition statique (le plan généré) sans explorer le cas de la composition au moment de l'exécution où les schémas de composition sont abstraits. les seuls travaux sont ceux de [El Falou, 2010] et [Omrana, 2014] qui proposent des méthodes dynamiques. On peut en déduire que la dynamicité reste une des limites majeures de ces approches.

Nous notons aussi que ces approches proposent des solutions de composition qui sont dépendantes des modèles sémantiques de description des services web, ainsi du niveau de composabilité considéré qui se limite souvent à l'appariement des entrées/-sorties et ne considèrent pas les pré-/post-conditions des services web. Seul [Lécué, 2008b] prend en compte cette métrique dans le processus de matching.

Pour ce qui est du contexte, nous constatons que la plupart des techniques de composition issu de la planification de l'IA produisent un service composite constitué d'un ensemble de services atomiques sans considération des informations contextuelles qui constituent un moyen efficace pour assurer le dynamisme dans le processus de composition. On ne trouve que les travaux de [Qiu et al., 2007], [Mokhtar et al., 2005] et [Lamine et al., 2017] qui prennent en considération le contexte. Aussi, peu de travaux utilisent les préférences de l'utilisateur dans la recherche des plans de composition.

Nous constatons aussi que la plupart des architectures proposées reposent sur une centralisation de la composition. En effet, ces méthodes centralisées n'assurent pas toujours le passage à l'échelle (scalabilité) à cause de leurs complexités exponentielles. Et donc une conception centralisée du problème de composition est une limite car les fournisseurs de services peuvent être réticents à fournir une description détaillée du fonctionnement de leurs services.

3.4 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur la composition automatique des services web. Nous avons identifié principalement 8 catégories d'approches de composition automatique : les approches basées sur le calcul de situation, celles basées sur la planification hiérarchique, celles basées sur le langage PDDL, celles basées sur la planification à base de règles, celles basées sur la preuve par théorème, celles basées sur le planificateur MBP, celles basées sur le graphe de planification et enfin, celles basées sur la planification distribuée.

Les propriétés non-fonctionnelles d'un service web constituent un ensemble d'attributs de qualité de service (QoS) qui représentent des critères cruciaux dans la sélection des services participants à une composition. Plusieurs approches de composition à base de QoS ont été proposées dans la littérature. L'objectif du prochain chapitre est d'étudier les approches les plus pertinentes.

Chapitre 4

Composition des services web sensible à la QoS

Sommaire

4.1 Introduction

La sélection des services composites « QoS-aware service composition », est l'une des problématiques les plus importantes de l'architecture orientée service (SOA). Elle a fait l'objet de plusieurs travaux de recherche comme en témoigne la littérature.

Étant donné un modèle de workflow composé d'un ensemble d'activités abstraites, la sélection de services composites répond à la question : comment sélectionner efficacement un service approprié pour chaque activité ?. En effet, chaque activité peut avoir plusieurs services candidats identiques de point de vue fonctionnalité et qui ont différentes propriétés non fonctionnelles. Le but de la phase de sélection est de concrétiser un workflow en un workflow exécutable qui satisfait les critères non-fonctionnels. La figure 4.1 présente une illustration de processus de sélection à base de QoS. Dans cette figure, chaque cercle creux représente un service abstrait (activité), qui est une unité fonctionnelle non-exécutable et doit être instanciée par l'un de ses services concrets candidats correspondants qui sont représentés par des cercles pleins [Canfora et al., 2005], [Fki, 2015].

Au début, un workflow prédéfini est constitué d'un ensemble d'activités abstraites, et chaque activité peut être réalisée par l'un des services concrets candidats après l'étape de découverte de service pour chaque activité. Les services concrets dans chaque groupe candidat d'une activité offrent typiquement des caractéristiques non fonctionnelles distinctes. Ainsi il s'agit, pour chaque activité, de sélectionner le meilleur service concret de chaque groupe candidat, générant ainsi un véritable service composite exécutable avec les propriétés non fonctionnelles nécessaires.

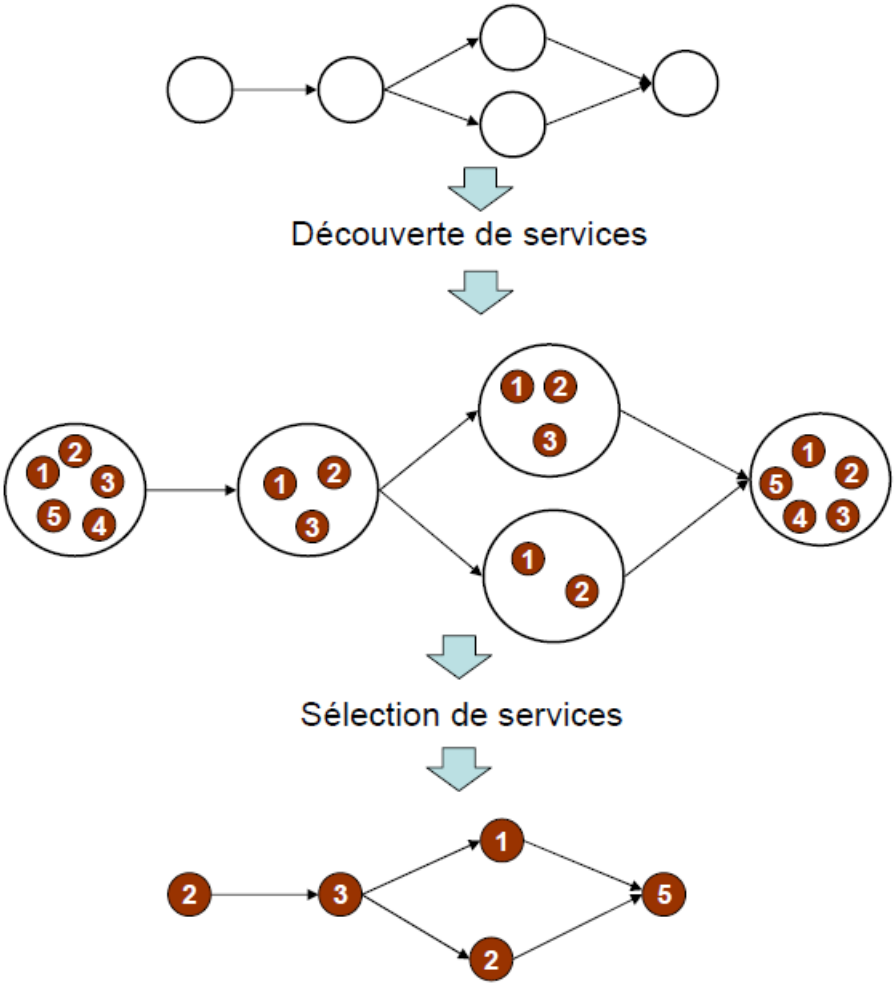

Figure *4.1:* *Sélection des services composites à base de QoS*

Compte tenu de la diversité des services fonctionnellement équivalents mais ayant des niveaux de QoS différents, le problème clé à résoudre dans le contexte de la composition de services, est comment choisir de manière optimale les services qui satisfont le mieux les exigences de l'utilisateur, c'est-à-dire au niveau de QoS requis par l'utilisateur. Dans le contexte de la composition de services à base de QoS, on distingue deux catégories d'approches : (1) les approches basées sur l'optimisation locale et (2) les approches basées sur l'optimisation globale [Zeng et al., 2004] :

- **L'optimisation locale**

  Les approches de sélection locale permettent de déterminer le service candidat optimal pour chaque service abstrait de la composition, et ce, indépendamment des autres services. En effet, les services candidats sont classés en termes de QoS en utilisant la technique SAW (Simple Additive Weighting) qui agrège les attributs de QoS en une seule valeur d'utilité [Zeleny, 1976]. La sélection de services est ensuite transformée en un problème d'optimisation mono-objectif pour déterminer le service candidat ayant la meilleure valeur d'utilité. Bien que ces approches soient très efficaces en raison de leur complexité en temps de calcul qui est polynomial [Zeng et al., 2004], elles ne garantissent en aucun cas d'aboutir à un service composite satisfaisant les contraintes globales de QoS

imposées par l'utilisateur.

- **L'optimisation globale**
  Elle considère les contraintes de QoS au niveau du service composite dans son ensemble. L'optimisation globale évalue les services composites afin de déterminer la meilleure combinaison en termes de QoS qui satisfait les contraintes globales. L'optimisation globale est un problème NP-difficile, ce qui engendre la difficulté d'aboutir à une solution optimale en un temps raisonnable [Garey and Johnson, 1979]. Les approches proposées ont tendance à se focaliser alors sur la recherche des solutions proches de l'optimum.

Le but du présent chapitre consiste tout d'abord à étudier et classifier les principales approches de composition à base de QoS de la littérature en tenant compte de la méthode de résolution utilisée. Ensuite, nous comparons les approches étudiées en termes de prise en compte de certaines exigences telles que l'algorithme de sélection utilisé, le type de la stratégie adoptée, la scalabilité, etc. Enfin, nous terminerons le chapitre par une synthèse.

4.2 Etude des approches de composition à base de QoS

La sélection de services avec QoS pour la création de services composites est un problème NP-difficile [Canfora et al., 2005], [Yu et al., 2007] ; plus précisément, elle est associée à la variante du problème de sac à dos multidimensionnel à choix multiple (MMKP : multi-dimension multi-choice knapsack problem) où les dimensions sont représentées par les différents paramètres de QoS. Elle a été largement abordée dans la littérature. Les approches peuvent être classées selon la méthode de résolution utilisée [Jatoth et al., 2017], [Strunk, 2010], [Khanouche et al., 2016]. On distingue les approches exactes, les approches heuristiques, les approches méta-heuristiques et les approches basées sur la dominance au sens de Pareto.

4.2.1 Les approches exactes

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

la programmation linéaire en nombres entiers (ILP) a été utilisée pour résoudre le problème de la sélection optimale de services [Zeng et al., 2004], [Liu et al., 2012], [Ardagna and Pernici, 2007]. Dans [Zeng et al., 2004], deux approches de sélection de services tenant compte des contraintes de QoS sont proposées pour la composition de services. La première approche effectue une sélection locale de services sans considérer les contraintes globales de QoS imposées au service composite. Pour chaque service abstrait (c'est-à-dire classe de services) de la composition, une technique d'aide à la décision multicritère (MCDM – Multiple Criteria Decision Making) est utilisée afin de trouver le service concret optimal. Bien que la QoS soit optimisée au niveau de chaque classe de services, l'approche ne conduit pas nécessairement à une QoS globale

optimale. Un autre avantage de cette solution est le temps de calcul qui est polynomial. Pour pallier les limites de la sélection locale, la deuxième approche proposée par les auteurs tient compte des contraintes globales de QoS imposées au service composite, et formule la recherche de la composition optimale comme un problème de Programmation Linéaire en Nombres Entiers avec un ensemble de variables, une fonction objectif à maximiser, et un ensemble de contraintes à satisfaire. La composition maximisant la fonction objectif et satisfaisant les contraintes globales de QoS est sélectionnée. Bien que cette approche assure une QoS globale optimale, elle engendre un temps de calcul important ; la complexité en temps de calcul est exponentielle dans certains cas.

[Ardagna and Pernici, 2007] étendent le modèle de programmation linéaire pour inclure les contraintes locales. Dans ce modèle, les contraintes globales sont exprimées sur la composition entière, et par l'utilisateur final, tandis que les contraintes locales peuvent être spécifiées par le concepteur de la composition (au niveau des classes). Dans le même contexte, [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 la classe X n'est pas compatible avec le service j de la classe Y. L'expérimentation montre que l'ajout des conflits, provoque une augmentation de 13% dans le temps d'exécution de l'approche à base de MIP.

Certaines approches de composition à base de QoS utilisent des méthodes d'optimisation basées sur la décomposition des contraintes de QoS [Alrifai et al., 2012], [Sun and Zhao, 2012] où le problème d'optimisation globale est divisé en plusieurs sous-problèmes d'optimisation. Des contraintes locales de QoS sont déduites à partir des contraintes globales. Pour chaque service abstrait de la composition, une sélection locale est ensuite effectuée pour trouver les services candidats satisfaisant les contraintes locales de QoS correspondantes. Dans [Alrifai et al., 2012], les auteurs proposent une approche pour la composition de services avec QoS qui comprend deux phases : 1) une décomposition des contraintes globales de QoS et 2) une sélection locale. Dans la première phase, les contraintes globales de QoS sont décomposées en un ensemble de contraintes locales pour chaque service abstrait. Dans la deuxième phase, une sélection locale est effectuée pour trouver le meilleur service en termes de QoS qui satisfait les contraintes locales. Les contraintes locales sont utilisées comme bornes supérieures pour les valeurs de QoS des services candidats. Les services qui dépassent ces bornes sont ignorés lors de la sélection. Une liste de services candidats respectant les contraintes de QoS est alors créée et triée par valeurs d'utilité décroissantes.

Dans [Sun and Zhao, 2012], une autre approche de composition de services basée sur la décomposition des contraintes de QoS, est proposée. Elle comprend trois étapes principales : une étape de décomposition, une étape de recherche, et une étape d'amélioration. Dans la première étape, l'utilité d'un service composite, ou utilité globale, est déterminée à partir de l'utilité des services candidats, ou utilité locale ; les contraintes relatives aux services candidats, ou contraintes locales, sont obtenues à partir des contraintes imposées par l'utilisateur au service composite, appelées aussi contraintes globales. Dans la deuxieme étape, un service candidat satisfaisant les contraintes locales est sélectionné pour chaque classe de services. S'il existe au moins un service candidat satisfaisant les contraintes locales, la classe de services correspondante est marquée comme satisfaite. Dans le cas contraire, la classe est marquée comme insatisfaite. Tandis que dans la troisième étape, l'utilité globale est améliorée en réajustant les contraintes locales de toutes les classes de services. L'algorithme se

termine si aucune nouvelle amélioration n'est obtenue pour l'utilité globale.

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

[Gabrel et al., 2014] présentent une méthode pour trouver la solution optimale pour la composition de services web transactionnels en utilisant un graphe de dépendance et la programmation linéaire 0-1.

4.2.2 Les approches heuristiques

La complexité en temps de calcul des approches exactes augmente exponentiellement avec la taille du problème. Pour pallier cette limite, d'autres approches formulent le problème de sélection de services avec QoS comme un problème d'optimisation combinatoire [Yu et al., 2007] ou un problème de recherche dans un graphe [Yu et al., 2007] ; [Llinás and Nagi, 2015]. Dans le modèle combinatoire, la sélection de services est modélisée comme un problème de sac à dos multidimensionnel à choix multiples (MMKP – Multi-dimensional Multi-choice Knapsack Problem) [Martello and Toth, 1987] alors que le modèle de graphe définit la sélection de services comme un problème du plus court chemin sous contraintes (MCSP – Multi-Constrained Shortest Path problem) [Khanouche et al., 2016].

Dans [Yu et al., 2007], les auteurs proposent des heuristiques pour le modèle combinatoire et d'autres pour le modèle de graphe. Concernant le modèle combinatoire, pour les structures de composition séquentielles, l'heuristique WS-HEU est proposée afin de résoudre le problème MMKP. Cette heuristique est une amélioration de l'algorithme HEU [Khan et al., 2002]. Elle consiste en la recherche d'une solution réalisable initiale qui est ensuite optimisée à travers un processus de remplacement des services ayant la valeur d'utilité la plus faible, appelés services non-réalisables. Pour les structures de composition complexes, chaque composition abstraite a plusieurs chemins d'exécution dont chacun est caractérisé par une probabilité d'exécution. Une heuristique WFlow est proposée pour résoudre le problème 0-1 IP ; elle a deux variantes qui sont basées sur deux fonctions objectif différentes. La première, WFlow-EU, suit le même principe que l'heuristique WS-HEU mais le processus de remplacement de services est différent. La deuxième variante, WFlow-HP, consiste à optimiser la composition concrète ayant la probabilité d'exécution la plus élevée, et à trouver une solution réalisable pour toutes les autres compositions alternatives. Concernant le modèle de graphe, pour les structures séquentielles de composition, l'approche MCSP, basée sur l'algorithme du plus court chemin à origine unique (SSSP – Single-Source Shortest Paths algorithm) [Cormen et al., 2001], est proposée afin de résoudre le problème de composition de services avec contraintes. Il s'agit ici de trouver le service composite maximisant la valeur d'utilité et satisfaisant les contraintes de QoS imposées par l'utilisateur. En effet, la composition est représentée par un graphe acyclique orienté (DAG – Directed Acyclic Graph), et le service composite obtenu correspond au chemin optimal parcourant le graphe de la source à la destination.

L'approche proposée dans [Llinás and Nagi, 2015] permet de sélectionner plusieurs services candidats pouvant être exécutés séquentiellement afin d'améliorer la QoS du service composite. Pour ce faire, la composition de services est modélisée comme un

problème de recherche, dans un graphe acyclique orienté (DAG – Directed Acyclic Graph), du plus court chemin satisfaisant certaines contraintes. L'algorithme MCSP-FP (MCSP-FP – Multiple Constrained Shortest Path-Feasibility Potential) proposé améliore l'algorithme du plus court chemin à origine unique (SSSP – Single-Source Shortest Paths) par l'introduction du concept de réalisabilité potentielle en termes d'exigences de QoS. L'algorithme MCSP-FP est différent de l'algorithme MCSP proposé dans [Yu et al., 2007], dans la mesure où seuls les chemins potentiellement réalisables sont sauvegardés pour la recherche du chemin optimal. L'amélioration apportée par l'algorithme MCSP-FP est due à l'utilisation de conditions plus restrictives sur la faisabilité d'un chemin.

[Akbar et al., 2006] proposent une autre heuristique, C-UHE, et évaluent sa performance et son optimalité contre plusieurs heuristiques, y compris l'algorithme M-UHE. Les résultats montrent que la C-UHE est meilleure par rapport à M-UHE en termes de temps d'exécution. Cependant, les expériences montrent également que M-UHE est la meilleure en termes de degré d'optimalité, tandis que l'optimalité de C-HEU diminue à mesure que le nombre de candidats par classe augmente. Les expériences montrent aussi que l'algorithme C-UHE fonctionne mieux dans le cas où l'objectif à maximiser (par exemple, la valeur d'utilité de la composition de services) n'est pas proportionnel aux besoins en ressources (i.e., les valeurs de QoS des services Web). Et de ce fait elle ne peut être appliquée pour la sélection de services composés à base de qualité (puisque l'utilité dépend des qualités de service).

Les auteurs dans [Xia et al., 2011] proposent une sélection globale de composition de services en adoptant 4 structures de flux de contrôle : parallélisme, séquence, choix conditionnels, et les boucles, la requête est formalisée en BPEL, elle possède 5 critères de QoS. L'algorithme est baptisé "qssac", il peut donner un résultat proche de l'optimal. Pour réduire le temps d'exécution, ils groupent les services similaires en termes de QoS à l'aide d'un algorithme nommé optics (qui est moins sensible aux bruits et ayant peu de problèmes d'initialisation, optics se base sur la densité), on obtient M classes (et leurs représentants) pour chaque tâche, ensuite l'algorithme énumère toutes les compositions possibles pour 2 ou K tâches du document BPEL, et les trie à l'aide d'une fonction objectif. Dans la même optique, [Luo et al., 2011] ont proposé un algorithme Heuristic HCE pour QoS aware web service composition qui satisfait les contraintes QoS de l'utilisateur. [Comes et al., 2010] ont proposé un modèle de recherche heuristique pour la sélection de compositions de services basée sur la QoS. [Li et al., 2014] ont discuté une approche efficace et fiable pour la sélection de la composition optimale des services à base de paramètres de confiance.

[Lecue and Mehandjiev, 2009], [Klein et al., 2011] utilisent l'algorithme *hill climbing* (Méthode de Descente) pour réduire la complexité de temps de calcul et comparent leur méthode avec celle utilisant LIP.

[Feng1/2 et al., 2013] utilisent les algorithmes d'apprentissage par renforcement pour trouver l'ensemble de solutions Pareto optimales qui satisfait les paramètres QoS multiples et les préférences de l'utilisateur.

Plusieurs autres travaux ont proposé plusieurs algorithmes heuristiques [Luo et al., 2011], [Comes et al., 2010] et [Do Prado et al., 2013] pour réduire la complexité du temps dans la recherche locale ou globale liée à la composition des services web.

4.2.3 Les approches méta-heuristiques

Les métaheuristiques ont connu une popularité importante durant les vingt dernières années. Leur utilisation dans de nombreuses applications montre leur efficacité pour résoudre des problèmes complexes et de grande taille en des temps raisonnables [Talbi, 2009]. Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux familles : les métaheuristiques à base de solution unique (S-métaheuristiques) et celles à base de population de solutions (P-métaheuristiques). Les méthodes de la première famille (telles que les algorithmes de recherche locale [Papadimitriou and Steiglitz, 1998], de recherche tabou [Glover, 1989], de recuit simulé [Černỳ, 1985], etc.) consistent à manipuler et améliorer une seule solution, tant que cela est possible. Par contre, dans les métaheuristiques à base de population (telles que les algorithmes évolutionnaires, algorithmes à essaim de particules, etc.), un ensemble de solutions, nommé population, évolue en parallèle. Ces deux familles ont des caractéristiques complémentaires : les S-métaheuristiques sont axées sur l'exploitation de l'espace de recherche, elles ont la capacité d'intensifier la recherche dans des régions locales prometteuses (afin de trouver une meilleure solution). Les P-métaheuristiques sont orientées exploration, elles permettent une meilleure diversification de l'espace de recherche [Talbi, 2009].

Les méta-heuristiques sont les techniques d'optimisations les plus utilisées pour la résolution des problèmes combinatoires NP-difficiles. Elles ont été très adoptées pour résoudre le problème de composition des services à base de QoS [Canfora et al., 2005], [Chen and Wang, 2007].

Le premier algorithme génétique utilisé pour résoudre le problème de composition à base de QoS est proposé dans [Canfora et al., 2005]. La composition y est réalisée sur la base de caractéristiques à la frontière du fonctionnel et du non-fonctionnel (QoS). Elle se décline, d'un point de vue architectural, en un évaluateur de QoS ("QoS evaluator") et un composant de liaison tardive ("late binding"). Etant donné une composition S spécifiée par un langage de description de workflow (par ex BPEL4WS) de n services abstraits $S = (S1, S2, ..., Sn)$, chaque service abstrait Si peut être lié à un service parmi les m services concrets qui sont fonctionnellement équivalents. Un service concret est alors sélectionné pour l'affectation globale si sa QoS propre, ajoutée à celle des précédents choix, ne viole pas les contraintes globales imposées par le processus. Pour cela il fait appel à des fonctions d'agrégation de QoS qui permettent d'obtenir la valeur globale pour un processus d'une propriété de QoS à partir des multiples QoS locales des services.

Dans [Yu et al., 2013], les auteurs ont proposé un algorithme génétique dans lequel ils ont présenté le problème sous forme d'un arbre. De même, [Liu et al., 2010] ont adopté un algorithme génétique amélioré utilisant l'optimisation par colonies de fourmis pour sélectionner les anticorps de la population initiale afin d'améliorer la convergence. Dans [Xiangbing et al., 2012], les auteurs modélisent aussi le problème des services web en utilisant l'ontologie(WSMO) et appliquent ensuite l'algorithme génétique qui minimise le temps de recherche pour trouver l'optimum.

Dans [Elsayed et al., 2017], les auteurs proposent une nouvelle approche qui combine l'utilisation de l'algorithme génétique et *Q-learning* pour trouver la composition optimale. La performance des AG dépend de la population initiale, de sorte que le *Q-learning* est utilisé pour générer la population initiale afin d'améliorer l'efficacité de l'AG. Ils ont implémenté l'approche proposée sur la plate-forme .NET Framework 4.7 avec le langage de programmation C#. Les résultats expérimentaux montrent l'ef-

ficacité de l'approche de composition proposée par rapport à l'algorithme *Q-learning* et l'algorithme génétique.

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. D'autres chercheurs ont adopté des algorithmes d'optimisation pour trouver une solution optimale ou quasi-optimale. Dans [Wang et al., 2010], les auteurs proposent l'optimisation par colonies de fourmis (ACO) pour choisir une composition proche de l'optimale, ils analysent les performances en variant plusieurs paramètres (le taux d'évaporation de la phéromone, la population initiale...). Dans [Wu and Zhu, 2013], la composition de services tenant compte de la QoS et des propriétés transactionnelles est modélisée comme un problème de construction de chemin dans un graphe acyclique orienté (DAG). Les propriétés transactionnelles sont des propriétés qui garantissent qu'un service est exécuté de façon fiable. Les sommets du graphe représentent les services candidats et les arcs les liens entre les candidats de deux classes de services adjacentes. La métaheuristique d'optimisation par colonies de fourmis (ACO) [Dorigo et al., 2006] est utilisée pour rechercher dans le DAG le meilleur chemin (ou le chemin proche de l'optimum) satisfaisant les contraintes transactionnelles et de QoS globales. Dans l'algorithme ACO, à chaque arc est associé un poids, appelé phéromone. Plusieurs fourmis intelligentes sont créées. Chacune d'entre elles construit une solution du problème (c'est-à-dire un chemin) en traversant le DAG du noeud source au noeud puit. Lors de cette construction, chaque service candidat est sélectionné avec une probabilité liée à l'utilité du service et l'intensité de la phéromone associée à l'arc correspondant. Chaque chemin construit correspond à une instance du service composite, c'est-à-dire, un service composite concret. Lorsque toutes les fourmis achèvent de parcourir l'ensemble des chemins, l'utilité de chaque chemin est évaluée en se basant sur un score de QoS et une valeur indicative sur la satisfaction des contraintes transactionnelles et de QoS. Dans [Xia et al., 2008], les auteurs présentent également 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. [Yang et al., 2010] proposent une hybridation de l'ACO et les algorithmes génétiques. L'algorithme génétique a pour objectif d'ajuster les paramètres d'ACO, qui sera utilisé dans la phase de sélection, les auteurs notent une accélération considérable au niveau du temps d'exécution de l'algorithme proposé par rapport à la version originale. [Hadjila, 2014] présentent l'algorithme de "sélection clonale" pour la résolution du problème de sélection. Ils considèrent une fonction mono-objectif qui manipule 5 paramètres QoS. Les auteurs montrent que la sélection clonale s'avère plus intéressante que l'algorithme génétique.

Dans [Hwang et al., 2015], le problème de sélection de services est traité en représentant les valeurs de QoS des services comme des variables aléatoires discrètes avec des fonctions masse de probabilité. L'approche proposée comprend quatre phases :

1. Tout d'abord, les contraintes globales de QoS sont décomposées en contraintes locales à satisfaire au niveau de chaque service abstrait de la composition. Cette décomposition se base sur les valeurs moyennes de QoS associées aux services abstraits (classes de services). Pour chaque attribut de QoS, la valeur moyenne de QoS associée à une classe de services est calculée à partir des valeurs moyennes de QoS de ses services candidats. En effet, la valeur de QoS d'un service candidat est une variable aléatoire. La valeur moyenne d'un attribut de

QoS est alors le scalaire dont la probabilité cumulative est égale à 0,5.

2. Ensuite, pour chaque service abstrait de la composition, une sélection locale est effectuée pour déterminer le service candidat maximisant la probabilité de satisfaire les contraintes locales de QoS obtenues à l'étape précédente. L'ensemble des services candidats présélectionnés constitue la sélection de services initiale.
3. Pour chaque attribut de QoS, la méthode proposée dans [Hwang et al., 2007] est ensuite appliquée afin de calculer la distribution de probabilité du service composite à partir des services présélectionnées à l'étape précédente (i.e., sélection de services initiale). En effet, la conformité de QoS globale (c'est-à-dire la probabilité de satisfaire les contraintes de QoS globale) du service composite est calculée à partir de la distribution de probabilité de ses attributs de QoS.
4. Enfin, la méthode d'optimisation par "recuit simulé" est appliquée de manière itérative pour substituer certains services présélectionnées afin de maximiser la probabilité de satisfaire les contraintes de QoS globale (c'est-à-dire conformité de QoS). Autrement dit, lorsque la conformité de QoS globale n'est pas satisfaisante, la sélection de services initiale est ajustée par la substitution de certains services candidats, puis la distribution de probabilité du service composite est recalculée. Cette étape est réitérée jusqu'à ce que la conformité de QoS globale devienne maximale.

Dans [Chen and Wang, 2007] les auteurs utilisent l'optimisation en essaims particulaires PSO [Eberhart and Kennedy, 1995], selon les auteurs, PSO est plus rapide et passe mieux à l'échelle 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. [Liao et al., 2011] présentent un l'algorithme NichePSO et considèrent le problème QoSWSC comme un problème multimodal, l'algorithme proposé permet de localiser et de raffiner plusieurs solutions en même temps en considérant de multiple contraintes globales. Liu. et all [Liu et al., 2011] ont présenté un algorithme d'optimisation d'essaim de particules quantique 'hqpso' pour résoudre le problème QoSWSC ; l'algorithme proposé utilise la théorie quantique dans la représentation du problème QoSWSC et l'algorithme PSO. Dans [Liu and Yin, 2009], les auteurs proposent les réseaux de petri coloré (CPN) pour résoudre le problème QoSWSC. [Zhao et al., 2012] adoptent la version discrète de PSO qui est basée sur les systèmes immunitaires artificiels.

Plusieurs travaux [Esfahani et al., 2012], [Jafarpour and Khayyambashi, 2010] et [Mohammed et al., 2014] ont utilisé l'optimisation par Harmonie pour résoudre le problème de sélection. Dans [Esfahani et al., 2012], les auteurs proposent un mécanisme de sélection efficace basé sur un algorithme de recherche d'harmonie social en considérant les attributs de QoS. Afin de démontrer l'efficacité de l'algorithme proposé, il est comparé à un algorithme génétique et les résultats correspondants ont révélé que le présent algorithme consomme moins de temps en recherchant la solution optimale. [Mohammed et al., 2014] proposent une approche mono-objectif basée sur l'algorithme de recherche d'harmonie, permettant d'obtenir une composition presque optimale, en tirant parti de plusieurs opérateurs tels que la recherche locale, la sélection aléatoire ..., etc. Les résultats expérimentaux sont très encourageants et montrent que l'algorithme proposé est plus efficace que les algorithmes génétiques et l'optimisation par colonies d'abeilles.

Certains chercheurs ont adopté la recherche par colonies d'abeilles artificielles (ABC) comme dans [Kousalya et al., 2011], la recherche par coucou (CS) comme

dans [Chifu et al., 2010] ou la recherche par l'auto migration organisée (SOMA) comme dans [Amine, 2017] et [Krithiga, 2012].

4.2.4 Les approches basées sur la dominance au sens de Pareto

Dans cette classe d'approches, le problème de sélection des services composites est résolu en utilisant l'optimalité de Pareto. Un ensemble Pareto est un ensemble de solutions réalisables qui ont au moins un objectif optimisé tout en gardant les autres inchangés [Chankong and Haimes, 2008].

[Khanouche et al., 2016] proposent EQSA (Energy-centered and QoS-aware services selection) un algorithme de sélection de services centré sur l'énergie et sensible à la QoS dans le contexte de la composition de services à large échelle. L'un des principaux aspects qui différencie la solution proposée des autres approches de la littérature, est sa capacité de passage à l'échelle dans le contexte des environnements orientés services. L'algorithme EQSA permet de réaliser des économies d'énergie en réduisant légèrement le niveau de QoS sans affecter la satisfaction de l'utilisateur. Pour ce faire, les services offrant le niveau de QoS requis pour la satisfaction des besoins de l'utilisateur, sont d'abord pré-sélectionnés en utilisant la méthode d'optimisation lexicographique. En introduisant le concept de dominance relative au sens de Pareto, les meilleurs services candidats sont ensuite sélectionnés pour le processus de composition. Les résultats de simulations mettent en évidence les avantages et les performances de l'algorithme EQSA. L'algorithme EQSA supporte le passage à l'échelle en termes de temps de sélection et permet de trouver des solutions proches à environ 98 % de l'optimum ; il engendre une durée de vie de la composition qui est très proche du cas optimal en ne nécessitant qu'une quantité supplémentaire d'énergie relativement acceptable.

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

Dans [Trummer et al., 2014], le problème qui consiste à sélectionner les compositions de services Pareto optimales, est traité en utilisant une approche d'approximation polynomiale A-FPTAS (pour Approximations by Fully Polynomial Time Approximation Scheme). En effet, un sous-ensemble de compositions dont les valeurs de QoS sont représentatives de celles des compositions Pareto Optimales, est sélectionné comme une approximation de l'ensemble Pareto de compositions. La métrique $\xi - error$ est utilisée afin d'estimer la précision de l'approximation [Zitzler et al., 2003].

Pour calculer cette métrique, chaque composition de l'ensemble Pareto est associée à la composition de l'ensemble d'approximation ayant des valeurs de QoS proches. De manière plus précise, la métrique $\xi - error$ est définie comme étant la distance maximale en termes de valeurs de QoS entre une composition Pareto optimale et sa meilleure approximation.

Dans [Jin et al., 2014], un modèle de service physique (pour Physical Service Model) est proposé pour décrire les caractéristiques des services fournis par les dispositifs des environnements IoT (Internet of Things). Ces caractéristiques consistent en quatre attributs de QoS : disponibilité (AT), zone de service (SA), temps de traitement (PT), et réputation (RP). Une fonction d'évaluation est définie par les auteurs afin de calculer l'utilité de chaque attribut de QoS du service physique ; elle tient compte des valeurs de QoS publiées dans la description du service et des exigences de l'utilisateur. En fonction des besoins de l'utilisateur, les services physiques candidats (CPS – Candidate Physical Services) sont évalués en agrégeant leurs utilités individuelles de QoS. L'algorithme de sélection de services comprend trois phases : une phase de pré-tri des services candidats, une phase de filtrage basée sur la dominance absolue au sens de Pareto, et une phase de tri final permettant la sélection des meilleurs services.

[Li et al., 2010] utilisent eux aussi la théorie de chaos et la combinent avec l'algorithme ACO et proposent l'algorithme "Multi objectif chaos ant coloni" (MOCACO) qui considère une optimisation multi-objectif du problème QoSWSC. Dans [Wang and Hou, 2008] les auteurs proposent un algorithme génétique en utilisant une optimisation multi-objectif, ils proposent une fonction objectif pour chaque QoS considéré qui sont au nombre de 3. Contrairement au travail de [Canfora et al., 2005] dans lequel les auteurs codent les chromosomes en utilisant une représentation binaire.

Une approche récente dans [Zhao et al., 2017] a été proposée pour résoudre le problème de la composition des services web prenant en compte les conflits et les dépendances entre les services. Les auteurs utilisent la théorie du chaos et la combine avec un algorithme génétique (CGA). Une stratégie de réparation est utilisée pour traiter les individus de la population initiale qui violent les différentes contraintes. Après la sélection, (les opérations de croisement et mutation), l'algorithme peut produire des individus infaisables qui violent les contraintes imposées entre les services, d'où la nécessité d'une nouvelle fonction de fitness pour traiter le problème de cette population qui a des individus infaisables. Ensuite, une petite perturbation chaotique est imposée à la population, ce qui peut aider l'algorithme génétique à s'échapper de l'optimum local et accélérer la convergence.

Une approche de sélection de services est proposée pour minimiser le risque de violation des contraintes de QoS lorsqu'une recomposition est requise [Al-Helal and Gamble, 2014]. En effet, le concept de remplaçabilité a été introduit comme métrique dans le processus de sélection de services pour réduire le risque de violation des contraintes de QoS durant l'exécution du plan de composition. Ce risque est le résultat des changements de la QoS qui nécessitent une re-sélection de services. La re-sélection a un impact direct sur la composition dès lors que le remplacement d'un service, dont les valeurs de QoS ont conduit à la violation d'au moins une contrainte, engendre un temps de calcul supplémentaire. Toutefois, en raison de la prolifération des services, le calcul de la remplaçabilité devient coûteux en temps de calcul. Afin d'améliorer le processus de composition et de resélection, les services candidats sont filtrés en utilisant la dominance de Pareto basée sur la QoS. Ce filtrage ne supprime aucun service pertinent pour la composition ; la recherche des remplaçants d'un ser-

vice donné, est limitée à l'ensemble de services candidats qui ne sont dominés par aucun autre service de la même classe. Ainsi, la remplaçabilité est calculée en un temps plus réduit.

4.3 Synthèse

Le tableau 4.1 synthétise les approches présentées dans ce chapitre en les classant selon les critères suivants : le type d'approche, la méthode de résolution utilisée, les attributs QoS traités ((Temps de réponse(Tr), Coût (c), Réputation (R), Disponibilité (D), Fiabilité(F), Sécurité (S), Throughput (Th)), la stratégie d'optimisation (locale ou globale), les contraintes QoS, le passage à l'échelle et la méthodologie d'évaluation (simulation, data-set,...).

Travaux étudiés	**Type d'approche**	**Méthode de résolution**	**Stratégie d'optimisation**	**Attributs QoS**	**Contraintes QoS**	**Passage à l'échelle**	**Méthodologie d'évaluation**
[Zeng et al., 2004]	Approches exactes	Integer Linear Programming (LIP)	Locale+Globale	C, Tr ,R ,D	Oui	Non	Prototype expérimental
[Ardagna and Pernici, 2007]		Mixed Integer Programming (MIP)	Globale	Tr, Th, D, C	Oui	Non	Simulation utilisant des données synthétiques
[Alrifai et al., 2012]		Décomposition des contraintes de QoS	Globale	Tr, Th, D, C	Oui	Oui	QWS Dataset [Al-Masri and Mahmoud, 2008] et données synthétiques.
[Sun and Zhao, 2012]		Décomposition des contraintes de QoS	Globale	Tr, D, C	Oui	Oui	Simulation utilisant des données synthétiques
[Rosenberg et al., 2009]		Constraints Optimization Problem (COP)	Globale	Tr, Th, D, C	Oui	Non traité	Non spécifié
[Gabrel et al., 2014]		Graphe de dépendance+ LIP 0-1	Globale		Oui	Non traité	Non spécifié
[Yu et al., 2007]	Approches heuristiques	Algorithme WS-UHE	Globale	C, Tr, D	Oui	Non	Simulation utilisant des données synthétiques
[Llinás and Nagi, 2015]		Algorithme MCSP-FP	Globale	C, Tr	Oui	Oui	Simulation utilisant des données synthétiques
[Akbar et al., 2006]		Algorithme C-UHE	Globale	C, Tr	Oui	Non	Non spécifié

Travaux étudiés	**Type d'approche**	**Méthode de résolution**	**Stratégie d'optimisation**	**Attributs QoS**	**Contraintes QoS**	**Passage à l'échelle**	**Méthodologie d'évaluation**
[Xia et al., 2011]	Approches heuristiques	Algorithme *optics*	Globale	C, Tr, F, Taux de succès	Oui	Non	Prototype éxpérimental
[Luo et al., 2011]		Algorithme HCE	Globale	C, Tr, D	Oui	Non	Non spécifié
[Klein et al., 2011]		Algorithme *Hill climbing*	Globale	C, Tr, D, F	Non	Non	Non spécifié
[Feng1/2 et al., 2013]		Apprentissage par renforcement	Globale	C, Tr, D, F	Oui	Non	Non spécifié
[Canfora et al., 2005]	Approches méta-heuristiques	Algorithme génétique	Globale	C, Tr,D, F	Oui	Non	Simulation utilisant des données synthétiques
[Xiangbing et al., 2012]		Algorithme génétique	Globale	C, Tr, D, F	Oui	Non	Non spécifié
[Elsayed et al., 2017]		Algorithme génétique+ *Q-learning*	Globale		Non	Non	Non spécifié
[Wang et al., 2010]		Algorithme ACO	Globale		Oui	Non	Non spécifié
[Wu and Zhu, 2013]		Algorithme ACO	Globale		Oui	Non	QWS Dataset [Al-Masri and Mahmoud, 2008]
[Xia et al., 2008]		Algorithme ACO	Globale		Oui	Non traité	Simulation utilisant des données synthétiques
[Yang et al., 2010]		Algorithme hybride (génétique + ACO)	Globale		Oui	Non traité	Simulation utilisant des données synthétiques

Travaux étudiés	**Type d'approche**	**Méthode de résolution**	**Stratégie d'optimisation**	**Attributs QoS**	**Contraintes QoS**	**Passage à l'échelle**	**Méthodologie d'évaluation**
[Hadjila, 2014]	Approches méta-heuristiques	Sélection clonale	Globale	C, Tr, F, D, R	Oui	Non	Simulation utilisant des données synthétiques
[Hwang et al., 2015]		Apprentissage automatique+ Algorithme recuit simulé	Globale		Oui	Non	Données synthétiques basée sur le Dataset de [Al-Masri and Mahmoud, 2008]
[Chen and Wang, 2007]		Algorithme PSO	Globale		Oui	Oui	Non spécifié
[Liao et al., 2011]		Algorithme Niche PSO	Globale	C, Tr	Oui	Non traité	Simulation utilisant des données synthétiques
[Liu et al., 2011]		Algorithme hqpso+ théorie quantique	Globale		Oui	Non	Simulation utilisant des données synthétiques
[Zhao et al., 2012]		Algorithme de sélection clonale et PSO	Globale	C, Tr,D, F	Oui	Non traité	Prototype expérimental
[Esfahani et al., 2012]		Algorithme de recherche d'harmonie	Globale	C, Tr	Oui	Non	Simulation utilisant des données synthétiques
[Mohammed et al., 2014]		Algorithme de recherche d'harmonie	Globale	C, Tr,D, F, R	Oui	Non	Simulation utilisant des données synthétiques
[Kousalya et al., 2011]		Algorithme ABC	Globale	D, F, Tr, C	Oui	Non	Simulation utilisant des données synthétiques
[Amine, 2017]		Algorithme SOMA	Globale	C, Tr,D, F, R	Oui	Non	Prototype expérimental

Travaux étudiés	**Type d'approche**	**Méthode de résolution**	**Stratégie d'optimisation**	**Attributs QoS**	**Contraintes QoS**	**Passage à l'échelle**	**Méthodologie d'évaluation**
[Khanouche et al., 2016]	Approches basées sur la dominance au sens de Pareto	Algorithme EQSA	Globale	Qualité énergétique	Non	Oui	Simulation utilisant des données synthétiques
[Chen et al., 2015]		Algorithme DPSA	Globale	Non mentionné	Oui	Non	Dataset [Zhang et al., 2011] et données synthétiques
[Trummer et al., 2014]		Algorithme A-FPTAS	Globale	Non mentionné	Oui	Non	QWS Dataset [Al-Masri and Mahmoud, 2008]
[Jin et al., 2014]		Une fonction d'utilité+ un modèle de service physique	Globale		Oui	Non	Données synthétiques basées sur le Dataset de [Alrifai et al., 2012].
[Li et al., 2010]		Algorithme multi-objectif chaos ant coloni (MOCACO)	Globale	Tr,D, R, C	Oui	Non	Simulation utilisant des données synthétiques
[Zhao et al., 2017]		Algorithme multi-objectif chaos génétique	Globale	Non mentionné	Oui	Non	Simulation utilisant des données synthétiques
[Al-Helal and Gamble, 2014]		Une fonction d'utilité+ un mécanisme de remplaçabilité de services	Globale	Non mentionné	Oui	Non traité	QWS Dataset [Al-Masri and Mahmoud, 2008]

Tableau 4.1: *Classification des approches de composition à base de QoS*

L'état de l'art montre que le problème de composition à base de QoS (QoS-aware web service composition) a été largement abordé dans la littérature. Cependant, la plupart des approches modélisent la composition comme un workflow de n services abstraits où chaque service abstrait peut être lié à un service parmi les m services concrets candidats qui sont fonctionnellement équivalents [Canfora et al., 2005], [Hadjila, 2014], [Mohammed et al., 2014], [Esfahani et al., 2012], [Zhao et al., 2017].

D'après le tableau comparatif 4.1, nous pouvons constater que la majorité des travaux utilisent les méta-heuristiques pour la résolution de problème de composition à base de QoS. En effet, nous pouvons remarquer qu'un nombre élevé de papiers utilisent les algorithmes génétiques, du fait de leurs popularité. Se positionnent juste après, les algorithme PSO qui sont suivis par les algorithmes ACO, puis les autres méta-heuristiques avec plus au moins un pourcentage équivalent. Nous remarquons aussi que d'autres algorithmes sont peu utilisés du fait de leurs récente apparition comme les algorithmes de recherche par colonies d'abeilles artificielles (ABC) ou par harmonie. Cependant l'algorithme par recherche d'harmonie est très performant dans la résolution des problèmes académiques comme l'ingénierie structurelle [Bekdaş and Nigdeli, 2011] ou encore l'ingénierie des ressources de l'eau [Geem, 2009]. Par contre, il demeure encore peu connu et peu exploité par la communauté des services web.

D'autre part, si nous comparons les méta-heuristiques, il est évident que chaque méta-heuristique propose des avantages que l'on cherche à maximiser et des inconvénients que l'on cherche à minimiser. Partant de ce principe, beaucoup de papiers proposent la combinaison des méthodes de résolution des problèmes comme dans [Yang et al., 2010] et [Zhao et al., 2012] afin de tirer profit des points forts de chacune et de proposer des alternatives plus efficaces et plus performantes.

Aussi, nous pouvons constater que dans la plupart des papiers étudiés, les attributs de QoS considérés sont des critères quantitatifs et qui sont le temps de réponse, le coût, la fiabilité et la disponibilité.

Dans [Chen et al., 2015], [Trummer et al., 2014], [Jin et al., 2014] la dominance de Pareto basée sur la QoS est utilisée pour filtrer les services dominés et obtenir l'ensemble des compositions Pareto optimales. Cependant, le nombre de compositions augmente exponentiellement avec le nombre de services conduisant ainsi à un temps de calcul excessif [Yu and Bouguettaya, 2013]. En plus, dans les travaux décrits ci-dessus, les services candidats non-dominés peuvent avoir une faible qualité par rapport à certains attributs de QoS, et il peut ainsi en résulter un service composite offrant une qualité insuffisante par rapport aux exigences de l'utilisateur.

Notre travail diffère de celui des approches étudiées ci-dessus du fait qu'il transforme le problème de composition en un problème de planification (il n'y a pas un plan de composition préalablement établi) dont le but est de trouver une composition proche de l'optimum en un temps réduit en comparaison aux approches proposées dans [Chen et al., 2015] ; [Al-Helal and Gamble, 2014] ; [Trummer et al., 2014] ; [Jin et al., 2014] ; [Yu and Bouguettaya, 2013], où toutes les compositions Pareto optimales sont déterminées. En plus, afin de garantir un service composite répondant aux contraintes de l'utilisateur en termes de QoS, les services candidats qui ne satisfont pas les valeurs seuils de QoS, c'est-à-dire les limites exigées, ne seront pas retenus pour la composition.

4.4 Conclusion

Nous avons vu dans le présent chapitre que, selon la méthode de résolution utilisée, les approches de sélection de services avec QoS peuvent être classées en quatre catégories : les approches exactes, les approches heuristiques, les approches méta-heuristiques, et enfin les approches basées sur la dominance au sens de Pareto.

En général, nous pouvons conclure que les méta-heuristiques sont mieux adaptées à cette problématique par rapport aux autres (selon l'étude présentée dans [Jatoth et al., 2017], plus de 50% des approches "QoS-aware" se concentrent sur des méta-heuristiques), puisque les méthodes exactes ne supportent pas le passage à l'échelle à cause de leur complexité exponentielle. En plus, les approches heuristiques ne sont pas généralisables pour toutes les sortes de fonctions objectifs. D'autre part, l'optimisation à base de Pareto n'est plus efficace lorsque le nombre de critères de QoS est grand. Par conséquent la meilleure classe d'algorithmes qui gère tous ces compromis (nombre donné d'attributs de QoS, complexité polynomiale, quasi-optimalité de solutions, adaptabilité) sera la classe des méta-heuristiques. Le prochain chapitre est consacré aux approches de composition hybrides (qui combinent les techniques de planification et les techniques d'optimisation).

Chapitre 5

Composition des services web fondée sur les techniques de planification et les techniques d'optimisation (QoS-aware automatic web service composition)

Sommaire

5.1 Introduction

Comme nous l'avons mentionné précédemment, la composition des services web est un sujet académique qui couvre un champ important, impliquant et profitant de plusieurs domaines et techniques de l'informatique (par exemple le web sémantique, l'intelligence artificielle, les ontologies...) [Wang and Huang, 2014], [Kona et al., 2008], [Klusch et al., 2005]. Elle spécifie quels services ont besoin d'être invoqués, dans quel ordre, quelles sont les données à échanger, et comment traiter les situations d'exceptions. La composition de services peut être vue comme un mécanisme qui permet l'intégration des services dans une application [Benatallah et al., 2005]. Cette intégration peut se faire de plusieurs manières résultant en un ensemble de classes de composition de services : manuelles, semi-automatiques ou automatiques tout en implantant des compositions statiques ou dynamiques. Techniquement, ces différentes catégories peuvent être réalisées par plusieurs approches de composition de services.

Dans le chapitre 3, nous avons exposé les travaux qui traitent la composition automatique comme un problème de planification où le service composite recherché correspond à un système états-transitions, qui a des états multiples et représente des transitions à partir d'un état initial acceptant les *inputs* d'un utilisateur jusqu'à un état final fournissant les *outputs* et les effets requis. Dans le système de transition, les actions applicables correspondent à des services disponibles. Si les pré-conditions et les *inputs* nécessaires d'une action sont satisfaits dans un certain état, la transition de cet état à un autre état peut être effectuée par l'application de l'action [Fki, 2015]. Cependant la plupart des approches étudiées cherchent à trouver des plans de

composition valides sans considération de la qualité de service globale de la solution recherchée. Par conséquent, ces méthodes seraient incapables de trouver des compositions optimales en satisfaisant les préférences de l'utilisateur en terme de qualité de service. De plus, la majorité de ces méthodes ne passent pas à l'échelle à cause de la complexité des algorithmes proposés.

Dans le chapitre 4, nous avons présenté un état de l'art des travaux qui traitent la composition des services web à base de QoS (QoS-aware service composition). Ces travaux peuvent aboutir à des solutions optimales (ou quasi-optimales) et de bonne qualité en un temps raisonnable. Cependant, la plupart des approches proposées modélisent la composition comme un workflow abstrait c'est-à-dire se basent sur l'existence préalable des plans de compositions des services abstraits où chaque service abstrait peut être lié à un service concret parmi l'ensemble des services concrets qui sont fonctionnellement équivalents. Les caractéristiques de QoS assurées par le service composite ou exigées par l'utilisateur peuvent être une seule valeur représentative, qui est calculée par l'intermédiaire de certaines fonctions d'agrégation de QoS [Zeng et al., 2004].

Dans le présent chapitre, nous nous intéressons au problème "QoS-aware automatic service composition". Cette problématique a attiré beaucoup d'attention ces dernières années [Bartalos and Bieliková, 2009], [Yan et al., 2009], [Jiang et al., 2010]. Elle combine les techniques de planification et les techniques d'optimisation afin de trouver la composition recherchée d'une manière automatique. Ainsi, les solutions trouvées prennent en considération non seulement l'aspect fonctionnel des services web mais aussi l'aspect non fonctionnel regroupant un ensemble de paramètres de qualité de service (temps de réponse, fiabilité, etc). Dans un premier temps, nous présentons l'essentiel des travaux qui traitent cette problématique. Ensuite, nous comparons les différents travaux en termes de la prise en compte de certaines exigences des environnements orientés service. Enfin, nous terminons ce chapitre par une conclusion générale portant sur la partie de l'état de l'art.

5.2 Etat de l'art

Cette section est consacrée à un état de l'art sur le problème "QoS-aware automatic web service composition" qui est un problème NP hard [Pisinger, 1995], [Yu et al., 2007]. Les travaux dans ce domaine modélisent le problème de composition de services web en adoptant des techniques de planification et des techniques d'optimisation. Les approches de composition fondées sur ces techniques ne sont pas très nombreuses dans la littérature.

Dans [Yachir et al., 2012], une approche de composition à deux phases est proposée : une phase offline et une phase online. Durant la phase offline, un graphe global AGoSC (pour Abstract Graph of Service Composition) est construit automatiquement en utilisant quatre règles définies par les auteurs, à savoir : la règle de marquage, la règle d'égalité, la règle d'inclusion simple, et la règle d'inclusion complexe. Cette construction consiste à relier les entrées et sorties de tous les services abstraits disponibles. Les services et les paramètres qui n'apportent aucune valeur ajoutée sont éliminés de sorte que le graphe résultant soit global et optimal en termes de fonctionnalités offertes. Le graphe résultant est global car il intègre toutes les fonctionnalités disponibles, et il est optimal car aucune fonctionnalité n'est dupliquée grâce à l'élimination des paramètres et des services redondants. Dans la

phase online, un sous-graphe SubGoSC (pour Sub-Graph of Service Composition) est extrait automatiquement du graphe global AGoSC dans le but de répondre à des besoins spécifiques lorsqu'un événement est détecté, tel que la hausse inhabituelle de température, la présence d'un objet suspect, etc. Cette extraction se repose sur une technique orientée objectif (*goal-driven* technique) utilisant une règle dite de démarquage. Ainsi, un sous-graphe contient exactement les fonctionnalités nécessaires pour réaliser la tâche qui lui a été assignée. Le modèle de sélection de services proposé comprend trois phases :

1. Estimation de la QoS ;
2. Sélection de services avec apprentissage ;
3. Invocation de services.

Tout d'abord, la phase d'estimation de la QoS consiste à évaluer la qualité globale d'un service concret à partir des valeurs des paramètres de qualité statique (SQP – Static Quality Parameters) et dynamique (DQP – Dynamic Quality Parameters). Les paramètres de qualité statique, comme par exemple le coût, restent inchangés pendant une longue durée de temps alors que les paramètres de qualité dynamique, comme par exemple la fiabilité, changent en fonction du contexte d'utilisation. Dans [Yan et al., 2012], un graphe de composition de services est construit en se basant sur une technique de planification. L'algorithme de Dijkstra est ensuite utilisé pour rechercher dans le graphe obtenu, le meilleur chemin en termes de QoS satisfaisant les exigences fonctionnelles de l'utilisateur.

[Lécué, 2008a] 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 chaînage 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 pré-conditions et les post-conditions des services). L'outil proposé est testé sur 03 collections : Télécommunication scénario, E-Tourisme scénario, 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 coûteuse du deuxième algorithme, est celle du chaînage arrière, de même, il est utile de noter que, la phase d'optimisation est peu coûteuse en termes de temps d'exécution (moins de 1 sec).

Une autre approche prenant en compte l'aspect contexte est proposée dans [Kouicem et al., 2014], elle est basée sur une architecture à trois agents : un agent de planification, un agent de sélection, et un agent d'orchestration de services. Le rôle de l'agent de planification consiste à générer un plan de composition de services abstrait en utilisant l'algorithme de planification par chaînage arrière. Cet algorithme produit une succession de services abstraits pouvant fournir les sorties requises par la requête de l'utilisateur à partir des entrées de services. Deux heuristiques ont été

introduites afin d'optimiser la taille du plan (c'est-à-dire, le nombre de services abstraits) ainsi que le temps de génération du plan. L'agent de planification génère, en collaboration avec l'agent de sélection, un workflow de services concrets. L'agent de sélection détermine, pour chaque service abstrait du plan de composition, le meilleur service candidat en se basant sur les paramètres de QoS et les attributs du contexte. Ce choix s'effectue en deux étapes. La première étape repose sur un algorithme de filtrage pour pré-sélectionner tous les services candidats sensibles au contexte courant de l'utilisateur. La deuxième étape consiste à sélectionner, parmi les services pré-sélectionnés dans l'étape précédente, le meilleur candidat en se basant sur une fonction d'utilité exprimée comme une somme pondérée des paramètres de QoS et des attributs contextuels. Trois paramètres sont considérés dans le modèle de QoS : le temps de réponse, la fiabilité et la disponibilité. Dans le modèle du contexte, deux attributs sont pris en compte : la localisation de l'utilisateur et le niveau de la batterie du dispositif hébergeant le service. Les valeurs des attributs de QoS sont mises à jour après chaque invocation de services en utilisant un algorithme d'apprentissage par renforcement basé sur la réponse du service invoqué.

L'agent d'orchestration se charge de la gestion des défaillances pouvant se produire lors de l'exécution du workflow. En effet, les services concrets sont invoqués à la volée et leur comportement est imprévisible. Une exception ou une défaillance peut alors rendre l'exécution d'un service impossible, et une re-sélection du meilleur service en termes de valeur d'utilité, est alors nécessaire pour poursuivre l'exécution de la composition.

les auteurs dans [Li et al., 2011] utilisent une structure de graphe orienté représentant les dépendances entre les services et leurs entrées et sorties. Ce graphe est construit au préalable avant la phase de composition dans le but d'optimiser les temps de recherche lors de l'exécution des requêtes clients. Les noeuds représentent des services ou des paramètres (entrées ou sorties). Chaque arc du graphe relie un noeud service « Si » à un noeud paramètre « Pi » et est pondéré en utilisant un poids global de qualité de service calculé auparavant et associé au service en question. Par ailleurs, chaque noeud paramètre peut être lié à une variété de noeuds services. Dans ce travail, la sélection des services, qui peuvent constituer des plans de composition répondant à la requête, est effectuée au moyen des techniques du chaînage arrière "backward chaining" et de la recherche en profondeur d'abord "depth first search", tout en tenant compte de la sémantique des services. Le choix du meilleur plan de composition obtenu est déterminé sur la base de la meilleure qualité de service offerte.

A la différence des recherches de [Li et al., 2011] où la sélection des services pouvant constituer des plans de composition est effectuée au moyen des techniques de chaînage arrière, [Ying, 2010] utilise la technique du chaînage avant "Forward chaining" pour construire un graphe orienté de composition de services maximal dit« Gmax ». La construction de ce graphe s'initialise à partir des entrées de la requête utilisateur. Le noeud de départ du « Gmax » est le noeud dont les sorties sont équivalentes aux paramètres d'entrée de la requête. Les deux étapes de découverte de services et de matching des entrées et sorties de ces services sont, par la suite, altérées jusqu'à l'obtention des paramètres de sorties de la requête utilisateur parmi ceux des services découverts. Le graphe « Gmax » inclut l'ensemble des sous-graphes (plans de composition) qui répondent à la requête client. A partir de ce graphe maximum, un ou plusieurs graphes minimums « Gmin » peuvent être extraits et proposés comme des plans de composition différents satisfaisant la requête utilisateur. L'extraction d'un « Gmin » à partir d'un « Gmax » est fondée sur la détection des noeuds partagés

(noeuds attachés à plusieurs autres noeuds qui les précèdent) et l'élimination des noeuds redondants.

Les auteurs dans [Qiu et al., 2007] se sont basés sur les diagrammes d'états (statecharts) et les graphes acycliques directs DAG (Direct Acyclic Graph) afin de représenter les plans de composition de services qui sont construits par l'approche de planification HTN. Dans leur travail, les auteurs proposent une extension d'OWL-S, nommée OWL-SC (OWL-S Context) permettant d'intégrer le contexte à la description des services Web. Les auteurs distinguent trois catégories de contexte : le contexte de l'utilisateur (U-Context), le contexte du service Web (W-Context) et le contexte de l'environnement (E-Context). Chaque catégorie de contexte est représentée par une ontologie et est intégrée aux ontologies existantes d'OWL-S.

Nahrstedt et al. [Nahrstedt et al., 2001] utilisent l'idée des algorithmes de planification globale pour la composition dynamique de services. Les auteurs proposent un modèle qui calcule un plan initial. Par la suite, ce plan est révisé si nécessaire lors de l'exécution en fonction du contexte. Saif et al. [Saif et al., 2003] intègrent les informations contextuelles dans un mécanisme de planification dans un environnement intelligent. Les auteurs indiquent que la manière pour satisfaire un objectif particulier de la planification peut dépendre du contexte.

[Barreiro Claro, 2006] propose un canevas appelé SPOC (Semantic based Planning Optimized Compositions) pour la composition automatique de services web dédiés à la réalisation de devis, il a été élaboré en quatre étapes :

- la découverte de services : cette phase permet d'identifier les services à partir d'une ontologie de services UDDI. Dans cette phase, SPOC utilise le web sémantique comme langage de description des services Web ou l'auteur propose une ontologie du domaine qui organise les services web selon une similarité sémantique. Une fois une tâche définie, le processus de découverte cherche dans cette ontologie les services Web qui la réalisent.
- la planification : cette phase concerne la composition automatique, elle permet d'ordonner les tâches à réaliser et de choisir les services web qui peuvent réaliser chaque tâche. Dans SPOC, Claro utilise JESS (Java Expert System Shell), qui est un moteur de règles de planification, pour déterminer les services Web qui participeront à la composition ainsi que leur ordonnancement.
- l'exécution : cette phase envoie une requête à chaque service pour obtenir des informations comme le coût, la durée,etc.
- l'optimisation : cette phase a pour but de proposer à l'utilisateur un certain nombre de compositions optimisées selon des critères de qualité prédéfinis (e.g., le coût, le prix, le chiffre d'affaire, la réputation) en utilisant l'algorithme génétique NSGA-II (Non-Dominated Sorting Genetic Algorithm).

Dans [Pop et al., 2009] les auteurs appliquent un algorithme de sélection clonale pour sélectionner la composition optimale. Leurs approche modélise la composition du service web comme un processus multi-couches dans lequel une structure de graphe de planification (Enhanced Planning Graph) est créée, et augmentée avec une matrice de liens sémantiques. Ils ont amélioré le graphe de planification classique avec les nouveaux concepts de cluster de services et les liens de similarité sémantique. Les liens de similarité sémantique sont définis entre les services sur différentes couches de graphe et sont stockés dans une matrice de liens sémantiques. Pour calculer le degré de correspondance sémantique entre services, ils ont adopté les mesures de recherche d'information : Rappel, Précision et $F\_Measure$. L'algorithme de sélection clonale

utilise le graphe de planification amélioré (EPG) et la matrice de liens sémantiques pour sélectionner la solution optimale en utilisant les attributs QoS et la qualité sémantique comme critères de sélection.

Dans [Salomie et al., 2011] les auteurs proposent une amélioration du travail présenté en [Pop et al., 2009], 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).

Dans [Pop et al., 2011b], [Chifu et al., 2015], les auteurs proposent toujours, une composition à deux phases, la première phase permet la génération de graphe de planification (EGP) augmenté par les liens sémantiques. La deuxième phase (celle de l'optimisation) emploie l'algorithme Firefly pour le premier travail et l'algorithme ACO (Ant Colony Optimization) pour le deuxième travail. Les résultats obtenus sont très satisfaisants en termes d'optimalité. Dans [Pop et al., 2011a], les auteurs reprennent le travail de [Chifu et al., 2011] et proposent un algorithme hybride qui combine les principes de la recherche tabou et l'apprentissage par renforcement avec l'algorithme cuckoo search. [Salomie et al., 2014] proposent le même principe d'hybridation que le travail de [Chifu et al., 2011] mais avec l'algorithme Firefly.

Dans [Jiang et al., 2014], [Deng et al., 2014], une méthode de composition est proposée pour trouver les k meilleurs (appelé Top-k) web services composites en termes de QoS. Ce problème est formulé comme un problème de recherche des k meilleurs chemins dans un graphe acyclique dirigé (DAG – Directed Acyclic Graph). Le DAG est obtenu à partir du graphe de dépendances des services qui représente les liens entre les services faisant partie de la composition. Pour trouver les k meilleurs services composites, la méthode proposée dans [Jiang et al., 2014] se base sur un algorithme incrémental nommé "Key-Path-Loose" (KPL) (avec 100% de précision) alors que la solution proposée dans [Deng et al., 2014] combine une méthode de recherche avec retour arrière et une méthode de recherche en profondeur ; ces deux méthodes sont exécutées en parallèle pour améliorer le temps de composition.

[Wagner et al., 2012] proposent une méthode qui combine une technique de planification et une approche de sélection des services en tirant parti des caractéristiques communes des services web. Ils utilisent une structure de données qui organise des services fonctionnellement similaires en clusters et calcule la QoS de chaque cluster. Puis l'outil de planification compose des workflows composés de ces clusters, en prenant en compte la qualité de service globale des clusters. Ainsi, l'utilité en général et la fiabilité des workflows composés sont significativement améliorées.

L'auteur dans [Bartalos, 2011] propose un processus de composition composé de trois étapes. Les deux premières étapes cherchent des services consommant les entrées fournies (dans la requête de la composition) et les services fournissant les résultats demandés, respectivement. Cependant, dans la plupart des cas, seuls ces services (initiaux et finaux) ne sont pas suffisants pour fournir l'ensemble des fonctionnalités exigées par l'utilisateur, nécessitant ainsi de découvrir et de composer des services intermédiaires. Dans la troisième phase, une stratégie de chaînage avant basée sur l'algorithme de graphe de planification est appliquée afin de sélectionner et composer les services utilisables. La solution est extraite en appliquant une stratégie de recherche avec retour arrière pour chaque entrée des services apparaissant dans la couche d'actions finale. Le plan de composition résultant comprend des constructeurs séquentiels et parallèles.

Dans [Weise et al., 2007] les auteurs proposent trois algorithmes pour la composition automatique des services web fondés sur leurs descriptions sémantiques : l'algorithme IDDFS (Iterative Deepening Depth First Search), un algorithme heuristique et un algorithme génétique. Les paramètres d'E/S des services web sont annotés par des concepts sémantiques. L'utilisateur exprime ses besoins via une requête qui est elle aussi caractérisée par des concepts d'entrée et des concepts de sortie. Afin de satisfaire une requête donnée, les concepts d'entrée du service offert (composite) doivent être plus généraux que les concepts d'entrée de la requête. En revanche les concepts de sortie du service offert doivent être plus spécifiques que les concepts de sortie de la requête. L'algorithme IDDFS est spécialement rapide à trouver des solutions si le répertoire des services web est de taille petite ou moyenne. L'algorithme heuristique est plus efficace et convient pour un répertoire de services web plus grand ; et l'algorithme génétique approprié aux répertoires de services web de taille particulièrement grande.

Dans [Talantikite et al., 2009] les auteurs proposent une approche automatique pour la découverte et la composition des services web sémantique, dans laquelle un réseau d'interaction est créé où les nœuds représentent les services web sémantiques et les liens d'interaction, appelés « liens de dépendance sémantique », sont calculés en utilisant les opérateurs de mise en correspondance sémantiques introduits par [Paolucci et al., 2002]. Sachant qu'ils cherchent avant tout à satisfaire les buts de la requête, ils proposent l'utilisation d'un algorithme de chaînage arrière à une passe pour la recherche automatique de compositions. Une solution est sélectionnée parmi l'ensemble des plans de compositions retournés à partir d'un critère de qualité. Pour la partie expérimentale, les performances de l'algorithme sont testées sur un réseau construit à partir d'une collection de 60 descriptions artificielles. Dans ce travail, les auteurs se concentrent sur des descriptions sémantiques. Les données considérées sont artificielles et la collection de descriptions est de petite taille.

[Rodriguez-Mier et al., 2017] présentent une approche hybride pour la composition automatique de services web générant des compositions optimales en terme de QoS, et un nombre minimal des services qui participent à la composition. L'approche comporte quatre étapes principales :

1. La génération du graphe de composition pour une requête donnée ;
2. Le calcul de la composition optimale qui minimise une fonction mono-objectif de QoS ;
3. L'optimisation en plusieurs étapes pour réduire l'espace de recherche en identifiant les services équivalents et dominés ;
4. Une recherche hybride (locale et globale) pour extraire la QoS optimale avec le nombre minimum de services.

Une validation de l'approche proposée avec le data-set Web Service Challenge 2009-2010 montre que : 1) la combinaison de l'optimisation globale et locale est une technique puissante pour extraire des compositions optimales dans divers scénarios ; et 2) la stratégie hybride fonctionne mieux que les travaux de l'état de l'art, obtenant des solutions avec un nombre minimal de services et une valeur de QoS optimale.

Dans [Khanouche et al., 2016], les auteurs ont proposé un algorithme de composition CQCA (Clustering-based and QoS-aware services Composition Algorithm) basé sur le partitionnement des services et sensible à la QoS. L'approche proposée comprend trois phases. La première phase consiste à partitionner (regrouper) les services candidats en clusters (groupes), chacun représentant un niveau de QoS. L'objectif

de ce partitionnement est de calculer la valeur d'utilité de chaque service candidat qui permettra de filtrer les services candidats ayant une faible QoS. En utilisant la méthode d'optimisation lexicographique, la deuxième phase consiste à sélectionner les services candidats offrant le niveau de QoS requis pour satisfaire les contraintes globales de QoS. La troisième phase consiste, quant à elle, à trouver la composition de services quasi-optimale, autrement dit, le service composite satisfaisant les contraintes de QoS et offrant la meilleure valeur d'utilité en termes de QoS. Les performances obtenues en simulation montrent que l'approche de composition CQCA engendre des compositions proches de l'optimum et réduit considérablement l'espace de recherche, et par conséquent, le temps de composition.

[Fki, 2015] présente une approche de composition dans laquelle la génération du schéma de composition est effectuée en partie au moment de l'exécution en ayant recours aux services abstraits fournis au moment de la conception. L'utilisation des services abstraits permet une certaine flexibilité et adaptabilité sans avoir besoin de construire une composition de services à partir de zéro au moment de l'exécution. Le processus de composition qui a été proposé se compose principalement de quatre étapes. La première étape prend une structuration des besoins de l'utilisateur matérialisée par un graphe d'intentions et l'enrichit pour expliciter les relations implicites. Le résultat de cette étape permet de générer un schéma de composition initial en construisant le flux de contrôle déduit du graphe des intentions enrichi, puis en sélectionnant les services abstraits adéquats. Le choix de ces services est basé sur le matching sémantique et le degré d'affinité sémantique entre les services abstraits. La troisième étape consiste à générer le schéma de composition final à l'aide d'un mécanisme de raffinement des services abstraits en utilisant des techniques de matching sémantique et en tenant compte du contexte de l'utilisateur. Enfin, le plan d'exécution est généré en tenant compte des contraintes non-fonctionnelles fournies dans la spécification des intentions.

Dans [Yan et al., 2012], les auteurs proposent d'appliquer l'algorithme de Dijkstra sur le graphe de planification étendu qui est généré par l'approche GraphPlan avec un changement minimum. En particulier, ils utilisent deux types de graphes de planification, à savoir le graphe de planification partiellement labellisé (PLPG) et le Graphe pondéré en couches (LWG). Dans ce travail, les auteurs proposent une extension de l'algorithme de Dijkstra pour permettre d'avoir des nœuds de traitement ayant des sources multiples. Le GraphPlan proposé comporte deux phases, la phase de construction du graphe de planification et la phase d'extraction de la solution. Cette dernière phase consiste essentiellement, dans une recherche en arrière à partir de la couche d'objectif, qui utilise l'information obtenue dans la première phase (c'est-à-dire la phase de construction). Dans un travail plus récent [Chen and Yan, 2014] des auteurs de [Yan et al., 2012], l'approche GraphPlan, combinée à l'algorithme de Dijkstra, est également utilisée. Les auteurs proposent un moyen de modifier l'accessibilité du graphe pour faciliter l'utilisation de l'algorithme de Dijkstra et réduire la possibilité d'une explosion combinatoire. L'avantage significatif de cette approche est de trouver la solution globale optimale pour tous les types de critères de qualité de service, mais pour un seul critère. Cependant, pour atteindre ses résultats, cette approche réalise un ensemble d'étapes, à savoir la génération d'un marquage de graphe de planification (PLPG), la conversion en un graphe pondéré en couches et la génération de plan. Ces étapes prennent un temps considérable, qui pourrait même être en un temps exponentiel.

Dans [Baccar et al., 2013], les auteurs proposent une approche de composition

automatique en deux phases. La première phase consiste à produire un plan abstrait de composition basé sur le langage DDL (Dynamic Description Logics). Ce plan abstrait est ensuite concrétisé en un plan exécutable en sélectionnant les instances de service web appropriées en utilisant les propriétés non fonctionnelles des services.

Dans [Gabrel et al., 2018], les auteurs proposent une approche basée sur une formulation d'ordonnancement avec contraintes AND/OR, utilisant une structure de graphe orienté afin de résoudre le problème "QoS-aware Automatic Syntactic Service Composition". Deux critères de qualité de service ont été utilisés : le temps d'exécution et le débit. Ils ont mené une série d'expérimentation pour évaluer la performance de l'approche de composition proposée en utilisant la collection de données *Web Service Challenge 2009*.

Dans [Bali, 2017], l'auteur propose deux contributions. La première contribution est une approche de gestion de contexte basée sur une représentation sémantique du contexte en utilisant des langages standard tels que RDF, RDF(S). Elle est caractérisée par sa grande flexibilité, simplicité et permet au développeur du service de mettre son point de vue concernant le contexte qui doit être utilisé tout en évitant l'exigence d'utiliser un modèle de contexte unifié. De plus, il a introduit la notion de journal de changements, qui garde seulement les changements de contexte significatifs. Pour réaliser les parties actives de cette architecture, il s'est basé sur la technique des agents grâce à leur efficacité dans les systèmes distribués. La deuxième contribution consiste à proposer une approche de composition permettant de satisfaire à la fois les exigences fonctionnelles et les exigences non fonctionnelles (qualité de service QoS), tout en minimisant le nombre de services de la composition. Cette approche est basée sur une structure de graphe de planification amélioré. Les résultats expérimentaux montrent une amélioration significative du temps d'exécution de la composition et même du nombre de services lorsqu'on travaille sur une grande échelle des services (jusqu'à 20000 de services).

5.3 Synthèse

Le tableau 5.1, représente un récapitulatif et une comparaison des travaux étudiés dans ce chapitre. Nous avons choisi comme critères : la technique de planification et/ou d'optimisation, le modèle sémantique, les contraintes globales QoS, le contexte (Voir Section 02 du Chapitre 03) et la méthodologie d'évaluation (simulation, data-set,...)..

Travaux étudiés	**Technique de planification et/ou d'optimisation**	**Modèle sémantique**	**Contraintes globales QoS**	**Contexte**	**Méthodologie d'évaluation**
[Yachir et al., 2012]	Graphe global de composition AGoSC (Abstract Graph of Service Composition)	OWL-S	Non	Oui	Prototype expérimental
[Lécué, 2008a]	Algorithme récursif de régression "Ra4c"	DAML-S	Non	Non	3 scénarios : Télécommunication, E-Tourisme et E-santé
[Kouicem et al., 2014]	Algorithme de planification par chaînage arrière	OWL-S	Non	Oui	Non spécifié
[Li et al., 2011]	Algorithme de planification par chaînage arrière et une recherche en profondeur (depth search) dans un graphe orienté	OWL-S	Non	Non	Simulation utilisant des données synthétiques
[Ying, 2010]	Algorithme de planification par chaînage avant	OWL-S	Non	Non	Simulation utilisant des données synthétiques
[Qiu et al., 2007]	Diagramme d'états+ Graphe acyclique direct (DAG) de composition	OWL-S	Non	Oui	Non spécifié
[Barreiro Claro, 2006]	un moteur de règles de planification JSS (Java Expert System Shell)+ Algorithme NSGA-II (Non-Dominated Sorting Genetic Algorithm)	Non spécifié	Non	Non	Prototype expérimental

Travaux étudiés	**Technique de planification et/ou d'optimisation**	**Modèle sémantique**	**Contraintes globales QoS**	**Contexte**	**Méthodologie d'évaluation**
[Pop et al., 2009]	Graphe de planification amélioré (EPG)+ Algorithme de sélection clonale	SAWSDL	Non	Non	Évaluation basée sur des scénarios
[Salomie et al., 2011]	Graphe de planification amélioré (EPG)+ Algorithme de sélection clonale+ Apprentissage par renforcement	SAWSDL	Non	Non	Évaluation basée sur des scénarios
[Pop et al., 2011b]	Graphe de planification amélioré (EPG)+ Algorithme Firefly	SAWSDL	Non	Non	Évaluation basée sur des scénarios
[Chifu et al., 2015]	Graphe de planification amélioré (EPG)+ Algorithme ACO (Ant Colony Optimization)	OWL-S	Non	Non	Évaluation basée sur des scénarios
[Pop et al., 2011a]	Graphe de planification amélioré (EPG)+ Algorithme hybride (Cuckoo+ recherche tabou)+ Apprentissage par renforcement	OWL-S	Non	Non	Évaluation basée sur des scénarios
[Salomie et al., 2014]	Graphe de planification amélioré (EPG)+ Algorithme hybride (Firefly+ Cuckoo)	OWL-S	Non	Non	Évaluation basée sur des scénarios
[Jiang et al., 2014]	Algorithme KPL (Key-Path-Loose)	OWL-S	Non	Non	Web Service Challenge 2009 dataset [1]

1. http ://ws-challenge.georgetown.edu/wsc09/

Travaux étudiés	**Technique de planification et/ou d'optimisation**	**Modèle sémantique**	**Contraintes globales QoS**	**Contexte**	**Méthodologie d'évaluation**
[Deng et al., 2014]	Techniques de recherche avec retour arrière et de recherche en profondeur	OWL-S	Non	Non	Web Service Challenge 2009 dataset+ CWSC2011 dataset [2]
[Rodriguez-Mier et al., 2017]	Graphe de planification	OWL-S	Non	Non	Web Service Challenge 2009 dataset
[Khanouche et al., 2016]	Algorithme CQCA (Clustering-based and QoS-aware services Composition Algorithm)	OWL-S	Oui	Non	Simulation utilisant des données synthétiques
[Fki, 2015]	Graphe de planification	OWL-S	Oui	Oui	Une étude de cas : la gestion intelligente du bâtiment des équipements consommateurs et producteurs
[Yan et al., 2012]	Algorithme de Dijkstra sur le graphe de planification	OWL-S	Non	Non	Web Service Challenge 2009 dataset
[Gabrel et al., 2018]	Algorithme d'ordonnancement avec contraintes AND/OR sur un graphe orienté	/	Non	Non	Web Service Challenge 2009 dataset
[Bali, 2017]	Graphe de planification amélioré	OWL-S	Non	Non	Web Service Challenge 2009 dataset
[Baccar et al., 2013]	Approche basée sur le Langage DDL	OWL-S	Non	Non	Non spécifié

2. http ://debs.ict.ac.cn/cwsc2011/

Travaux étudiés	Technique de planification et/ou d'optimisation	Modèle sémantique	Contraintes globales QoS	Contexte	Méthodologie d'évaluation
Notre approche	**Graphe de planification amélioré (EPG)+ Algorithme Harmony Search**	**SAWSDL**	**Oui**	**Non**	**Web Service Challenge 2009 dataset**

Tableau 5.1: *Classification des approches de composition hybrides*

L'état de l'art montre que le problème "QoS-aware automatic web service composition" a été largement abordé dans la littérature. Comme il est indiqué dans le tableau comparatif, on note que la majorité des approches utilisent les techniques de chaînage en avant ou en arrière comme dans [Kouicem et al., 2014], [Li et al., 2011], [Ying, 2010] et [Deng et al., 2014]. D'autres travaux modélisent la composition comme un graphe acyclique direct (DAG) [Yachir et al., 2012], [Bali, 2017], [Salomie et al., 2014] [Rodriguez-Mier et al., 2017]. Cependant, peu de travaux considèrent les contraintes globales QoS de l'utilisateur. Seuls les travaux de [Khanouche et al., 2016] et [Fki, 2015] prennent en compte cette métrique dans le processus de composition.

Pour trouver une solution optimisant un certain nombre d'attributs de qualité de service, nous trouvons dans la littérature des approches qui adoptent les méta-heuristiques comme techniques d'optimisation. Par exemple dans [Pop et al., 2009], les auteurs utilisent un algorithme de sélection clonale, l'algorithme Firefly a été utilisé dans [Pop et al., 2011b] ou encore l'algorithme ACO dans [Chifu et al., 2015]. A notre connaissance l'algorithme Harmony Search n'est pas très utilisé pour trouver la composition quasi-optimale. De plus, la plupart de ces approches optimisent généralement deux ou trois attributs de QoS.

Nous notons aussi que la plupart des travaux se focalisent sur des structures de contrôle séquentielles et ne traitent pas d'autres structures de contrôle complexes impliquant les boucles, et les structures conditionnelles. Nous pouvons remarquer aussi que peu de travaux considèrent le contexte pendant le processus de composition. Avec l'utilisation grandissante des technologies émergentes telles que l'intelligence ambiante et l'Internet des Objets (Internet of Things), le challenge est de fournir aux utilisateurs des applications adaptées qui prennent en compte le changement de leur contexte d'utilisation.

L'objectif de notre travail diffère de celui des approches proposées dans [Khanouche et al., 2016] et [Fki, 2015] du fait qu'il consiste à trouver une composition proche de l'optimum en un temps raisonnable en prenant en compte cinq paramètres de qualité de service à savoir le temps d'exécution, le coût, la fiabilité, la disponibilité et la réputation. En fait, nous avons modélisé la composition des services web comme un graphe de planification [Pop et al., 2009] en prenant en compte les services répondant à des exigences fonctionnelles similaires. Ainsi le service composite recherché répond aux exigences de l'utilisateur en termes de QoS, où les services candidats qui ne satisfont pas les valeurs seuils de QoS, c'est-à-dire les limites exigées, ne seront pas retenus pour la composition.

5.4 Conclusion

Nous avons vu dans le présent chapitre l'essentiel des travaux qui traitent le problème "QoS-aware automatic web service composition". Nous avons constaté que pratiquement toutes les approches combinent les techniques de planification et les techniques d'optimisation pour trouver une composition optimale ou quasi-optimale répondant à la requête de l'utilisateur. Toutefois, ces approches souffrent de limitations lorsqu'il s'agit de trouver une solution optimale satisfaisant les exigences de l'utilisateur en termes de QoS. Pour pallier ces limitations, l'objectif du prochain chapitre consiste à présenter notre approche de composition automatique de services web sensible à la QoS. Cette approche combine : (1) un mécanisme de *matching* sémantique pour calculer la similarité sémantique entre les différents services consti-

tuant le service composite recherché et (2) une stratégie d'optimisation multi-critères de QoS utilisant l'algorithme Harmony Search satisfaisant les contraintes globales non-fonctionnelles de l'utilisateur. Cette proposition sera développée en détail dans le prochain chapitre.

Chapitre 6

Composition automatique des services web basée sur l'algorithme Harmony Search et les contraintes non-fonctionnelles de l'utilisateur

Sommaire

6.1 Introduction

Le problème de la composition de services web reste un des défis de la recherche, lorsqu'un seul service web ne satisfait pas à certaines exigences (fonctionnelles et/ou non-fonctionnelles), l'objectif est alors de combiner automatiquement d'autres services pour répondre pleinement à ces exigences [Kil, 2010]. Nous avons présenté dans le chapitre 3, les principales approches de composition automatique et nous avons constaté que le graphe de planification est la technique de planification la plus utilisée [Wang and Huang, 2014], [Yan et al., 2012], [Zheng and Yan, 2008], [Li et al.,

2010]. En effet, le graphe de planification modélise l'espace de recherche du problème et se base sur l'idée de transformer un problème de planification en un problème d'exploration, dans lequel l'espace d'état est exploré à partir de l'état initial en cherchant le but. Ses notions sont très compatibles avec le problème de la composition du service ; ils incluent le concept d'action et de proposition et les concepts d'entrée/sortie. Cependant, la plupart des approches étudiées sont incapables de produire des solutions optimales (ou quasi-optimales) satisfaisant les contraintes non-fonctionnelles (QoS) de l'utilisateur.

En termes de QoS, la composition reste encore un challenge ouvert en raison de la nature dynamique des services due aux changements aléatoires pouvant se produire dans de tels environnements, comme par exemple, l'apparition ou disparition de services, la dégradation de la qualité d'un service à cause de la mobilité des utilisateurs ou des dispositifs, les attentes des utilisateurs en termes de QoS qui peuvent évoluer avec le temps. Nous avons présenté dans le chapitre 4 un ensemble de travaux qui traitent le problème "QoS-aware service composition", par exemple [Canfora et al., 2005], [Xiangbing et al., 2012], [Hadjila, 2014], [Mohammed et al., 2014], [Amine, 2017]. Néanmoins, la majorité de ces approches supposent que les schémas de composition sont statiques et prédéfinis. Cela exige par ailleurs la transformation minutieuse des règles métier et des relations en un modèle de processus particulier. Dans le cas où plusieurs alternatives existent pour atteindre un objectif, ces alternatives doivent être énumérées et examinées de sorte que la stratégie optimale peut être sélectionnée. En outre, il n'est pas toujours possible de prédéfinir les schémas de processus pour des domaines d'application sophistiqués et volatils.

Dans le contexte de la composition automatique, la sélection d'un service composite tenant compte de la QoS connu sous le nom "QoS-aware automatic service composition" a été une problématique de recherche importante depuis des années. Beaucoup de travaux ont été dédiés à ce type de sélection, par exemple [Bartalos and Bieliková, 2009], [Yan et al., 2009], [Jiang et al., 2010], [Pop et al., 2009], [Chifu et al., 2015], [Salomie et al., 2014], [Rodriguez-Mier et al., 2017], [Bali, 2017]. A l'issue de l'étude comparative des principales approches présentées dans le chapitre 5, nous constatons que ces approches ont atteint un niveau de maturité satisfaisant si nous évaluons leur efficacité pour la composition des services web. Elles adoptent notamment des techniques de planification et/ou d'optimisation qui s'avèrent efficaces et adaptées au problème de composition de services web. Cependant elles souffrent de limitations lorsqu'il s'agit de trouver une solution optimale satisfaisant les exigences de l'utilisateur en termes de QoS.

A partir de ce constat, le présent chapitre présente une approche efficace pour résoudre le problème "QoS-aware automatic service composition" en satisfaisant les contraintes non-fonctionnelles (QoS) de l'utilisateur. Dans notre proposition, nous avons modélisé ce problème comme un graphe de planification [Pop et al., 2009] pour identifier la composition optimale (ou quasi-optimale) dans un temps raisonnable sans forcément parcourir tout l'espace de recherche. Comme critères de sélection nous avons considéré non seulement les paramètres non-fonctionnels (QoS) des services constituant la composition recherchée mais aussi la similarité sémantique entre eux. Nous avons utilisé 5 critères de QoS : le temps de réponse (*response time*), le coût (*cost*), la réputation (*reputation*), la fiabilité (*reliability*), et la disponibilité (*availability*). Ainsi, les services candidats qui ne satisfont pas les valeurs seuils de QoS, c'est-à-dire les limites exigées par l'utilisateur, ne seront pas retenus pour la composition. Afin de mieux expliquer nos motivations, nous présentons dans la section qui

suit un exemple.

6.1.1 Exemple de motivation

Supposons que nous ayons un utilisateur qui souhaite planifier un voyage comprenant la réservation d'un billet d'avion, la réservation d'une chambre d'hôtel et le traitement du paiement. Il est impossible qu'un seul service web accomplisse une tâche aussi complexe car il comprend plusieurs sous-tâches (réservation d'une chambre d'hôtel, réservation d'un billet d'avion et traitement des paiements).

Considérons que toutes ces sous-tâches sont publiées par des fournisseurs en tant que services web et que notre objectif est de les composer pour planifier ce voyage. Dans de telles situations, la composition du service est nécessaire. Elle combine plusieurs services web afin de créer un service composite qui est considéré par l'utilisateur comme un service web unique qui satisfait la requête de l'utilisateur.

Nous avons aussi anticipé qu'il y a plusieurs concurrents pour fournir chacun de ces services web (des services qui présentent les mêmes fonctionnalités mais des paramètres de QoS différents). Par exemple, les clients doivent choisir l'entreprise la plus appropriée qui propose des offres de paiements en ligne (le coût global ne doit pas dépasser le budget prédéfini). En plus, l'utilisateur a des exigences en termes de QoS comme par exemple que le temps d'exécution des services web soit réduit au minimum.

L'exemple présenté dans la Figure 6.1 est inspiré de celui cité dans [Shehu et al., 2014]. Nous avons étendu l'exemple avec la prise en compte des contraintes globales de QoS. L'utilisateur se présente avec une séquence de tâches qui doivent être exécutées pour compléter la requête. Dans ce cas, les offres de paiement en ligne, les offres de crédit, la réservation d'hôtel et la réservation de billet sont des tâches à effectuer via des appels de service web afin de répondre à la requête de l'utilisateur. Pour chacune de ces tâches, il existe de nombreux services web alternatifs (services candidats) capables de les exécuter, par ex. A1, A2 et A3 sont les services candidats capables de remplir la tâche `Offres de paiement en ligne`. Un service web est sélectionné pour chaque tâche en fonction de sa fonctionnalité et de sa qualité de service, puis orchestré avec des services web sélectionnés parmi d'autres tâches. Afin de créer cette composition d'une manière automatique, chaque service web est décrit par un ensemble de propriétés fonctionnelles (concepts ontologiques qui annotent ses entrées/sorties). Par exemple, un service web qui cherche à trouver un hôtel pourrait avoir ses paramètres d'entrée annotés avec les concepts "DestinationCity" et "HotelType" et son paramètre de sortie annoté avec le concept "HotelName". En plus de sa description (fonctionnelle) sémantique, chaque service est associé à un ensemble de valeurs pour les propriétés QoS suivantes : disponibilité, fiabilité, réputation, coût, et temps de réponse.

6.1.2 Contributions

Ce travail a pour but de favoriser l'intégration des préférences des utilisateurs en termes de QoS dans le processus de la composition de services web. Notre contribution pour résoudre le problème "QoS-aware automatic service composition" est basée sur un graphe de planification amélioré [Pop et al., 2009] et un algorithme Harmony Search [Geem et al., 2001] pour sélectionner la composition optimale ou quasi-optimale [Bekkouche et al., 2017]. L'utilisateur peut spécifier plusieurs contraintes

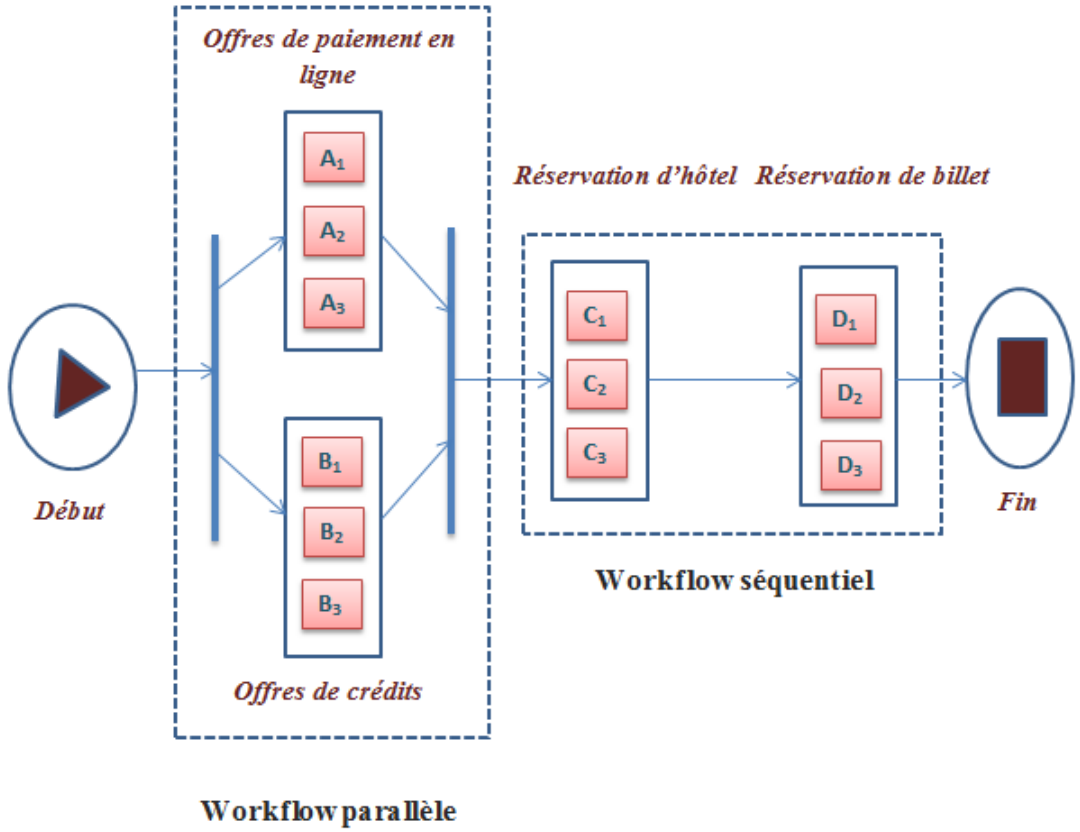

Figure *6.1:* *Scénario de planification de voyage pour la composition des services web sémantiques*

non-fonctionnelles. Ainsi, notre méthode est capable de trouver un service composite qui optimise la valeur de QoS globale, tout en satisfaisant les contraintes spécifiées. Nous résumons ci-dessous nos principales contributions :

1. *Mécanisme de Matching Sémantique* : le "matching" ou "appariement" est le moyen d'identifier un degré de similitude entre les concepts sémantiques (d'entrée et/ou de sortie) [He and Chang, 2003]. Il constitue une étape cruciale dans n'importe quel processus de composition. Dans notre travail, nous proposons un nouveau mécanisme de matching en se basant sur les types de matching définis dans [Klusch and Kapahnke, 2008]. Et cela dans le but de calculer les scores de similarité sémantique entre les services constituant le service composite recherché sur les différents niveaux du graphe de planification.
2. *Algorithme de sélection de la composition optimale (ou quasi-optimale)* : beaucoup de travaux ont été proposés pour trouver la composition optimale ou quasi-optimale dans le contexte de la composition automatique [Chifu et al., 2010], [Pop et al., 2009], [Pop et al., 2011b], [Chifu et al., 2015] et [Salomie et al., 2014]. Dans notre thèse, nous proposons un *algorithme Harmony Search* qui à notre connaissance n'a pas été utilisé pour résoudre le problème "QoS-aware automatic service composition". Il a pour particularité de prendre en compte non seulement les contraintes non fonctionnelles de l'utilisateur mais aussi la qualité sémantique des différents services constituant la solution recherchée.
3. *Comparaison de l'algorithme de sélection avec d'autres méta-heuristiques* : nous

comparons notre algorithme de sélection basé sur *l'algorithme Harmony Search* avec ses deux variantes développées postérieurement : *Improved Harmony Search (IHS)* [Mahdavi et al., 2007] et *Global Best Harmony Search (GHS)* [Omran and Mahdavi, 2008] dans le but d'améliorer le processus de sélection en termes de temps d'exécution de l'extraction de la solution d'une part et de taux d'optimalité d'une autre part.

Le reste du chapitre est organisé comme suit. La section 6.2 contient notre formulation des différentes notions de base du problème étudié, telles que les concepts de service web sémantique, la QoS, les contraintes globales de QoS, la requête et le problème de la composition automatique du service sensible au QoS, etc. La section 6.3, présente en détails notre approche de composition automatique qui est basée sur une structure de graphe de planification amélioré et un algorithme de recherche d'harmonie en décrivant le mécanisme de matching sémantique proposé ainsi que le processus de sélection. Enfin, dans la section 6.4, nous concluons ce chapitre en invoquant les perspectives.

6.2 Formalisation du problème de la composition des services web

Un plan de composition décrit la manière dont plusieurs services sont combinés afin de créer un service composite offrant une nouvelle fonctionnalité qu'aucun service atomique ne peut fournir individuellement. La composition de services avec QoS consiste à sélectionner un service composite maximisant une fonction objectif de QoS tout en satisfaisant plusieurs contraintes imposées aux valeurs des attributs de QoS. À cette fin, nous présentons nos définitions formelles des notions et concepts de base que nous utilisons pour nous attaquer au problème "QoS-aware automatic service composition".

Définition 1 - Qualité d'un service QoS.
Un attribut de qualité de service Q^k est une valeur qui représente une propriété non-fonctionnelle d'un service web. Ces attributs peuvent être divisés en attributs quantitatifs comme par exemple le coût, le temps de réponse, la réputation, la fiabilité et la disponibilité, et des attributs qualitatifs comme par exemple la sécurité, la confidentialité et le confort [Zeng et al., 2004].

Soit $\langle\ Q^1, Q^2, ...Q^n\ \rangle$ le vecteur des attributs de QoS d'un service web où Q^k représente la valeur du $k^{ième}$ attribut. Les attributs de QoS peuvent être classés en deux catégories :

1. un attribut positif (par ex. disponibilité, fiabilité) a une influence positive sur la QoS ; en d'autres termes, plus la valeur de l'attribut augmente, plus la QoS augmente ;
2. un attribut négatif (par ex. temps de réponse, coût) a un impact négatif sur la QoS ; autrement dit, plus la valeur de l'attribut augmente, plus la QoS diminue.

Ainsi, dans le contexte de la composition de services, les attributs positifs d'un service composite doivent être maximisés alors que ses attributs négatifs doivent être minimisés.

Définition 2 - Contraintes globales de QoS.
Les exigences non-fonctionnelles de l'utilisateur sont exprimées sous la forme d'un ensemble de contraintes globales $G = \langle g_1, g_2, ...g_n \rangle$ imposées sur les valeurs de QoS du service composite ; g_k $(0 \leq k \leq n)$ représente la contrainte imposée sur le $k^{ième}$ attribut de QoS et n le nombre des attributs de QoS. Ces contraintes représentent, d'une part, les bornes supérieures imposées aux valeurs des attributs de QoS négatifs du service composite et, d'autre part, les bornes inférieures pour les valeurs des attributs de QoS positifs du service composite.

Définition 3 - Ontologie.
Une ontologie est défini comme un 2-tuple Γ=$\langle$C,R$\rangle$, où :

1. C est l'ensemble des concepts représentés par des nœuds dans l'ontologie.
2. R est l'ensemble des relations hiérarchiques représentées par des arcs dans l'ontologie. Si un concept c_2 est un sous-concept direct d'un autre concept c_1, il peut être noté $c_2 \sqsubseteq c_1$, (voir Définition 5).

Dans le problème de la composition automatique des services, il est plus pratique de supposer que les services sont sans état (*stateless*) plutôt que des processus avec un état (*stateful*, c'est-à-dire qu'ils ont un comportement interne et des données stockées) [Khanouche et al., 2016]. Par conséquent, les services sont décrits par leurs paramètres d'entrées / sorties à travers des messages dans des descriptions WSDL, qui n'incluent pas de description comportementale spécifiant l'ordre dans lequel les opérations doivent être appelées. Par conséquent, pour une raison de simplicité, nous supposons que chaque service n'a qu'une seule opération.

Définition 4 - Service web sémantique.
Etant donné une ontologie Γ=$\langle$C,R$\rangle$, un service web sémantique est défini comme un triplet w_i=$\langle$ I_{w_i}, O_{w_i}, QoS_i $\rangle$, où :

1. $I_{w_i} \subseteq C$ représente l'ensemble des concepts qui décrivent sémantiquement les paramètres d'entrée du service web.
2. $O_{w_i} \subseteq C$ représente l'ensemble des concepts qui décrivent sémantiquement les paramètres de sortie du service Web.
3. QoS_i est un n-uplet $\langle Q_i^1, Q_i^2, ...Q_i^n \rangle$, où chaque Q_i^j indique la valeur correspondante d'un attribut QoS de n-uplet $\langle Q^1, Q^2, ...Q^n \rangle$.

Selon [Chabeb, 2011] le degré de similarité sémantique entre une requête donnée et un service composite est déterminé en fonction de la sémantique des éléments des descriptions à apparier. Dans la littérature, il existe principalement trois approches de matching :

- **IO-*matching*** : dit aussi "IO-*matching* de profil de service". Ce type d'appariement est déterminé à partir des données sémantiques des paramètres de service : les entrées : *inputs* (I) et les sorties : *outputs* (O). Ce type d'appariement est adopté dans [Paolucci et al., 2002].
- **PE-matching** : Ce type d'appariement est déterminé à partir d'appariements sur des pré-conditions (P) et des effets (E) des services et des requêtes. Ce type d'appariement est adopté dans PCEM [Klusch, 2008] avec des pré-conditions et des effets spécifiés en Prolog.

- **IOPE-matching** : Ce type d'appariement est déterminé à partir d'appariements sur les données sémantiques des *inputs* (I), des *outputs* (O), des préconditions (P) et des effets (E) des services et des requêtes. Cet appariement est adopté dans [Bartalos, 2011].

Pour évaluer le degré de similarité entre une requête donnée et un service composite (candidat) nous proposons un mécanisme de matching (appariement) sémantique basé sur les concepts ontologiques annotant les paramètres d'entrée/sortie des services web participant à la composition ainsi que les concepts ontologiques annotant les paramètres d'E/S de la requête (IO-matching). Pour cela, nous avons utilisé les différents types de matching introduits dans [Klusch and Kapahnke, 2008] :

- **Exact** ($\equiv$) : Une entrée $i_{w_j} \in I_{w_j}$ d'un service w_j correspond à une sortie $o_{w_i} \in O_{w_i}$ d'un service w_i avec un degré de matching *exact* si les deux concepts sont équivalents ; dénoté par, $i_{w_j} \equiv o_{w_i}$.
- **Plugin** ($\sqsubseteq$) : Une entrée $i_{w_j} \in I_{w_j}$ d'un service w_j correspond à une sortie $o_{w_i} \in O_{w_i}$ d'un service w_i avec un degré de matching *plugin* si i_{w_j} est un sous-concept direct de o_{w_i} ; dénoté par, $i_{w_j} \sqsubseteq o_{w_i}$.
- **Subsume** ($\sqsubset$) : Une entrée $i_{w_j} \in I_{w_j}$ d'un service w_j correspond à une sortie $o_{w_i} \in O_{w_i}$ d'un service w_i avec un degré de matching *subsume* si i_{w_j} est un sous-concept indirect de o_{w_i} ; dénoté par, $i_{w_j} \sqsubset o_{w_i}$.
- **Subsumed by** ($\sqsupseteq$) : Une entrée $i_{w_j} \in I_{w_j}$ d'un service w_j correspond à une sortie $o_{w_i} \in O_{w_i}$ d'un service w_i avec un degré de *subsumed by* si i_{w_j} est un super-concept direct of o_{w_i} ; dénoté par, $i_{w_j} \sqsupseteq o_{w_i}$.
- **Fail** ($\perp$) : si aucun des types de matching précédents n'est trouvé, les deux concepts sont incompatibles et le matching a un degré *fail* ; dénoté par, $i_{w_j} \perp o_{w_i}$.

Définition 5 - Matching Compatible .
Étant donnés deux ensembles de concepts O_{w_i} et I_{w_j} pour les services w_i et w_j respectivement, un *matching compatible* entre O_{w_i} et I_{w_j} existe si $\forall\ i_{w_j} \in I_{w_j}$, $\exists\ o_{w_i} \in O_{w_i}$ $(i_{w_j} \equiv o_{w_i} \vee i_{w_j} \sqsubset o_{w_i} \vee i_{w_j} \sqsubseteq o_{w_i} \vee i_{w_j} \sqsupseteq o_{w_i})$, dénoté par, $O_{w_i} \otimes I_{w_j}$.

Avant de présenter le problème de la composition d'une manière formelle, nous devons expliquer la requête de service du demandeur (qui peut être une application ou un utilisateur). Une requête R peut être vue comme un modèle du service demandé.

Définition 6 - Requête de Composition.
Étant donnés une ontologie Γ=⟨C,R⟩, une requête de composition est définie comme un quadruplet R=$\langle I_R, O_R, G, W\rangle$, où :

1. $I_R \subseteq C$ représente l'ensemble des paramètres d'entrée de la requête.
2. $O_R \subseteq C$ représente l'ensemble des paramètres de sortie de la requête.
3. G=$\langle\ g_1, g_2, ...g_n\ \rangle$ est un n-uplet de contraintes globales de QoS pour $\langle Q^1, Q^2, ...Q^n\rangle$.
4. $W = (W_{Q_{ind}}, W_{Q_{glob}}, W_{sem})$ tel que $W_{Q_{ind}}$,
   $W_{Q_{glob}}, W_{sem} \in [0, 1]$ est l'ensemble des poids fournis par l'utilisateur tels que :
   - $W_{Q_{ind}} = \langle W_{Q^1}, W_{Q^2}, ...W_{Q^n}\rangle$ représente les préférences de l'utilisateur concernant la valeur d'un attribut de QoS pour une solution de composition.
   - $W_{Q_{glob}}, W_{sem}$ représentent respectivement les préférences de l'utilisateur concernant :

- La valeur globale de QoS d'une solution de composition ;
- La valeur du score global de similarité sémantique d'une solution de composition.

Définition 7 - Problème "QoS-aware Automatic Service Composition". Étant donnés un ensemble de services web sémantiques W et une requête de composition R=$\langle I_R, O_R, G, W\rangle$, le problème de composition considéré dans notre travail consiste à rechercher une solution de composition optimale qui définit un ordre d'invocation sur un ensemble de services web (un workflow) $\{w_1, w_2, ..., w_n\}$ et satisfait les conditions suivantes :

1. $\{I_R \cup O_{w_1} \cup ... \cup O_{w_i}\} \otimes I_{w_{i+1}} (1 \leq i \leq n-1)$.
2. $\{O_{w_1} \cup O_{w_2} \cup ... \cup O_{w_n}\} \otimes O_R$.
3. La valeur globale de QoS de la solution est optimale.
4. Les contraintes globales de G sont satisfaites.

6.3 Approche proposée

Dans cette section, nous présentons en détail l'approche de composition des services web proposée. L'idée principale de cette approche est d'exploiter les techniques de planification et les techniques d'optimisation présentées dans le chapitre 3 et 4 dans le but de définir une méthodologie de composition automatique de services web. Notre approche modélise le problème "QoS-aware Automatic Service Composition" comme un processus multi-couches qui crée une structure de graphe de planification. Le graphe de planification est l'une des techniques de planification en intelligence artificielle qui sont considérées comme les plus performantes pour résoudre le problème de la composition de service web [Yan et al., 2010]. Il représente l'espace de toutes les solutions de composition candidates, pour une requête donnée.

Les étapes impliquées dans le mécanisme de composition que nous avons proposé sont présentées dans la Figure 6.2. Le processus de composition est déclenché par une requête de composition qui spécifie les besoins fonctionnels et non-fonctionnels de l'utilisateur. Les besoins fonctionnels sont exprimés par des paramètres d'E/S qui sont annotés par des concepts appartenant à une ontologie de domaine. Les besoins non-fonctionnels sont exprimés par des préférences en terme de QoS, par exemple, le coût total du service composite recherché ne doit pas dépasser une certaine valeur. Ensuite on génère le graphe de planification relativement à la requête. Dans cette étape, on cherche tous les liens sémantiques qui se trouvent entre les services sur les différents niveaux du graphe. Pour cela, nous proposons un IO-matching pour calculer les scores de similarité. Une fois tous les plans de composition trouvés, on procède à l'étape de sélection. Le but est de trouver une composition quasi-optimale sans forcément parcourir tout l'espace de recherche. Pour cette fin, nous avons adopté l'algorithme Harmony Search pour sélectionner le meilleur résultat en respectant les contraintes globales de QoS imposées par l'utilisateur.

Dans ce qui suit, nous expliquons chaque étape de l'approche proposée en fonction de la formulation du problème présenté dans la section précédente.

6.3.1 Spécification des contraintes de l'utilisateur

Dans cette phase, l'utilisateur peut spécifier sa requête de composition en termes d'exigences fonctionnelles et non fonctionnelles :

1. Des concepts ontologiques représentant la description sémantique des entrées et des sorties du service web composite.
2. Un ensemble de poids qui indiquent les préférences de l'utilisateur concernant l'importance des attributs QoS et la qualité sémantique établie entre les services impliqués dans la composition. Nous considérons deux catégories de poids : une catégorie fait référence à la pertinence des attributs QoS par rapport à la qualité sémantique, tandis que l'autre catégorie établit la pertinence de chaque attribut QoS individuel.
3. Un ensemble de contraintes de QoS globales sur la solution de composition (par exemple, le prix du service composite ne dépassant pas 100$, la disponibilité du service composite n'étant pas inférieure à 99,9%, etc.).

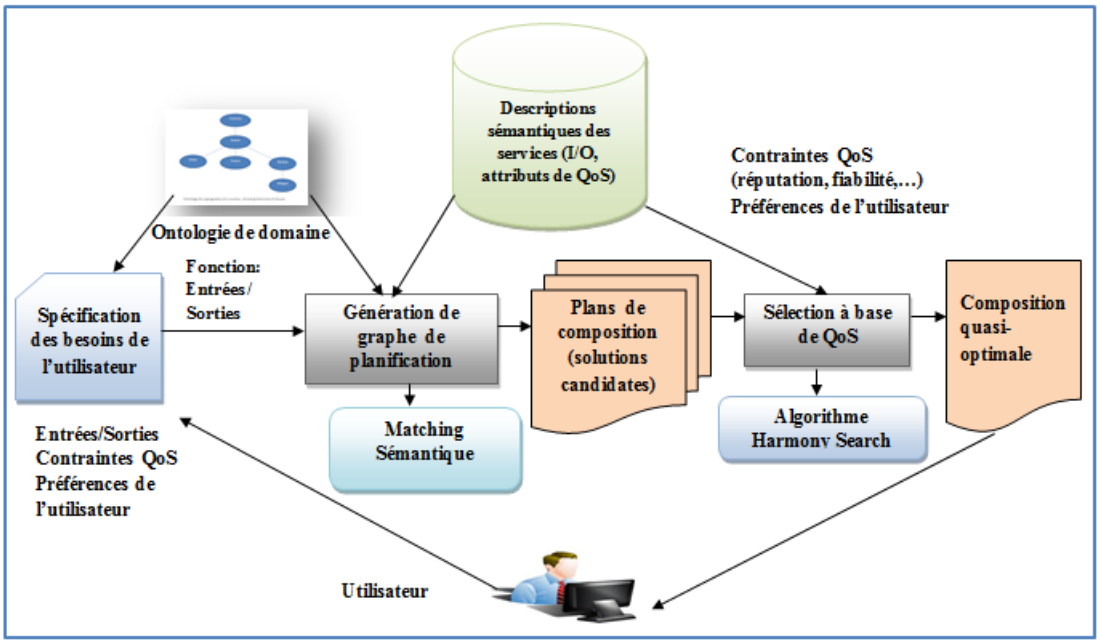

Figure *6.2:* *Approche proposée pour la composition des services web*

6.3.2 Génération de graphe de planification

Un graphe de planification est un graphe orienté structuré en niveaux qui permet d'organiser d'une manière efficiente l'espace de recherche d'un problème de planification. La solution de ce dernier est obtenue en explorant son graphe de planification [Ghallab et al., 2004], [Russell et al., 1995]. Formellement, un graphe de planification est défini comme suit [Ghallab et al., 2004] :

$G = (L_0, L_1, ..., L_n)$

G : est un graphe de planification.

n : est le nombre de niveaux que comporte le graphe de planification G.

L_0 : représente le premier niveau (level) du graphe qui est composé d'une seule couche nommée P_0.

Cette couche comporte toutes les propositions de l'état initial s_0 d'un problème de planification.

$L_{1 \leq i \leq n}$: représente un niveau ayant le rang i, chacun est composé de deux couches : A_i suivie par P_i.

A_i : est l'ensemble des actions, dont les pré-conditions sont incluses dans la couche de propositions P_{i-1}. P_i : est l'union de l'ensemble P_{i-1} et des ensembles des effets positifs de chaque action a de la couche A_i.

Les effets négatifs ne sont pas supprimés lors de la création d'une nouvelle couche de propositions. Les actions appartenant à la couche A_i sont reliées par des arêtes à leurs pré-conditions qui appartiennent à la couche P_{i-1}. Il existe aussi des arêtes qui partent de la couche A_i vers la couche P_i. Il y a donc deux types d'arêtes, des arêtes entrantes (de P_{i-1} vers A_i) et des arêtes sortantes (de A_i vers P_i). Le développement du graphe de planification s'arrête si le but est atteint ($g \subseteq P_i$) ou si deux niveaux consécutifs sont égaux (c'est-à-dire $P_{i-1} = P_i$ et $A_{i-1} = A_i$) [Chen, 2015].

Dans notre travail, nous avons adopté la structure du graphe de planification amélioré EPG (Enhanced Planning Graph) définie dans [Pop et al., 2009]. Ce graphe de planification fournit un espace de recherche unique où les connexions entre les couches sont exprimées de manière compacte. Il a des caractéristiques de solidité (*soundness*), de complétude, de terminaison, et peut être construit en temps polynomial [Blum and Furst, 1997]. Nous avons choisi cette structure du graphe de planification (EPG) pour les raisons suivantes :

- Il fournit l'ensemble des solutions de composition en réponse à la requête de composition.
- Il considère plusieurs services web offrant les mêmes fonctionnalités (ils sont regroupés dans des clusters voir Définition 10).
- Il prend en considération les attributs de QoS.

La construction du graphe de planification amélioré (EPG) prend en compte la similarité sémantique entre les paramètres de sortie de certains services et les paramètres d'entrée d'autres services. Nous définissons d'abord comment calculer les scores de similarité sémantique entre les services, puis nous donnons les concepts de base et les principes de l'EPG.

6.3.2.1 Calcul de score de similarité

Pour évaluer la similarité sémantique entre deux services, nous définissons d'abord une fonction de matching pour calculer le degré de matching entre les concepts décrivant les paramètres d'entrée/sortie des services.

Définition 8 - Degré de Matching.
Étant donné deux concepts $o_{w_i} \in O_{w_i}$ et $i_{w_j} \in I_{w_j}$ pour les services w_i et w_j respectivement, le degré de matching (DoM) entre o_{w_i} et i_{w_j} retourne une valeur dans [0,1]

tels que :

$$DoM(o_{w_i}, i_{w_j}) = \begin{cases} 1 & \text{if } i_{w_j} \equiv o_{w_i} \\ 0.9 & \text{if } i_{w_j} \sqsubseteq o_{w_i} \\ 0.8 & \text{if } i_{w_j} \sqsubset o_{w_i} \\ 0.7 & \text{if } i_{w_j} \sqsupseteq o_{w_i} \\ 0 & \text{Autrement} \end{cases} \tag{6.1}$$

Définition 9 - Score de Similarité Sémantique.
Le score de similarité sémantique(Sim_s) entre les concepts de sortie O_{w_i} du service w_i et les concepts d'entrée I_{w_j} du service w_j est calculé comme suit :

$$Sim_s(O_{w_i}, I_{w_j}) = \frac{\sum_{k=1}^{|I_{w_j}|} \max_{l=1}^{|O_{w_i}|} DoM(o^l_{w_i}, i^k_{w_j})}{|I_{w_j}|} \tag{6.2}$$

Où $o^l_{w_i} \in O_{w_i}$ et $i^k_{w_j} \in I_{w_j}$.

6.3.2.2 Structure de graphe de planification amélioré (EPG)

Le graphe de planification amélioré (EPG) [Pop et al., 2009] est obtenu en faisant la correspondance entre les concepts du graphe de planification en IA [Russell et al., 1995] et les concepts de la composition des services web sémantiques donnés précédemment d'une part, et en introduisant d'autres concepts comme le "*cluster de services*" et le "*littéral*" d'autre part :

Définition 10 - Cluster de Services.
Un cluster de services SC regroupe les services qui ont le même ensemble de paramètres d'entrée et qui sont dans l'une des relations de matching suivantes : *exact*, *plugin*, *subsume* or *subsumed by*. Formellement [Pop et al., 2009] :

$\forall w_i, w_j \in SC, \forall i_{w_i} \in I_{w_i}, \forall i_{w_j} \in I_{w_j}$, on a : $i_{w_i} \equiv i_{w_j} \vee i_{w_i} \sqsubseteq i_{w_j} \vee i_{w_i} \sqsubset i_{w_j} \vee i_{w_i} \sqsupseteq i_{w_j}$.

Définition 11 - Littéral.
Soit W un répertoire de services web, un littéral est un ensemble de concepts qui décrit sémantiquement un ensemble de paramètres de sortie et/ou d'entrée d'un ensemble de services web. Formellement [Pop et al., 2009] :

l=$l_i \mid \forall w_i \in W, l_i \in \{I_{w_i} \cup O_{w_i}\}$.

Le Tableau 6.1 montre la correspondance entre les concepts de graphe de planification en IA et les concepts de composition de services web sémantiques [Pop et al., 2011b].

Tableau 6.1: *Correspondance entre les concepts de graphe de planification et les concepts de composition de services web sémantiques [Pop et al., 2011b]*

Concepts de graphe de planification en IA	**Concepts de composition des services web sémantiques**
Graphe de planification	Graphe de planification amélioré
Action	Concept ontologique annotant une opération de service
Pré-condition	Concept ontologique annotant un paramètre d'entrée de service
Effet positif	Concept ontologique annotant un paramètre de sortie de service
État initial (A_0, L_0)	(ϕ, Concepts ontologiques décrivant les paramètres d'entrée fournis par l'utilisateur)
Littéraux but (Lg)	Concepts ontologiques décrivant les paramètres de sortie fournis par l'utilisateur
Ensemble d'actions (A_i)	Ensemble de clusters de services pour lesquels les paramètres d'entrée sont dans les littéraux L_{i-1}
Ensemble des littéraux (L_i)	Union de l'ensemble des littéraux L_{i-1} et l'ensemble des paramètres de sortie des services dans A_i

La construction du graphe de planification amélioré EPG (extension de graphe niveau par niveau) est un processus itératif. A chaque étape, un nouveau niveau constitué d'un tuple $\langle A_i, L_i \rangle$ est ajouté au graphe où A_i représente un ensemble de *clusters de services* et L_i est un *littéral*. Dans le tuple $\langle A_0, L_0 \rangle$ du niveau 0, A_0 est un ensemble de services vide et L_0 contient l'ensemble des paramètres d'entrée de la requête. Pour chaque niveau $i > 0$, A_i est un ensemble de clusters de services pour lesquels les paramètres d'entrée sont des concepts dans L_{i-1}, et L_i est un *littéral* obtenu en ajoutant les sorties des services dans A_i à l'ensemble L_{i-1} (Voir Tableau 6.1).

La construction du l'EPG s'arrête si les paramètres de sortie de la requête sont contenus dans le *littéral* actuel ou bien lorsque les ensembles de *clusters de services* et *littéraux* sont les mêmes sur deux nivaux consécutifs. Une solution de composition extraite du graphe de planification consiste en un ensemble de services tel qu'un service est sélectionné à partir de chaque cluster de chaque niveau. Dans ce qui suit, nous donnons les définitions formelles du "*graphe de planification pour la composition des services*" et une "*solution de composition*".

Définition 12 - Graphe de Planification pour la Composition des Services. Un graphe de planification pour la composition des services $SCPG$ (Service Composition planning Graph) est défini comme un ensemble de tuples [Pop et al., 2009] :

$SCPG$=$\{\langle A_i, L_i\rangle\}$, $0 \leq i \leq n$ où :

- $i \in [1...n]$ représente l'indice du niveau lay_i dans $SCPG$.
- $A_i = \{sc_j^i \mid sc_j^i \in lay_i\}$ représente l'ensemble des clusters de services du niveau lay_i dans $SCPG$.
- $L_i = L_{i-1} \cup \{c \mid c \in \cup\, O_{w_k},\ avec\ w_k \in sc_j^i\ et\ sc_j^i \in lay_i\}$, $i \geq 1$.

Note : (A_0, L_0) est un cas particulier où $A_0 = \phi$ et $L_0 = I_R$.

Définition 13 - Solution de Composition.
Une solution de composition *sol* pour le $SCPG$ contient un service de chaque cluster appartenant à un niveau du graphe.

Formellement [Pop et al., 2009] : $\forall\ lay_i \in SCPG \wedge \forall sc_j^i \in lay_i,\ \exists!\ w_{jk}^i \in sc_j^i \mid w_{jk}^i \in sol$ où :

- i représente l'indice du niveau lay_i dans $SCPG$;
- sc_j^i est le cluster j sur le niveau lay_i dans $SCPG$;
- w_{jk}^i est le service k dans le cluster j sur le niveau lay_i dans $SCPG$.

Un exemple de graphe de planification pour la requête de composition suivante : $I_R = \{c1, c2\}$ et $O_R = \{c4, c6, c7, c8\}$ est montré dans la Figure 6.3. Comme nous pouvons le voir, ce graphe contient 5 clusters de services (entourés de cercles) et 3 littéraux ($1^{ère}, 3^{eme}$ et $5^{eme} colonnes$).

Une solution de composition pour le graphe de la Figure 6.3 consiste à des sous-ensembles de services : $sol = < \{w_{11}^1 || w_{21}^1\}\ ,\ \{w_{11}^2 || w_{21}^2 || w_{31}^2\} >$.

$\{w_{11}^1 || w_{21}^1\}$ signifie que ces deux services web s'exécutent en parallèle. Il en est de même pour $\{w_{11}^2 || w_{21}^2 || w_{31}^2\}$.

Les services qui contribuent à l'extension du graphe de planification $SCPG$ sont fournis par un processus de découverte qui consiste à trouver les services web appropriés dans un répertoire de services, basé sur la matching sémantique entre les entrées des services et l'ensemble des littéraux du niveau précédent. *La matrice des liens sémantiques (MSL)* stocke les *liens sémantiques* établis entre les services sur les différents niveaux du graphe. Un service web pourrait être lié à plusieurs autres services web.

Définition 14 - Lien de Similarité Sémantique.
Un lien de similarité sémantique entre deux services w_i, w_j est défini comme un tuple [Pop et al., 2009] :

sl_{ij}=$(V, Sim_s(O_{w_i}, I_{w_j}))$ où V est un ensemble de paires de paramètres de sortie du service w_i et les paramètres d'entrée du service w_j tels que :

V=$\{(o_{w_i}, i_{w_j}) \mid DoM(o_{w_i}, i_{w_j}) > 0, o_{w_i} \in O_{w_i}, i_{w_j} \in I_{w_j}\}$.

Définition 15 - Matrice de Liens Sémantiques.
Une matrice de liens sémantiques est définie dans [Pop et al., 2009] :

SLM=$[slm_{ij}], i = 1, ...n; j = 1, ...m$; tels que :

$$slm_{ij} = \begin{cases} (\phi, 0), & \text{si } Sim_s(O_{w_i}, I_{w_j}) = 0 \\ sl_{ij}, & \text{Autrement} \end{cases} \tag{6.3}$$

où sl_{ij} représente le lien de similarité sémantique entre le service sur la ligne i et le service sur la colonne j.

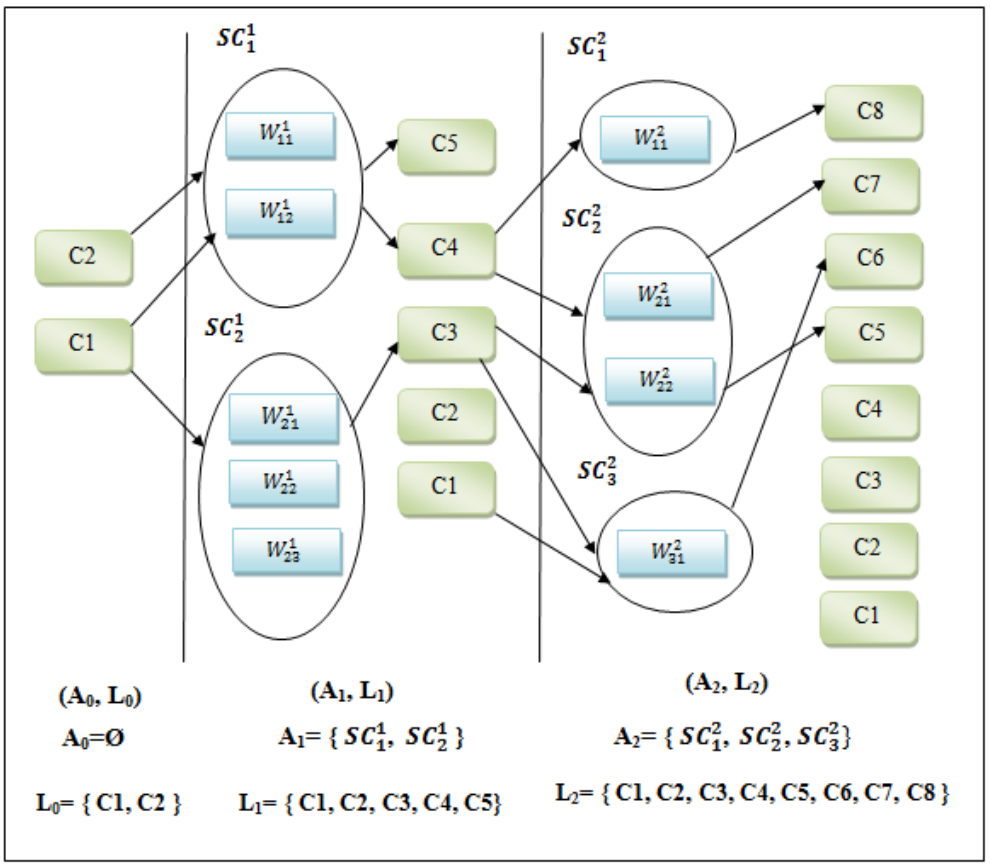

Figure *6.3:* *Exemple de Graphe de planification pour la composition des services,* $SCPG$=$\{\langle A_0, L_0\rangle, \langle A_1, L_1\rangle, \langle A_2, L_2\rangle\}$

6.3.3 Sélection des services web composites à base de QoS

Le problème de sélection des services web composites consiste en la sélection d'un service pour chaque tâche (dans notre cas un service de chaque cluster) de telle manière que les valeurs de QoS composées soient optimisées et les contraintes globales de l'utilisateur soient respectées. Un tel processus est formulé comme un problème d'optimisation mono-objectif afin de trouver une solution quasi-optimale. Les algorithmes Méta-heuristiques deviennent plus appropriés pour ce type de problèmes.

Ces dernières années, de nouvelles méta-heuristiques sont apparues dans le domaine de l'optimisation comme Harmony search Algorithme (HS) [Geem et al., 2001] Self-Organising Migrating Algorithm (SOMA) [Zelinka, 2004], Cuckoo Search (CS) [Yang and Deb, 2009], Invasive Weed optimization (IWO) [Mehrabian and Lucas, 2006], Biogeography Based Optimization (BBO) [Simon, 2008], Bat-inspired Algorithm [Yang, 2010] et Fruit Fly [Pan, 2012]. Ces algorithmes se sont révélés être très efficaces en surpassant de loin les autres dans la résolution de problèmes d'optimisation dans beaucoup de domaines. Certains ces algorithmes ont été utilisé avec efficacité pour résoudre le problème "QoS-aware automatic service composition". Cependant, pour autant que nous sachions, l'algorithme HS n'a pas été exploité dans ce problème. Alors que, beaucoup de travaux démontrent sa performance dans des problèmes d'optimisation [Kim and Geem, 2015], comme le problème de distribution d'eau [Geem, 2000], la vision artificielle, la segmentation d'images [Moh'd Alia et al.,

2009b], le routage de véhicules, la composition musicale [Geem and Choi, 2007], les jeux [Geem, 2007], ou encore le traitement multi-sources des désastres naturels [Bekkouche and Fizazi, 2016].

L'algorithme HS est considéré comme un outil d'optimisation très réussi grâce à sa technique de calcul souple et sa capacité à exploiter la nouvelle solution proposée (l'harmonie) par la synchronisation de l'espace de recherche à la fois avec l'intensification et la diversification environnementale d'optimisation parallèle [Moh'd Alia et al., 2009a]. Dans cette section, nous décrivons notre algorithme Harmony Search (HS) pour résoudre le problème "QoS-aware automatic service composition". L'espace de recherche comprend l'ensemble des solutions de composition de service candidates générées par le graphe *SCGP* décrit dans la phase précédente.

6.3.3.1 Principe de l'algorithme Harmony search

L'algorithme Harmony Search (Recherche par Harmonie /ou Fouille par Harmonie) est un algorithme d'optimisation méta-heuristique basé sur une analogie dérivée du domaine musical, développée par ZW Geem, J.H. Kim et G. Loganathan en 2001, pour résoudre des problèmes d'optimisation combinatoire [Geem et al., 2001]. A l'inverse de certaines méta-heuristiques qui s'inspirent des phénomènes naturels, cette méthode a été inspirée par l'observation des musiciens quand ils coopèrent ensemble pour produire un morceau de musique avec une parfaite harmonie. Pour y parvenir, ils procèdent par improvisations successives en utilisant leurs instruments et en se basant sur leurs expériences musicales [Yang, 2009], [Moh'd Alia and Mandava, 2011], [Geem and Choi, 2007].

Le processus de recherche de solutions optimales aux problèmes d'ingénierie est analogue à cette recherche efficace pour un parfait état d'harmonie. Le tableau 6.2 présente une comparaison entre l'improvisation musicale et l'optimisation [Ricart et al., 2011], [Geem, 2010].

Tableau 6.2: *Comparaison entre l'improvisation musicale et l'optimisation*

Facteur de comparaison	**Improvisation musicale**	**Optimisation**
Meilleur état	Parfaite harmonie	Optimum global
Estimée par	Esthétique standard	Fonction objectif
Estimation avec	Tons des instruments	Valeurs des variables
Produite par	Musiciens	Variables de décisions
Unité de traitement	Chaque pratique	Chaque itération

Dans le processus d'optimisation, le meilleur état trouvé (c.-à-d. optimum global) est déterminé par la valeur de la fonction objectif, tout comme l'algorithme HS. Les performances musicales d'un état parfait de l'harmonie trouvée (c'est-à-dire harmonie agréable) sont déterminées par l'estimation esthétique. Le ton de chaque instrument de musique détermine la qualité esthétique, tout comme la valeur de la fonction objective est déterminée par l'ensemble des valeurs attribuées à chaque variable de décision [Yang, 2009], [Lee and Geem, 2005].

La Figure 6.4 montre la structure de la mémoire d'harmonie (HM), qui est le noyau de l'algorithme HS, ainsi que l'analogie entre l'improvisation de la musique et l'optimisation d'un problème [Ricart et al., 2011], [Geem, 2010].

Dans l'improvisation de la musique, chaque joueur donne n'importe quel ton dans la gamme possible, constituant un vecteur d'harmonie. Si tous les tons forment une bonne harmonie, cette expérience est stockée dans la mémoire de chaque joueur. La possibilité de faire une bonne harmonie est augmentée pour la prochaine fois. De même, dans les techniques d'optimisation, chaque variable de décision est choisie parmi les valeurs d'un intervalle donné pour former un vecteur de solution. Si toutes les valeurs des variables de décisions font une bonne solution, cette expérience est stockée dans la mémoire de chaque variable. La possibilité pour faire une bonne solution est également augmentée la prochaine fois [Geem, 2010].

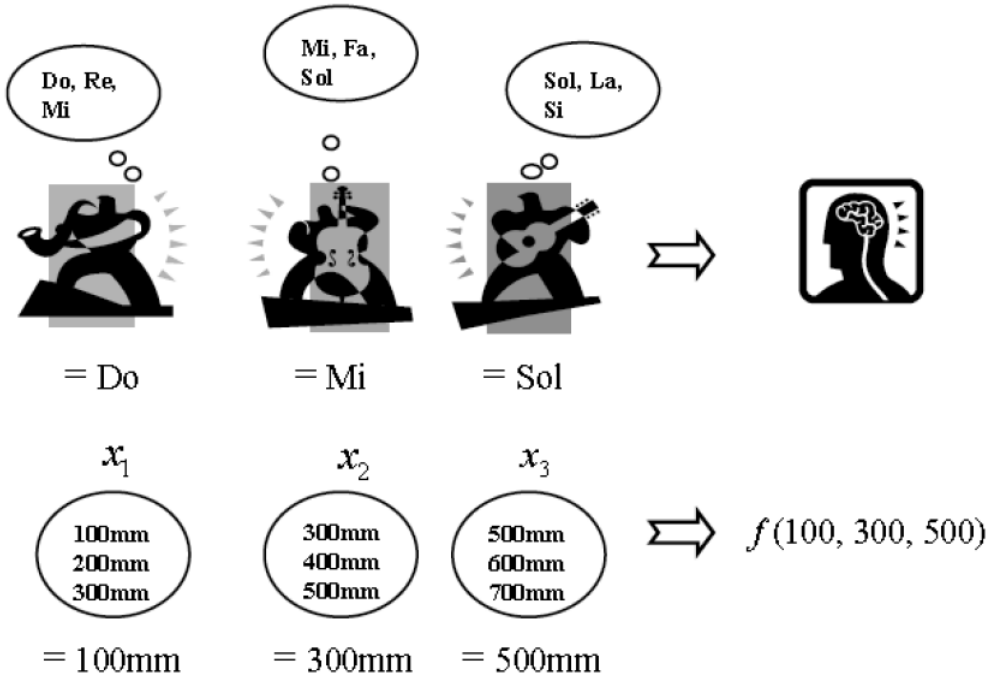

Figure *6.4:* *Analogie entre l'improvisation et l'optimisation*

Considérons un trio de jazz composé de saxophone, de contrebasse et de guitare. Il existe certaines valeurs de ton préférables dans chaque mémoire des musiciens : le saxophoniste, $\{Do, Ré, Mi\}$, le contrebassiste, $\{Mi, Fa, Sol\}$, et le le guitariste, $\{Sol, La, Si\}$. Si le saxophoniste joue au hasard $\{Do\}$ sur sa mémoire $\{Do, Ré, Mi\}$, contrebassiste $\{Mi\}$ sur $\{Mi, Fa, Sol\}$ et guitariste $\{Sol\}$ sur $\{La, Fa, Do\}$, cette harmonie $\{Sol, Si, Do\}$ engendre une autre harmonie. Si cette nouvelle harmonie est meilleure de la pire harmonie existante dans HM, la nouvelle harmonie est incluse dans HM et la pire est exclue. Cette procédure est répétée jusqu'à ce que l'harmonie idéale soit trouvée.

Dans un contexte d'optimisation réelle, chaque musicien peut être remplacé par une variable de décision $(X1, X2, X3)$. Les tons sonores préférés peuvent être remplacés par des variables de valeurs préférées. Si chaque variable de décision représente le diamètre du tuyau d'un arc entre deux nœuds, il y a certain nombre de diamètres préférés. Si on choisit pour $X1$, $\{100mm\}$ dans $\{100mm, 200mm, 300mm\}$, pour $X2$, $\{300mm\}$ dans $\{300mm, 400mm, 500mm\}$, et pour $X3$, $\{500mm\}$ dans

$\{500mm, 600mm, 700mm\}$. Ces valeurs $(100mm, 300mm, 500mm)$ font un autre vecteur de solution. Si ce nouveau vecteur est meilleur que le pire vecteur dans HM, le nouveau vecteur est inclus dans la HM et le pire vecteur en est exclu. Cette procédure est répétée jusqu'à ce que certains critères de terminaison soient satisfait.

Quand un musicien improvise une musique [Moh'd Alia and Mandava, 2011], le plus souvent il suit l'une des trois règles :

1. Jouer n'importe quel ton ou note de musique extrait de sa mémoire.
2. Jouer un ton ou une note de musique adjacente à n'importe quel son mémorisé.
3. Composer un nouveau ton ou note de musique aléatoire à partir de l'ensemble de tous les tons possibles.

De même par analogie, la formulation de ces trois options dans un processus d'optimisation qualitative deviennent, pour chaque valeur de variable de décision choisie dans l'algorithme HS [Belmadani et al., 2009]. Elle suit l'une des trois règles pour trouver une nouvelle Harmonie [Chakraborty et al., 2009] , [Geem et al., 2001], [Khader et al., 2014] :

1. *Considérations de la mémoire d'harmonies* : Choisir n'importe quelle valeur de la mémoire d'harmonie (Harmony Memory 'HM') avec une probabilité HMCR (Harmony Memory Consideration Rate/ Taux de considération de la mémoire Harmonie).
2. *Ajustements de ton* : Choisir une valeur adjacente à n'importe quelle valeur dans la HM avec une probabilité PAR (Pitch Adjustment Rate/ Taux d'ajustement de ton).
3. *Recherche aléatoire* : Choisir une valeur totalement aléatoire dans la gamme des valeurs possibles avec une probabilité 1-HMCR (Randomisation /Randomization).

La procédure de la solution de l'algorithme HS est contrôlée à l'aide de quatre paramètres [Moh'd Alia et al., 2009a] ; [Belmadani et al., 2009] qui sont :

- La taille de la mémoire d'harmonies *(HMS : Harmony Memory Size)*
- Le taux de considération de HM *(HMCR : Harmony Memory Considiring Rate)*
- Le taux d'ajustement du ton *(PAR : Pitch Adjusting Rate)*
- Le nombre d'itérations *(IT)*.

La solution d'un problème d'optimisation utilisant l'algorithme HS est basée sur les cinq étapes suivantes [Chakraborty et al., 2009] ; [Khader et al., 2014] ; [Moh'd Alia and Mandava, 2011] ; [Nakamura et al., 2011] :

Étape 1 : Initialisation des paramètres du problème à optimiser et d'algorithme HS. Initialiser les paramètres HS : nombre de variables de décisions, la gamme de valeurs *HMS* (c'est-à-dire le nombre de vecteurs dans la solution de la mémoire de l'harmonie), *HMCR* où $HMCR \in [0, 1]$, *PAR* où $PAR \in [0, 1]$ et le critère d'arrêt (c'est-à-dire le nombre d'itérations *(IT)*).

Étape 2 : Initialisation de la mémoire d'harmonies (HM). Initialiser *HM* en générant les vecteurs de solutions possibles de variables de décisions (l'ensemble de l'espace de recherche du problème à optimiser) et selon le paramètre *HMS*.

Étape 3 : Improvisation d'une nouvelle harmonie. Improviser une nouvelle harmonie (un nouveau vecteur de solution) qui utilise *HM*, la recherche aléatoire et les trois règles d'improvisation de l'algorithme HS selon les principes suivants [Moh'd Alia et al., 2009a] :

- Sélectionner la solution stockée dans HM avec une probabilité de taux de considération de l'harmonie mémoire $(HMCR) \in [0, 1]$.
- Si elle est choisie aléatoirement dans HM, la faire varier aléatoirement dans un voisinage proche avec une probabilité du taux d'ajustement du ton $(PAR) \in [0, 1]$, sinon on la laisse inchangée.
- Si elle n'est pas choisie à partir HM alors on la génère aléatoirement.

Étape 4 : Mise à jour de la mémoire d'harmonie (HM). Si le nouveau vecteur de solution improvisée est mieux que la pire des solutions dans HM, elle remplace la pire des solutions par la nouvelle (en termes de la valeur de la fonction objectif).

Étape 5 : Vérifier le critère d'arrêt. Vérifier la convergence : Si la convergence ou le critère d'arrêt n'ont pas été respectés, passer à l'étape 3, sinon donner la meilleure solution dans HM comme vecteur de solution optimale (ou quasi-optimale) du problème.

6.3.3.2 Adaptation de l'algorithme Harmony Search au problème de composition

Dans notre problème de composition de services web, HS est adapté comme suit :

- une *harmonie* représente une solution de composition candidate,
- un *ton (pitch) musical* est représenté par un cluster de services,
- une *esthétique standard* est représentée par une fonction de fitness qui évalue les solutions candidates.

L'algorithme HS utilise les propriétés fonctionnelles et non fonctionnelles (attributs QoS) des services web afin de trouver la solution de composition optimale. Une telle composition doit :

- Optimiser une fonction qui calcule le score de similarité sémantique entre les services.
- Optimiser la valeur globale de QoS de la solution.
- Satisfaire les contraintes globales de QoS.

a) Fonction de Fitness

La fonction de Fitness F est une fonction multi-critères qui optimise :

1. Le score global de similarité sémantique de la solution.
2. La valeur agrégée de QoS de la solution.

Définition 16 - Score Global de Similarité Sémantique .
Le score global de similarité sémantique (Sim_g) de la solution de composition *sol* est calculé comme suit :

$$Sim_g(sol) = \frac{\sum_{k=1}^{nlinks} Sim_s(O_{w^l_{qi}}, I_{w^p_{rj}})}{nlinks} \tag{6.4}$$

Où :

- $O_{w^l_{qi}}$ représente les concepts de sortie du service i dans le cluster q du niveau l ;

- $I_{w_{rj}^p}$ représente les concepts d'entrée du service j dans le cluster r du niveau p ;
- w_{qi}^l, w_{rj}^p sont des services qui appartiennent à la solution sol, $p = l + 1$;
- nLinks est le nombre total de liens de similarité sémantique (Voir Définition 14) dans la solution de composition sol.

La QoS d'un service composite dépend de sa structure ; c'est-à-dire, du plan de composition, et de la fonction utilisée pour l'agrégation des attributs de QoS. En termes de structure, les services atomiques peuvent être connectés en utilisant différents types de structures : séquentielle, parallèle, conditionnelle et boucle [Yu et al., 2007], [Zeng et al., 2004], [Zheng et al., 2013]. Des structures plus complexes peuvent être conçues en combinant les quatre structures de base précédentes. Dans une structure séquentielle, cf. figure 6.5 (a), les services $S_1, ...S_m$ sont exécutés dans un ordre donné alors que dans une structure parallèle, les services $S_1, ...S_m$ sont exécutés simultanément, cf. figure 6.5 (b). Dans une structure conditionnelle, cf. figure 6.5 (c), un service $S_i (0 \leq i \leq m)$ est choisi pour exécution avec une probabilité p_i parmi m services. Dans une structure en boucle, un service S_i est exécuté k fois, cf. figure 6.5 (d).

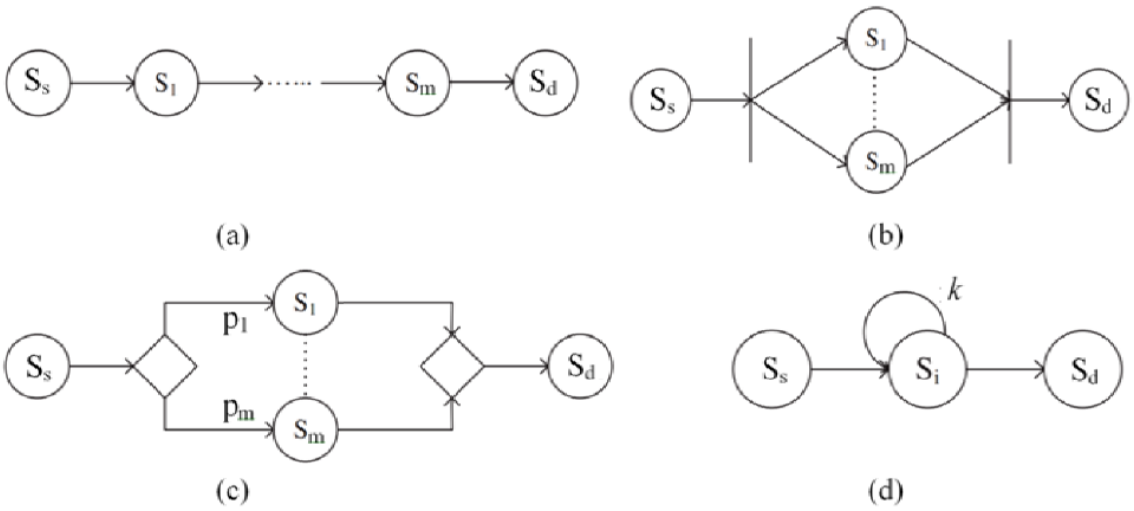

Figure *6.5:* *Les quatre structures de base utilisés pour la composition de services [Zheng et al., 2013]*

Dans notre travail, nous avons considéré cing attributs de QoS : temps de réponse, coût, disponibilité, fiabilité, et réputation. Le Tableau 6.3, décrit la liste des attributs de QoS utilisés [Geebelen et al., 2014], [Xiong et al., 2009], [Ardagna and Pernici, 2007].

Tableau 6.3: *Description des attributs de QoS*

Attribut de QoS	**Description**
Temps de réponse	Désigne la durée prise par un service pour répondre à une requête. Cette durée est susceptible d'inclure le temps de transmission engendré par le réseau lors de l'invocation du service.
Coût	Cet attribut désigne le prix qu'un client paie à chaque invocation d'un service.
Disponibilité	Indique l'aptitude d'un service à assurer sa fonctionnalité pendant un temps donné. La valeur de la disponibilité s'exprime le plus souvent sous la forme d'un pourcentage.
Fiabilité	Appelée aussi le taux d'exécution avec succès, cet attribut représente la capacité du service à réaliser correctement les tâches spécifiées dans sa description fonctionnelle.
Réputation	C'est une mesure de crédibilité d'un service web (elle est mesurée avec les feedbacks des utilisateurs).

Différents opérateurs peuvent être utilisés pour agréger les attributs de QoS d'un service composite : 1) l'addition (par ex. le coût) ; 2) le produit (par ex. la disponibilité) ; 3) l'opérateur min/max (par ex. le temps d'exécution dans le cas des structures parallèles). Le Tableau 6.4 montre les fonctions d'agrégations appliquées à cinq propriétés QoS [Zeng et al., 2004], [Alrifai et al., 2012], [Zeng et al., 2004].

QoS	**Séquentiel**	**Parallèle**	**Boucle**	**Conditionnel**
Temps de réponse	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$
Coût	$\sum_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\sum_{j=1}^{n} q(S_j)$	$\max_{j=1}^{n} q(S_j)$
Disponibilité	$\Pi_{j=1}^{n} q(S_j)$	$\Pi_{j=1}^{n} q(S_j)$	$\Pi_{j=1}^{n} q(S_j)$	$min_{j=1}^{n} q(S_j)$
Réputation	$\frac{1}{n}\sum_{j=1}^{n} q(S_j)$	$\frac{1}{n}\sum_{j=1}^{n} q(S_j)$	$\frac{1}{n}\sum_{j=1}^{n} q(S_j)$	$min_{j=1}^{n} q(S_j)$
Fiabilité	$\min_{j=1}^{n} q(S_j)$	$\min_{j=1}^{n} q(S_j)$	$q(S_j)$	$\min_{j=1}^{n} q(S_j)$

Tableau 6.4: *Les fonctions d'agrégation des propriétés [Alrifai et al., 2012] QoS*

Définition 17 - Score QoS d'une Solution de Composition.
La valeur QoS globale d'une solution de composition est calculée comme suit :

$$QoS(sol) = \sum_{k=1}^{n} w_{Q_k} * Q_k \tag{6.5}$$

où Q_k est la valeur QoS agrégée pour le $k^{éme}$ attribut QoS.

w_{Q_k} est le poids associé à chaque attribut QoS (son degré d'importance est défini par les préférences de l'utilisateur), n est le nombre des attributs QoS et $\sum_{k=1}^{n} w_{Q_k} = 1$.

En combinant les deux scores définis par les formules 6.4 et 6.5, on obtient la fonction objectif permettant l'évaluation d'une solution de composition définie comme suit :

$$F(sol) = W_{Q_{glob}} * QoS(sol) + W_{sem} * Sim_g(sol) \tag{6.6}$$

Où les poids $W_{Q_{glob}}, W_{sem} \in [0, 1]$ représentent les préférences de l'utilisateur concernant :

- La valeur globale de QoS d'une solution candidate ;
- Le score global de similarité sémantique d'une solution candidate.

De plus la fonction objectif $F(sol)$ doit favoriser l'évolution vers la satisfaction des contraintes globales de l'utilisateur, ainsi les compositions qui ne satisfont pas ces contraintes sont pénalisées. Plusieurs fonctions de pénalité sont proposées dans la littérature (fonctions statiques, dynamiques, adaptatives,etc). Nous avons choisi une fonction statique adoptée par [Lécué, 2009]. Notre choix est justifié par le fait que les autres fonctions ne donnent pas d'amélioration significative (elles augmentent le temps d'exécution)(voir formule 6.7).

$$F'(sol) = F(sol) - (\sum_{k=1}^{n} (\frac{\Delta Q}{g_k^{max} - g_k^{min}}))^2 \tag{6.7}$$

Où g_k^{max} et g_k^{min} sont respectivement la valeur maximale (borne supérieure) et la valeur minimale (borne inférieure) de la k^{th} contrainte globale. n représente le nombre de contraintes globales et ΔQ est défini par la formule 6.8.

$$\Delta Q = \begin{cases} Q_k - g_k^{max} & \text{si } Q_k > g_k^{max} \\ 0 & \text{si } g_k^{min} \leq Q_k \leq g_k^{max} \\ g_k^{min} - Q_k & \text{si } Q_k < g_k^{min} \end{cases} \tag{6.8}$$

b) L'algorithme Harmony Search pour la sélection de composition optimale

Notre algorithme de sélection basé sur la recherche d'harmonie (voir Algorithme 1) détermine la solution optimale en considérant l'ensemble des solutions générées par le graphe de planification de composition $SCPG$. Les entrées de l'algorithme de sélection sont :

- Le $SCPG$ pour une requête de composition,
- La fonction objectif F' (voir la formule 6.7),
- Les paramètres ajustables HMS, $HMCR$, PAR,
- *pitch_num* dénote l'ensemble des clusters de services,
- *pitch_bounds* représente l'ensemble des services dans un cluster donné,
- *itération_max* est le nombre maximum d'itérations.

L'algorithme procède comme suit :

Algorithme 1 Algorithme Harmony Search

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

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

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

Step 1. Initialiser la mémoire d'harmonies HM avec des solutions aléatoires générées par le graphe de planification $SCPG$ *(Line 1)*. La mémoire HM est codée comme une matrice où chaque colonne représente un cluster de services $pitch\_j \in pitch\_num$ et chaque ligne est une harmonie (solution). Les solutions initialisées dans cette étape sont améliorées dans un processus itératif qui est exécuté jusqu'à ce que le nombre maximum d'itérations soit atteint *(Lignes 3-23)*.

Step 2. Evaluer les harmonies dans HM selon F' *(Line 2)*.

Step 3. Improviser (créer) une nouvelle harmonie comme suit *(Line 3-23)* :

(i) Chaque composant $pitch\_j$ de la solution est initialisé avec un service web w_i pris d'une harmonie aléatoire h tel que $h \in HM$ avec une probabilité $HMCR$ *(Line 6-7)*.

(ii) Avec une probabilité PAR, le service web choisi précédemment est remplacé par un service web voisin w'_i du même cluster de services tel que $F'(w_1, ...w'_i...w_n)$ est maximisé *(Line 8-12)*.

(iii) Si les instructions i) and ii) ne sont pas exécutées, le composant $pitch\_j$ est initialisé avec un service web aléatoire w_j, tel que $w_j \in pitch\_bounds(pitch\_j)$ *(Line 13-14)*.

(iv) Nous mettons à jour le composant $pitch\_j^{th}$ de l'harmonie en utilisant la valeur de $Pitch$ *(Line 16)*.

Step 4. Évaluer la solution construite notée $Harmony$ *(Line 18)*.

Step 5. La plus mauvaise harmonie de HM est remplacée par $Harmony$ si cette dernière est plus efficace en termes d'optimalité *(Lines 19-22)*.

Step 6. Retourner la meilleure solution de HM ($Best\_Harmony$) *(Lines 24-25)*.

6.4 Conclusion

L'approche proposée dans ce chapitre traite le challenge de la composition automatique des services web sémantiques avec contraintes globales de QoS. Le problème est modélisé comme un graphe de planification amélioré en prenant en compte les services répondant à des exigences fonctionnelles similaires. La méthode proposée permet de trouver une solution de composition quasi-optimale dans un temps raisonnable. Une formalisation du problème est d'abord présentée en introduisant les concepts nécessaires.

En se basant sur la structure de graphe de planification amélioré, les services sont groupés dans des clusters. Un mécanisme de *matching* sémantique est proposé pour calculer la similarité sémantique entre les services sur les différents niveaux du graphe. Ensuite, une technique d'optimisation multi-critères est appliquée à l'ensemble des compositions candidates générées par le graphe. L'algorithme Harmony Search a été adopté pour trouver la composition quasi-optimale en satisfaisant les contraintes de l'utilisateur en termes de QoS. Dans le chapitre de validation qui suit, nous allons présenter, la faisabilité et l'efficacité de l'application de notre approche proposée.

Chapitre 7

Implémentation et expérimentations

Sommaire

7.1 Introduction

Dans le but de tester et de valider notre approche de composition des services web, nous avons implanté le processus de composition décrit dans le chapitre précédent. Dans ce chapitre, nous présentons l'architecture de l'environnement de composition en détaillant les différents modules le constituant. Ensuite nous présentons les outils et les technologies que nous avons utilisés pour la conception et l'exécution de la plateforme de composition. Puis, nous présentons la base de services utilisée pour évaluer l'environnement. Enfin, nous montrons les différentes expérimentations menées tout en discutant les résultats obtenus et présentant les menaces à la validité (Threats to validity) [de Oliveira Barros and Dias-Neto, 2011].

7.2 Mise en œuvre de l'environnement de composition

7.2.1 Architecture de l'environnement de composition

Dans ce qui suit, nous présentons l'environnement de composition suivant l'approche définie dans la Figure 6.2. Notre implantation de l'environnement de composition, illustrée sur la Figure 7.1 est constituée des modules suivants :

- *Module de génération de graphe de planification amélioré :* le point d'entrée de ce module est un ensemble de besoins fonctionnels et non-fonctionnels de l'utilisateur exprimé sous forme d'une requête (voir section 6.3.1). Ce module construit le graphe de planification amélioré (EPG) en prenant en compte la similarité sémantique entre les paramètres de sortie de certains services et les paramètres d'entrée d'autres services. Pour effectuer le matching sémantique entre les paramètres d'E/S (annotés par des concepts appartenant à une ontologie de domaine) de deux services web, ce module s'appuie sur le module de matching sémantique.
- *Module de matching sémantique :* ce module effectue le matching sémantique entre les concepts ontologiques des services selon les niveaux de matching (exact, subsume, plugin, subsumed by, fail)(voir section 6.3.2.1). Il utilise le raisonneur Pellet [1] ainsi que l'API Jena [2] pour inférer les relations entre les concepts. Ce module utilise une ontologie de domaine.
- *Module de sélection à base de QoS :* ce module génère le plan de composition quasi-optimale de telle manière que les valeurs de QoS composées soient optimisées et les contraintes globales de l'utilisateur soient respectées. Le point d'entrée de ce module est l'ensemble de solutions candidates générées par le graphe EGP. Ce module implémente l'algorithme Harmony Search décrit dans la section 6.3.3.2.

7.2.2 Outils, langages et technologies utilisés

Les modules de l'architecture de composition ont été implémentés en Java (JDK 1.7.0) et l'environnement de développement intégré *Eclipse*. Nous avons eu recours à divers outils et bibliothèques externes, qui sont également écrits en Java. Le Tableau 7.1 résume tous les langages et les outils ainsi que les technologies qui interviennent dans la conception des modèles utilisés et dans l'exécution de la plateforme de composition.

1. http ://www.mindswap.org/2003/pellet/
2. https ://jena.apache.org/documentation/ontology/

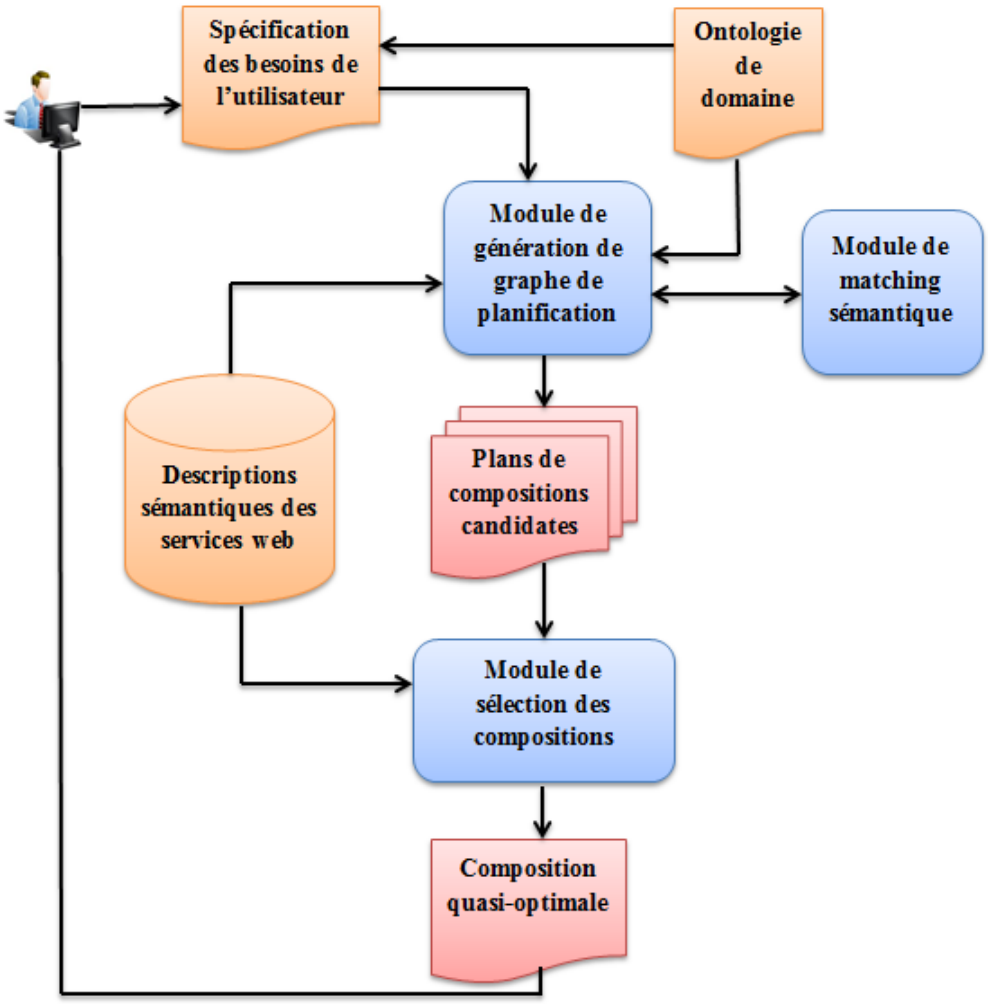

Figure 7.1: *Architecture de l'environnement de composition*

Tableau 7.1: *Outils, langages et technologies utilisés dans l'environnement de composition*

Tâche	**Outils/langages/technologies**
Programmation	Java 1.7
Environnement de développement	Eclipse
Gestion d'ontologies	Protégé 3.2.1, OWL, Pellet 2.0, Jena
Gestion des fichiers XML	JDOM

Dans ce qui suit, nous détaillons chacun des outils et langages utilisés pour la manipulation des données ainsi que l'implémentation de l'environnement de composition.

- *Le langage de programmation Java :* notre choix s'est porté sur le langage Java, et cela parce que Java est un langage orienté objet simple ce qui réduit les risques d'incohérence ; il est portable, il peut être utilisé sous Windows, sous Linux, sous

Macintosh et sur d'autres plateformes sans aucune modification, enfin il possède une riche bibliothèque de classes comprenant des fonctions diverses telles que les fonctions standards, le système de gestion de fichiers, les fonctions multimédia et beaucoup d'autres fonctionnalités.

- *Eclipse :* on a opté pour *Eclipse* comme environnement de développement car il possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :
  - Une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plugins ;
  - Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS ;
  - Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT.
- *L'éditeur Protégé*[3] développé par l'Université de Stanford en collaboration avec l'Université de Manchester, est le standard de facto pour la création et l'édition d'ontologies OWL. Son code source, libre, est écrit en Java, et il admet des extensions sous forme de plugins. Il existe de nombreux plugins, par exemple pour la visualisation des ontologies
- *L'API OWL*[4] est l'implantation de référence pour la création, la manipulation et la sérialisation d'ontologies OWL. OWL-API est un projet libre mené par l'université de Manchester qui supporte pleinement OWL 2. Il fournit également plusieurs interfaces pour l'accès à des moteurs d'inférence d'une façon transparente.
- *L'API Pellet :* [Sirin et al., 2007] cette API permet le raisonnement sur les ontologies formalisées avec OWL, elle offre des mécanismes d'inférences basés sur les logiques de description.
- *L'API Jena* [McBride et al., 2004] est une API Java qui contient des classes et des interfaces pour l'interaction avec les modèles RDF et OWL et les raisonneurs comme Pellet. Jena permet la création, la mise à jour et la suppression des triplets RDF ainsi que l'interrogation basée sur SPARQL.
- *L'API JDOM :* [Hunter and McLaughlin, 2007] une API open source Java, vue comme un modèle de documents objets dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. JDOM propose aussi une intégration de SAX, DOM, XSLT et XPath. Un document XML est encapsulé dans un objet de type Document. Les éléments d'un document sont encapsulés dans des classes dédiées : Element, Attribute, Text, Processing Instruction, Namespace, Comment, DocType, EntityRef, CDATA. JDOM permet aussi de vérifier que les données contenues dans les éléments respectent la norme XML.

7.3 Description de la base des services

Nous avons utilisé la collection de données de la compétition sur le WSC *Web Service Challenge 2009*[5] qui est utilisée dans de nombreux travaux sur la composition

3. http ://protege.stanford.edu/
4. http ://owlapi.sourceforge.net/
5. http ://ws-challenge.georgetown.edu/wsc09/

automatique de services web, tels que ceux de [Yan et al., 2012], [Chen and Yan, 2014], [Yan et al., 2008]. Cette collection de données est bien adaptée au problème de composition des services web, car elle implique un grand nombre de services web (˜4000 instances), un grand nombre de concepts (˜40 000 concepts), et différentes tailles de la solution (3 à 10 services). Elle contient trois fichiers :

- Un fichier *WDSL* qui contient un ensemble de descriptions sémantiques des services web annotés (les paramètres d'entrée/sortie sont annotés par des concepts ontologiques). Le nombre de services varie entre 500 à 4000.
- Un fichier *OWL* contenant la hiérarchie des concepts dans l'ontologie. Le nombre de concepts varie entre 5000 et 40 000 concepts.
- Un fichier *WSLA* décrit les attributs QoS des services web. Dans la collection de données, deux attributs QoS seulement sont considérés : le temps de réponse et le débit. Afin de prendre en considération d'autres attributs QoS, nous avons enrichi ce fichier avec 4 attributs : le coût, la disponibilité, la fiabilité et la réputation comme suggéré dans [Yu and Bouguettaya, 2009] : Coût : [0, 30]$, Disponibilité : [0.7, 1]%, Fiabilité : [0.5, 1]% , Réputation : [0, 5]%, et Temps de réponse : [0, 300] ms.

7.4 Expérimentations

Dans cette section, nous analysons les performances de notre approche de composition automatique des services web. Nous menons une série d'expérimentations qui répond aux questions suivantes :

- **QR1 :** Quelle est la performance de notre méthode d'optimisation en termes d'optimalité et temps d'exécution ?
- **QR2 :** Etudier l'impact des contraintes QoS sur la qualité des solutions de composition trouvées ?
- **QR3 :** Est-ce que les deux variantes de l'algorithme HS, IHS et GHS permettent d'améliorer le processus de sélection ?

Les expérimentations ont été réalisées dans une machine sous Windows 7 dont les performances sont : 1.80 GHz Intel i5 core CPU, et 6 GB de RAM.

7.4.1 Expérimentation pour QR1

Afin de répondre à la première question de recherche, nous présentons notre méthodologie pour évaluer l'algorithme de sélection ainsi qu'une analyse des résultats expérimentaux.

7.4.1.1 Méthode d'évaluation

La convergence d'un algorithme d'optimisation vers la solution optimale est influencée par un ensemble de paramètres qui sont spécifiques à l'algorithme. Nous pensons que la meilleure façon d'évaluer un algorithme d'optimisation consiste d'abord à trouver les valeurs optimales des paramètres, ensuite évaluer la performance de l'algorithme en utilisant la configuration optimale de ces paramètres. Pour trouver les valeurs optimales des paramètres ajustables, les deux étapes suivantes doivent être abordées [Salomie et al., 2014] :

Dans un premier temps, une recherche exhaustive doit être effectuée pour identifier le score de la solution de composition optimale (ou quasi-optimale). Ce score est en outre utilisé pour identifier la configuration la plus appropriée des paramètres ajustables ; ce qui garantit que la solution optimale ou quasi-optimale est obtenue sans parcourir tout l'espace de recherche.

Dans un second temps, il faut faire varier les valeurs des paramètres pour identifier leurs valeurs optimales. Au cours de ces expérimentations, le *temps d'exécution* et la *déviation* de la solution optimale retournée par l'algorithme sont comparés avec le *temps d'exécution* et la *solution optimale* retournée par la recherche exhaustive.

7.4.1.2 Résultats expérimentaux

Nous avons testé notre approche sur trois scénarios avec des complexités différentes. Dans le Tableau 7.2, nous avons spécifié pour chaque scénario : (i) l'identifiant du scénario, (ii) la configuration du graphe en termes de nombre de niveaux (donné par le nombre de sous-ensembles), le nombre de clusters pour chaque niveau (donné par la cardinalité de chaque sous-ensemble) et le nombre de services par cluster (donné par la valeur de chaque élément dans un sous-ensemble), (iii) la complexité de l'espace de recherche en termes de nombre de solutions possibles encodées dans le graphe de planification (le nombre a été obtenu en comptant le nombre de solutions générées dans une recherche exhaustive), (iv) la valeur de fitness optimale globale en effectuant une recherche exhaustive, et (v) le temps d'exécution pour trouver la solution lors d'une recherche exhaustive.

Tableau 7.2: *Les configurations du graphe pour chaque Scenario*

Scenario Id	**Configuration du graphe**	**Complexité de l'espace de recherche**	**Fitness optimale**	**Temps d'exécution (heure :min :s)**
A	Niveau1 :{3 5 6} Niveau2 :{6 4 3} Niveau3 :{4 6 5}	777600	0.891	00 :02 :10
B	Niveau1 :{2 5 4 6} Niveau2 :{6 4 6 4} Niveau3 :{4 5}	2764800	0.950	00 :07 :47
C	Niveau1 :{6 6 3 3} Niveau2 :{5 6 5 3} Niveau3 :{6 5 6}	26244000	0.920	01 :13 :18

Les paramètres ajustables de l'algorithme HS sont les suivants : *HMS*, *HMCR* et *PAR*. Nous avons fait varier les valeurs de ces paramètres en considérant les mêmes plages que celles utilisées dans les travaux de [Kim and Geem, 2015] : $HMS \in \{200, 400\}$, $HMCR \in \{0, 70, 0.95\}$, et $PAR \in \{0.01, 0.30\}$. Les Tableaux 7.3, 7.4 et 7.5 illustrent un fragment des meilleurs résultats obtenus (fitness optimale moyenne

(*Fitness*), temps d'exécution moyen (*Time*), déviation moyenne (*Déviation*)) en faisant varier les valeurs des paramètres ajustables pour les scénarios A, B et C. Chaque ligne du tableau représente une valeur moyenne des résultats obtenus en exécutant l'algorithme 50 fois sur la même configuration de paramètres. Les lignes surlignées en gras indiquent la configuration optimale des paramètres ajustables (meilleur compromis entre fitness et temps d'exécution).

En analysant les résultats expérimentaux, nous concluons que l'algorithme de sélection HS présente une bonne performance en termes d'optimalité (la valeur de fitness optimale moyenne dépasse 0,8 dans le cas du scénario B). En outre, il renvoie en moyenne la solution optimale ou quasi-optimale en 7 secondes (dans le cas des trois scénarios), ce qui est un temps d'exécution acceptable. Cela confirme la capacité de l'algorithme HS à explorer l'espace des solutions dans un temps raisonnable.

Tableau 7.3: *Fragment des meilleurs résultats pour le scénario A*

#	HMS	HMCR	PAR	Fitness	Time	Déviation
1	250	0.70	0.1	0.720	5000	0.171
2	250	0.70	0.2	0.725	5500	0.166
3	250	0.70	0.3	0.730	5500	0.161
4	250	0.80	0.1	0.720	5000	0.171
5	250	0.80	0.2	0.726	5500	0.165
6	250	0.80	0.3	0.735	5500	0.156
7	250	0.95	0.1	0.720	5800	0.171
8	250	0.95	0.2	0.735	6100	0.156
9	250	0.95	0.3	0.730	6200	0.161
10	300	0.70	0.1	0.729	5500	0.162
11	300	0.70	0.2	0.740	5500	0.151
12	300	0.70	0.3	0.741	6000	0.15
13	300	0.80	0.1	0.729	4000	0.162
14	300	0.80	0.2	0.730	4100	0.161
15	300	0.80	0.3	0.732	5000	0.159
16	300	0.95	0.1	0.733	4000	0.158
17	**300**	**0.95**	**0.2**	**0.745**	**5000**	**0.146**
18	300	0.95	0.3	0.744	5910	0.147
19	400	0.70	0.1	0.730	5000	0.161
20	400	0.70	0.2	0.731	6000	0.160
21	400	0.70	0.3	0.732	6100	0.159
22	400	0.80	0.1	0.720	6500	0.171
23	400	0.80	0.2	0.725	6800	0.166
24	400	0.80	0.3	0.725	6800	0.166
25	400	0.95	0.1	0.700	7000	0.191
26	400	0.95	0.2	0.710	7000	0.181
27	400	0.95	0.3	0.720	7100	0.171

Tableau 7.4: *Fragment des meilleurs résultats pour le Scénario B*

#	**HMS**	**HMCR**	**PAR**	**Fitness**	**Time**	**Déviation**
1	250	0.70	0.1	0.8	4500	0.15
2	250	0.70	0.2	0.8	4510	0.15
3	250	0.70	0.3	0.810	4520	0.14
4	250	0.80	0.1	0.810	5000	0.14
5	250	0.80	0.2	0.820	5200	0.13
6	250	0.80	0.3	0.830	5300	0.12
7	250	0.95	0.1	0.80	5100	0.15
8	250	0.95	0.2	0.80	5000	0.15
9	250	0.95	0.3	0.810	5100	0.14
10	300	0.70	0.1	0.810	5000	0.14
11	300	0.70	0.2	0.830	5500	0.12
12	300	0.70	0.3	0.820	5200	0.13
13	300	0.80	0.1	0.828	4900	0.122
14	300	0.80	0.2	0.830	5000	0.12
15	300	0.80	0.3	0.835	5500	0.115
16	300	0.95	0.1	0.82	6000	0.13
17	300	0.95	0.2	0.84	6200	0.11
18	**300**	**0.95**	**0.3**	**0.85**	**6300**	**0.1**
19	400	0.70	0.1	0.821	6300	0.129
20	400	0.70	0.2	0.828	6400	0.122
21	400	0.70	0.3	0.810	6500	0.14
22	400	0.80	0.1	0.829	6800	0.121
23	400	0.80	0.2	0.83	6900	0.12
24	400	0.80	0.3	0.831	7000	0.119
25	400	0.95	0.1	0.810	7100	0.14
26	400	0.95	0.2	0.820	7200	0.13
27	400	0.95	0.3	0.825	7200	0.125

Tableau 7.5: *Fragment des meilleurs résultats pour le Scénario C*

#	HMS	HMCR	PAR	Fitness	Time	Déviation
1	250	0.70	0.1	0.77	5000	0.15
2	250	0.70	0.2	0.770	5000	0.15
3	250	0.70	0.3	0.772	5100	0.148
4	250	0.8	0.1	0.776	5200	0.144
5	250	0.8	0.2	0.78	5300	0.14
6	250	0.8	0.3	0.782	5310	0.138
7	250	0.95	0.1	0.79	5300	0.13
8	250	0.95	0.2	0.79	5300	0.13
9	250	0.95	0.3	0.78	5200	0.14
10	300	0.70	0.1	0.771	5000	0.149
11	300	0.70	0.2	0.772	5100	0.148
12	300	0.70	0.3	0.782	5200	0.138
13	300	0.8	0.1	0.77	5000	0.15
14	300	0.8	0.2	0.783	6000	0.137
15	300	0.8	0.3	0.773	5900	0.147
16	300	0.95	0.1	0.79	6700	0.13
17	300	0.95	0.2	0.782	6400	0.138
18	300	0.95	0.3	0.783	6500	0.137
19	400	0.70	0.1	0.787	6600	0.133
20	300	0.70	0.2	0.789	6700	0.131
21	300	0.70	0.3	0.789	6700	0.131
22	400	0.8	0.1	0.790	6800	0.13
23	400	0.8	0.2	0.792	6800	0.128
24	400	0.8	0.3	0.793	6830	0.127
25	400	0.95	0.1	0.789	6800	0.131
26	**400**	**0.95**	**0.2**	**0.800**	**7000**	**0.12**
27	400	0.95	0.3	0.790	7000	0.13

Afin d'évaluer la performance de l'algorithme de sélection proposé, nous avons utilisé le *graphe de fitness* [Floreano and Mattiussi, 2008], qui fournit des informations sur les performances de l'algorithme sur plusieurs exécutions en utilisant la même configuration de paramètres ajustables et des populations initiales différentes (générées aléatoirement). Pour qu'un algorithme soit efficace, il est souhaitable que la valeur de fitness moyenne des individus dans la population ainsi que la meilleure valeur de fitness varient peu au cours de plusieurs exécutions de l'algorithme. Dans

les Figures 7.2, 7.3, et 7.4 nous dressons les *graphes de fitness* pour les trois scénarios considérés en utilisant la configuration optimale des paramètres ajustables.

En analysant les Figures 7.2, 7.3, et 7.4, nous pouvons remarquer que le nombre de clusters ainsi que leur taille (c'est-à-dire le nombre de services contenus dans chaque cluster) ont une influence sur le processus de convergence. Par exemple, dans le scénario A qui contient 09 clusters, l'algorithme converge après 60 itérations. Cependant, pour le Scénario B qui contient 10 clusters, l'algorithme converge après 70 itérations. Enfin pour le Scénario C qui contient 11 clusters l'algorithme convergera après 75 itérations. On peut également remarquer que la variation de la moyenne et de la meilleure valeur de fitness est faible lorsqu'on exécute l'algorithme plusieurs fois en utilisant la même configuration des paramètres pour chaque scénario. Cela prouve l'efficacité de notre méthode de sélection pour identifier la solution de composition quasi-optimale dans un problème de composition de services web sémantiques sensible aux QoS.

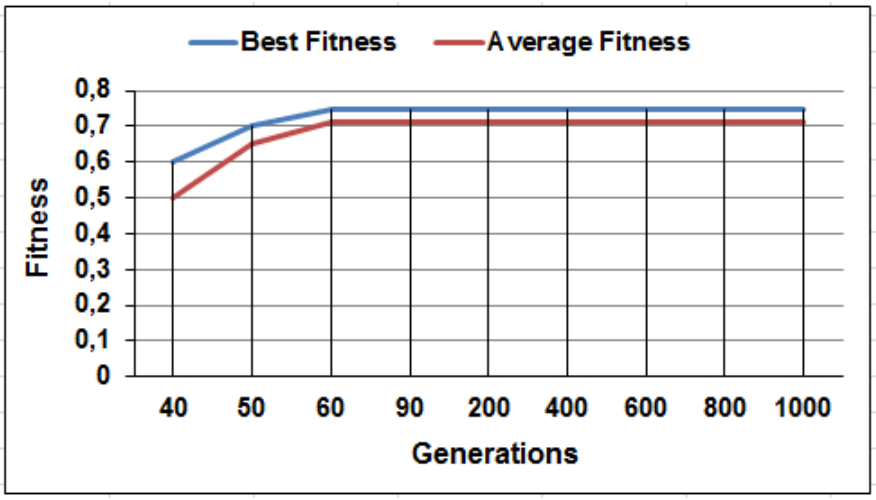

Figure 7.2: *Fitness vs. Générations (Scenario A)*

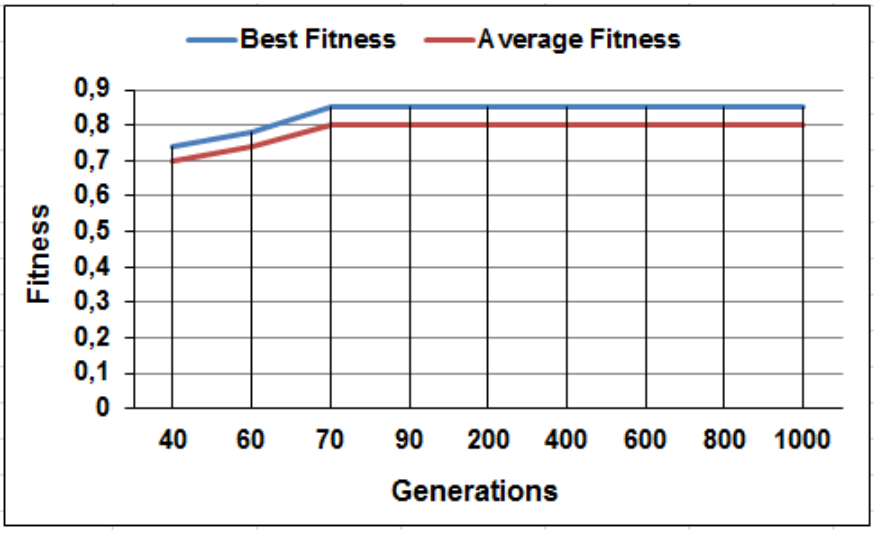

Figure 7.3: *Fitness vs. Générations (Scenario B)*

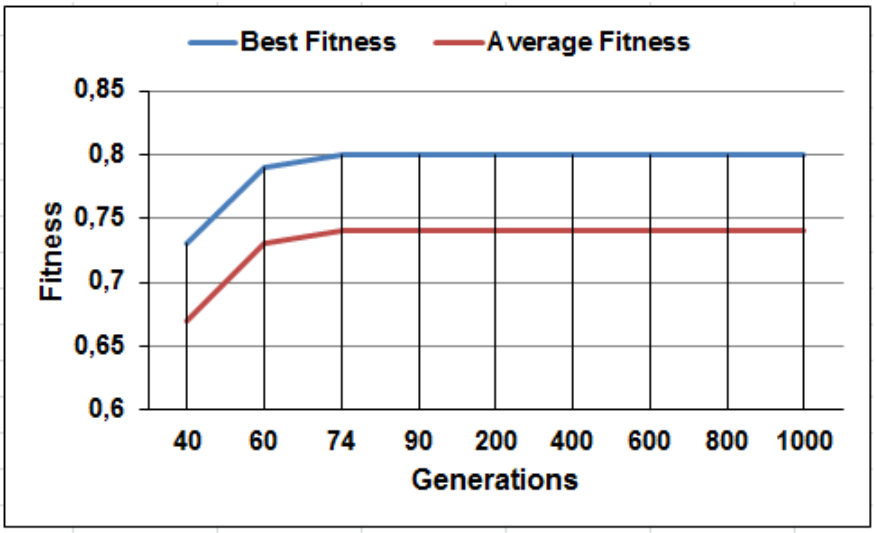

Figure *7.4:* *Fitness vs. Générations (Scenario C)*

7.4.2 Expérimentation pour QR2

Pour évaluer l'effet des contraintes globales de QoS sur la qualité des solutions de composition, nous avons mené deux expérimentations. Dans la première expérimentation, nous avons dressé le *graphe de fitness*, en premier lieu sans prendre en considération les contraintes, ensuite nous les avons intégrées dans le processus de sélection. Les Figures 7.5, 7.6, and 7.7 illustrent les résultats obtenus dans les trois scénarios envisagés en utilisant la même configuration optimale que dans la section précédente. Nous pouvons remarquer que la performance de l'algorithme HS en termes d'optimalité sans tenir en compte les contraintes globales est meilleure que sa performance avec contraintes (la meilleure valeur de fitness dépasse 0.9 dans le scénario B).

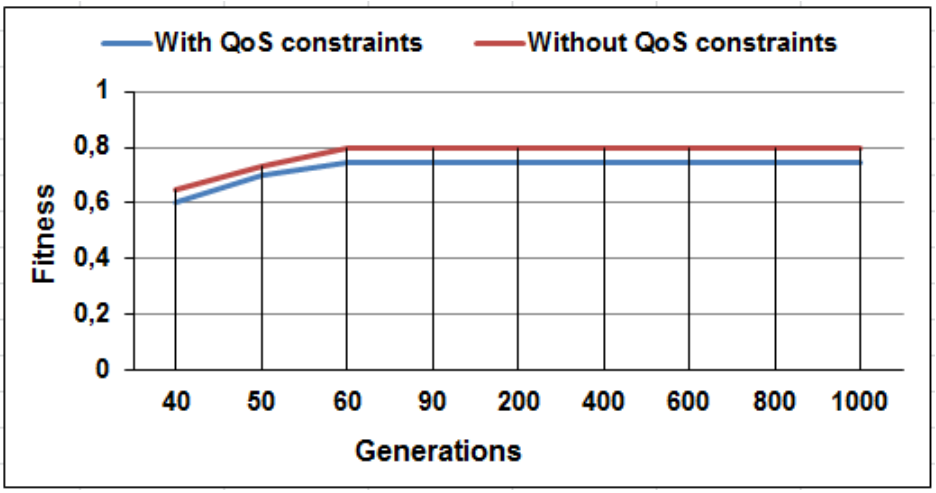

Figure *7.5:* *Fitness vs. Générations (Scenario A)*

Dans la deuxième expérimentation, nous avons évalué les performances de notre méthode de sélection par rapport au nombre de contraintes de QoS. Pour ce faire, nous avons fait varier le nombre de contraintes de 1 à 5 (rappelons que le nombre d'attributs

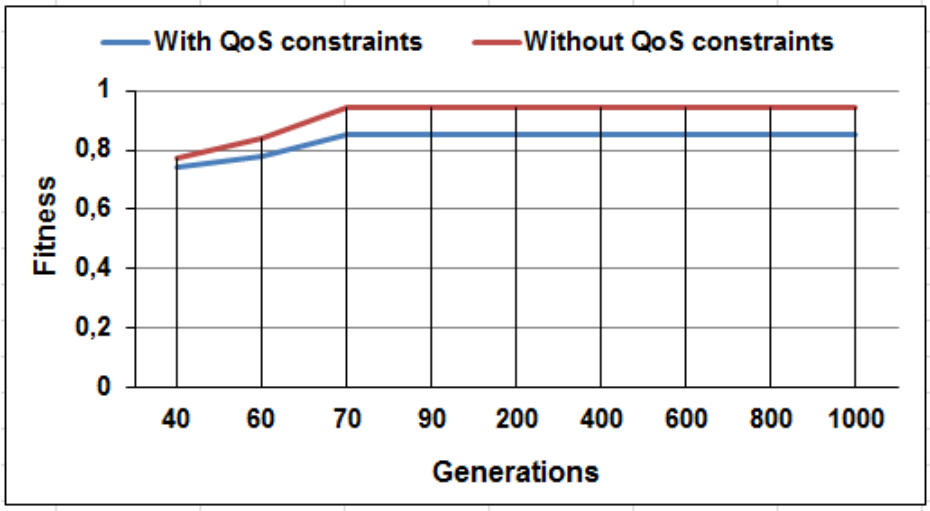

Figure *7.6:* *Fitness vs. Générations (Scenario B)*

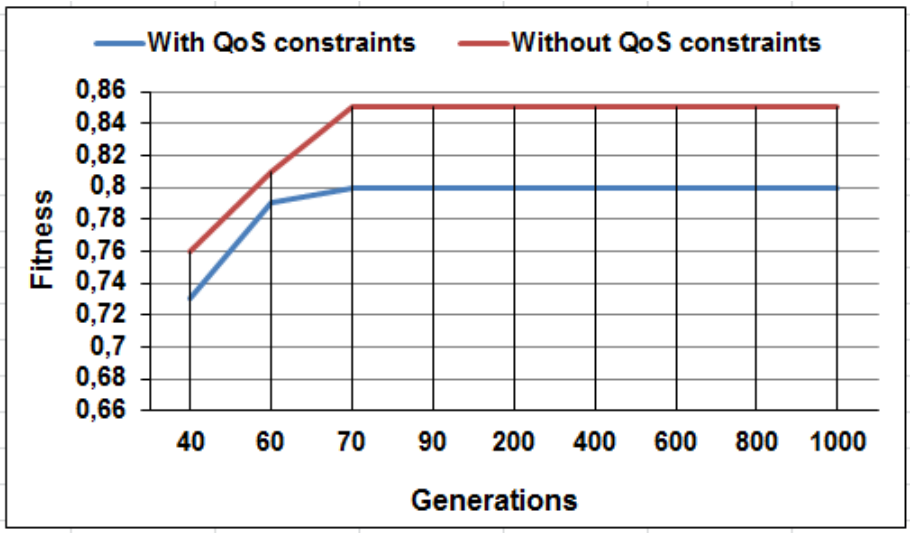

Figure *7.7:* *Fitness vs. Générations (Scenario C)*

QoS considérés dans notre travail est 5). Les résultats de cette expérience sont montrés dans la Figure 7.8. Dans les trois scénarios, nous observons que la deuxième contrainte QoS ajoute un peu de coût par rapport à la première contrainte QoS. Cependant, le coût de la troisième, la quatrième et la cinquième contrainte de QoS augmenteront progressivement par rapport au coût de calcul précédent. En résumé, nous concluons que le temps d'exécution pour trouver la solution de composition optimale augmente linéairement avec le nombre de contraintes dans les trois scénarios. Les résultats obtenus dans les deux expérimentations ont démontré que le problème de trouver la composition qui optimise la valeur de QoS globale tout en satisfaisant plusieurs contraintes de QoS est beaucoup plus difficile que d'optimiser juste la valeur de QoS globale, en fait le problème est NP-difficile. Toute solution exacte à ce problème a une complexité de calcul exponentielle. C'est la raison pour laquelle nous avons proposé une méthode d'optimisation basée sur un algorithme méta-heuristique.

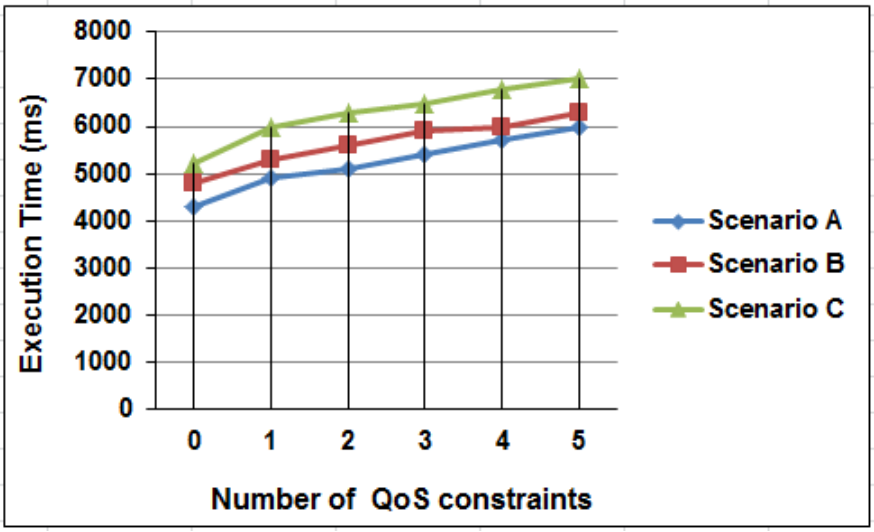

Figure *7.8:* *Temps d'exécution vs. Nombre de contraintes de QoS*

7.4.3 Expérimentation pour QR3

Afin de répondre à la troisième question de recherche, nous avons comparé l'algorithme HS avec ses deux variantes IHS (Improved HS) [Mahdavi et al., 2007] et GHS (Global Best HS) [Omran and Mahdavi, 2008]. L'algorithme IHS est un nouvel algorithme de recherche d'harmonie proposé en 2007 par Mahdavi et al. [Mahdavi et al., 2007]. Cet algorithme applique une méthode pour générer de nouveaux vecteurs de solution basée sur l'ajustement dynamique des paramètres *PAR* et *HMCR* (au lieu de considérer leurs valeurs fixes comme dans l'algorithme HS original). Et cela dans le but d'améliorer la performance de l'algorithme HS en accélérant le processus de convergence. Les paramètres *PAR* et *HMCR* changent dynamiquement avec le nombre de générations. Ils sont calculés en utilisant les formules 7.1 et 7.2.

$$HMCR(t) = HMCR_{max} - \frac{HMCR_{max} - HMCR_{min}}{T_{max}} * t \tag{7.1}$$

$$PAR(t) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{T_{max}} * t \tag{7.2}$$

De plus, l'algorithme GHS [Omran and Mahdavi, 2008] a été inspiré par l'algorithme PSO (Particule Swarm Optimization) [Kennedy, 2011], les auteurs reprennent les mêmes étapes que dans l'algorithme IHS à l'exception que dans la phase de l'improvisation de la nouvelle harmonie, ils sélectionnent la meilleure harmonie (en termes de fitness) dans la mémoire d'harmonie (*HM*).

Les trois algorithmes de sélection ont été évalués de manière comparative. Nous avons comparé l'algorithme HS avec ces deux variantes IHS et GHS selon les critères suivants : valeur de fitness optimale moyenne (*Fitness*), temps d'exécution moyen(*Temps d'exécution*) et déviation moyenne (*Déviation*). Pour les algorithmes IHS et GHS, nous avons effectué la même procédure dans la section 7.4.1 pour établir les valeurs optimales des paramètres ajustables dans les trois scénarios considérés. Les résultats expérimentaux obtenus par les versions de l'algorithme HS sont présentés respectivement dans le Tableau 7.6 et Figure 7.9 pour le scénario A, dans le Tableau

Tableau 7.6: *Analyse Comparative (Scenario A)*

Algorithme	**Fitness**	**Temps d'exécution**	**Déviation**
HS (Harmony Search)	0.745	6000	0.146
IHS (Improved Harmony Search)	0.80	5900	0.091
GHS (Global Best Harmony Search)	0.87	7500	0.021

Tableau 7.7: *Analyse Comparative (Scenario B)*

Algorithme	**Fitness**	**Temps d'exécution**	**Déviation**
HS (Harmony Search)	0.850	6300	0.10
IHS (Improved Harmony Search)	0.90	6000	0.05
GHS (Global Best Harmony Search)	0.947	8000	0.03

7.7 et Figure 7.10 pour le scénario B, et enfin dans le tableau 7.8 et la Figure 7.11 pour le scénario C.

Concernant le scénario A, on remarque que les trois variantes de l'algorithme HS ont presque la même durée de convergence. L'algorithme GHS est plus efficace que l'algorithme IHS et l'algorithme IHS est plus efficace que l'algorithme HS (en termes de convergence). En revanche, dans le scénario B, nous observons différentes durées de convergence. L'algorithme GHS a la plus longue durée de convergence (90 générations) mais il assure la meilleure valeur de fitness (0,947), par contre IHS a la plus petite durée de convergence (60 générations), mais il donne le deuxième valeur de fitness 0.9. Enfin HS présente un temps de convergence moyen (70 générations), mais il donne la plus petite valeur de fitness qui est égale à 0,85. Le scénario C confirme les mêmes constatations du scénario B, c'est-à-dire que l'algorithme IHS a la plus petite durée de convergence et que le GHS a la plus longue durée de convergence, mais qu'il assure la meilleure valeur de fitness. L'algorithme HS reste en troisième position. Nous remarquons également que la valeur de fitness la plus élevée obtenue

Tableau 7.8: *Analyse Comparative (Scenario C)*

Algorithme	**Fitness**	**Temps d'exécution**	**Déviation**
HS (Harmony Search)	0.800	7000	0.12
IHS (Improved Harmony Search)	0.89	6500	0.03
GHS (Global Best Harmony Search)	0.919	9000	0.01

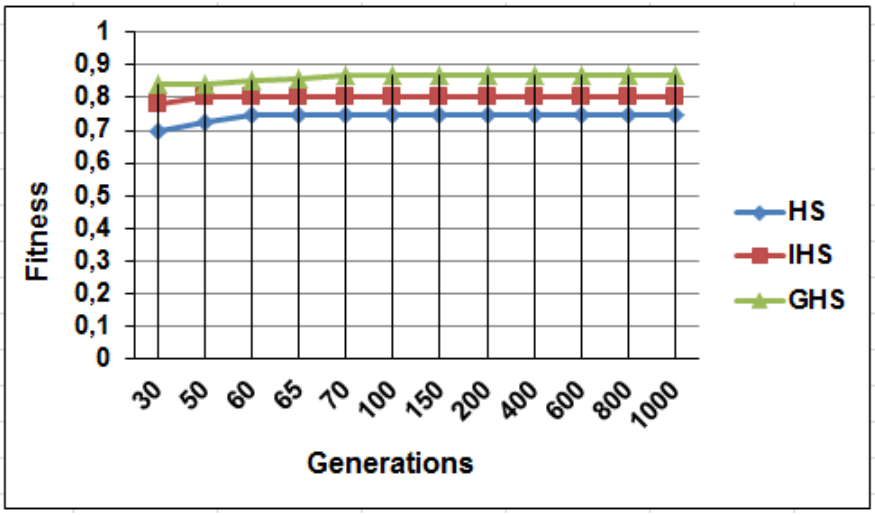

Figure *7.9:* *Fitness vs. Générations (Scenario A)*

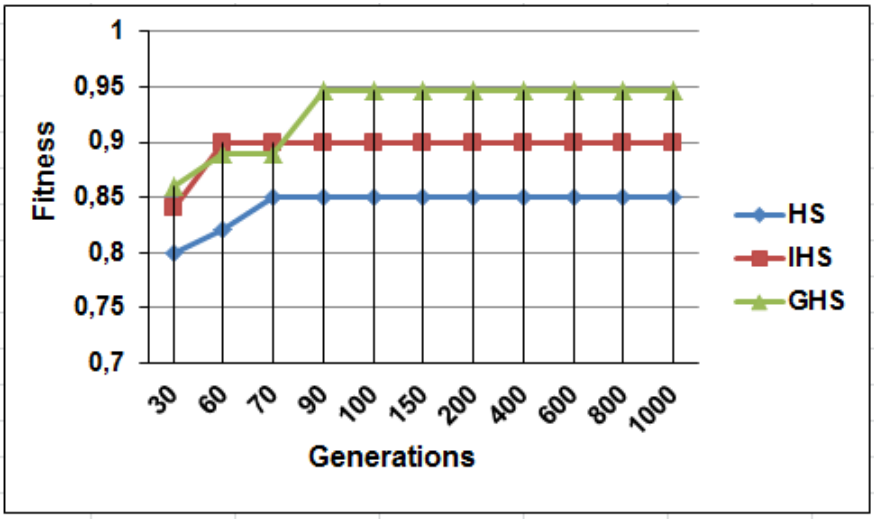

Figure *7.10:* *Fitness vs. Générations (Scenario B)*

dans ce scénario (0,91) est inférieure à la valeur de fitness la plus élevée obtenue dans le scénario B (0,947).

En résumé, nous pouvons dire que les deux variantes IHS et GHS ont apporté des améliorations significatives en termes de fitness et de temps d'exécution par rapport à la version originale de l'algorithme. Considérant les trois scénarios, nous pouvons conclure que l'algorithme GHS est plus efficace que les deux autres algorithmes en termes d'optimalité, mais en termes de temps il prend un temps additionnel minime pour converger (9 secondes dans le scénario C). Cela peut s'expliquer par le fait que le GHS prend un peu plus de temps pour rechercher la meilleure harmonie dans le processus d'improvisation. En revanche, les valeurs de temps d'exécution de IHS sont meilleures que celles de HS et GHS (moins de 7 secondes dans les trois scénarios), ce qui implique que IHS fournit une meilleure convergence que les algorithmes HS et GHS. Cela est dû au mécanisme d'ajustement dynamique des paramètres de l'algorithme qui est bénéfique pour échapper aux solutions optimales locales et améliorer le processus de convergence.

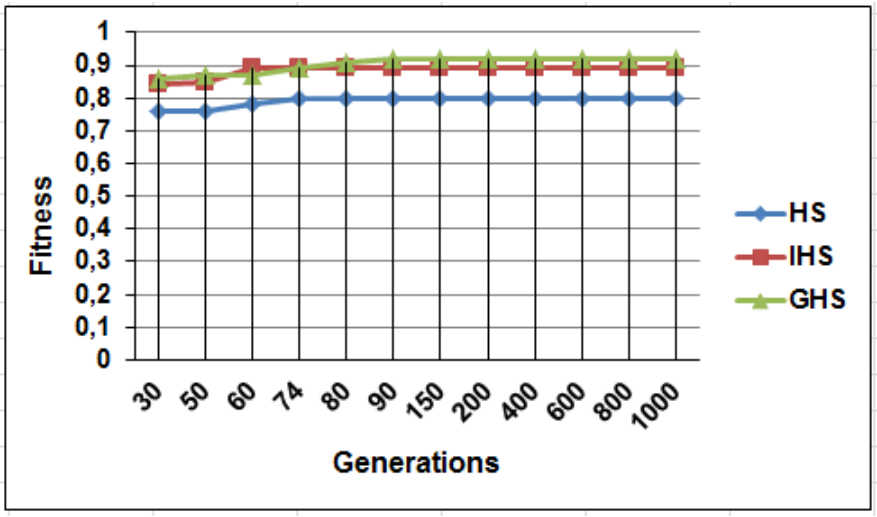

Figure *7.11:* *Fitness vs. Générations (Scenario C)*

7.5 Menaces à la validité (Threats To Validity)

Les résultats obtenus à travers cette étude sont assez concluants. Cependant, il y a un certain nombre de considérations qu'il faudra évoquer en ce qui concerne ses limitations et sa validité. On distingue généralement 3 types de validité : la validité de construction *(construct validity)*, la validité interne *(internal validity)*, et la validité externe *(external validity)* [de Oliveira Barros and Dias-Neto, 2011], [Feldt and Magazinius, 2010].

- *Validité de construction :* c'est la justesse avec laquelle une mesure ou une manipulation mène à des résultats `CONFORMES AU CONCEPT` postulé dans l'hypothèse (en d'autres termes c'est la concordance entre un construit théorique et sa définition opérationnelle, et éventuellement le besoin réel de l'utilisateur) [Feldt and Magazinius, 2010].
- *Validité interne :* c'est le degré auquel les `FLUCTUATIONS` d'une variable dépendante (VD : qui a pour but d'expliquer ou de prédire) peuvent être `ATTRIBUÉES` exclusivement à l'effet de la variable indépendante (VI : variable expérimentale). Donc c'est le degré de `CONFIANCE` avec lequel on peut inférer un effet `CAUSAL` de la VI sur la VD [Feldt and Magazinius, 2010].
- *Validité externe :* c'est le degré auquel les résultats d'une étude peuvent être `généralisés` à des individus, des situations ou des procédés au-delà de l'étude [Feldt and Magazinius, 2010].

7.5.1 Validité de construction

Pour évaluer notre approche de composition nous avons utilisé le temps d'éxecution comme critère de coût et une fonction objectif constituée de trois termes (la compatibilité sémantique, la performance en QoS agrégée, et la satisfaction de contraintes non-fonctionnelles) comme fonction de performance. Puisque l'utilisateur final exige en général moins de 10 minutes pour avoir la solution quasi-optimale, le critère de temps d'exécution est utile pour qualifier toute approche de composition.

Concernant le critère de performance, nous remarquons que la validité pratique d'une composition dépend de 3 facteurs :

En premier lieu, il faut que le flux de données entre les composants de la solution soit correct et ceci est garanti par le premier terme (aspect sémantique) de la fonction objectif. En deuxième lieu, l'utilisateur préfère en général une composition ayant de bonnes valeurs de QoS et ceci est pris en compte dans le deuxième terme qui représente l'aspect non fonctionnel. Enfin la faisabilité d'une composition est conditionnée par sa satisfaction des contraintes globales et pour cela nous avons introduit le troisième terme dans la fonction objectif.

Concernant les pondérations relatives aux critères de QoS, nous avons pris des valeurs équitables pour les raisons suivantes :

- L'utilisateur final ne précise pas en général des préférences entre les attributs de QoS et le contexte d'utilisation, par conséquent le scénario d'équitabilité est la solution la plus favorable.
- L'adoption d'une optimisation multi-objectif (ou au sens de Pareto) peut engendrer une explosion combinatoire lorsque les critères de QoS sont nombreux (curse of dimensionality). Ceci peut provoquer la violation des contraintes temps réel de l'exécution du prototype de composition (on ne peut plus parcourir toutes les compositions skyline puisque leur nombre est grand et le temps d'exécution ne respecte plus le délai).

7.5.2 Validité interne

Concernant le paramètre *HMCR* de l'algorithme Harmony Search, nous constatons qu'une valeur inférieure à 0.70 donne un comportement presque aléatoire à notre recherche et ceci nous donne des solutions non-faisables parce que les contraintes globales sont susceptibles d'être violées. En plus l'aspect sémantique de la solution sera médiocre puisque la chance de trouver deux concepts pris aléatoirement dans l'ontologie et qui ont un lien sémantique est très faible.

D'autre part, une valeur supérieure à 0.95 donne plus de chance à l'algorithme d'être pris par un optimum local et par la suite nous ne pouvons explorer le reste de l'espace de recherche (puisque les nouvelles solutions sont quasiment contrôlées par les harmonies de la mémoire).

Pour le paramètre d'ajustement *PAR*, nous remarquons que les meilleures performances sont données par l'intervalle $\{0.01, 0.30\}$, puisque la recherche locale exige que les voisins soient partiellement similaires et ceci ne peut être fait avec un taux d'ajustement élevé. En ce qui concerne la taille de la mémoire *HMS*, nous avons remarqué que les valeurs 300, 400 engendrent une solution quasi-optimale, et cela pour les trois scénarios d'expérimentations (A, B et C).

7.5.3 Validité externe

Plusieurs paramètres peuvent étendus pour la généralisation des résultats de notre prototype de composition sur de nouvelles collections de données :

- Le nombre de QoS : nous avons pris 5 paramètres de QoS (la fiabilité, la disponibilité, la réputation en plus du coût et temps d'exécution qui étaient prédéfinis dans la collection de données). Bien que d'autres collections de données ("QWS Dataset" [Al-Masri and Mahmoud, 2008]) utilisent 11 attributs de QoS, nous

remarquons qu'elles sont peu exploitables (c'est-à-dire qu'elles ont des valeurs de QoS non à jour ou manquantes ou même des services non disponibles). En plus la taille de ces collections est relativement faible (ex. la taille de "QWS Dataset" est 365 services)

- Le nombre de services : bien que nous avons utilisé 500 à 4000 instances de services, nous remarquons que ces valeurs ne peuvent confirmer le passage à l'échelle de notre modèle puisque les moteurs de recherche actuels de services (tels que Programmable Web) comptent de 40.000 services et de ce fait, nous envisageons de tester notre approche sur des instances plus consistantes.
- La taille de l'ontologie : nous avons utilisé une ontologie synthétique qui comporte 5000 concepts et ceci n'est pas suffisant pour confirmer l'efficacité de notre modèle, puisque des ontologies existantes possèdent plus de valeurs (par exemple l'ontologie SUMO [6] contient plus de 13000 concepts et WORDNET [7] compte plus de 175000 concepts. Par conséquent nous envisageons l'extension de ces ontologies pour évaluer davantage notre prototype de composition.
- L'utilisation des requêtes concrètes d'utilisateurs : en pratique c'est rare de trouver des services possédant un nombre d'entrées/sorties dépassant 6 ou 7 services. De même les compositions les plus rencontrées en pratique ne dépassent pas 5 ou 6 services (les services d'une base synthétique peuvent avoir plus de 10 entrées/sorties). Par conséquent, il sera plus facile d'explorer l'espace de recherche avec des requêtes concrètes puisque la probabilité de trouver 2 services réels et composables est plus grande que de trouver 2 services synthétiques et composables. En plus la taille des compositions à base de services synthétiques est en moyenne plus grande que la taille des compositions réelles.

7.6 Conclusion

Dans ce chapitre, nous avons décrit la mise en œuvre de l'approche de composition proposée. Nous avons présenté l'environnement de composition des services qui est constitué de trois modules : le module de génération de graphe de planification amélioré, le module de matching sémantique et le module de sélection à base de QoS. Nous avons également présenté les outils et langages ainsi que la base de services utilisés pour l'implémentation et la validation de notre contribution. Les résultats des expérimentations que nous avons réalisées montrent que notre approche de composition est capable de trouver une solution de composition proche de l'optimale qui répond aux besoins et préférences de l'utilisateur en terme de QoS avec un temps d'exécution acceptable. Ceci grâce à l'utilisation du matching sémantique entre services lors de génération de graphe de planification d'une part, et grâce à l'adoption de l'algorithme HS pour la sélection des solutions d'autre part. En fait, les points forts de cet algorithme sont ses opérateurs d'improvisation (considération de la mémoire, ajustement du ton (pitch) et considération aléatoire) qui jouent un rôle important dans la réalisation de l'équilibre entre les deux grands extrêmes pour tout algorithme d'optimisation, l'intensification et la diversification. Les comparaisons de cet algorithme avec ses deux variantes (IHS et GHS) montrent des performances plus

6. http ://www.adampease.org/OP/
7. https ://www.w3.org/2006/03/wn/wn20/

améliorées en termes d'optimalité et temps d'exécution. Dans le prochain chapitre, nous présenterons nos conclusions et perspectives.

Chapitre 8

Conclusion et perspectives

Le développement rapide des systèmes d'information distribués et la diffusion de l'accès à Internet ont entraîné le développement de nouveaux paradigmes d'interaction entre applications. Ainsi, les paradigmes à base de composants ont évolué vers le paradigme orienté service qui s'articule autour du concept du service Web. La caractéristique importante des services Web est leur propriété de composabilité qui permet de créer des services plus complexes en combinant des services plus élémentaires en tenant compte des besoins des utilisateurs. La composition de services impliquant la capacité de sélectionner, de coordonner, d'interagir, et de faire interopérer des services Web existants constitue une tâche complexe. Cette complexité est renforcée quand il s'agit d'intégrer dynamiquement des services à la demande, et de les composer automatiquement pour répondre à des exigences qui ne sont pas réalisées par les services existants. Durant cette thèse, nous avons porté notre attention à cette problématique. En effet, nous pensons qu'une approche pour la composition de services doit offrir le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et les préférences de l'utilisateur en termes de QoS. C'est dans ce contexte que s'inscrit notre travail de thèse.

8.1 Synthèse

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

Après la spécification du problème de la composition automatique des services web, nous avons élaboré un état de l'art détaillé survolant les différentes approches qui permettent sa résolution à savoir, le calcul de situation, la planification hiérarchique (HTN), le système PDDL, la planification basée sur les règles, la preuve par théorème automatique, le planificateur MBP, la planification à base de graphe, et la planification distribuée. La classification des travaux de l'état de l'art a permis d'une part de dégager les atouts et les limites de chaque approche, et d'autre part de mieux positionner notre approche de composition pour voir comment il serait possible d'améliorer les critères de performance.

Ensuite, nous avons analysé, modélisé et formulé le problème de sélection des services web composés à base de QoS (QoS-aware service composition). Nous avons présenté les approches d'optimisation dédiées à la résolution de ce problème. Elles sont groupées en 4 catégories qui sont : les approches de résolutions exactes, les heuristiques approximatives, les méta-heuristiques et les approches basées sur la dominance au sens de Pareto.

Par la suite nous avons présenté l'essentiel des travaux qui traitent la problématique "QoS-aware automatic service composition". Ensuite, nous avons comparé les différents travaux en termes de prise en compte de certaines exigences des environnements orientés services.

La deuxième partie est consacrée aux contributions où nous avons décrit notre proposition. Notre approche modélise le problème "QoS-aware Automatic Service Composition" comme un processus multi-couches qui crée une structure de graphe de planification. Le graphe de planification est l'une des techniques de planification en intelligence artificielle qui sont considérées comme les plus performantes pour résoudre le problème de la composition de service web. Il représente l'espace de toutes les solutions de composition candidates, pour une requête donnée. Ensuite, nous décrivons notre algorithme Harmony Search (HS) adopté pour trouver la composition optimale ou quasi-optimale. L'espace de recherche comprend l'ensemble des solutions de composition de service candidates générées par le graphe planification. Une fonction de fitness est proposée pour évaluer l'ensemble des solutions candidates prenant en considération la qualité sémantique ainsi que les critères QoS des services web. L'ensemble des expérimentations menées montre l'efficacité de l'approche proposée en termes d'optimalité et temps d'exécution.

8.2 Perspectives

Par ailleurs, nous pensons que notre travail ouvre d'autres voies de recherche. Plusieurs extensions peuvent être proposées. Ces perspectives répondent à un objectif principal qui est l'amélioration de la composition des services web sémantiques. Les perspectives que nous exposons représentent des améliorations de l'approche de composition des services web proposée.

- Propriétés non-fonctionnelles complexes : la représentation des propriétés non fonctionnelles que nous avons adoptée est une représentation simple qui couvre un large éventail des propriétés non-fonctionnelles pouvant être exprimées sous forme de contraintes simples. Nous envisageons de travailler sur des représentations complexes.
- Enrichissement des descriptions sémantiques par les pré-conditions et les post-conditions : il y a un consensus sur le fait que la sémantique basée sur les inputs/outputs des services est nécessaire. Cependant, il n'est pas encore décidé que ce niveau de sémantisation est suffisant. Puisque le but de cette dernière est de capturer le comportement des services, des informations plus complexes sont requises. Il semble que pour la plupart des services et beaucoup de scénarios de composition réels, de bons résultats peuvent être obtenus si les pré-/post-conditions sont prises en compte pour mieux exprimer les fonctionnalités des services et rendre ainsi plus intelligent le schéma de *matching*.
- Prendre en compte la pluralité des ontologies de domaine durant le processus de *matching* : cela nous permettra d'effectuer un *matching* entre deux concepts appartenant à des ontologies différentes puisque les services à composer sont souvent décrits par des ontologies différentes.
- Utilisation d'autres structures de contrôle : nous envisageons également d'étendre l'évaluation des performances de l'approche de composition de services proposée notamment en considérant d'autres structures de composition (les choix

conditionnels, les boucles, etc).

- Prise en compte du contexte : nous avons également l'intention d'enrichir les descriptions de services web avec les informations contextuelles qui sont cruciales dans les systèmes orienté services. En effet, les informations contextuelles (comme par exemple, l'état émotionnel de l'utilisateur, son centre d'attention, sa localisation, son orientation, la date et le temps où il évolue, etc) jouent un rôle important dans la personnalisation de la composition des services pour fournir un résultat satisfaisant à l'utilisateur.
- Utilisation d'autres algorithmes d'optimisation : nous prévoyons d'utiliser des algorithmes d'optimisation mono-objectif récents comme l'optimisation des mauvaises herbes envahissantes (Invasive Weed optimization : IWO), l'optimisation basée sur la biogéographie (Biogeography Based Optimization : BBO) et les algorithmes inspirés de la chauve-souris (Bat-inspired Algorithms : BA). Ou encore utiliser des algorithmes d'optimisation multi-objectif comme MOGA (Multiple Objective Genetic Algorithm), NSGA (Non dominate Sorting Genetic Algorithm) et SPEA (Strength Pareto Evolutionary algorithm).
- Enfin, nous avons l'intention de tester notre approche de composition sur des scénarios plus complexes et avec des utilisateurs et ontologies concrètes.

Liste des publications

- Journaux internationaux avec comité de lecture

[1] Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine,C., Hadjila, F., and Merzoug, M. (2017). Qos-aware optimal and automated semantic web service composition with user's constraints. Service Oriented Computing and Applications, 11(2) :183–201.

- Conférences internationales avec comité de lecture

[2] Amina Bekkouche, Sidi Mohamed Benslimane, A Genetic Algorithm approach to QoS-based Semantic Web service Composition. International Conference on Information Systems and Technologies (ICIST'2013), March 22 - 24, 2013, Tangier, Morocco.

[3] Amina Bekkouche, Sidi Mohamed Benslimane, Fethallah Hadjila, Merzoug Mohamed, Towards Semantic Web Service Composition using Immune Algorithm. Colloque sur l'Optimisation et les systèmes d'Information (COSI'2013). 9 au 11 Jun 2013, Algiers, Algeria.

- Chapitre d'un livre

[4] Amina Bekkouche, Sidi Mohamed Benslimane, A Genetic Algorithm approach to QoS-based Semantic Web service Composition. Book chapter in Business Intelligence and Mobile Technology Research : An Information Engineering Perspective, edt. In Sean Eom and Mohamed Ridda Laouar (Cambridge Scholars Publishing) ISBN-10 :1-4438-5507-3, pp 142- 159, 2014.

Bibliographie

[Abowd et al., 1999] Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

[Agarwal et al., 2008] Agarwal, V., Chafle, G., Mittal, S., and Srivastava, B. (2008). Understanding approaches for web service composition and execution. In *Proceedings of the 1st Bangalore annual Compute conference*, page 1. ACM.

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

[Al-Helal and Gamble, 2014] Al-Helal, H. and Gamble, R. (2014). Introducing replaceability into web service composition. *IEEE Transactions on Services Computing*, 7(2) :198–209.

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

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

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

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

[Arch-int et al., 2017] Arch-int, N., Arch-int, S., Sonsilphong, S., and Wanchai, P. (2017). Graph-based semantic web service composition for healthcare data integration. *Journal of healthcare engineering*, 2017.

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

[Baccar et al., 2013] Baccar, S., Rouached, M., and Abid, M. (2013). A user requirements oriented semantic web services composition framework. In *Services (SERVICES), 203 IEEE Ninth World Congress on*, pages 333–340. IEEE.

[Bali, 2017] Bali, A. (2017). *Une approche basée agents pour l'adaptation des services web*. PhD thesis, Université Mohamed Khider-Biskra.

[Barreiro Claro, 2006] Barreiro Claro, D. (2006). *Spoc : un canevas pour la composition automatique de services web dédiés à la réalisation de devis*. PhD thesis, Angers.

[Barros et al., 2005] Barros, A., Dumas, M., and Oaks, P. (2005). Standards for web service choreography and orchestration : Status and perspectives. In *International Conference on Business Process Management*, pages 61–74. Springer.

[Bartalos, 2011] Bartalos, P. (2011). Effective automatic dynamic semantic web service composition. *Inf. Sci. and Technol. Bulletin ACM Slovakia*, 3(1) :61–72.

[Bartalos and Bieliková, 2009] Bartalos, P. and Bieliková, M. (2009). Semantic web service composition framework based on parallel processing. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 495–498. IEEE.

[Bekdaş and Nigdeli, 2011] Bekdaş, G. and Nigdeli, S. M. (2011). Estimating optimum parameters of tuned mass dampers using harmony search. *Engineering Structures*, 33(9) :2716–2723.

[Bekkouche et al., 2017] Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine, C., Hadjila, F., and Merzoug, M. (2017). Qos-aware optimal and automated semantic web service composition with user's constraints. *Service Oriented Computing and Applications*, 11(2) :183–201.

[Bekkouche and Fizazi, 2016] Bekkouche, I. and Fizazi, H. (2016). A new image clustering method based on the fuzzy harmony search algorithm and fourier transform. *Journal of Information Processing Systems*, 12(4).

[Bellwood et al., 2002] Bellwood, T., Capell, S., Clement, L., Colgrave, J., Dovey, M., Feygin, D., Kochman, A., Macias, P., Novotny, M., Paolucci, M., et al. (2002). Universal description, discovery and integration specification (uddi) 3.0. *Online : http ://uddi. org/pubs/uddi-v3. 00-published-20020719. htm*, 10.

[Belmadani et al., 2009] Belmadani, A., Benasla, L., and Rahli, M. (2009). Etude d'un dispatching economique environnemental par la méthode harmony search. *Acta Electrotehnica*, 50(1) :44–48.

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

[Bertoli et al., 2001] Bertoli, P., Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2001). Mbp : a model based planner. In *Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*.

[Blum and Furst, 1997] Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial intelligence*, 90(1) :281–300.

[Bourdon, 2007] Bourdon, J. (2007). Multi-agent systems for the automatic composition of semantic web services in dynamic environments. *Rapport de master, École des Mines de Saint Etienne-G2I & Université Joseph Fourier*.

[Bucchiarone and Gnesi, 2006] Bucchiarone, A. and Gnesi, S. (2006). A survey on services composition languages and models. In *International Workshop on Web Services–Modeling and Testing (WS-MaTe 2006)*, page 51.

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

[Casati and Shan, 2002] Casati, F. and Shan, M.-C. (2002). Event-based interaction management for composite e-services in eflow. *Information Systems Frontiers*, 4(1) :19–31.

[Cauvet and Guzelian, 2008] Cauvet, C. and Guzelian, G. (2008). Business process modeling : A service-oriented approach. In *Hawaii international conference on system sciences, proceedings of the 41st annual*, pages 98–98. IEEE.

[Černỳ, 1985] Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1) :41–51.

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

[Chakraborty et al., 2009] Chakraborty, P., Roy, G. G., Das, S., Jain, D., and Abraham, A. (2009). An improved harmony search algorithm with differential mutation operator. *Fundamenta Informaticae*, 95(4) :401–426.

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

[Charif and Sabouret, 2007] Charif, Y. and Sabouret, N. (2007). Coordination in introspective multi-agent systems. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 412–415. IEEE Computer Society.

[Chen, 2015] Chen, M. (2015). *QoS-aware Service Composition and Redundant Service Removal*. PhD thesis, Concordia University.

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

[Chen and Yan, 2014] Chen, M. and Yan, Y. (2014). Qos-aware service composition over graphplan through graph reachability. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 544–551. IEEE.

[Chen et al., 2016] Chen, N., Cardozo, N., and Clarke, S. (2016). Goal-driven service composition in mobile and pervasive computing. *IEEE Transactions on Services Computing*.

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

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

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

[Chifu et al., 2015] Chifu, V. R., Salomie, I., Pop, C. B., Niculici, A. N., and Suia, D. S. (2015). Exploring the selection of the optimal web service composition through ant colony optimization. *Computing and Informatics*, 33(5) :1047–1064.

[Chinnici et al., 2007] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C recommendation*, 26 :19.

[Chollet, 2009] Chollet, S. (2009). *Orchestration de services hétérogènes et sécurisés*. PhD thesis, Université Joseph-Fourier-Grenoble I.

[Claro et al., 2005] Claro, D. B., Albers, P., and Hao, J.-K. (2005). Approaches of web services composition. In *International Conference on Enterprise Information Systems*, pages 208–213.

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

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms second edition.

[Dal Lago et al., 2002] Dal Lago, U., Pistore, M., and Traverso, P. (2002). Planning with a language for extended goals. In *AAAI/IAAI*, pages 447–454.

[de Oliveira Barros and Dias-Neto, 2011] de Oliveira Barros, M. and Dias-Neto, A. C. (2011). 0006/2011-threats to validity in search-based software engineering empirical studies. *RelaTe-DIA*, 5(1).

[Deng et al., 2014] Deng, S., Huang, L., Tan, W., and Wu, Z. (2014). Top-k automatic service composition : A parallel method for large-scale service sets. *IEEE Transactions on Automation Science and Engineering*, 11(3) :891–905.

[Deng et al., 2013] Deng, S., Wu, B., Yin, J., and Wu, Z. (2013). Efficient planning for top-k web service composition. *Knowledge and information systems*, 36(3) :579–605.

[Do Prado et al., 2013] Do Prado, P. F., Nakamura, L. H., Estrella, J., Santana, M. J., and Santana, R. H. (2013). A performance evaluation study for qos-aware web services composition using heuristic algorithms. In *ICDS 2013, The Seventh International Conference on Digital Society*, pages 53–58.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4) :28–39.

[Dustdar and Papazoglou, 2008] Dustdar, S. and Papazoglou, M. P. (2008). Services and service composition–an introduction (services und service komposition–eine einführung). *IT-Information Technology*, 50(2) :86–92.

[Dustdar and Schreiner, 2005] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *International journal of web and grid services*, 1(1) :1–30.

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

[El Falou, 2010] El Falou, M. (2010). *Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multi-agents*. PhD thesis, Université de Caen.

[Elsayed et al., 2017] Elsayed, D. H., Nasr, E. S., Alaa El Din, M., and Gheith, M. H. (2017). A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 537–546. Springer.

[Erl, 2008] Erl, T. (2008). *Soa : principles of service design*, volume 1. Prentice Hall Upper Saddle River.

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

[Feldt and Magazinius, 2010] Feldt, R. and Magazinius, A. (2010). Validity threats in empirical software engineering research-an initial survey. In *SEKE*, pages 374–379.

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

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

[Fielding and Taylor, 2000] Fielding, R. T. and Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation.

[Fki, 2015] Fki, E. (2015). *Sélection et composition flexible basée services abstraits pour une meilleure adaptation aux intentions des utilisateurs*. PhD thesis, Univerisité Toulouse 1 Capitole.

[Floreano and Mattiussi, 2008] Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence : theories, methods, and technologies*. MIT press.

[Gabrel et al., 2018] Gabrel, V., Manouvrier, M., Moreau, K., and Murat, C. (2018). Qos-aware automatic syntactic service composition problem : complexity and resolution. *Future Generation Computer Systems*, 80 :311–321.

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

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

[Gardarin et al., 1993] Gardarin, G., Gardarin, G., Gardarin, G., Gardarin, G., and Informaticien, F. (1993). *Maîtriser les bases de données : modèles et langages*. Eyrolles.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). A guide to the theory of np-completeness. *WH Freemann, New York*, 70.

[Geebelen et al., 2014] Geebelen, D., Geebelen, K., Truyen, E., Michiels, S., Suykens, J. A., Vandewalle, J., and Joosen, W. (2014). Qos prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset. *Information Sciences*, 268 :397–424.

[Geem, 2000] Geem, Z. W. (2000). *Optimal design of water distribution networks using harmony search*. PhD thesis, Korea University.

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

[Geem, 2009] Geem, Z. W. (2009). Harmony search optimisation to the pump-included water distribution network design. *Civil Engineering and Environmental Systems*, 26(3) :211–221.

[Geem, 2010] Geem, Z. W. (2010). State-of-the-art in the structure of harmony search algorithm. In *Recent advances in harmony search algorithm*, pages 1–10. Springer.

[Geem and Choi, 2007] Geem, Z. W. and Choi, J.-Y. (2007). Music composition using harmony search algorithm. In *Workshops on Applications of Evolutionary Computation*, pages 593–600. Springer.

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

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning : theory and practice*. Elsevier.

[Gharzouli, 2011] Gharzouli, M. (2011). *Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer*. PhD thesis, Doctoral Thesis. University of Mentouri, Constantine, Algeria.

[Glover, 1989] Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3) :190–206.

[Gruber, 1995] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing ? *International journal of human-computer studies*, 43(5-6) :907–928.

[Gustavo et al., 2004] Gustavo, A., Casati, F., Kuno, H., and Machiraju, V. (2004). Web services : concepts, architectures and applications.

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

[Han et al., 2009] Han, Y., Wang, J., and Zhang, P. (2009). Business-oriented service modeling : A case study. *Simulation Modelling Practice and Theory*, 17(8) :1413–1429.

[Hatzi et al., 2012] Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D., and Vlahavas, I. (2012). An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing*, 5(3) :319–332.

[He and Chang, 2003] He, B. and Chang, K. C.-C. (2003). Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 217–228. ACM.

[Helin et al., 2005] Helin, H., Klusch, M., Lopes, A., Fernández, A., Schumacher, M., Schuldt, H., Bergenti, F., and Kinnunen, A. (2005). Context-aware business application service co-ordination in mobile computing environments. In *AAMAS05 workshop on Ambient Intelligence-Agents for Ubiquitous Computing*.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P. F., and Van Harmelen, F. (2003). From shiq and rdf to owl : The making of a web ontology language. *Web semantics : science, services and agents on the World Wide Web*, 1(1) :7–26.

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

[Hunter and McLaughlin, 2007] Hunter, J. and McLaughlin, B. (2007). Jdom v1. 1 api specification. *Retrieved October*, 11 :2008.

[Hwang et al., 2015] Hwang, S.-Y., Hsu, C.-C., and Lee, C.-H. (2015). Service selection for web services with probabilistic qos. *IEEE transactions on services computing*, 8(3) :467–480.

[Hwang et al., 2007] Hwang, S.-Y., Wang, H., Tang, J., and Srivastava, J. (2007). A probabilistic approach to modeling and estimating the qos of web-services-based workflows. *Information Sciences*, 177(23) :5484–5503.

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

[Jatoth et al., 2017] Jatoth, C., Gangadharan, G., and Buyya, R. (2017). Computational intelligence based qos-aware web service composition : A systematic literature review. *IEEE Transactions on Services Computing*, 10(3) :475–492.

[Jiang et al., 2014] Jiang, W., Hu, S., and Liu, Z. (2014). Top k query for qos-aware automatic service composition. *IEEE Transactions on Services Computing*, 7(4) :681–695.

[Jiang et al., 2010] Jiang, W., Zhang, C., Huang, Z., Chen, M., Hu, S., and Liu, Z. (2010). Qsynth : A tool for qos-aware automatic service composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 42–49. IEEE.

[Jin et al., 2014] Jin, X., Chun, S., Jung, J., and Lee, K.-H. (2014). Iot service selection based on physical service model and absolute dominance relationship. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pages 65–72. IEEE.

[Juric et al., 2006] Juric, M. B., Mathew, B., and Sarang, P. G. (2006). *Business process execution language for web services : an architect and developer's guide to orchestrating web services using BPEL4WS*. Packt Publishing Ltd.

[Kavantzas et al., 2005] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., and Barreto, C. (2005). Web services choreography description language version 1.0. *W3C candidate recommendation*, 9 :290–313.

[Kellert and Toumani, 2003] Kellert, P. and Toumani, F. (2003). Les web services sémantiques. *Web sémantique, Action spéci_que*, 32.

[Kennedy, 2011] Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer.

[Khader et al., 2014] Khader, A. T., Abusnaina, A. A., Shambour, Q., et al. (2014). Modified tournament harmony search for unconstrained optimisation problems. In *Recent Advances on Soft Computing and Data Mining*, pages 283–292. Springer.

[Khan et al., 2002] Khan, S., Li, K. F., Manning, E. G., and Akbar, M. M. (2002). Solving the knapsack problem for adaptive multimedia systems. *Stud. Inform. Univ.*, 2(1) :157–178.

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

[Kil, 2010] Kil, H. (2010). Efficient web service composition : from signature-level to behavioral description-level.

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

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

[Klusch, 2008] Klusch, M. (2008). Semantic web service coordination. *CASCOM : Intelligent Service Coordination in the Semantic Web*, pages 59–104.

[Klusch et al., 2005] Klusch, M., Gerber, A., and Schmidt, M. (2005). Semantic web service composition planning with owls-xplan. In *Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*, pages 55–62.

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

[Klyne et al., 2004] Klyne, G., Carroll, J. J., and McBride, B. (2004). Resource description framework (rdf) : Concepts and abstract syntax. w3c recommendation, feb. 2004.

[Kona et al., 2008] Kona, S., Bansal, A., Blake, M. B., and Gupta, G. (2008). Generalized semantics-based service composition. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 219–227. IEEE.

[Kouicem et al., 2014] Kouicem, A., Chibani, A., Tari, A., Amirat, Y., and Tari, Z. (2014). Dynamic services selection approach for the composition of complex services in the web of objects. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 298–303. IEEE.

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

[Krithiga, 2012] Krithiga, R. (2012). Qos-aware web service selection using soma. *Global Journal of Computer Science and Technology*.

[Lamine et al., 2017] Lamine, R. B., Jemaa, R. B., and Amor, I. A. B. (2017). Graph planning based composition for adaptable semantic web services. *Procedia Computer Science*, 112 :358–368.

[Lämmermann, 2002] Lämmermann, S. (2002). *Runtime service composition via logic-based program synthesis*. PhD thesis, Mikroelektronik och informationsteknik.

[Lécué, 2008a] Lécué, F. (2008a). *Composition de Services Web : Une Approche basée Liens Sémantiques*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne.

[Lécué, 2008b] Lécué, F. (2008b). Web service composition : Semantic links based approach. *Ecole des Mines de Saint-Etienne, Saint-Etienne, France*.

[Lécué, 2009] Lécué, F. (2009). Optimizing qos-aware semantic web service composition. In *International Semantic Web Conference*, pages 375–391. Springer.

[Lecue and Mehandjiev, 2009] Lecue, F. and Mehandjiev, N. (2009). Towards scalability of quality driven semantic web service composition. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 469–476. IEEE.

[Lee and Geem, 2005] Lee, K. S. and Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization : harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194(36-38) :3902–3933.

[Lemos et al., 2012] Lemos, F., Abbaci, K., Grigori, D., Hadjali, A., Bouzeghoub, M., Liétard, L., and Rocacher, D. (2012). Intégration de préférences dans la découverte et la sélection des services web. *ISI*, 17(5).

[Li et al., 2011] Li, J., Chen, S., Li, Y., and Zhang, Q. (2011). Semantic web service automatic composition based on service parameter relationship graph. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 1773–1778. IEEE.

[Li et al., 2014] Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., and Chen, D.-r. (2014). An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, 269 :238–254.

[Li et al., 2010] Li, W., Dai, X., and Jiang, H. (2010). web services composition based on weighted planning graph. In *Networking and Distributed Computing (ICNDC), 2010 First International Conference on*, pages 89–93. IEEE.

[Liao et al., 2011] Liao, J., Liu, Y., Zhu, X., Xu, T., and Wang, J. (2011). Niching particle swarm optimization algorithm for service composition. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE.

[Liu et al., 2010] Liu, H., Zhong, F., Ouyang, B., and Wu, J. (2010). An approach for qos-aware web service composition based on improved genetic algorithm. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 1, pages 123–128. IEEE.

[Liu et al., 2012] Liu, M., Wang, M., Shen, W., Luo, N., and Yan, J. (2012). A quality of service (qos)-aware execution plan selection approach for a service composition process. *Future Generation Computer Systems*, 28(7) :1080–1089.

[Liu and Yin, 2009] Liu, X. and Yin, Z. (2009). Web service composition with global constraint based on discrete particle swarm optimization. In *Web Mining and Web-based Application, 2009. WMWA'09. Second Pacific-Asia Conference on*, pages 183–186. IEEE.

[Liu et al., 2011] Liu, Y., Miao, H., Li, Z., and Gao, H. (2011). Qos-aware web services composition based on hqpso algorithm. In *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, pages 400–405. IEEE.

[Llinás and Nagi, 2015] Llinás, G. A. G. and Nagi, R. (2015). Network and qos-based selection of complementary services. *IEEE Transactions on Services Computing*, 8(1) :79–91.

[Luo et al., 2011] Luo, Y.-s., Qi, Y., Hou, D., Shen, L.-f., Chen, Y., and Zhong, X. (2011). A novel heuristic algorithm for qos-aware end-to-end service composition. *Computer Communications*, 34(9) :1137–1144.

[MacKenzie et al., 2006] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12 :18.

[Mahdavi et al., 2007] Mahdavi, M., Fesanghary, M., and Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, 188(2) :1567–1579.

[Marconi and Pistore, 2009] Marconi, A. and Pistore, M. (2009). Synthesis and composition of web services. In *SFM*, volume 5569, pages 89–157. Springer.

[Martello and Toth, 1987] Martello, S. and Toth, P. (1987). Algorithms for knapsack problems. *North-Holland Mathematics Studies*, 132 :213–257.

[Martin et al., 2007] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with owl-s. *World Wide Web*, 10(3) :243–277.

[McBride et al., 2004] McBride, B., Boothby, D., and Dollin, C. (2004). An introduction to rdf and the jena rdf api. *Retrieved August*, 1 :2007.

[McDermott et al., 1998] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language.

[McDermott, 2002] McDermott, D. V. (2002). Estimated-regression planning for interactions with web services. In *AIPS*, volume 2, pages 204–211.

[McGuinness et al., 2004] McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10) :2004.

[McIlraith and Son, 2002] McIlraith, S. and Son, T. C. (2002). Adapting golog for composition of semantic web services. *KR*, 2 :482–493.

[McIlraith and Martin, 2003] McIlraith, S. A. and Martin, D. L. (2003). Bringing semantics to web services. *IEEE Intelligent systems*, 18(1) :90–93.

[Medjahed et al., 2003] Medjahed, B., Bouguettaya, A., and Elmagarmid, A. K. (2003). Composing web services on the semantic web. *The VLDB Journal—The International Journal on Very Large Data Bases*, 12(4) :333–351.

[Mehrabian and Lucas, 2006] Mehrabian, A. R. and Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological informatics*, 1(4) :355–366.

[Milanovic and Malek, 2004] Milanovic, N. and Malek, M. (2004). Current solutions for web service composition. *IEEE Internet Computing*, 8(6) :51–59.

[Mitra et al., 2003] Mitra, N., Lafon, Y., et al. (2003). Soap version 1.2 part 0 : Primer. *W3C recommendation*, 24 :12.

[Mohammed et al., 2014] Mohammed, M., Chikh, M. A., and Fethallah, H. (2014). Qos-aware web service selection based on harmony search. In *ISKO-Maghreb : Concepts and Tools for knowledge Management (ISKO-Maghreb), 2014 4th International Symposium*, pages 1–6. IEEE.

[Moh'd Alia et al., 2009a] Moh'd Alia, O., Mandava, R., Ramachandram, D., and Aziz, M. E. (2009a). Harmony search-based cluster initialization for fuzzy c-means segmentation of mr images. In *TENCON 2009-2009 IEEE Region 10 Conference*, pages 1–6. IEEE.

[Moh'd Alia et al., 2009b] Moh'd Alia, O., Mandava, R., Ramachandram, D., and Aziz, M. E. (2009b). A novel image segmentation algorithm based on harmony fuzzy search algorithm. In *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*, pages 335–340. IEEE.

[Moh'd Alia and Mandava, 2011] Moh'd Alia, O. and Mandava, R. (2011). The variants of the harmony search algorithm : an overview. *Artificial Intelligence Review*, 36(1) :49–68.

[Mokhtar et al., 2005] Mokhtar, S. B., Fournier, D., Georgantas, N., and Issarny, V. (2005). Context-aware service composition in pervasive computing environments. In *International Workshop on Rapid Integration of Software Engineering Techniques*, pages 129–144. Springer.

[Na-Lumpoon et al., 2014] Na-Lumpoon, P., Fauvet, M.-C., and Lbath, A. (2014). Toward a framework for automated service composition and execution. In *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*, pages 1–8. IEEE.

[Nahrstedt et al., 2001] Nahrstedt, K., Xu, D., Wichadakul, D., and Li, B. (2001). Qos-aware middleware for ubiquitous and heterogeneous environments. *IEEE Communications magazine*, 39(11) :140–148.

[Nakamura et al., 2011] Nakamura, R., Pereira, C., Papa, J. P., and Falcao, A. (2011). Optimum-path forest pruning parameter estimation through harmony search. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 181–188. IEEE.

[Nau et al., 2003] Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2 : An htn planning system. *Journal of artificial intelligence research*, 20 :379–404.

[Omran and Mahdavi, 2008] Omran, M. G. and Mahdavi, M. (2008). Global-best harmony search. *Applied mathematics and computation*, 198(2) :643–656.

[Omrana, 2014] Omrana, H. (2014). *Vers une Composition Dynamique des Services Web : une approche de Composabilité Offline*. PhD thesis, UNIVERSITÉ MOHAMMED V AGDAL–ECOLE MOHAMMADIA D'INGENIEURS.

[Oppong and Khaddaj, 2010] Oppong, E. and Khaddaj, S. (2010). Provision of qos for grid enabled service oriented architectures. In *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on*, pages 118–123. IEEE.

[Osman et al., 2005] Osman, T., Thakker, D., and Al-Dabass, D. (2005). Bridging the gap between workflow and semantic-based web services composition. *6789@ ABCDE FGHC6D• I*, page 13.

[O'sullivan et al., 2002] O'sullivan, J., Edmond, D., and Ter Hofstede, A. (2002). What's in a service ? *Distributed and Parallel Databases*, 12(2-3) :117–133.

[Pan, 2012] Pan, W.-T. (2012). A new fruit fly optimization algorithm : taking the financial distress model as an example. *Knowledge-Based Systems*, 26 :69–74.

[Paolucci et al., 2002] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002). Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347. Springer.

[Papadimitriou and Steiglitz, 1998] Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization : algorithms and complexity*. Courier Corporation.

[Papazoglou et al., 2008] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-oriented computing : a research roadmap. *International Journal of Cooperative Information Systems*, 17(02) :223–255.

[Pellier and Fiorino, 2009] Pellier, D. and Fiorino, H. (2009). Un modèle de composition automatique et distribuée de services web par planification. *Revue des Sciences et Technologies de l'Information-Série RIA : Revue d'Intelligence Artificielle*, 23(1) :13–46.

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10) :46–52.

[Pessini, 2014] Pessini, E. C. (2014). Expressiveness of automatic semantic web service composition approaches : A survey based on workflow patterns. *Revista de Informática Teórica e Aplicada*, 21(1) :45–76.

[Phan and Hattori, 2006] Phan, M. and Hattori, F. (2006). Automatic web service composition using congolog. In *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, pages 17–17. IEEE.

[Pisinger, 1995] Pisinger, D. (1995). Algorithms for knapsack problems.

[Pistore and Traverso, 2001] Pistore, M. and Traverso, P. (2001). Planning as model checking for extended goals in non-deterministic domains. In *IJCAI*, volume 1, pages 479–486.

[Pistore et al., 2005] Pistore, M., Traverso, P., and Bertoli, P. (2005). Automated composition of web services by planning in asynchronous domains. In *ICAPS*, volume 5, pages 2–11.

[Ponnekanti and Fox, 2002] Ponnekanti, S. R. and Fox, A. (2002). Sword : A developer toolkit for web service composition. In *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*, volume 45.

[Pop et al., 2009] Pop, C. B., Chifu, V. R., Salomie, I., and Dinsoreanu, M. (2009). Immune-inspired method for selecting the optimal solution in web service composition. In *International Workshop on Resource Discovery*, pages 1–17. Springer.

[Pop et al., 2011a] Pop, C. B., Chifu, V. R., Salomie, I., and Vlad, M. (2011a). Cuckoo-inspired hybrid algorithm for selecting the optimal web service composition. In *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pages 33–40. IEEE.

[Pop et al., 2011b] Pop, C. B., Rozina Chifu, V., Salomie, I., Baico, R. B., Dinsoreanu, M., and Copil, G. (2011b). A hybrid firefly-inspired approach for optimal semantic web service composition. *Scalable Computing : Practice and Experience*, 12(3) :363–370.

[Qiu et al., 2007] Qiu, L., Chang, L., Lin, F., and Shi, Z. (2007). Context optimization of ai planning for semantic web services composition. *Service Oriented Computing and Applications*, 1(2) :117–128.

[Qiu et al., 2006] Qiu, L., Shi, Z., and Lin, F. (2006). Context optimization of ai planning for services composition. In *e-Business Engineering, 2006. ICEBE'06. IEEE International Conference on*, pages 610–617. IEEE.

[Rao and Su, 2004] Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *SWSWPC*, volume 3387, pages 43–54. Springer.

[Ricart et al., 2011] Ricart, J., Hüttemann, G., Lima, J., and Barán, B. (2011). Multiobjective harmony search algorithm proposals. *Electronic Notes in Theoretical Computer Science*, 281 :51–67.

[Rodriguez-Mier et al., 2017] Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2017). Hybrid optimization algorithm for large-scale qos-aware service composition. *IEEE transactions on services computing*, 10(4) :547–559.

[Rodriguez-Mier et al., 2016] Rodriguez-Mier, P., Pedrinaci, C., Lama, M., and Mucientes, M. (2016). An integrated semantic web service discovery and composition framework. *IEEE transactions on services computing*, 9(4) :537–550.

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1) :77–106.

[Rosenberg et al., 2009] Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., and Dustdar, S. (2009). An end-to-end approach for qos-aware service composition. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 151–160. IEEE.

[Ross-Talbot and Fletcher, 2006] Ross-Talbot, S. and Fletcher, T. (2006). Web services choreography description language : Primer. *World Wide Web Consortium, Working Draft*.

[Russell et al., 1995] Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25 :27.

[Saif et al., 2003] Saif, U., Pham, H., Paluska, J. M., Waterman, J., Terman, C., and Ward, S. (2003). A case for goal-oriented programming semantics. In *Workshop on System Support for Ubiquitous Computing (UbiSys' 03), 5th International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, WA, USA*.

[Salomie et al., 2010] Salomie, I., Chifu, V. R., Harsa, I., and Gherga, M. (2010). Web service composition using fluent calculus. *International Journal of Metadata, Semantics and Ontologies*, 5(3) :238–250.

[Salomie et al., 2014] Salomie, I., Chifu, V. R., and Pop, C. B. (2014). Hybridization of cuckoo search and firefly algorithms for selecting the optimal solution in semantic web service composition. In *Cuckoo Search and Firefly Algorithm*, pages 217–243. Springer.

[Salomie et al., 2011] Salomie, I., Vlad, M., Chifu, V. R., and Pop, C. B. (2011). Hybrid immune-inspired method for selecting the optimal or a near-optimal service composition. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 997–1003. IEEE.

[Schantz and Schmidt, 2001] Schantz, R. E. and Schmidt, D. C. (2001). Middleware for distributed systems : Evolving the common structure for network-centric applications. *Encyclopedia of Software Engineering*, 1 :1–9.

[Shadbolt et al., 2006] Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The semantic web revisited. *IEEE intelligent systems*, 21(3) :96–101.

[Shehu et al., 2014] Shehu, U., Epiphaniou, G., and Safdar, G. A. (2014). A survey of qos-aware web service composition techniques. *International Journal of Computer Applications*.

[Sheng et al., 2014] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition : A decade's overview. *Information Sciences*, 280 :218–238.

[Shiaa et al., 2008] Shiaa, M. M., Fladmark, J. O., and Thiell, B. (2008). An incremental graph-based approach to automatic service composition. In *Services Computing, 2008. SCC'08. IEEE International Conference on*, volume 1, pages 397–404. IEEE.

[Simon, 2008] Simon, D. (2008). Biogeography-based optimization. *IEEE transactions on evolutionary computation*, 12(6) :702–713.

[Singh and Huhns, 2006] Singh, M. P. and Huhns, M. N. (2006). *Service-oriented computing : semantics, processes, agents*. John Wiley & Sons.

[Sirin et al., 2003] Sirin, E., Hendler, J., and Parsia, B. (2003). Semi-automatic composition of web services using semantic descriptions. In *1st Workshop on Web Services : Modeling, Architecture and Infrastructure*, pages 17–24.

[Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet : A practical owl-dl reasoner. *Web Semantics : science, services and agents on the World Wide Web*, 5(2) :51–53.

[Sirin et al., 2004] Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004). Htn planning for web service composition using shop2. *Web Semantics : Science, Services and Agents on the World Wide Web*, 1(4) :377–396.

[Sohrabi et al., 2009] Sohrabi, S., Prokoshyna, N., and McIlraith, S. A. (2009). Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling : Foundations and Applications*, pages 319–334. Springer.

[Srivastava and Koehler, 2003] Srivastava, B. and Koehler, J. (2003). Web service composition-current solutions and open problems. In *ICAPS 2003 workshop on Planning for Web Services*, volume 35, pages 28–35.

[Standard, 2007] Standard, O. (2007). Wsbepl ver. 2.0. *Available on : http ://docs. oasisopen. org/wsbpel/2.0/os/wsbpel-v2. 0-OS. html*.

[Strunk, 2010] Strunk, A. (2010). Qos-aware service composition : A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE.

[Sun and Zhao, 2012] Sun, S. X. and Zhao, J. (2012). A decomposition-based approach for service composition with global qos guarantees. *Information Sciences*, 199 :138–153.

[Szyperski et al., 1999] Szyperski, C., Bosch, J., and Weck, W. (1999). Component-oriented programming. In *European Conference on Object-Oriented Programming*, pages 184–192. Springer.

[Talantikite et al., 2009] Talantikite, H. N., Aissani, D., and Boudjlida, N. (2009). Semantic annotations for web services discovery and composition. *Computer Standards & Interfaces*, 31(6) :1108–1117.

[Talbi, 2009] Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons.

[Tang et al., 2013] Tang, X., Tang, F., Bing, L., and Chen, D. (2013). Dynamic web service composition based on service integration and htn planning. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 307–312. IEEE.

[Taylor, 1997] Taylor, D. A. (1997). *Object technology : a manager's guide*. Addison-Wesley Longman Publishing Co., Inc.

[Trummer et al., 2014] Trummer, I., Faltings, B., and Binder, W. (2014). Multi-objective quality-driven service selection—a fully polynomial time approximation scheme. *IEEE Transactions on Software Engineering*, 40(2) :167–191.

[Ulrich et al., 2005] Ulrich, K., Stern, M., Ries, B. K., et al. (2005). A classifcation of issues and approaches in automatic service composition.

[Vinoski, 2004] Vinoski, S. (2004). An overview of middleware. In *International Conference on Reliable Software Technologies*, pages 35–51. Springer.

[Wagner et al., 2012] Wagner, F., Klein, A., Klöpper, B., Ishikawa, F., and Honiden, S. (2012). Multi-objective service composition with time-and input-dependent qos. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 234–241. IEEE.

[Waldinger, 2000] Waldinger, R. (2000). Web agents cooperating deductively. In *International Workshop on Formal Approaches to Agent-Based Systems*, pages 250–262. Springer.

[Wang and Hou, 2008] Wang, J. and Hou, Y. (2008). Optimal web service selection based on multi-objective genetic algorithm. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, volume 1, pages 553–556. IEEE.

[Wang et al., 2010] Wang, R., Ma, L., and Chen, Y. (2010). The application of ant colony algorithm in web service selection. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4. IEEE.

[Wang and Huang, 2014] Wang, Y. and Huang, J. S. (2014). Method of semantic web service discovery and automatic composition. *Journal of Emerging Technologies in Web Intelligence*, 6(3) :298–304.

[Weise et al., 2007] Weise, T., Bleul, S., Geihs, K., and Allee, W. (2007). Web service composition systems for the web service challenge-a detailed review. *University of Kassel, FB16, Distributed Systems Group, Wilhelmshöher Allee*, 73 :34121.

[Wu et al., 2006] Wu, D., Sirin, E., Hendler, J., Nau, D., and Parsia, B. (2006). Automatic web services composition using shop2. Technical report, MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE.

[Wu and Zhu, 2013] Wu, Q. and Zhu, Q. (2013). Transactional and qos-aware dynamic service composition based on ant colony optimization. *Future Generation Computer Systems*, 29(5) :1112–1119.

[Xia et al., 2011] Xia, Y., Chen, P., Bao, L., Wang, M., and Yang, J. (2011). A qos-aware web service selection algorithm based on clustering. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 428–435. IEEE.

[Xia et al., 2008] Xia, Y.-m., Chen, J.-l., and Meng, X.-w. (2008). On the dynamic ant colony algorithm optimization based on multi-pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 630–635. IEEE.

[Xiangbing et al., 2012] Xiangbing, Z., Hongjiang, M., and Fang, M. (2012). An optimal approach to the qos-based wsmo web service composition using genetic algorithm. In *International Conference on Service-Oriented Computing*, pages 127–139. Springer.

[Xiong et al., 2009] Xiong, P., Fan, Y., and Zhou, M. (2009). Web service configuration under multiple quality-of-service attributes. *IEEE Transactions on Automation Science and Engineering*, 6(2) :311–321.

[Yachir, 2014] Yachir, A. (2014). *Composition dynamique de services sensibles au contexte dans les systèmes intelligents ambiants*. PhD thesis, Université Paris-Est.

[Yachir et al., 2012] Yachir, A., Amirat, Y., Chibani, A., and Badache, N. (2012). Towards an event-aware approach for ubiquitous computing based on automatic service composition and selection. *Annals of telecommunications-Annales des télécommunications*, 67(7-8) :341–353.

[Yan et al., 2012] Yan, Y., Chen, M., and Yang, Y. (2012). Anytime qos optimization over the plangraph for web service composition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1968–1975. ACM.

[Yan et al., 2010] Yan, Y., Poizat, P., and Zhao, L. (2010). Repairing service compositions in a changing world. *Software Engineering Research, Management and Applications 2010*, pages 17–36.

[Yan et al., 2008] Yan, Y., Xu, B., and Gu, Z. (2008). Automatic service composition using and/or graph. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pages 335–338. IEEE.

[Yan et al., 2009] Yan, Y., Xu, B., Gu, Z., and Luo, S. (2009). A qos-driven approach for semantic service composition. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 523–526. IEEE.

[Yang, 2009] Yang, X.-S. (2009). Harmony search as a metaheuristic algorithm. In *Music-inspired harmony search algorithm*, pages 1–14. Springer.

[Yang, 2010] Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer.

[Yang and Deb, 2009] Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE.

[Yang et al., 2010] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. *Journal of Computational Information Systems*, 6(8) :2617–2622.

[Ying, 2010] Ying, L. (2010). A method of automatic web services composition based on directed graph. In *Communications and Mobile Computing (CMC), 2010 International Conference on*, volume 1, pages 527–531. IEEE.

[Yu and Bouguettaya, 2009] Yu, Q. and Bouguettaya, A. (2009). *Foundations for efficient web service selection*. Springer Science & Business Media.

[Yu and Bouguettaya, 2013] Yu, Q. and Bouguettaya, A. (2013). Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4) :776–789.

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1) :6.

[Yu et al., 2013] Yu, Y., Ma, H., and Zhang, M. (2013). An adaptive genetic programming approach to qos-aware web services composition. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1740–1747. IEEE.

[Zeleny, 1976] Zeleny, M. (1976). The theory of the displaced ideal. In *Multiple criteria decision making Kyoto 1975*, pages 153–206. Springer.

[Zelinka, 2004] Zelinka, I. (2004). Soma—self-organizing migrating algorithm. In *New optimization techniques in engineering*, pages 167–217. Springer.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5) :311–327.

[Zhang et al., 2011] Zhang, Y., Zheng, Z., and Lyu, M. R. (2011). Exploring latent features for memory-based qos prediction in cloud computing. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*, pages 1–10. IEEE.

[Zhao et al., 2012] Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., and Fan, Y. (2012). An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing*, 12(8) :2208–2216.

[Zhao et al., 2017] Zhao, Y., Tan, W., and Jin, T. (2017). Qos-aware web service composition considering the constraints between services. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pages 229–232. ACM.

[Zheng et al., 2013] Zheng, H., Zhao, W., Yang, J., and Bouguettaya, A. (2013). Qos analysis for web service compositions with complex structures. *IEEE Transactions on Services Computing*, 6(3) :373–386.

[Zheng and Yan, 2008] Zheng, X. and Yan, Y. (2008). An efficient syntactic web service composition algorithm based on the planning graph model. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 691–699. IEEE.

[Zitzler et al., 2003] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers : An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2) :117–132.

[Zribi, 2014] Zribi, S. (2014). *La gouvernance SOA pour une approche de conception de Système d'Information de Médiation : réconciliation non-fonctionnelle de services pour mettre en œuvre les processus métier*. PhD thesis, Ecole des Mines d'Albi-Carmaux.