RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

**UNIVERSITÉ ABOU-BEKR BELKAID - TLEMCEN**

# THÈSE

Présentée à :

FACULTÉ DES SCIENCES – DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT EN SCIENCES

Spécialité: Informatique

Par :

**Mme HALFAOUI Amal épouse GHERNAOUT**

Sur le thème

---

# La sélection des services web dans une composition à base de critères non fonctionnels.

---

Soutenue publiquement le 26 Février 2017 à Tlemcen devant le jury composé de :

| | | | |
|---|---|---|---|
| Mr CHIKH Amine | Professeur | Université de Tlemcen | Président |
| Mme BABA HAMED Latifa | Professeur | Université d'Oran | Examinatrice |
| Mr CHIKH Azeddine | Professeur | Université de Tlemcen | Examinateur |
| Mr BENSLIMANE Sidi Mohammed | Professeur | ESI de Sidi Bel Abbès | Examinateur |
| Mme DIDI Fedoua | Maître de Conférences A | Université de Tlemcen | Directrice de thèse |
| Mr HADJILA Fethallah | Maître de Conférences B | Université de Tlemcen | Invité |

*Laboratoire de Recherche en Informatique de Tlemcen (LRIT)*
*BP 119, 13000 Tlemcen - Algérie*

# Remerciements

Mes vifs remerciements vont à toutes les personnes ayant contribué de près ou de loin au bon déroulement et à l'aboutissement de cette thèse, tant sur le plan professionnel que sur le plan personnel.

En premier lieu, je tiens à exprimer ma profonde gratitude à ma directrice de thèse, Madame DIDI Fedoua, Maitre de conférence à UABT Tlemcen et directrice de notre laboratoire LRIT, qui m'a donné la chance de réaliser ces travaux et de les mener à bien ainsi que pour la confiance et la liberté qu'elle m'a accordées durant toutes ces années.

Je ne saurai témoigner toute ma gratitude à Monsieur Fethallah HADJIlA, Maitre de conférence à la Faculté des Sciences de Tlemcen et membre du laboratoire LRIT, qui m'a initiée au sujet de cette thèse. Je suis heureuse de lui adresser mes vifs remerciements pour l'intérêt qu'il a manifesté à ce travail en acceptant la charge de suivre de près ces travaux. Je voudrais lui exprimer ma profonde reconnaissance pour l'aide qu'il m'a constamment octroyée tout au long de ce travail.

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, Madame BABA HAMED Latifa Professeur à l'université d'Oran, et Monsieur BENSLIMANE Sidi Mohamed Professeur et directeur de l'Ecole Supèrieur 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.

Je remercie chaleureusement ma famille pour son soutien, son écoute et ses encouragements tout au long de cette thèse. Mes parents, mes frères, ma sœur, ma belle famille et surtout mon adorable époux.

En espérant n'oublier personne, j'adresse, enfin, un grand merci à mes collègues enseignants du département d'informatique.

À la mémoire de mon oncle tonton Halim,
à mon cher mari Tariq,
à mes parents,
à toute ma famille. . .

**Résumé**

Le marché de commercialisation des services Web sur internet ne cesse d'augmenter, ce qui résulte en un nombre de plus en plus croissant de services offrant des fonctionnalités équivalentes. De ce fait, la sélection d'un service web approprié pour une tâche particulière est devenue un défi difficile pour l'utilisateur. Cependant, ces services diffèrent dans leurs qualités de services (QoS). Ces dernières deviennent alors cruciales pour l'utilisateur, car elles permettent de l'aider à choisir parmi les services Web qui sont fonctionnellement équivalents. Toutefois, il n'est toujours pas évident de choisir des services intéressants qui permettent de satisfaire les multiples contraintes QoS. De plus, il devient difficile de satisfaire l'exigence complexe d'un utilisateur par juste un seul service web individuel ou atomique, ce qui a conduit les concepteurs à composer les services Web existants. Par conséquent, la sélection n'est plus le simple choix d'un seul service atomique parmi tant d'autres pour une seule tâche, mais la sélection devient une opération complexe qui consiste à combiner un ensemble de services choisis pour chaque tâche, afin de former la composition qui satisfait au mieux les critères QoS de l'utilisateur.

Nous nous intéressons, dans cette thèse, à la problématique de sélection des services Web dans une composition sur la base des besoins non fonctionnels (QoS), communément connue sous le nom QoSWSC. La problématique QoSSWC est considérée comme un problème d'optimisation multi-objectif. Nous nous intéressons, plus particulièrement, dans nos travaux, à deux catégories d'approches d'optimisations, à savoir, la première qui se base sur les techniques de bases de données qui utilisent la dominance de Pareto et la deuxième qui repose sur les méta-heuristiques.

Nous proposons dans la première catégorie d'approches deux contributions qui reposent sur la fuzzification de la relation de dominance selon Pareto. La première permet la sélection des Top-k services Web basée sur la dominance floue. La deuxième permet de calculer les services Skylines sur la base de la dominance floue.
Nous proposons, dans la deuxième catégorie, trois contributions. Dans la première contribution, nous adaptons la récente méta-heuristique SOMA (Self-Organizing Migrating Algorithm) et nous présentons l'algorithme QoS-SOMA qui est la version discrète de SOMA. Cette dernière est mieux adaptée au problème de sélection des services Web composés. Nous proposons, dans la deuxième contribution, une amélioration de l'algorithme QoS-SOMA en utilisant la notion de dominance floue dans la phase de la sélection locale. Dans la troisième contribution, Nous proposons une approche de sélection des services Web composés, basée sur la recherche Tabou et le calcul des Skylines. L'algorithme de Skyline est utilisé dans le but de réduire l'espace de recherche de la sélection.

Les résultats expérimentaux montrent que les méta-heuristiques SOMA et Tabou améliorées par la dominance floue et le Skyline sont plus prometteuses que les versions standards.

**Mots clés :** La sélection des services Web, Skyline, Optimisation combinatoire, Dominance floue, Méta-heuristiques, Qualité de service, SOMA.

**Abstract**

The Web services market does not stop growing; this results in a more and more increasing number of functionally similar services. Therefore, the selection of an appropriate web service for a particular task has become a difficult challenge for the user. However, these services differ in their qualities of services (QoS) which constitute a crucial aspect for selecting web services among functionally similar components. Nevertheless, it remains difficult to select an interesting Web service which meets the multiple QoS constraints from a large number of candidates. In addition, the satisfaction of the complex requirement is rarely done by an individual Web service, what led the designers to compose the existing Web services.

Therefore, the selection is not any more, the simple choice of an atomic service for a single task among so many others, but the selection becomes a decision problem on which component services should be selected for each task so that user's QoS criteria are best satisfied**.**

In this thesis, we address the composite web service selection problem based on QoS known as QoS-aware web service selection 'QoSWSC'. This problem is considered as a multiobjective optimization problem. We mainly focus our research on two categories for solving the QoSWSC problem: the first one is based on the databases techniques that use the pareto dominance, and the second is based on the metaheuristics algorithms.

We propose in the first category, two major contributions that are based on the fuzzification of Pareto dominance to compare functionally similar services. The first contribution selects the top-k web services. The second one computes the service skylines.

We propose in the second category, three contributions. In the first one, we adapt a recent stochastic optimization algorithm called Self Organizing Migrating Algorithm (SOMA) and present QoS-SOMA algorithm which is the discrete version of SOMA. In the second contribution, we improve the current version by using the fuzzy dominance comparison in the step of local search. In the third contribution, we propose an approach that is based on Tabu search and the skyline. The skyline is used to reduce the search space of the QoSWSC problem.

The experimental results show that the hybridization of metaheuristics with the fuzzy domination and the skyline outperform the standard versions.

**Key words**: Web service selection, Skyline, Combinatorial optimization, Fuzzy dominance, Metaheuristics , Quality of service, SOMA.

## ملخص

ان تسويق خدمات الويب على الانترنت في زيادة مستمرة مما يؤدى الى تزايد عدد الخدمات التي تقدم نفس الوظيفة. ولذلك، فان إنتقاء خدمة الويب الملائمة لوظيفة معينة ،أصبح تحديا صعبا بالنسبة للمستخدم. ومع ذلك، فإن هذه الخدمات تختلف في معايير الجودة غير الوظيفية (QoS) المتعلقة بالجانب التقني للتطبيق الإلكتروني. هذه المعايير النوعية أصبحت حاسمة للمستخدم لأنها تسمح للمساعدة في الإختيار من بين خدمات الويب التي تتعادل وظيفيا. لكن، يبقى من الصعب إختيار الخدمات المناسبة التي تلبي معايير الجودة المتعددة. وعلاوة على ذلك، فإنه أصبح من الصعب أيضا تلبية إحتياجات المستخدم المعقدة عن طريق تطبيق فردي واحد ، مما أدى المبرمجين لإعادة إستعمال التطبيقات الفردية الموجودة و دمجها في صيغة خدمة ويب مركبة.

نتيجة لذلك، لم تعد عملية الانتقاء خيارا بسيطا لخدمة ويب واحدة من بين العديد ، ولكن الانتقاء هو عملية معقدة تنطوي على الجمع بين مجموعة من الخدمات الفردية التي يتم اختيارها لكل مهمة، لتشكيل خدمة ويب مركبة ترضي جميع متطلبات معايير الجودة غير الوظيفية للمستخدم.

نتطرق في هذه الأطروحة إلى إشكالية اختيار خدمات الويب لتكوين خدمة ويب مركبة بناء على متطلبات الجودة غير الوظيفية، المعروفة باسم (QoSWSC). تنتمي الاشكالية إلى فئة مشاكل الأمثلة متعددة الأهداف (المعروف أيضا باسم البرمجة متعددة الاهداف او الأمثلة متعددة المعايير (Optimisation multi-objectifs). نتطرق في عملنا، بالأخص إلى فئتين لحل إشكاليتنا، الأولى تعتمد على تقنيات قواعد البيانات التي تستخدم هيمنة باريتو (Dominance de Pareto) والثانية على أساس خوارزميات الادلة العليا(Méta-heuristiques).

نقترح فى الفئة الاولى مساهمتين تعتمدان على علاقة fuzzification لهيمنة باريتو. الاولى تسمح باختيار Top_k خدمات الويب. اما الثانية تمكن من حساب Skyline.
نقترح، في الفئة الثانية، ثلاث مساهمات. في الأولى ، نقوم بتعديل الخوارزمية الحديثة في مجال الادلة العليا SOMA (self-organizing migrating algorithm) ونقدم الخوارزمية 'QoS- SOMA' المناسبة المسماة لحل مشكلة QoSWSC. نقترح في المساهمة الثانية، تحسين الخوارزمية باستعمال(FuzzyDominance). في المساهمة الثالثة، نقدم منهجية تعتمد على خوارزمية 'Tabu Search' وحساب Skyline. تستخدم Skyline من أجل تقليص منطقة البحث.



**كلمات البحث**: اختيار خدمات الويب, حساب Skyline, البرمجة متعددة الاهداف, معايير الجودة غير الوظيفية , خوارزميات الادلة العليا (Méta-heuristiques), SOMA.

# Table des matières

# Table des figures

# Liste des tableaux

# Liste des Algorithmes

# Liste des abréviations et des sigles

**ABC** Artificial Bee Colony

**ACO** Ant Colony Optimization

**BPEL** Business Process Execution Language

**BP** Business Process

**CS** Cuckoo Search

**EC** Evolutionary Computation

**GA** Genetic Algorithm

**GRASP** Greedy Randomized Adaptive Search Procedure

**HS** Harmony Search

**ILP** Integer Lineaire Porogramming

**MCDM** multiple criteria decision making

**MIP** Mixed Integer Programming

**MMKP** multi-dimension multi-choice knapsack problem

**PMO** Problème Multi-objectif

**PSO** Particle Swarm Optimization

**QoSWSC** QoS-aware web service composition

**QoS** Quality of Service

**REST** REpresentational State Transfer

**RPC** Remote Procedure Calls

**SA** Simulated annealing

**SLA** Service Level Agreements

**SOAP** Simple Object Access Protocol

**SOA** Service Oriented Architecture. (Fr : architecture orientée service)

**SOMA** Self Organizing Migrating Algorithm

**SW** Service Web

**TS** Tabu Search

**UDDI** Universal Description Discovery and Integration

**URI** Uniform Resource Identifier

**WS-CDL** Web Services Choreography Description Language

**WSC** Web Service Composition

**WSDL** Web Service Description Langage

**WWW** World Wide Web Consortium

**XML** eXtensible Markup Language

Chapitre 1

# Introduction générale

*"On peut toujours apprendre ce qu'on ne sait pas,*
*non ce qu'on croit savoir."*

Gustave Thibon

## Sommaire



## 1.1 Contexte

Actuellement, l'enjeu principal des entreprises est de faire face au problème de compétitivité. En effet, les entreprises doivent faire face aux défis concurrentiels et aux changements rapides de l'environnement auxquels elles doivent continuellement réagir. Pour cela, elles doivent assurer l'intégration et l'interopérabilité de leurs applications au niveau interne et au niveau partenaire (Business To Business). Ces applications sont généralement appelées processus métiers. Pour gérer et automatiser le cycle de vie de ces processus, les entreprises adoptent l'architecture orientée service (SOA : Service Oriented Architecture)

Selon le Gartner Group [1], plus de 75 % des projets d'entreprise reposent sur l'architecture SOA. Cette forte convergence des entreprises vers SOA montre les capacités de cette dernière à offrir des solutions aux différents problèmes dont des technologies orientées composants comme les intergiciels conventionnels CORBA (Common Object Request Broker Architecture) [Portal, 2009] et DCOM (Distributed Component

1. http ://www.gartner.com/

Object Model) [Horstmann and Kirtland, 1997] ont été incapables de résoudre. En effet, l'architecture SOA [Erl, 2004] est caractérisée par son couplage faible, son indépendance par rapport aux plateformes, et sa grande capacité d'intégration et de réutilisation.

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

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

- SOAP (Simple Object Access Protocol) [Protocol, 2003] est le protocole permettant de structurer les messages échangés entre les SW.
- WSDL (Web Service Description Langage) [Chinnici et al., 2007] est la spécification qui définit les interfaces des SW.
- UDDI (Universal Description Discovery and Integration) [Clement et al., 2004] est une spécification de publication et de localisation de SW.

Les services Web possèdent un cycle de vie qui ramifie le modèle d'interaction de l'architecture SOA et en particulier l'étape de consultation de services. Nous avons en l'occurrence la phase de :

- **la découverte** : qui permet de chercher et trouver les services adéquats pour une requête .
- **la composition** : cette phase permet de combiner plusieurs SW atomiques pour répondre à une requête, car il est devenu de plus en plus rare de satisfaire cette requête par un seul service.
- **la sélection et réutilisation** : cette phase permet de choisir un SW pour chaque tâche dans la phase de composition. Elle permet aussi de choisir le service Web composite (composé) qui satisfait les critères globaux de la requête.

## 1.2 Objectifs et problématiques

En raison du succès de SOA comme solution pour l'interopérabilité, la réutilisation et la mondialisation [Yu et al., 2008], il y a eu ces dernières années, un grand flux de services web déployés sur le Web. Si on lance une requête de recherche de services pour envoyer des sms, par exemple dans le moteur de recherche de services Web Titan [2], on a un retour de plus de 200 services. Titan regroupait déjà en 2012 plus de 15 969 services Web accessibles à partir de plusieurs fournisseurs et depuis, ce chiffre n'a cessé d'augmenter [Wu et al., 2012].

Face à cette croissance exponentielle en nombre et en fonctionnalités des services sur Internet, l'utilisateur est face à la difficulté de choisir manuellement entre un nombre important de services retournés par la requête. Il est devenu essentiel de proposer un système qui permet au client de choisir les meilleurs services qu'il désire sans intervention manuelle.

Bien qu'un seul service web ait sa propre valeur pour ses utilisateurs, la fonctionnalité offerte par les services Web individuels est limitée. Il est très rare de répondre aux besoins des clients par un seul service. Par conséquent, nous sommes obligés de composer plusieurs services afin de satisfaire une requête. 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.

La composition des services Web est un processus complexe qui fait intervenir plusieurs activités telles que la découverte, la composabilité, la sélection ou encore l'exécution. La phase de sélection dans la composition consiste à choisir un service de chaque classe de services. Généralement, la composition est considérée, dans un premier temps, comme un workflow abstrait, c-à-d une composition de tâches auxquelles aucun service n'est encore affecté.

La phase de sélection (voir Figure 1.1) consiste à chercher les instances concrètes (services concrets) qui peuvent remplacer des nœuds abstraits. Ces services doivent satisfaire autant que possible les exigences QoS et les préférences des utilisateurs.

La sélection est une étape cruciale de laquelle dépend le résultat de la composition. Cette sélection n'est pas évidente, car il s'agit de choisir des services Web parmi un nombre important d'alternatives de services proposés par la phase de découverte. Ce problème est une instance de problèmes combinatoires, c'est une version multidi-

2. //ccnt.zju.edu.cn :8080/

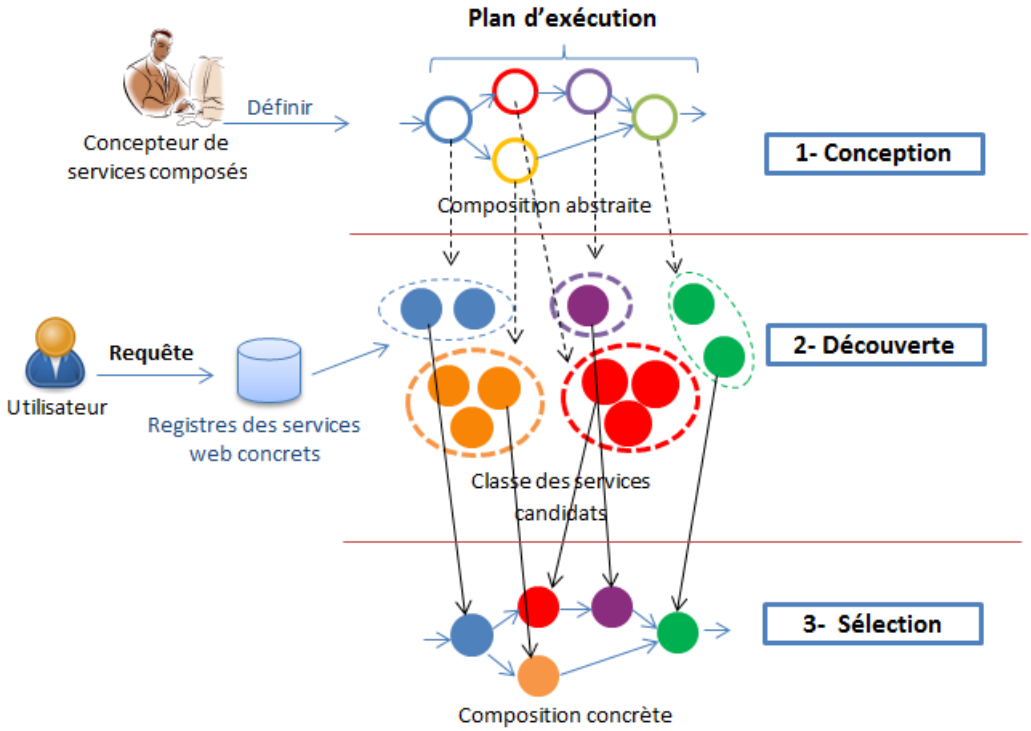



**Figure *1.1:*** *L'étape de sélection dans la phase de composition*

mensionnelle du problème du sac à dos connu pour être NP-difficile. Par conséquent, toute solution exacte à ce problème nécessite un temps d'exécution exponentiel, et de ce fait, le temps réel des utilisateurs n'est pas garanti. Ce problème sera présenté en détail dans le chapitre 3.

Nous nous intéressons dans cette thèse plus particulièrement à la phase de sélection des services qui permet de fournir une composition de services Web. Afin de mieux cerner les problématiques liées à cette phase et ainsi expliquer nos motivations, considérons l'exemple suivant :

Supposons que nous ayons un utilisateur qui souhaite acheter une voiture y, fabriquée par un constructeur d'un pays x. L'utilisateur souhaite aussi contracter une assurance automobile z. Un tel service n'existe pas, par conséquent, nous sommes obligés de composer un ensemble de services pour répondre à la requête du client. L'exemple présenté dans le Tableau 1.1 est inspiré de celui cité dans [Benouaret et al., 2011a]. Il présente un système e-commerce qui offre aux utilisateurs l'achat d'automobiles. Nous avons étendu l'exemple en ajoutant l'aspect non fonctionnel des services Web. Un service web est caractérisé pas un aspect fonctionnel qui décrit ce que le service doit faire, et un aspect non fonctionnel qui décrit comment le service peut le faire. Cet aspect non fonctionnel est représenté par les paramètres de Qualité de

| **Service** | **Fonctionnalité** | **Contraintes fonctionnelles** | **N-fonctionnelles** | | |
|---|---|---|---|---|---|
| | | | $q_1$ (€) | $q_2$ (ms) | $q_3$ (ms) |
| $S_{11}(i(x), o(y)$ | Retourne les voitures y fabriquées par x | - | 8 | 120 | 70 |
| $S_{21}(i(x), o(y, z, t))$ | Retourne les voitures y avec leurs prix z et la garantie t pour un constructeur automobile x | z ∈ [5000,7000], t ∈ [12,24] | 1 | 140 | 50 |
| $S_{22}(i(x), o(y, z, t))$ | | z ∈[5500,7000], t ∈[12,18] | 1 | 145 | 35 |
| $S_{23}(i(x), o(y, z, t))$ | | z ∈[15000,20000], t ∈[6,24] | 3 | 160 | 30 |
| $S_{31}(i(x), o(y, z))$ | Retourne la puissance y et la consommation z pour la voiture x | y ∈[50,70], z ∈[4,10] | 3 | 170 | 40 |
| $S_{32}(i(x), o(y, z))$ | | y ∈ [50,90], z ∈ [8,12] | 8 | 150 | 50 |
| $S_{33}(i(x), o(y, z))$ | | y ∈ [200,400], z ∈ [10,20] | 4 | 160 | 30 |

***Tableau 1.1:*** *Exemple de Services Web [Benouaret et al., 2011a].*

Services ($QoS$) qui sont indépendants de sa fonctionnalité comme le prix, le temps de réponse etc.

Dans notre exemple, chaque service $S_{ij}$ a des paramètres d'entrée $i()$ et des paramètres de sortie $o()$. Les services fournissant la même fonctionnalité appartiennent à la même classe ($S_{21}, S_{22}, S_{23}$ appartiennent à la classe $S_2$). Chaque service contient :

1. des contraintes fonctionnelles sur les données qu'il manipule. Par exemple, le prix de la voiture retournée par $S_{22}$ est compris entre 5500 et 7000 €.
2. des contraintes non fonctionnelles, relatives à la qualité de service. Dans notre exemple, nous nous limitons à trois qualités $q_1$, $q_2$, $q_3$ qui correspondent respectivement aux prix, temps de réponse et disponibilité. Ces dernières, ainsi que d'autres paramètres $QoS$ seront définis et détaillés dans le chapitre suivant.

Supposons qu'un utilisateur x souhaite acheter une voiture française avec une puissance supérieure à 60 chevaux et une consommation entre [10,11]. Il souhaite aussi contracter une garantie entre 12 et 18 mois.

L'utilisateur va invoquer le service $S_{11}$, il peut invoquer un ou plusieurs services appartenant à la classe $S_2$. Finalement, il invoque un ou plusieurs services appartenant à la classe $S_3$. Plus le nombre de services augmente par classe, plus le choix devient fastidieux pour l'utilisateur. La qualité de services Web (QoS) devient alors cruciale pour choisir des services web parmi des composants fonctionnellement équivalents. L'utilisateur peut aussi exprimer des préférences sur la composition qu'on appelle des

contraintes QoS 'globales' par exemple : le prix total de la composition ne doit pas dépasser 10 €. c-à-d la somme des prix ($q_1$) de tous les services composants ne doit pas dépasser cette contrainte.

La sélection à base de Qualités de Services consiste à choisir parmi les services Web découverts de chaque tâche, ceux qui répondent au mieux aux exigences de l'utilisateur sur la base des besoins non fonctionnels de QoS. La sélection de services Web basée sur la QoS dépend de la spécification adoptée lors de la définition des critères QoS et du profil QoS du service Web. Deux catégories importantes peuvent alors être considérées [Moghaddam and Davis, 2014] : les approches basées sur la négociation et celles basées sur l'optimisation.
Nous nous intéressons dans cette thèse aux approches basées sur l'optimisation qui assument typiquement un profil de QoS prédéterminé où les QoS ne sont pas personnalisables. Dans ce cas, le problème de sélection de services devient un problème d'optimisation. Le problème d'optimisation de la sélection de services Web basée sur la QoS est apparu avec différentes appellations comme *'QoS-driven or QoS-aware Web Service Composition' QoSWSC* ou *'Web service Composition Optimization'*.

Le problème QoSWSC est l'un des plus importants de l'architecture orientée service. Il constitue aussi l'une des étapes les plus décisives dans le processus de composition. QoSWSC est un problème d'optimisation multi-objectif. Effectivement, la majorité des travaux actuels prend en compte, dans la sélection des services Web, plusieurs paramètres de QoS. Dans ce cas, face au caractère multidimentionnel des critères de QoS, comment faire pour trouver la composition qui permet à l'utilisateur :

**Q1** - de choisir des services dans la phase locale, pour chaque tâche, ceux offrant un compromis entre les différentes contraintes de QoS : il n'est pas intéressant de choisir des services qui ont certains paramètres de QoS avec des valeurs très élevées (optimales) et d'autres nulles.

**Q2** - de satisfaire les contraintes globales : les services choisis dans la phase locale pour chaque tâche, doivent satisfaire les contraintes sur la composition.

Pour répondre à la première question Q1, nous avons étudié les travaux qui ont utilisé le principe de dominance de Pareto qui est connu pour comparer deux vecteurs d'objectifs de nature contradictoire, dans notre cas, les données QoS sont de nature contradictoire comme le temps, la fiabilité. Ces travaux reposent entre autre sur les techniques orientées bases de données qui utilisent la notion de dominance comme les requêtes Skyline et Top-k, pour classer et filtrer les services Web.

Pour répondre à la deuxième question Q2, nous avons étudié les travaux utilisant les méta-heuristiques, connues pour leurs performances dans la résolution des problèmes Np-difficiles.

Nous nous intéressons dans cette thèse à ces deux méthodes qui sont beaucoup utilisées dans la résolution du problème QoSWSC. Nous définissons dans ce qui suit, les problématiques liées à ces deux catégories de travaux.

**P1 : La sélection des services Web basée sur les techniques orientées bases de données (utilisant la dominance)**.

De nouveaux travaux sur la résolution du problème QoSWSC ont adopté des approches qui reposent sur les techniques avancées des bases de données pour le traitement de requêtes complexes. Effectivement, deux grandes techniques ont reçu une importance croissante de la part des chercheurs et ont fait l'objet de plusieurs travaux : les requêtes Top-k et les requêtes Skyline. Récemment, ces deux dernières ont été combinées en une troisième technique appelée requêtes Top-k dominating. Les requêtes Top-k dominating tirent leurs avantages des requêtes Top-k et celles de Skyline, évitant ainsi leurs inconvénients.

Les requêtes Skyline et les requêtes Top-k dominating représentent des outils performants pour analyser des données multicritères, ce qui facilite la prise de décisions intelligentes face à des données de grande dimension, où l'utilisateur à des difficultés pour formuler ses préférences. Elles permettent d'extraire l'ensemble des points les plus pertinents quand différents critères, souvent conflictuels, sont pris en compte. Elles s'appuient sur le principe de dominance de Pareto.

La majorité des travaux, utilisant les techniques de Skyline et Top-k dominating, repose sur le principe de Pareto dominance. Ce dernier ne différentie pas entre les services Web avec un bon compromis et ceux avec un mauvais compromis. Nous désignons par le terme 'Bon compromis' les services qui présentent un ensemble de qualités de services avec des valeurs équilibrées.

Nous désignons par le terme 'mauvais compromis' des services qui ont des valeurs nulles ou proches du zéro pour certains paramètres, et des valeurs très élevées pour d'autres. En effet, ces services ne sont pas très intéressants pour la composition. D'autre part, le calcul du Skyline peut conduire à deux scénarios : soit (i) un nombre assez important de services sont retournés (cas où les paramètres QoS augmentent), ce qui complique davantage la tâche de sélection

en augmentant les choix de l'utilisateur, soit (ii) un nombre minime de services sont retournés, ce qui peut être insuffisant pour la prise de décisions des utilisateurs. En plus, le Skyline élimine certains objets qui peuvent être intéressants dans quelques paramètres de QoS. Le Skyline ne permet pas aussi d'ordonner les services car les objets appartenant aux Skyline sont incomparables entre eux.

Notre objectif est de proposer une approche flexible permettant de choisir et classer les meilleurs services offrant un compromis entre leurs QoS et écarter ceux qui peuvent sembler les meilleurs alors qu'ils présentent un très mauvais compromis entre leurs QoS.

**P2 : La sélection des services web basée sur les méta-heuristiques**.

Les algorithmes méta-heuristiques sont devenus appropriés à la résolution de la problématique QoSWSC. Ces dernières années, un ensemble de nouveaux algorithmes d'optimisation a été proposé par la communauté des chercheurs dans le domaine d'optimisation pour résoudre des problèmes académiques très connus comme le problème du voyageur de commerce, le problème du Sac à dos,..,etc.

Parmi ces algorithmes, nous citons l'algorithme d'optimisation Self-Organising Migrating Algorithm SOMA [Zelinka, 2004], cuckoo search [Yang and Deb, 2009], Fire Fly [Yang, 2009]. Ces derniers n'ont pas été très exploités dans le domaine des services web bien qu'ils offrent des résultats prometteurs.

D'autre part, la tendance des travaux actuels est d'utiliser des algorithmes hybrides qui sont des combinaisons entre méta-heuristiques et heuristiques ou des méta-heuristiques et approches exactes. Ces derniers tentent d'améliorer des méta-heuristiques en tirant avantage des autres techniques et méthodes de résolution.

L'objectif donc de cette thèse est de proposer des solutions qui améliorent la qualité des services choisis pour participer dans le processus de compositions en se basant sur des approches qui : d'un côté adaptent les nouvelles méta-heuristiques qui n'ont pas été encore très exploitées dans l'optimisation du problème QoSWSC. D'un autre côté, améliorer les méta-heuristiques les plus utilisées ou encore proposer des approches hybrides pour mieux répondre aux problèmes de QoSWSC.

## 1.3 Contributions

Les majeures contributions de cette thèse sont :

1. La proposition de deux solutions qui sont des contributions en réponse à la première problématique P1, qui consistent en :

   (a) **C1 : La sélection des Top-k services Web basée sur la dominance floue.**

   Pour pallier les inconvénients des techniques du Skylines et Top-k dominating qui reposent sur la notion de Pareto dominance, nous proposons une approche [Halfaoui et al., 2015] qui repose sur la définition du principe de *'Fuzzy dominated Score'* (le score dominé flou), nous proposons la fonction $FDed()$ qui est une fuzzification du principe de Pareto Dominance. Alors que le principe de Pareto Dominance ne permet pas toujours une comparaison entre les vecteurs des paramètres de QoS, notre approche offre une alternative pour classer les services incomparables au sens de Pareto dominance. Elle permet aussi l'inclusion de services web avec un bon compromis entre des paramètres de QoS. Elle fournit ainsi aux utilisateurs les services web les plus pertinents. En effet, notre proposition est une extension de la relation de Pareto dominance qui permet d'exprimer le degré avec lequel un service est dominé par un autre, sur ses paramètres de QoS. D'autre part, les travaux QoSWSC, basés sur les requêtes Top-k dominating, utilisent tous un classement selon le score de la relation *'est dominant'* de Pareto. Cette relation associe à chaque service 's', le nombre de services que ce dernier domine. Notre proposition effectue un classement de services qui s'appuie sur la relation *'est dominé par'* qui est l'inverse de la relation *'est dominant'* de Pareto. Elle permet d'associer à chaque service 's', le nombre de services qui dominent ce dernier. Cette dernière approche s'avère donner un meilleur classement par rapport à la première relation. Les étapes de cette proposition sont résumées dans l'algorithme : '$FTop\text{-}KS$'.

   (b) **C2 : La sélection des services web Skyline basée sur la dominance floue.**

   La majorité des travaux utilisant les algorithmes de calcul de Skyline repose sur la dominance de Pareto au sens strict. Nous proposons une approche, qui permet de calculer le Skyline des services web d'une classe en

utilisant l'algorithme SFS (Sort First Skyline) [Chomicki et al., 2005] et la dominance floue. Plus précisément, nous utilisons le concept proposé dans la contribution C1, '$FDed()$'. Cette approche permet de relaxer le Skyline et de prendre en considération certains services qui étaient initialement écartés par les algorithmes standards de Skyline. L'algorithme :'$\alpha$-$DSkyS$-$SFS$' présente toutes les étapes de cette proposition.

2. Nous proposons aussi, trois autres approches qui sont des contributions en réponse à la problématique P2.

   (a) **C3 : La sélection des services web basée sur la méta-heuristque SOMA.**

   Nous proposons dans cette approche [Halfaoui et al., 2017] l'algorithme 'QoS-SOMA' qui est une adaptation de la récente méta-heuristique SOMA (Self-Organizing Migrating Algorithm) [Zelinka, 2004] pour la résolution du problème QoSWSC. La version initiale de SOMA traite les problèmes à valeurs continues. Notre approche propose la version discrète adaptée au problème de sélection des services web composés qui présente des résultats prometteurs.

   (b) **C4 : La sélection des services web basée sur la méta-heuristque SOMA et la dominance floue.**

   Nous proposons une amélioration de l'algorithme QoS-SOMA proposé en C3 en utilisant la notion de dominance floue (Contribution C1) dans la phase de la sélection locale.

   (c) **C5 : La sélection des services web basée sur la recherche Tabou et le Skyline flou.**

   Nous proposons dans cette contribution une approche [Halfaoui et al., 2014] qui se déroule en deux phases :

   - Une phase locale : nous utilisons dans cette étape, la notion de Skyline flou (Contribution C2) pour réduire l'espace de recherche dans chaque tâche.
   - Une phase globale : l'espace de recherche ainsi réduit, nous avons besoin d'une méta-heuristique simple pour trouver les services composites et satisfaire les contraintes globales. Nous définissons l'algorithme 'QoS-$SkyTabou$' qui utilise la recherche Tabou [Glover, 1989]. L'algorithme a comme entrée l'ensemble de Skyline flou de chaque tâche.

## 1.4 Organisation du rapport

Le reste de ce rapport de thèse est constitué de deux parties principales : la première est intitulée « État de l'art », elle introduit le contexte dans lequel se place cette thèse et présente un état de l'art des travaux axés sur les deux problématiques P1 et P2. La seconde partie du document est intitulée « Contributions », elle présente l'ensemble des propositions énoncées dessus (C1,..,C5) et présente aussi le prototype de test développé ainsi que l'ensemble d'expérimentations effectuées. Le

<table>
<tr><th colspan="2">Partie 1 : Etat de l'art</th><th colspan="2">Partie 2 : Approches proposées</th></tr>
<tr><th>Problématiques</th><th>Contributions<br>(Chapitre 6)</th><th colspan="2">Expérimentations<br>(Chapitre 7)</th></tr>
<tr><td rowspan="2">P1 (Chapitre 5)</td><td>C1 (section 6.2.2)</td><td colspan="2" rowspan="2">Série d'expérimentations<br>E1 (section 7.5.1 )</td></tr>
<tr><td>C2 (section 6.2.3)</td></tr>
<tr><td rowspan="3">P2 (Chapitre 4)</td><td>C3 (section 6.3.3)</td><td colspan="2" rowspan="3">Série d'expérimentations<br>E2 (section 7.5.2)</td></tr>
<tr><td>C4 (section 6.3.4)</td></tr>
<tr><td>C5 (section 6.3.5)</td></tr>
</table>

***Tableau 1.2:*** *Correspondance entre problématiques et contributions ainsi que leurs chapitres et sections*

tableau 1.2 présente les chapitres couvrant les problématiques et les contributions correspondantes.

1. La partie I : État de l'art. Cette partie inclut quatre chapitres :
   - **Chapitre 2 : La technologie 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 en relation.
   - **Chapitre 3 : La sélection des services web dans une composition à base de QoS**

     Ce chapitre est consacré à l'analyse et à la définition de la problématique de sélection des services web dans une composition à base de paramètres de QoS (QoSWSC). Une classification des différentes approches existantes relatives à l'optimisation du problème QoSWSC est présentée. Cette classification regroupe 4 catégories qui sont : les approches de résolutions exactes, heuristiques, méta-heuristiques et celles utilisant les techniques de bases de données. Nous consacrons, la fin de ce chapitre à un état de l'art des travaux utilisant les approches exactes et heuristiques.

- **Chapitres 4 : La sélection de services web basée sur les méta-heuristiques**

  Ce chapitre est dédié à la classe des méta-heuristiques et leur utilisation dans l'optimisation du problème QoSWSC. Nous définissons, dans le début du chapitre, le principe des méta-heuristiques et leur classification. Nous présentons ensuite, un état de l'art comparatif des travaux sur le problème QoSWSC qui utilisent les méta-heuristiques.

- **Chapitres 5 : La sélection de services web basée sur les techniques orientées bases de données.**

  Ce chapitre est consacré aux techniques avancées des bases de données qui reposent sur les requêtes Top-k et Skyline et leurs applications dans le domaine de la sélection des services web. Dans un premier temps, nous définissons quelques notions de base, ensuite un état de l'art des travaux existants dans les deux domaines de Skyline et Top-k dominating et leurs applications dans la résolution de la problématique QoS est établi.

2. La partie II : Contributions. Cette partie regroupe deux chapitres :

   - **Chapitre 6 : Approches proposées.**

     Ce chapitre présente les cinq contributions énoncées plus haut. Pour chacune d'entre elles :

     – nous présentons nos motivations argumentées par rapport aux travaux cités dans l'état de l'art,

     – nous exprimons formellement nos besoins et nos objectifs et nous donnons des exemples explicatifs,

     – nous présentons les algorithmes proposés accompagnés, pour certains d'entre eux, d'exemples illustratifs.

   - **Chapitre 7 : Implémentation et expérimentations.**

     Ce chapitre présente le prototype de test que nous avons développé et nommé SoS-QoS 'Environnement de **S**élection et d'**O**ptimisation des **S**ervices web basée **QoS**'. Ce dernier implémente toutes les approches proposées et a été utilisé dans le but de les valider. Cette partie expose aussi une série d'expérimentations que nous avons réalisées sur une base de données réelles et une autre synthétique.

Nous terminons cette thèse avec un chapitre conclusion et perspectives.

# Partie I :
# État de l'art

Chapitre 2

# Technologies des services Web

## Sommaire

## 2.1 Introduction

Les services Web sont des technologies émergentes et prometteuses pour le développement, le déploiement et l'intégration des applications Internet. Dans ce chapitre, nous décrivons les Services Web en mettant en évidence la définition, les acteurs et les technologies des Services Web. Nous mettons aussi l'accent sur l'aspect non fonctionnel des services Web en présentant quelques paramètres de QoS. Nous terminons en donnant une vue générale de la composition des services Web et son cycle de vie, car nous nous intéressons dans cette thèse particulièrement à l'étape de sélection des services Web qui est une phase importante du processus de la composition.

## 2.2 Définition des services Web

Les services Web ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par le consortium W3C (World Wide Web Consortium). Plusieurs définitions sont alors utilisées pour présenter la technologie des Service WEB. Nous citons, celles d'IBM et W3C.

**Définition 2.1 - Service Web (Selon IBM [1]).**
Les services Web sont la nouvelle génération des applications web. Ce sont des applications auto-contenues, auto-descriptives et modulaires qui peuvent être publiées, localisées et invoquées depuis le web. Les services Web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services Web) peuvent le découvrir et l'invoquer.

**Définition 2.2 - Service Web (Selon W3C [2]).**
Un service Web est un composant logiciel identifié par une URI (Uniform Resource Identifier), et conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format exploitable par la machine (WSDL : Web Service Description Langage). D'autres systèmes interagissent

1. IBM Web services tutorial. Online : http ://www-106.ibm.com/developerworks/webservices/.
2. www.w3c.org

avec le Service Web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant HTTP (Hyper-Text Transfer Protocol) avec une sérialisation XML(eXtensible Markup Language) en même temps que d'autres normes du Web.

A partir de ces deux définitions on peut voir un service web comme étant une entité logicielle offrant une ou plusieurs fonctionnalités allant des plus simples aux plus complexes. Les services Web sont publiés, découverts et invoqués à travers le Web grâce à l'utilisation des protocoles de transport d'Internet comme infrastructure de communication et de standards ouverts : XML, SOAP, WSDL, UDDI. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI (Universal Description Discovery and Integration) pour lister les fournisseurs de services et les services disponibles. Cette interaction est représentée dans la figure 2.1

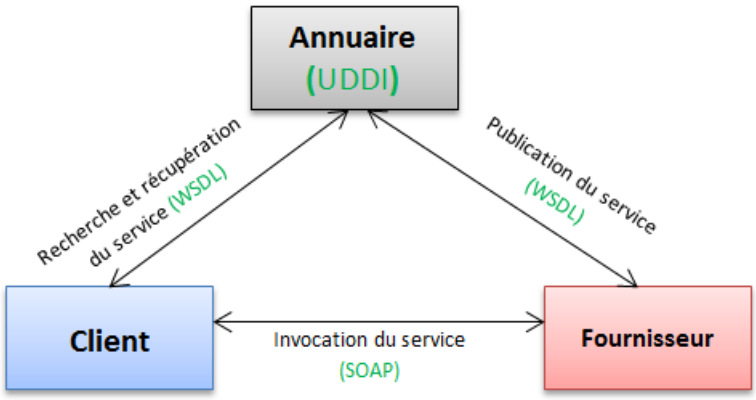



**Figure *2.1:*** *Interaction entre les différents acteurs des SW*

## 2.3 Les types des services Web

Les services Web peuvent être classés selon leur type d'invocation, leur description ou encore leur granularité.

1. **Selon le type d'invocation et le style de communication :**
   On distingue trois grands courants :
   (a) **Services Web de type RCP** : ils peuvent être vus comme l'application de RPC de (Remote Procedure Calls), sur le Web. Ce genre d'implémentation de service, même si elle est assez commune, n'est pas très recomman-

dée car les appels aux services sont souvent liés au langage d'implémentation. De ce fait, Le code du client est étroitement couplé avec la mise en œuvre du serveur.

(b) **Services Web basés sur REST** : ce type de services est en vogue ces dernières années et gagne de plus en en plus en popularité. Ils suivent l'architecture REST (REpresentational State Transfer), qui a été proposée par Roy Fielding [Fielding, 2000] comme un modèle pour le Web. L'interface de ces services est limitée à quatre méthodes HTTP : GET, POST, PUT et DELETE. L'ensemble des opérations du service est mis en œuvre en définissant différentes ressources sur lesquelles ces quatre méthodes peuvent être appelées. Les services Web basés sur REST n'ont pas forcément besoin de traiter les documents XML, ils utilisent des formats plus légers comme JavaScript Object Notation (JSON [3]).

(c) **Services Web basés sur SOAP** : ce type de services est le plus souvent utilisé dans l'industrie des sociétés commerciales du fait qu'elles aient construit de nombreux outils pour appuyer leur développement. SOAP est un protocole basé sur l'échange de messages XML pour les services Web, ce dernier sera décrit dans la section consacrée à la couche de communication des services Web (cf. 2.5.2).

2. **Selon leur description :**

(a) **Services Web Syntaxiques** : certains travaux qualifient aussi ces services par le terme 'classiques'. Ces derniers ne reposent que sur le profil fonctionnel sans intégration d'aucune autre description. La description syntaxique du profil fonctionnel contient les informations relatives aux méthodes proposées, leurs paramètres et les protocoles à utiliser. Ces informations permettent de faire l'appel du service.

(b) **Services Web Sémantiques** : un service Web est dit sémantique si sa description intègre, en plus de la description du profil fonctionnel, une dimension sémantique. Cette dimension englobe toute description complémentaire (du fournisseur, des objectifs du service Web, etc.) de la description fonctionnelle qui permet d'ajouter de l'information à la description classique du service Web. Cette information peut ensuite être traitée par des mécanismes (tels que l'inférence) afin d'en extraire de la connaissance.

3. Javascript object notation (json). http ://www.json.org/

Notons que l'aspect sémantique n'est pas traité dans cette thèse.

3. **Selon leur granularité :**
   (a) **Services Web atomiques** : un service est dit atomique, isolé ou seul s'il ne fait appel à aucun autre service Web dans son exécution.
   (b) **Services Web composites** : un service est dit composite ou composé s'il combine les fonctionnalités de plusieurs services Web au sein d'un même processus métier. Les services Web sont composés pour répondre à une demande complexe qu'un seul service ne pourrait satisfaire. La composition des services Web sera traitée dans la dernière section de ce chapitre, car nous nous intéressons dans cette thèse plus particulièrement à la sélection des services formant cette composition.

## 2.4 Les paramètres de Qualité des Services Web

Le sens de la Qualité de Service (QoS : Quality of Service) diffère d'une communauté à une autre. Dans la communauté des réseaux, la QoS peut traiter des questions comme les délais et les bandes passantes des liens du réseau, ou le nombre de paquets perdus dans les transmissions. Quand nous évoquons la QoS dans la communauté des services Web, nous raisonnons au niveau applicatif contrairement aux réseaux où les exigences en QoS concernent les couches basses des réseaux. Toutefois, même pour les paramètres QoS relatifs aux services web, là encore, nous trouvons aussi certains paramètres QoS relatifs au réseau. Par conséquent, il faut distinguer entre les paramètres QoS relatifs à l'application appelés aussi QoS de '*niveau application*' et ceux relatifs au réseau, tels que le paramètre délai de livraison de paquets, etc.

Le groupe de travail du W3C [Lee et al., 2003] a eu pour objectif d'identifier les paramètres de QoS applicables aux services Web. Ces paramètres QoS désignent l'aspect non-fonctionnel des services Web. Il n'existe pas un consensus bien précis au sujet de l'ensemble des paramètres QoS importants pour les services Web, mais la plupart des travaux de recherche, qui ont essayé d'identifier les paramètres QoS, ont pris en considération les paramètres définis par le W3C. Nous donnons ici une liste de certains de ces paramètres [Menascé, 2002], [Ran, 2003] les plus couramment utilisés par la communauté de chercheurs.

- **Le coût** (ou **le prix**) : désigne le prix que le client doit payer pour chaque invocation du service. Ce paramètre apparaît fréquemment dans les services Web commerciaux.

- **La latence** (aussi connue comme **le temps de réponse**) : ce paramètre est utilisé pour désigner le temps pris par un service pour répondre à une requête. Cela peut comprendre le retard causé par le réseau lors de l'appel au service Web.
- **Le débit** : le débit d'un service représente le nombre de demandes que le service est en mesure de traiter dans un intervalle de temps donné. Il peut inclure le débit maximal ou une fonction décrivant comment le débit varie en fonction de l'intensité de charge.
- **La fiabilité** (aussi connue comme **le taux d'exécution avec succès**) : ce paramètre désigne la capacité du service à accomplir sa fonction correctement pendant une période de temps spécifiée. Il peut être mesuré par le temps moyen entre pannes.
- **La disponibilité** : ce paramètre désigne la probabilité que le service soit actif. Il peut être exprimé par le ratio : $période active/période totale$. La période active représente le temps où le service est opérationnel en répondant aux demandes des clients. La période totale ($période active + période inactive$), elle désigne la durée durant laquelle le client souhaite que le service soit actif.
- **La sécurité** : la sécurité d'un service Web comporte différents aspects assurant que les échanges de messages entre le client et le service soient sécurisés. Plus précisément c'est un regroupement d'un ensemble de qualités à savoir : la confidentialité, le cryptage des messages et le contrôle d'accès.
- **La réputation** : c'est une mesure de la crédibilité du service. Elle dépend principalement des expériences d'utilisateurs finaux.

Ces paramètres peuvent être classés, selon leur mesure, en trois catégories [Araban and Sterling, 2004]

1. Métriques annoncées par le fournisseur : métriques dont les valeurs ont été annoncées par le fournisseur. Exemple : le coût d'un service.
2. Métriques évaluées par le consommateur : métriques dont les valeurs ont été données par l'utilisateur. Exemple la réputation.
3. Métriques observables : métriques dont les valeurs ont été obtenues par la surveillance ou le test. Comme exemple : le temps de réponse ou la disponibilité.

D'autres paramètres de QoS peuvent être trouvés dans la littérature, un grand nombre d'entre eux sont des variantes ou combinaisons des paramètres mentionnés ci-dessus.

## 2.5 Architecture en couches des services Web

Il existe deux types d'architectures pour représenter les services Web. La première est une architecture de base ou de référence qui permet de décrire les interactions et relations entre les trois acteurs principaux comme celle représentée dans la Figure 2.1. Cette architecture est traditionnellement utilisée pour les services Web isolés et reste insuffisante pour permettre une utilisation effective des services Web dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards. La seconde architecture vient compléter et étendre la première pour prendre en considération des processus plus complexes comme celui de la composition. Elle est constituée de plusieurs couches superposées, d'où son nom.

Différentes extensions de l'architecture de base ont été proposées dans la littérature. Le groupe architecture du W3C travaille activement à l'élaboration d'une architecture étendue standard. Nous présentons dans la figure 2.2 une vision globale de cette architecture en couches. Chaque couche de cette architecture s'appuie sur des langages, des protocoles et des modèles, nous distinguons des couches horizontales qui traitent les mécanismes fonctionnels et d'autres verticales pour les mécanismes non fonctionnels. Nous allons détailler dans ce qui suit les couches fonctionnelles ainsi que la couche QoS.

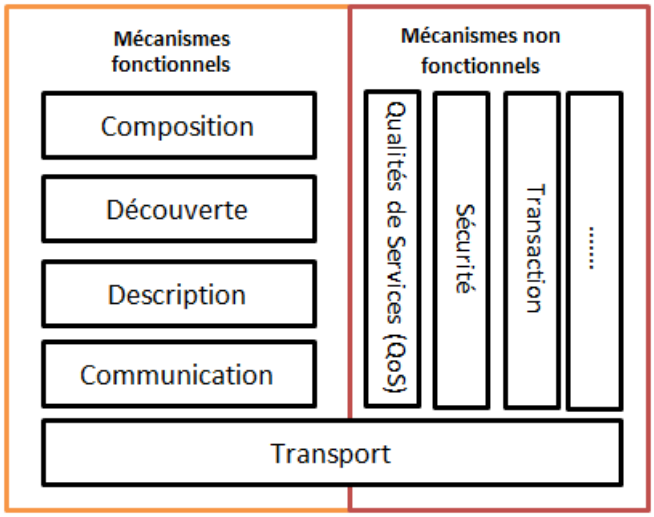



**Figure** ***2.2:*** *La pile des couches des Services Web*

### 2.5.1 Couche de transport des services

Cette couche regroupe les protocoles de transport de bas niveau, ces derniers servent à transporter les requêtes et les réponses échangées entre services. Le proto-

cole le plus utilisé est http, il est d'ailleurs recommandé par le consortium « Web Service Interoperability»), néanmoins d'autres implémentations peuvent utiliser d'autres protocoles comme HTTPS , SMTP , FTP , etc.

### 2.5.2 Couche de communication des services

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients, elle caractérise aussi le mode d'échange (s'il est bloquant ou non). Nous notons que selon le mode d'invocation, trois styles de services peuvent être distingués RPC, REST ou SOAP (cf. section 2.3). Les services Web SOAP restent actuellement les services les plus utilisés même si les services REST commencent à gagner du terrain. Tout au long de cette thèse, nous désignons par service web, les services Web SOAP, les deux autres ne font pas l'objet d'étude dans nos travaux. Nous présentons Le protocole SOAP qui est adopté comme un standard pour la messagerie entre les services Web.

- **SOAP (Simple Object Acces Protocol)**
  SOAP [Gudgin et al., 2003] a été développé par Microsoft et IBM. C'est un standard accepté par le W3C en 2000. SOAP est un protocole de communication basé sur XML qui permet aux services Web d'échanger des informations dans un environnement décentralisé et distribué en utilisant HTTP/SMPT/POP comme protocole de communication, tout en étant indépendant des plateformes de programmation utilisées. SOAP définit un ensemble de règles pour structurer les messages envoyés. Notons que la spécification du protocole SOAP ne donne aucune indication sur le mécanisme d'accès aux services Web. Cette indication est du ressort du langage WSDL qui sera défini dans la section suivante.
  SOAP est constitué des éléments suivants :
  - Entête des protocoles : les entêtes de protocoles (e.g. HTTP, SMTP, etc).
  - Enveloppe : définit un cadre général pour exprimer le contenu d'un message. Elle est elle même composée de deux autres éléments.
    * Entête : cet élément est optionnel. C'est un mécanisme d'extension qui fournit une manière de passer les informations en messages SOAP qui ne sont pas prises en compte directement par les applications. Il peut contenir, par exemple, des informations de sécurité (telles que les signatures électroniques), des informations transactionnelles, des informations de traçabilités,etc.

* Corps : est un élément obligatoire. Il contient les informations principales transmises dans un message SOAP. Il peut s'agir soit du nom de la méthode, avec les données correspondantes, ou d'un simple document XML pour le cas d'une requête. Le corps doit contenir aussi les valeurs de retour ou un message d'erreur pour le cas d'une réponse.

### 2.5.3 Couche de description des services

Cette couche permet de décrire le service web. Cette description concerne le profil fonctionnel/non fonctionnel du service web. Comme nous l'avons mentionné plus haut, les services Web peuvent être syntaxiques ou sémantiques, atomiques ou composites. La description est liée au type de service qui peut être, de ce fait :

1. une description syntaxique qui utilise des structures syntaxiques de type XML pour décrire les propriétés des services Web. Elle repose principalement sur le langage WSDL pour décrire la **structure** du service atomique. Elle repose aussi sur WS*-spécifications pour décrire le **comportement** au sein d'une composition. Nous citons la recommandation WS-CDL (Web Services Choreography Description Language) [Kavantzas, 2004] pour couvrir la description comportementale de la chorégraphie des services. Concernant le comportement du service lié à l'orchestration des services, OASIS (Organization for the Advancement of Structured Information Standards) propose le standard WS-BPEL (Web Services Business Process Execution Language) [Alves et al., 2006]. L'orchestration et la chorégraphie sont des modèles d'exécution de la composition. Ces deux modèles seront présentés dans la section composition des services Web. Nous présentons dans ce qui suit brièvement le standard WSDL.

   - **Web Services Description Language (WSDL)**
     WSDL [Chinnici et al., 2007] fournit un modèle et un format XML pour décrire des Services Web. WSDL permet de faire une séparation entre la description abstraite et concrète d'un service. Le niveau abstrait permet de répondre à la question : 'quelle' fonctionnalité est fournie ?. Tandis que la partie concrète permet de répondre à la question : 'comment' et 'ou' celle-ci est offerte ?. Par conséquent, le niveau abstrait regroupe les informations pouvant être réutilisées (non spécifiques à un service), tandis que le niveau concret est constitué de la description des protocoles d'accès au service Web (information particulière à un service). Le niveau abstrait

est utilisé principalement lors du processus de sélection. Il est constitué de types de données, des messages et des interfaces du document WSDL. Tandis que le niveau concret est seulement utilisé lors de l'invocation des méthodes du service Web. Il est constitué d'informations relatives aux ports d'accès et aux protocoles de communication ainsi que les liaisons de services (bindings).

2. une description sémantique qui est réalisée grâce à des annotations sémantiques de wsdl comme WSDL-S [Akkiraju et al., 2005] , SAWDL ou des ontologies de services comme l'ontologie OWLS [Martin et al., 2004] ou WSMO.

Nous définissons dans ce qui suit la facette structurelle et comportementale relatives à la description syntaxique.

### 2.5.4 Couche de découverte des Services

Un service web doit être d'abord référencé pour qu'il puisse être par la suite découvert et utilisé par des clients ou organisations. Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels. Nous décrivons dans ce qui suit le registre universel UDDI.

- **UDDI (Universal Description Discovery and Integration)**
  UDDI [Clement et al., 2004] (pour la version 3.02) est un standard pour décrire un annuaire de services Web. En plus de la description, UDDI est aussi un registre permettant de publier et de découvrir des services Web. Un registre UDDI est un registre basé sur XML qui contient des informations et des méta-données concernant les services Web. Trois types d'informations peuvent être décrits par ces registres XML :
  - Les pages blanches : décrivent les informations de contacts sur les entreprises. Elles sont utilisées pour trouver les services par contact car elles contiennent toutes les informations jugées pertinentes pour identifier l'organisation (telles que son nom, son adresse physique).
  - Les pages jaunes : décrivent des informations de classification de services. Elles sont utilisées pour trouver les services à l'aide des taxonomies car ces pages jaunes décrivent les services offerts par l'organisation, le type de services et les conventions d'utilisation de ces services.
  - Les pages vertes : donnent des informations techniques (telles que l'accès) des services. Elles sont utilisées pour trouver les services par leurs

caractéristiques techniques basées sur leur description WSDL.

### 2.5.5 Couche de composition des services

Cette couche sert a traiter le processus de composition qui permet d'agréger des services Web. La composition permet de construire de nouveaux services appelés services composites ou agrégats par assemblage de services déjà existants. Les services participant à la composition sont nommés services basiques ou élémentaires ou encore atomiques. En industrie, le concept de composition des services Web est réduit aux concepts d'orchestration et de chorégraphie. Dans ce contexte, les efforts industriels se sont concentrés sur la description de l'aspect comportemental des services Web composites, en développant des langages de composition spécifiques tels que WS-BPEL et WSCDL. Ces langages permettent de décrire des plans de composition (traduisant des comportements ou encore la manière dont les composants doivent être agrégés) pour qu'ils puissent, par la suite, être exécutés par des serveurs d'applications. Nous décrivons dans ce qui suit de manière brève les deux modèles de composition à savoir l'orchestration et la chorégraphie et nous donnerons les principes du langage BPEL (Business Process Execution Language) .

#### 2.5.5.1 Modèle comportemental du processus de la composition

Les propriétés comportementales décrivent la manière avec laquelle un service composite est construit. Ce comportement est lié au processus de la composition où l'information sur le comportement interne du service composite (la chorégraphie) et le comportement externe des services qui le composent (l'orchestration) est primordiale afin d'interagir avec le service composite.

- **Modèle d'orchestration**

  L'orchestration décrit le comportement externe d'un service composite. Elle désigne la manière dont les services invoqués sont agrégés au niveau de la composition afin de fournir une fonctionnalité plus complexe. L'orchestration permet de décrire l'enchaînement des services selon un canevas prédéfini [Sadiq and Racca, 2003], et de les exécuter comme étant un ensemble d'actions à réaliser par l'intermédiaire de services Web. Dans ce type de modèle, toutes les communications sont routées par un processus central appelé moteur d'orchestration ou moteur de workflow (voir figure 2.3).

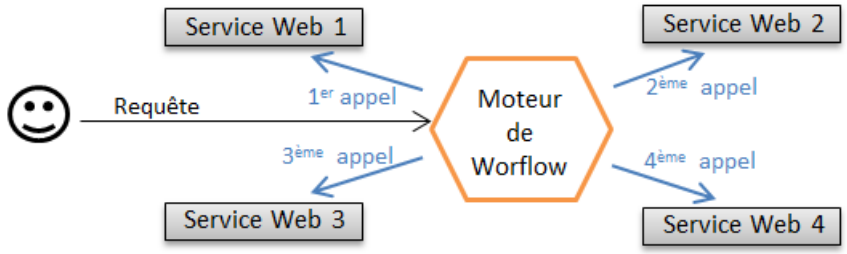



**Figure** ***2.3:*** *Vue générale de l'orchestration*

Le moteur d'orchestration aussi appelé 'coordinateur' prend le contrôle de tous les services web impliqués et coordonne l'exécution des différentes opérations des services composants qui participent dans le processus. Les services composants ne doivent pas être modifiés pour exécuter le workflow. Un exemple typique de langage d'orchestration est WS-BPEL [Jordan et al., 2007] (pour la version 2.0).

- **Modèle de chorégraphie**

  Contrairement à l'orchestration , il n'existe pas un coordinateur dans la chorégraphie, elle est basée sur la collaboration entre les services web (voir figure 2.4) où chaque partie de la composition possède un contrat ou des accords avec le reste des services qui dictent la manière de collaborer.

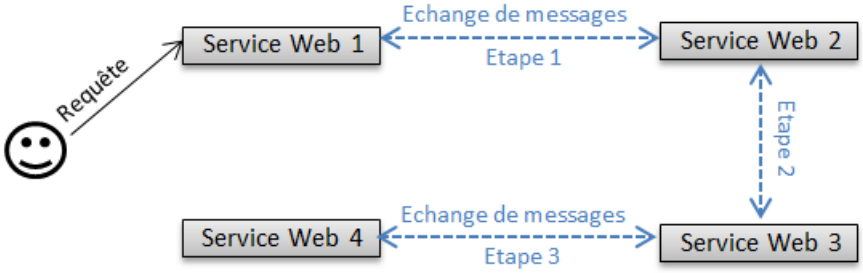



**Figure** ***2.4:*** *Vue générale de la chorégraphie*

La chorégraphie peut être considérée comme une composition dynamique [Peltz, 2003]. En effet, à chaque étape du processus de la composition, un service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance. Un exemple typique de langage de chorégraphie est la spécification WSDCL (Web Service Description Choregraphy Language) [Kavantzas, 2004], nous notons que ce type de compositions n'est pas très utilisé, de ce fait, il y a peu d'implémentations de la spécification WSDCL.

#### 2.5.5.2 Langage BPEL

Plusieurs langages de composition de services ont été proposés dans la littérature, mais un seul d'entre eux est devenu le formalisme le plus utilisé, c'est le standard BPEL ou BPEL4WS [Andrews et al., 2003] (pour la version 1.0) ou [Jordan et al., 2007] (pour la version 2.0). BPEL est basé sur XML et sur les Workflows. Il permet de modéliser les processus métiers en terme d'orchestration, en décrivant le flux de données (les variables) et le flux de contrôle (les activités simples et composées, les partenaires...), son but est de créer une fonctionnalité complexe qui réutilise les services existants. Il permet aussi de donner une vue centralisée de l'exécution de la composition. Ce langage distingue les processus abstraits des processus exécutables.

Le processus abstrait décrit les interactions publiques de messages entre les services web composants sans obligation d'indiquer le comportement interne de chacun d'eux. Par opposition, le processus exécutable montre l'ordre d'exécution des activités, des services invoqués, des messages échangés et des mécanismes de gestion des erreurs potentielles. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les services composants. Deux catégories d'activités peuvent être distinguées dans bpel, les activités primitives et les activités structurées. Les activités primitives (Receive, Reply, Assign, Throw et Wait) sont des activités simples utilisées pour décrire comment le processus effectué par le service Web composite doit réagir vis-à vis des messages échangés. Par ailleurs, les activités structurées (Sequence, Flow, while Switch ou Split) sont des activités complexes qui combinent plusieurs activités primitives pour décrire différents types de branchement dans un processus.

### 2.5.6 Couche de qualité de services

Il n'existe pas de langage dédié exclusivement à la spécification de la QoS des services web. Cependant, il est nécessaire d'avoir une spécification claire et non-ambigüe de la QoS du service afin de permettre leur évaluation. Deux possibilités sont offertes pour faire cette spécification.

#### 2.5.6.1 L'extension des langages de description des services par la QoS

Les langages de description de service comme UDDI et WSDL ne concernent que les aspects fonctionnels du service, de nombreuses propositions ont été faites pour renforcer ces spécifications afin de permettre la description des paramètres QoS du

service. Une des contributions les plus en avant, vient du groupe ouvert OASIS. Le groupe a proposé la prolongation du WSDL par la QoS appelée WSQDL (Web Service Quality Description Language). Nous trouvons aussi, l'extension de WSDL nommée Performance-enabled WSDL (P-WSDL) [D'Ambrogio and Bocciarelli, 2007] et l'extension d'UDDI nommée (UX) [Chen et al., 2004]. Dans ces deux formalismes, la QoS d'un service est modélisée comme un tuple de paramètres de QoS, en spécifiant une valeur (ou un intervalle de valeurs) pour chaque paramètre.

#### 2.5.6.2 Utilisation des contrats de QoS

On parle aussi de Service Level Agreements (SLA), les contrats sont des accords conclus entre le fournisseur et le client d'un service concernant la QoS du service. Les contrats peuvent préciser les obligations du fournisseur et du client. Par exemple, un contrat peut dire que "à condition que le client fasse au plus cinq demandes par seconde, le fournisseur assure que ces demandes soient traitées en moins de 100 millisecondes". La première partie de cette clause est une obligation que le client doit respecter et la seconde est une obligation du fournisseur. Un contrat peut avoir plusieurs clauses de ce genre, qui décrivent ensemble la QoS du service. Les contrats sont généralement négociés en off-line, avant que le service ne soit appelé par le client. Toute méthode de gestion de QoS impliquant des contrats est accompagnée par des techniques de monitoring, pour veiller à ce que les obligations dans le contrat soient respectées. WSLA [Keller and Ludwig, 2003] et WS-Agreement [Andrieux et al., 2007] sont deux cadres répandus pour la spécification des contrats pour les services Web.

## 2.6 La composition et les services web composites

La composition de services Web (WSC : Web Service Composition) est un processus complexe qui va de la découverte des services jusqu'à leur composition pour répondre à une requête particulière.

[Casati and Shan, 2002] définissent la composition des services Web comme la capacité de constituer de nouveaux services Web composites à valeur ajoutée en combinant des services Web existants et probablement offerts par différents fournisseurs.

[Benatallah et al., 2005], quant à eux, considèrent la composition des services Web comme un moyen efficace pour créer, exécuter et maintenir des services qui dépendent d'autres services.

[Martin et al., 2004] considèrent la composition comme le processus de sélection, de combinaison et d'exécution de services en vue d'accomplir un objectif donné.

Nous pouvons définir, de façon générale, la composition de services Web comme étant un processus complexe qui consiste à trouver des composants appropriés, de les sélectionner, de les combiner, et de fournir des informations d'ordre comportemental renseignant sur leur orchestration ou chorégraphie en vue de les rendre exécutables. Le résultat de cette composition est un service web composite faisant interagir un ensemble de services web afin de satisfaire la requête de l'utilisateur qui ne peut l'être par un service atomique.

### 2.6.1 Nature de la composition des services

La combinaison des services web atomiques, en vue de répondre à une requête complexe, peut être réalisée de manière statique ou dynamique avec un degré d'automatisation plus au moins différent. Ces critères permettent d'identifier la nature de la composition de services.

#### 2.6.1.1 Statique vs. dynamique

Dans la méthode statique, la combinaison des services Web est réalisée à la phase de conception du service composite. Les services Web sont reliés entre eux, avant d'être déployés. Dans ce genre d'approche, la substitution ou la modification d'un service composant nécessite d'apporter des modifications au niveau de la conception du service composite. De ce fait, ce type de composition est plus adéquat pour les environnements qui ne sont pas sujet à des changements rapides dans le temps.

Contrairement à l'approche statique, la composition dynamique permet de réaliser la combinaison des services Web au moment de l'exécution de la requête. L'approche de composition dynamique permet de sélectionner et combiner dynamiquement les services à partir des annuaires de services Web, tout en tenant compte des contraintes de l'utilisateur.

#### 2.6.1.2 Manuelle vs. semi-automatique ou automatique

En dehors de l'aspect statique ou dynamique de la composition, un autre aspect peut définir la nature d'une composition en se basant sur le degré de participation de l'utilisateur dans le schéma de composition des services Web. Le degré de participation

de l'utilisateur détermine le degrés d'automatisation de la composition qui peut aller de la composition manuelle à celle totalement automatisée.

La composition manuelle suppose l'intervention totale de l'utilisateur sans l'intervention d'outils spécialisés. L'utilisateur se servira d'un simple éditeur texte pour programmer toutes les étapes du processus de la composition. Ce type de composition exige de l'utilisateur qu'il ait le profil développeur, faute de quoi, la composition devient une tâche fastidieuse où d'importants efforts sont à fournir de la part d'un utilisateur novice.

Par opposition à la composition manuelle, la composition totalement automatique ne requiert pas l'intervention de l'utilisateur et prend en charge tout le processus de composition et le réalise automatiquement.

L'approche semi-automatique est un compromis entre l'approche manuelle et automatique. Elle permet à l'utilisateur de composer les services en s'appuyant sur des outils dédiés. Cette façon de faire permet à l'utilisateur de garder un certain contrôle et de superviser le processus de composition sans qu'il ait pour autant un profil de programmeur en s'appuyant sur l'assistance fournie par les outils graphiques.

### 2.6.2 Techniques de composition des services web

Les deux courants principaux dans les approches de WSC sont : les approches à base de workflow et les approches à base de planification en intelligence artificielle.

#### 2.6.2.1 Composition à base de Planification en intelligence artificielle

Dans les cas de WSC à base de planification, la composition de services est vue comme un problème de planification. Généralement, un problème de planification AI [Russell et al., 2003] est défini par les éléments suivants :

- la description de l'état initial($S_0$) ;
- la définition de l'état ($S$) qui est une description de l'ensemble des états possibles.
- l'état final ou l'état but ($G$) que l'on souhaite atteindre.
- l'ensemble des actions possibles ($A$) qui permettent le changement d'états.
- l'ensemble des transitions d'états possibles($R$). Il s'agit de trouver l'ordre des actions qui transformeront l'état initial à l'état final.

Dans le domaine de SWC, un service composite est représenté comme un état final qui doit être atteint et les services candidats sont représentés comme l'ensemble

des actions qui peuvent transformer les états. À la fin d'un exercice de planification réussi (l'existence d'une composition), un plan est produit. Ce plan constitue les services Web choisis et l'ordre de leur exécution de telle façon qu'ils produisent la fonctionnalité exigée.

#### 2.6.2.2 Composition à base de workflow

Les méthodes à base de workflows utilisent le principe qu'un service Web composite peut être considéré conceptuellement comme similaire à un Workflow. Par analogie à un workflow d'activités, un service composite correspond notamment à un ensemble de services atomiques ou composites interconnectés. De ce fait, l'exécution d'un service Web composite s'avère similaire à l'exécution d'un Worfklow dont les mécanismes assurent la flexibilité, l'adaptation et l'intégration de processus automatiques. Les frameworks de composition fondés sur la technique de Workflows ont été les premières solutions proposées pour la composition de services Web, ils peuvent être statiques ou dynamiques.

Dans le cas statique, l'utilisateur établit manuellement un modèle abstrait des tâches, en plus des dépendances. (i.e. le flux de contrôle, le flux de données, ..etc). Chaque tâche contient une clause ou question qui est employée pour rechercher le service concret qui l'accomplit. Ce travail est fait pendant la conception. Durant l'exécution, le système de composition instancie chaque tâche, en cherchant un service concret. La construction des schémas de composition de type statique s'appuie généralement sur des langages de description comportementale pour préciser les interactions entre fournisseurs de services Web et leurs clients. Nous citons à titre d'exemples les langages tels que BPEL [Jordan et al., 2007], ou BPMN [Group et al., 2006].

Par contre, les méthodes à base de workflows dynamiques génèrent les graphes de tâches instanciées à la volée (pendant le runtime). La composition à base de workflows dynamiques crée le modèle de tâches et l'instanciation des services de façon automatique. Cette composition est qualifiée comme étant orientée processus ou comportement [Rao and Su, 2004], tous les services Web considérés possèdent des états (statefull web services). Contrairement aux services web sans états (stateless web services), dont le schéma d'interaction est trop simplifié (i.e. de type requête réponse), les services web avec états ont une interaction complexe, et les résultats finaux ne sont pas complètement prévisibles (non déterministes). Cette interaction est généralement modélisée avec des systèmes formels, tels que les automates d'états

finis, les réseaux de petri,. . . ,etc. ) [Lécué, 2008].

L'interaction d'un service avec état représente un protocole multi-phases, qui peut englober des actions internes, des échanges de messages avec les partenaires, des accès aux bases de données. L'intérêt des méthodes formelles (systèmes formels) est de donner la possibilité de vérifier automatiquement des propriétés d'un système (code logiciel, modèle conceptuel,. . . ,etc.). Ils peuvent être adoptés pour générer les workflows dynamiques. Ils peuvent aussi vérifier et prouver la compatibilité de la composition avec les besoins de l'utilisateur.

Généralement, les compositions à base de planification AI ne reposent pas sur une vue abstraite de la composition à l'exception faite des travaux de [McIlraith and Son, 2002]. Au contraire, cette vue abstraite de la composition est un élément clé dans les approches WSC basées workflow.

Nous considérons dans cette thèse, la phase de sélection d'une composition basée sur les workflows. Pour cela, nous définissons dans la section suivante le cycle de vie d'une composition qui repose sur une solution typiquement workflow.

### 2.6.3 Cycle de vie d'une composition

La composition est un processus qui passe par plusieurs étapes. Nous présentons dans la figure 2.5 un cycle de vie qui montre l'enchainement des étapes d'une composition basée workflow inspirée du cycle de vie présentée dans [Moghaddam and Davis, 2014] où la vue abstraite et concrète d'une composition est mise en évidence.

Dans ce cycle de vie, la première étape est la spécification des besoins. Ces besoins sont définis sous forme de préférences fonctionnelles et non-fonctionnelles (QoS) ; elles concernent les préférences des utilisateurs par rapport aux services demandés. Les préférences sont décomposées (semi) automatiquement dans un processus abstrait (workflow) qui correspond à la composition abstraite. Cette dernière comprend un ensemble de tâches, chacune représente une fonctionnalité distincte. Les Qualités de Service (QoS) pour le BP aussi bien que pour chaque tâche participante sont aussi spécifiées. Durant l'étape de découverte, les services Web concrets qui correspondent aux exigences fonctionnelles et non fonctionnelles des tâches sont localisées. La localisation des services est faite à travers des recherches effectuées dans des registres tels que UDDI ou l'API 'programmable web' [4].

La phase de découverte de services permet de trouver à chacune des tâches parti-

4. //www.programmableweb.com

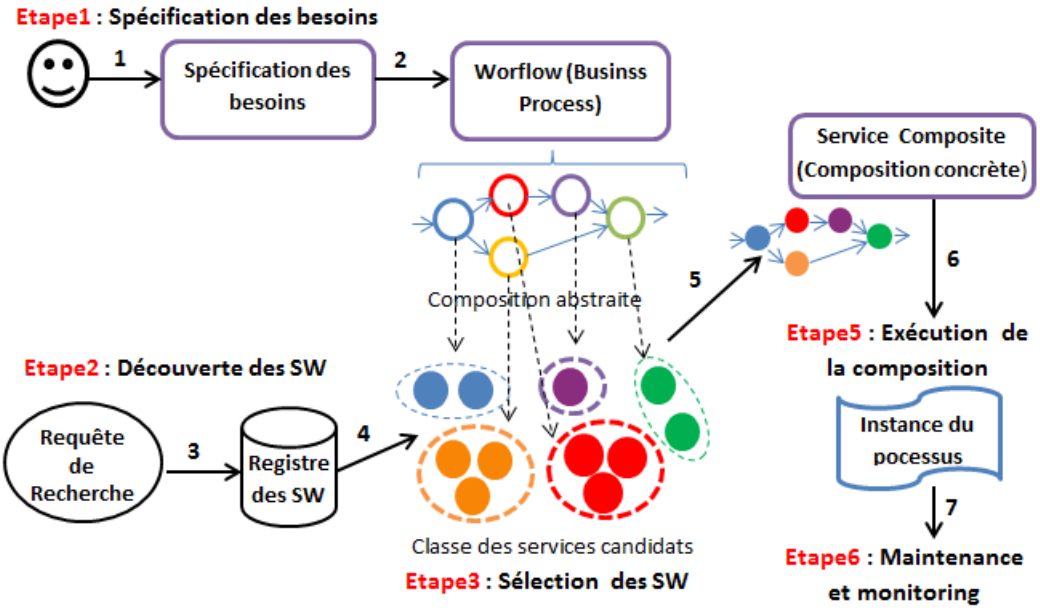



**Figure** ***2.5:*** *Cycle de vie d'une composition*

cipantes les services correspondants. À cette étape, il est très probable que plus d'un service candidat soit trouvé pour chaque tâche avec des attributs de QoS différents.

La sélection de services Web est l'étape qui vient juste après la découverte. À cette étape, une variété de techniques est proposée par la communauté de recherche pour aider le demandeur de service à choisir des services selon les exigences indiquées. Après avoir trouvé les services candidats pour toutes les tâches et lier chaque tâche à son service Web choisi, le service composite concret est créé.

Pendant l'étape d'exécution, une instance du processus est créée en exécutant le service composite. La dernière étape de monitoring permet de contrôler le processus continuellement pour de nouvelles réponses en cas d'un quelconque échec (service indisponible) dans le but de maintenir le bon déroulement de la composition.

La phase de la sélection des services, est considérée comme une étape fondamentale dans le cycle de vie de la composition. Elle fait l'objet de beaucoup de recherches par la communauté SOC (Service Oriented Computing). Elle à été adressée à deux niveaux : (i) pour une seule tâche ; (ii) pour le workflow en entier (la composition). La problématique de la sélection est au cœur des travaux de cette thèse. Le prochain chapitre fera l'objet d'étude et d'analyse de la problématique de la sélection des services web.

## 2.7 Conclusion

Nous avons présenté dans ce chapitre, une vue d'ensemble des services web tout en relatant leurs technologies et leur fonctionnement. Ces technologies améliorent l'intégration et l'interopérabilité entre services à travers l'infrastructure Web en utilisant différents standards XML : SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services Web. Nous avons abordé aussi l'aspect non fonctionnel des services web qui est défini par les paramètres QoS.

Nous avons enfin consacré la dernière section à la composition des services Web. Cette dernière permet de combiner des services existants pour former de nouveaux services plus élaborés. Nous nous sommes intéressés plus particulièrement à la composition qui se base sur les workflow. Nous avons détaillé le cycle de vie de ce type de composition dans le but de mettre en évidence la phase de sélection. Cette phase est une étape importante dans la composition des services Web et constitue une problématique de recherche majeure dans la communauté SOC. Elle fera l'objet du chapitre suivant.

Chapitre 3

# La sélection des services Web dans une composition à base de QoS

*"Un problème sans solution est un problème mal posé."*

Albert Einstein

## Sommaire

## 3.1 Introduction

Les tâches composantes d'une application web peuvent être exécutées à l'aide des services web. En particulier, pour une même tâche, nous pouvons découvrir plusieurs services web capables de l'exécuter. Par exemple, pour la tâche de vérification de données d'un client, plusieurs fournisseurs sur le marché des services web peuvent offrir de tels services (exp. Service Objects [1], CDYNE [2], Informatica [3], ou encore PostcodeAnywhere [4]). Cependant, ces services diffèrent dans leurs qualités de services. Nous nous trouvons alors face à un problème de sélection de services Web afin de choisir le service ou la composition de services qui offrent la meilleure qualité.

La sélection à base de Qualités de Services (QoS) consiste à choisir parmi les services Web découverts pour chaque tâche, ceux qui répondent au mieux aux exigences de l'utilisateur sur la base des besoins non fonctionnels QoS. La sélection des services Web basée sur la QoS dépend de la spécification adoptée lors de la définition des critères QoS et du profil QoS du service Web. Deux importantes catégories peuvent alors être considérées [Moghaddam and Davis, 2014] : la première est basée sur la négociation, la deuxième est basée sur l'optimisation.
Les approches basées sur la négociation permettent au profil QoS d'être flexible et négociable. Par contre, les approches basées sur l'optimisation assument typiquement un profil prédéterminé où les paramètres QoS ne sont pas personnalisables. Dans ce cas, le problème de sélection des services Web est souvent considéré comme un problème d'optimisation. Ce dernier est apparu sous différents noms comme *'QoS-driven or QoS-aware web service composition' QoSWSC* ou *'web service composition optimization'*. Dans notre travail, nous nous sommes intéressés à la deuxième catégorie en adoptant l'appellation 'QoSWSC' pour désigner le problème de sélection de services web basée sur les paramètres QoS, plus de détails sur la première catégorie peuvent être trouvés dans [Moghaddam and Davis, 2014].

La sélection des services Web est l'une des problématiques les plus importantes de l'architecture orientée service. Elle constitue aussi l'une des étapes les plus importantes dans le processus de composition. Nous commençons le chapitre par un exemple représentatif qui permet d'analyser et de définir cette problématique en relevant les principaux défis liés à cette dernière. Nous donnons ensuite une classification des dif-

1. www.serviceobjects.com
2. http ://cdyne.com/
3. https ://marketplace.informatica.com
4. http ://www.pcapredict.com/

férentes approches existantes relatives à l'optimisation du problème QoSWSC. Cette classification regroupe 4 catégories qui sont : les approches de résolutions exactes, heuristiques, méta-heuristiques et celles utilisant les techniques orientées bases de données. Nous présentons dans ce chapitre un état de l'art relatif à la catégorie des approches exactes et heuristiques, les deux prochains chapitres traitent respectivement les deux dernières catégories. Nous présentons aussi des tableaux répertoriant les travaux selon la stratégie utilisée et la méthode d'optimisation adoptée. Nous terminerons le chapitre par une synthèse.

## 3.2 Exemple représentatif du problème QoSWSC

La composition des services est le plus souvent modélisée comme des workflows au moment du design (conception). Un langage tel que BPEL [Jordan et al., 2007], ou BPMN [Group et al., 2006] peuvent être utilisés pour modéliser la représentation abstraite. Pour chaque tâche abstraite, nous découvrons un ensemble des services équivalents qui accomplissent les fonctions exigées. Au temps d'exécution, une sélection de service Web basée sur QoS est exécutée pour choisir un service pour chaque tâche tel que les exigences globales de l'utilisateur soient satisfaites.

Étant donné un workflow abstrait, notre question principale est : comment trouver la composition optimale ? ; c'est-à-dire, comment trouver l'ensemble de services candidats qui respectent les contraintes d'utilisateurs (les critères globaux) et optimisent les paramètres de QoS. Afin de mieux expliciter la problématique de sélection QoSWSC reprenons l'exemple cité dans notre proposition [Halfaoui et al., 2017]. L'exemple présente le service composite de traitement de fichiers :

Supposons qu'un étudiant ne maitrisant que la langue arabe trouve deux références, la première est un fichier écrit en anglais dans un format doc, la deuxième est un fichier écrit en arabe dans un format latex. Pour obtenir une meilleure compréhension, il veut traduire le premier fichier vers l'arabe et le fusionner avec le deuxième dans un format pdf. La demande de l'utilisateur est rarement satisfaite par un seul service Web , mais nécessite, par contre, plusieurs services. L'exemple est représenté dans la Figure 3.1 sous forme d'un diagramme d'activité UML qui spécifie l'ordre d'invocation des services composants (ici représentés par les tâches nommées $T_1..T_6$).

La composition commence par la traduction du premier fichier de l'anglais vers l'arabe et sa conversion du format doc à pdf, qui est faite en parallèle avec la conversion du latex à pdf du deuxième fichier. Après cela, les deux fichiers sont fusionnés

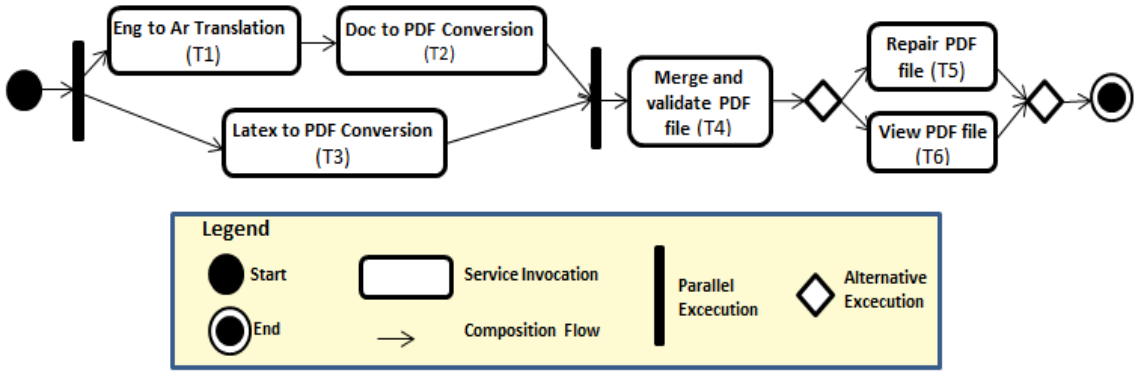



**Figure *3.1:*** *Un workflow abstrait : Le service de traitement des fichiers*

et vérifiés pour leur validité. Enfin, nous invoquons le dernier service pour visionner le fichier PDF ou, en cas d'erreur, pour réparer le fichier PDF. Il y a beaucoup de services web disponibles pour chaque tâche (par exemple, PDFFusionner, PDFMerge, etc pour $T_4$) mais ils diffèrent dans leurs attributs de QoS.

Le Tableau 3.1 montre les fournisseurs de services disponibles (les services candidats) pour chaque tâche et leurs attributs QoS correspondants. Nous notons que seulement un service peut être utilisé pour exécuter chaque tâche. Dans cet exemple, nous avons trois services candidats par tâche et 3 propriétés $QoS$ : $q_1, q_2, q_3$ (qui correspondent respectivement au coût, le temps d'exécution et la disponibilité). Les préférences de l'utilisateur peuvent concerner des contraintes locales (par exemple, choisir un service de chaque tâche avec le meilleur temps d'exécution) et des contraintes globales, comme le prix total de la composition n'excède pas 3 €. Les valeurs des propriétés $QoS$ globales pour les services composites dépendent de :

- La structure de la composition (parallèle, séquence, etc) ;
- Le service sélectionné pour chaque tâche.

Par exemple, supposons que le fichier fusionné ne contient pas d'erreurs, alors la tâche $T_5$ n'est pas requise. Nous aurons, dans ce cas, le plan d'exécution suivant : $< T_1, S_{13} >, < T_2, S_{21} >, < T_3, S_{31} >, < T_4, S_{42} >, < T_6, S_{61} >$. Où le service $S_{13}$ est choisi pour réaliser la tâche $T_1$, le service $S_{21}$ est choisi pour réaliser la tâche $T_2$, etc. Le coût total de la composition que nous venons de spécifier est : $CoûtTotal = q_1(S_{13}) + q_1(S_{21}) + q_1(S_{31}) + q_1(S_{42}) + q_1(S_{61}) = 0.3 + 1 + 0.4 + 0.9 + 0.9 = 3.5$ \$.

Le problème QoSWSC consiste en la sélection d'un service pour chaque tâche tel que les valeurs de QoS composées soient optimisées et les contraintes globales de l'utilisateur soient respectées. Ce problème devient particulièrement important quand le nombre de services candidats augmente ; notons que pour ce petit exemple

| **Task** | **Service** | $\mathbf{q_1}$(\$) | $\mathbf{q_2}$(milli-sec) | $\mathbf{q_3}$(%) |
|---|---|---|---|---|
| $T_1$ | $S_{11}$ | 0.8 | 120 | 80 |
| | $S_{12}$ | 1.5 | 10 | 98 |
| | $S_{13}$ | 0.3 | 10 | 98 |
| $T_2$ | $S_{21}$ | 1 | 140 | 70 |
| | $S_{22}$ | 1 | 145 | 90 |
| | $S_{23}$ | 3 | 160 | 75 |
| $T_3$ | $S_{31}$ | 0.4 | 150 | 99.9 |
| | $S_{32}$ | 0.3 | 200 | 98 |
| | $S_{33}$ | 1 | 150 | 86 |
| $T_4$ | $S_{41}$ | 0.0 | 200 | 99 |
| | $S_{42}$ | 0.9 | 120 | 90 |
| | $S_{43}$ | 0.3 | 120 | 90 |
| $T_5$ | $S_{51}$ | 0.7 | 50 | 99 |
| | $S_{52}$ | 0.3 | 100 | 98 |
| | $S_{53}$ | 0.8 | 15 | 77 |
| $T_6$ | $S_{61}$ | 0.9 | 200 | 99 |
| | $S_{62}$ | 0.7 | 15 | 98 |
| | $S_{63}$ | 0.3 | 10 | 98 |

***Tableau 3.1:*** *Services candidats pour le traitement de fichiers et leurs QoS*

nous avons $3^6$ combinaisons à examiner c-à-d 216 plans d'exécution à vérifier pour en choisir la plus optimale. Trouver la solution exacte prendra un temps exponentiel avec l'augmentation des services candidats par classe.

## 3.3 Analyse et définition du problème QoSWSC

Comme nous l'avons souligné précédemment, nous considérons le problème QoSWSC comme un problème d'optimisation. Un problème d'optimisation se définit comme la recherche du minimum ou du maximum (l'optimum) d'une fonction donnée, généralement nommée **fonction objectif ou fonction d'utilité**. Les variables de cette fonction sont souvent contraintes d'évoluer dans une certaine partie de l'espace de recherche. Nous aurons donc un problème d'optimisation sous contraintes. Généralement un problème d'optimisation est caractérisé par :

- **la nature de la fonction objectif à optimiser** : le problème à optimiser peut être, selon la nature de la fonction, soit linéaire ou non linéaire ;
- **le nombre de fonctions objectifs à optimiser** : si la fonction est scalaire nous parlons alors de problème mono-objectif. Par contre, si la fonction est vectorielle alors le problème est dit multi-objectif ;
- **la présence des contraintes** : nous parlons de problème avec ou sans contraintes.

- **sa taille** : selon la taille de l'espace de recherche à considérer, le problème est de petite ou de grande taille ;
- **le domaine des variables de décision** : selon le domaine de définition des variables de décision, le problème est soit **continu**, ou bien **discret**, dans le cas de variables d'entier. Quand le problème consiste en un nombre fini de permutations de nombre alors le problème est dit **combinatoire** ;

### 3.3.1 Modélisation multi-objectif vs. mono-objectif

Le problème QoSWSC peut être défini, selon la formulation de sa fonction objectif, comme un problème mono-objectif ou multi-objectif. Dans le cas où on considère un seul paramètre QoS nous serons face à un problème mono-objectif où le problème sera d'optimiser la qualité considérée. Par contre, la majorité des travaux actuels prend en compte, dans la sélection des services Web, plusieurs paramètres QoS. Dans ce cas, le problème QoSWSC sera considéré comme étant multi-objectif. Deux manières de traiter le problème multi-objectif QoSWSC alors sont à considérer :

1. La transformation du problème QoSWSC multi-objectif en un problème mono-objectif : le problème QoSWSC est transformé en un problème d'optimisation mono-objectif. Une des approches les plus utilisées est l'approche d'agrégation [Ishibuchi, 1998] qui consiste à définir une fonction objectif unique F comme étant la somme pondérée des différentes fonctions objectifs du problème initial. Dans ce cas de figure, deux phases sont à considérer :
    - Phase de normalisation : Elle consiste à ajuster les critères de qualité de valeurs positives (comme la disponibilité) et négatives (comme le temps de réponse) en appliquant des équations de normalisation afin de rendre les valeurs des critères de QoS homogènes.
    - Phase d'attribution de poids : il s'agit d'attribuer un coefficient de poids à chaque critère de QoS. Le coefficient de poids d'un QoS représente son importance. La fonction mono-objectif sert à calculer le score sous forme d'une somme pondérée pour chaque service web (phase locale) ou pour la composition totale (phase globale où les QoS correspondent aux qualités de la composition). Le service web qui a le score le plus élevé (dans le cas de maximisation du problème) est celui qui est choisi pour exécuter la tâche en question dans la phase locale et la composition qui a le score le plus haut est sélectionnée dans la phase globale.

2. L'expression du problème n'est pas modifiée : le problème QoSWSC reste un Problème Multi-Objectif (PMO) où la fonction objectif pourra être représentée par un vecteur de $N$ fonctions objectifs. Les contraintes sont représentées par des vecteurs regroupant les contraintes de chaque fonction objectif. Les paramètres QoS sont les critères multi-objectif à optimiser où chaque paramètre QoS à optimiser lui est associé une fonction objectif. Ces fonctions objectifs peuvent éventuellement être contradictoires selon les paramètres QoS à considérer, comme par exemple la qualité temps de réponse et la qualité fiabilité. L'objectif est alors d'optimiser simultanément l'ensemble des fonctions objectifs QoS du vecteur.

   Dans ce cas, La phase d'attribution de poids n'existe pas. En effet, la solution optimale ou de bonne qualité n'est plus une solution unique de nature numérique, mais un ensemble de solutions compromis entre les différents objectifs QoS à optimiser. Par contre, il est vital pour identifier ces meilleurs compromis, de définir une relation d'ordre entre ces éléments. La plus célèbre et la plus utilisée est la relation de dominance au sens Pareto. Par ailleurs, nous pouvons trouver aussi une résolution qui ne repose pas sur la notion Pareto comme la sélection parallèle ou lexicographique.

### 3.3.2 Stratégie de sélection locale vs globale

L'une des raisons principales qui rend le traitement du problème QoSWSC compliqué est la présence de deux types de contraintes QoS : les contraintes locales appliquées aux services composants et les contraintes QoS globales appliquées au service composite en entier. Idéalement, la solution recherchée devrait satisfaire ces deux contraintes avant le choix d'une solution optimale, ce qui accroit la difficulté du problème d'optimisation. Cette optimisation utilise deux stratégies principales : une stratégie de sélection locale et une stratégie de sélection globale. Chaque stratégie est définie en fonction de la nature des contraintes QoS imposées. Ces dernières permettent de définir les limites de fonctionnement d'un service web composant ou composite.

- La stratégie de **sélection locale** a pour objectif de choisir le meilleur service web pour chaque tâche individuelle à part entière. La stratégie de sélection locale considère seulement des contraintes de QoS relatives à chaque tâche sans pour autant considérer des contraintes de QoS globales exprimées pour l'ensemble

des tâches. L'optimisation du problème QosWSC, dans ce cas, traite les tâches abstraites du workflow de manière individuelle.

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

- La stratégie de **sélection globale** a pour but de choisir la composition de services web qui garantit la meilleure qualité globale en tenant compte des contraintes de QoS des préférences globales assignées pour l'ensemble des tâches. Le problème de sélection QoSWSC peut être modélisé comme un problème d'optimisation combinatoire. Il s'agit de trouver les services web candidats qui peuvent satisfaire une ou plusieurs contraintes de QoS globales tout en gardant comme objectif l'optimisation (maximiser ou minimiser) de la fonction d'utilité.

Ces dernières années, quelques travaux tentent d'utiliser les deux stratégies (Alrifai et al., 2012), (Surianarayanan et al., 2014), ce qui nous ramène à une troisième stratégie qui est la sélection hybride. Cette dernière, combine la sélection locale pour réduire l'espace de recherche dans la sélection globale et ne considérer ainsi qu'un sous intervalle des listes de services candidats de chaque classe tout en respectant les contraintes globales.

### 3.3.3 Complexité du problème QoSWSC

Il est très utile de comparer le problème QoSWSC à des classes de problèmes académiques d'optimisation célèbres, qui ont été examinés et analysés par un grand nombre de travaux dans différents domaines. Parmi les classes de problèmes connus, nous citons le problème du sac à dos, le problème du voyageur de commerce, les problèmes d'ordonnancement, le problème de coloration de graphes.

Le principe du problème du sac à dos consiste à modéliser une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou une partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

Le problème QoSWSC peut être vu comme un problème analogue à celui du sac à

dos, dans le sens où le problème QoSWSC consiste à sélectionner un sous-ensemble de services maximisant une fonction objectif scalaire pondérée de manière à maximiser le score de la qualité de la composition sous contraintes globales. Plus précisément, il est associé à la variante du problème de sac à dos multidimensionnel à choix multiples (MMKP : multi-dimension multi-choice knapsack problem) [Yu et al., 2007] où les dimensions sont représentées par les différents paramètres QoS.

En effet, la phase de sélection dans QoSWSC consiste à choisir pour chaque classe abstraite (ou tâche du workflow) $Ti$ de la composition un service $S_{ij}$ parmi m services $j = 1..m$ de la classe considérée. Sachant qu'il faudra aussi considérer plusieurs dimensions QoS. Dans le cas où on a $l$ tâches dans le workflow, nous aurons $m^l$ compositions à considérer tout en satisfaisant les contraintes globales. La classe MMKP appartient à la classe des problèmes combinatoires qui sont identifiés comme des problèmes NP-difficile [Parra-Hernandez and Dimopoulos, 2005].

### 3.3.4 Formulation du problème QoSWSC

Pour résumer, Le problème QoSWSC est un problème NP-Difficile qui est généralement défini comme un simple problème mono-objectif avec la maximisation des contraintes QoS locales/globales ou un problème multi-objectif avec la maximisation des contraintes QoS globales. Nous pouvons formuler le problème QoSWSC comme suit :
Soient :

- $CA = \{S_1, ..., S_n\}$ , une composition abstraite qui représente la requête de l'utilisateur, c-à-d les $n$ classes de services $S_i$ à consommer.
- $gc = \{gc_1, .., gc_n\}$, un ensemble de contraintes globales définies par l'utilisateur qui concerne les QoS de la composition.
- $C = \{s_{i1}, ..., s_{nj}\}$ une composition concrète, c-à-d nous remplaçons chaque classe $S_i$ par un service concret $S_{ij} \in S_i$
- $q(s_{ij}) = \{q_1, .., q_m\}$ vecteur des qualités du service $S_{ij}$.
- $Q(C) = \{Q_1(C), .., Q_m(C)\}$vecteur des qualités du service de la composition C.
- $U(C)$ la fonction objectif à minimiser ou maximiser. Elle peut être :
  - multi-objectif : dans ce cas, $U(C) = \{U_1..U_k\}$ où $U_k$ représente la $k_{ième}$ fonction à optimiser.
  - mono-objectif : dans ce cas, les $m$ attributs de QoS sont agrégés en une seule valeur.

- Le problème d'optimisation de QoSWSC est alors :
  - Global : si nous cherchons l'ensemble des compositions qui vont satisfaire les contraintes globales c-à-d $Q_k \geq gc_k, \forall k \in \{1..n\}$.
  - Local : si nous cherchons l'ensemble des compositions sans prendre en considération les contraintes globales $gc$ à optimiser, nous cherchons à optimiser les paramètres QoS dans chaque classe abstraite indépendamment des autres.
  - Globale et locale : si nous utilisons une optimisation locale et globale en même temps. Dans ce cas, notons que parfois une fonction mono-objectif est définie pour une optimisation globale et une fonction multi-objectif est définie dans l'optimisation locale ou l'inverse.

La modélisation du problème de sélection diffère selon la stratégie d'optimisation appliquée (locale ou globale) et la stratégie de modélisation du problème (fonction mono ou multi-objectif). Appliquer l'une de ces stratégies revient à choisir une méthode de résolution adéquate et à dérouler un algorithme de sélection approprié. Dans la section de l'état de l'art, nous donnerons une classification des travaux selon la méthode de résolution adoptée face aux problèmes QoSWSC. Nous présentons aussi des tableaux comparatifs des travaux selon la stratégie d'optimisation (Locale ou globale) ainsi que l'algorithme utilisé et les paramètres QoS traités.

## 3.4 Défis du problème QoSWSC

Les verrous lors de la sélection sont nombreux, parmi eux nous pouvons citer :

### 3.4.1 La scalabilité

En plus d'être considéré comme un problème NP-difficile, le problème QoSWSC est aussi considéré comme étant un problème de grande taille. Effectivement, nous assistons aujourd'hui à une explosion de services web sur le net où un grand nombre de fournisseurs de services rivalisent pour offrir la même fonctionnalité ; ce qui met à la disposition de l'utilisateur un nombre considérable de services Web. Ceci signifie, qu'il est peu probable qu'une solution optimale au problème soit trouvée dans un temps raisonnable. Nous verrons par la suite, dans la section de l'état des travaux que les approches approximatives comme les heuristiques et méta-heuristiques sont plus appropriées et utilisées pour l'optimisation du problème QoSWSC. D'autres travaux

tentent de réduire la taille des services candidats participants à la composition en utilisant les techniques orientées bases de données comme le Skyline.

### 3.4.2 Le passage du workflow au plan d'exécution

Les approches de composition de services à base de workflow supposent que le service composite exigé soit décrit à un niveau abstrait comme un processus de haut niveau dans le Business Process [Ardagna and Pernici, 2007]. Différents langages et modèles sont proposés et utilisés pour décrire la composition des services (ou plus précisément l'équivalent du workflow dans Busines process) comme le diagramme d'activité UML [Ardagna and Pernici, 2007], Diagramme d'état [Zeng et al., 2004], l'extension de BPEL [Agarwal and Jalote, 2010], ou YAWL [El Hadad et al., 2010].

Hormis la définition de la composition, différentes structures de contrôle dans le workflow sont à considérer comme le montre l'exemple que nous avons défini dans la section 3.2. Ces structures de contrôle peuvent être en *séquence*, *boucle*, *parallèle* ou *conditionnelle*. Certaines de ces structures de branchement, comme les structures conditionnelles ou en boucles, nécessitent un traitement particulier pour le calcul des Qualités de services de la compositions où différents chemins d'exécutions sont à considérer avec des probabilités pour le choix d'une tâche ou l'autre [Ardagna and Pernici, 2007].

### 3.4.3 Les fonctions d'agrégation

La mesure de la qualité du service composite représente un défi critique dans la sélection. Cette mesure est calculée par une fonction d'agrégation. La valeur agrégée d'un attribut QoS doit prendre en compte la valeur de l'attribut QoS de chaque service atomique participant dans la composition et la structure du workflow. Par exemple, pour la qualité prix, le prix global d'un service composite peut être défini comme la somme des prix de tous les services participants. Cependant, pour la qualité temps d'exécution, nous avons besoin d'une fonction plus complexe qui permet de considérer le temps d'exécution maximal parmi les services parallèles, d'additionner les temps d'exécution de services séquentiels et de combiner ces deux valeurs s'il y a des structures parallèles dans des branches liées de manière séquentielle.

Certains travaux ont proposé des fonctions d'agrégation pour certaines qualités comme les travaux de Jaeger et al. [Jaeger et al., 2004] où les auteurs considèrent les qualités temps d'exécution, le coût, le débit et leurs calculs sous différentes structures

du workflow. [Zeng et al., 2004] quant à eux ont proposé des fonctions d'agrégation pour les paramètres temps d'exécution, prix, durée d'exécution et la réputation pour les structures de base des workflow comme les structures en boucle, séquences et conditionnelles.

La majorité des travaux, pour ne pas dire tous, adopte les fonctions définies dans ces deux travaux pour calculer les Qualités de la composition. Nous utiliserons, nous aussi ces même fonctions, dans nos contributions. Elles seront présentées au moment de la formulation de notre contribution dans le chapitre 6.

### 3.4.4 Définition des poids d'attributs QoS

L'importance d'un attribut QoS par rapport à un autre est souvent marquée à travers l'attribution d'un poids. La majorité des travaux dans la littérature, suppose que le demandeur de service a une idée claire de l'importance d'un attribut de QoS qui le laisse assigner un poids scalaire à chaque critère QoS. Mais ceci peut ne pas être réaliste, d'autant plus que le nombre d'attributs de QoS impliqués dans les critères de sélection augmente. Pour pallier cette contrainte, certains travaux utilisent les techniques de bases de données comme le principe de Skyline. Nous donnerons un état de l'art de ces travaux dans le chapitre 5.

## 3.5 État de l'art des travaux sur le problème QoSWSC

Plusieurs efforts ont été définis pour faire face aux défis du problème de sélection. Comme nous l'avons mentionné précédemment, les travaux sur QoSWSC basés sur l'optimisation peuvent êtres examinés sous plusieurs angles selon la stratégie utilisée et/ou la formulation du problème et l'approche adoptée. En dehors du fait qu'elle soit locale ou globale, mono-objectif ou multi-objectif, nous avons constaté, en s'appuyant sur plusieurs papiers de type survol de la littérature du domaine QoSWSC [Strunk, 2010], [KAMAL et al., 2014], [Jatoth et al., 2015] que 4 grandes catégories peuvent être distinguées dans l'optimisation du problème QoSWSC, selon les approches utilisées qui peuvent se baser sur des méthodes de résolution :

1. Exactes.
2. Heuristiques (approximatives).
3. Méta-heuristiques (approximatives).
4. Techniques de bases de données.

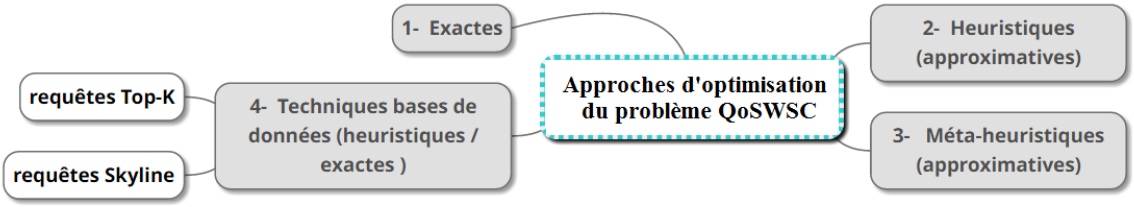



**Figure** ***3.2:*** *La classification des approches de résolution du problème de sélection des services Web*

Nous présentons dans cette section une classification des travaux selon la méthode de résolution adoptée face au problème QoSWSC. Nous présentons aussi des tableaux comparatifs des travaux selon la stratégie d'optimisation (Locale ou globale) ainsi que l'algorithme utilisé et les paramètres QoS traités. Nous détaillerons dans cet état de l'art les travaux relatifs au deux premières catégories (voir figure 3.2). Vu l'importance des travaux relatifs à la catégorie 3 et la catégorie 4, le prochain chapitre sera consacré à l'optimisation de QoSWSC par les méta-heuristiques, quant à ceux utilisant les techniques orientées bases de données, ils seront étudiés dans le chapitre 5. Nous notons aussi que nos travaux s'insèrent dans ces deux dernières catégories.

### 3.5.1 Approches exactes

Les méthodes exactes résolvent le problème d'optimisation d'une façon optimale. Chaque problème d'optimisation peut être résolu en utilisant une recherche exhaustive. Le plus souvent, ces méthodes utilisent des méthodes génériques comme branch and bound [Sen et al., 1988]. D'autres méthodes utilisent les techniques de la programmation par contraintes ou la programmation dynamique, la programmation linéaire entière (Integer Lineaire Porogramming, ILP) ou encore les techniques de programmation entière et mixte (Mixed Integer Programming ou MIP) .

La modélisation du problème de la composition par (ILP) ou (MIP) est une des approches dominantes dans les recherches basées sur l'optimisation, quelques exemples incluent les travaux de [Zeng et al., 2004] , [Kritikos and Plexousakis, 2009] et [Ardagna and Pernici, 2007]. Dans cette approche, une variable de décision binaire entière est assignée à chaque service. La valeur de cette dernière spécifie si ce service est choisi pour faire partie de la composition ou non.

[Zeng et al., 2004] utilisent des algorithmes d'optimisation dans la phase locale et globale. L'algorithme dans la phase d'optimisation locale sélectionne le service

optimal pour chaque tâche. L'algorithme dans la phase globale sélectionne le plan d'exécution optimal parmi tous les plans d'exécutions possibles en se basant sur la programmation entière.

[Yu and Lin, 2005b] proposent une méthode qui permet de satisfaire les contraintes globales, et ont considéré le problème comme un problème de sac à dos à choix multiples. Ils ont utilisé l'algorithme branch and bround pour retourner le résultat.

[Yu and Lin, 2005a] et [Liu et al., 2012] présentent une méthode en considérant de multiples critères QoS en utilisant l'algorithme branch and bround et traitent différents types de work-flows.

Dans [Huang et al., 2009], les auteurs découpent le plan d'exécution en parties divisibles et indivisibles séparées. Ils calculent une solution optimale pour chaque partie divisible, et utilisent l'algorithme récursif 'branch-and-bound' pour la découverte de solutions optimales pour les parties indivisibles.

[Nam et al., 2009] transforment le problème QoS-WSC en un problème de planification et utilisent l'algorithme 'learning-depth-first search' (LDFS), qui combine l'utilisation de l'algorithme de parcours en profondeur avec un apprentissage de manière itérative.

D'autres travaux [Jaeger and Muhl, 2006], [Liu et al., 2004] et [Zeng et al., 2003] appliquent la technique SAW (Simple Additive Weighting) dans la phase locale pour modéliser le problème QoSWSC. Dans [Jaeger and Muhl, 2006], les auteurs utilisent un algorithme qui s'inspire de l'algorithme Glouton.

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

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

[Rodriguez-Mier et al., 2011] utilisent l'algorithme A* [Stewart and White III, 1991] qui est connu pour réaliser de meilleures performances que les algorithmes de recherche de chemins dans les graphes en utilisant des heuristiques pour guider la

recherche.

Dans [Rosenberg et al., 2009] les auteurs ont proposé un modèle qui traite la sélection comme un problème d'optimisation par contraintes relaxées (COP [5]). 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. Un récapitulatif de ces travaux est donné dans le tableau 3.2.

| **Approches** | **Stratégie d'optimisation** | **Algorithme** | **QoS traités** |
|---|---|---|---|
| [Zeng et al., 2004] | locale + globale | LIP | C, Tr ,R ,D |
| [Kritikos and Plexousakis, 2009] | locale | MIP | |
| [Ardagna and Pernici, 2007] | globale | MIP | Tr,B , D, C |
| [Yu and Lin, 2005b] | globale | MMMKP | Tr , C , D, F |
| [Yu and Lin, 2005a] | globale | Branch-and-Bound | Tr , C, F, D |
| [Liu et al., 2012] | globale | Branch-and-Bound | Pas mentionnées |
| [Huang et al., 2009] | locale | Programmation dynamique | Tr, B |
| [Nam et al., 2009] | globale | LDFS | Pas mentionné |
| [Jaeger and Muhl, 2006] | locale | Algorithme Glouton | Tr, C, R, D |
| [Gao et al., 2006] | locale | Programmation dynamique | Tr, C, D, F |
| [Rodriguez-Mier et al., 2011] | globale | Algorithme A* | D, C, Tr, F |
| [Rosenberg et al., 2009] | globale | COP | Tr, F, C, D, S, B , Exactitude, Messagerie Fiable |

***Tableau 3.2:*** *Travaux utilisant les approches exactes avec la stratégie, l'algorithme utilisé et QoS traités (Temps de réponse(Tr), Coût (c), réputation (R), Disponibilité (D), Fiabilité(F), Sécurité (S), Débit (B) )*

L'un des avantages des approches exactes utilisant ILP, MIP ou COP est la disponibilité de beaucoup d'outils résolveurs open source et commerciaux comme LiSP [6] et

5. Constraint Optimization Problem
6. $http://ufasoft_lisp_studio.fr.downloadastro.com/$

choco [7]. Beaucoup de ces résolveurs sont très précis dans la découverte de la solution optimale pour des cas de problèmes de taille considérable. Cependant, une augmentation du nombre de services web candidats mène à l'augmentation du nombre de variables de décision, qui à son tour aboutit à l'explosion de l'espace de recherche et le nombre des conditions à vérifier. Donc, ces approches sont limitées par la taille du problème (le nombre des tâches) et l'espace des services candidats.

### 3.5.2 Approches heuristiques (approximatives)

Les approches heuristiques utilisent des algorithmes qui sont généralement créés par "l'expérience" pour un problème d'optimisation spécifique. Ils essayent de trouver une solution de qualité au problème dans un temps acceptable. La solution obtenue n'est pas la solution optimale mais une solution approchée (nous les appelons aussi des méthodes approximatives). Les algorithmes heuristiques tirent avantage des particularités du problème, contrairement aux algorithmes exacts qui prennent un grand temps pour obtenir la solution optimale, les algorithmes heuristiques obtiennent des solutions presque optimales dans un temps acceptable.

Plusieurs travaux sont inspirés des algorithmes heuristiques. Un des premiers algorithmes proposé pour la résolution du problème de sac à dos (et sa version MMKP [8]) est présenté par [Khan, 1998]. L'auteur propose une heuristique nommée UHE pour sa résolution, UHE utilise une mesure appelée la consommation des ressources agrégées, pour mettre à jour un élément de chaque groupe à chaque tour de sélection. Une modification de cette heuristique a été apportée par [Akbar et al., 2001] qui ont proposé l'algorithme M-UHE. [Yu et al., 2007] ont modifié et adapté l'heuristique M-UHE au contexte du service web en proposant deux algorithmes. Le premier appelé WS-UHE pour le modèle combinatoire et le deuxième appelé MCSP-K pour le modèle orienté graphe. En dépit de l'amélioration significative apportée par ces deux algorithmes par rapport aux algorithmes à solutions exactes, les deux algorithmes ne sont pas scalables (i.e. le temps d'exécution n'est plus temps réel sous un nombre énorme de services web). D'autres heuristiques ont été proposées par [Berbner et al., 2006], où les auteurs présentent trois nouveaux algorithmes : H1 RELAX IP, H2 SWAP, et H3 ANNEAL.

[Xia et al., 2011] proposent un algorithme nommé qssac qui peut trouver une solution proche de l'optimale (mais les auteurs n'ont pas mentionné les résultats d'ex-

7. http ://choco-solver.org/
8. multi-dimensional multiple choice knapsack problem

périmentations portant sur le degré d'optimalité). Les auteurs proposent un deuxième algorithme dans le but de réduire le temps d'exécution, nommé optics. Ce dernier regroupe les services similaires en termes de QoS en M classes (et leurs représentants) pour chaque tâche, ensuite il énumère toutes les compositions possibles pour 02 ou K tâches du document BPEL, et les trie à l'aide d'une fonction objectif.

[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.

Quant à [Moustafa and Zhang, 2013] et [Feng et al., 2013], ils utilisent les algorithmes d'apprentissage par renforcement pour trouver l'ensemble de solutions Pareto optimales qui satisfait les facteurs QoS multiples et les exigences de l'utilisateur.

[Li and Wen, 2012] adoptent la stratégie gagnant-gagnant (win-win stratégie) et proposent un algorithme qui porte le nom de la stratégie elle même.

Plusieurs autres travaux ont proposé plusieurs algorithmes heuristiques [Luo et al., 2011], [Li et al., 2010] , [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. Un récapitulatif des approches que nous venons de citer est présenté dans le tableau comparatif 3.3.

### 3.5.3 Approches méta-heuristiques

Les méthodes heuristiques sont nécessaires pour résoudre des problèmes de grande taille et/ou des problèmes avec un nombre de critères supérieur à deux. Parmi ces heuristiques, nous trouvons les méta-heuristiques qui, contrairement aux algorithmes heuristiques spécifiques à un problème donné, fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes. Les méta-heuristiques sont des approches approximatives, elle sont les méthodes d'optimisation les plus utilisées pour la résolution des problèmes combinatoires NP-difficile. Elles sont aussi très exploitées pour la résolution des problèmes QoSWSC. Nous donnerons, dans le prochain chapitre, les concepts de bases et présenterons une classification des algorithmes utilisés dans un état de l'art sur leurs utilisations dans l'optimisation du problème QoSWSC.

| **Approches** | **Stratégie d'optimisation** | **Algorithmes** | **QoS traités** |
|---|---|---|---|
| [Yu et al., 2007] | globale | Algorithme WS-UHE | C, Tr, D |
| [Xia et al., 2011] | globale | Algorithme optics | C, Tr, F, Taux de succès |
| [Berbner et al., 2006] | globale | MIP heuristic | C, Tr, F, D |
| [Lecue and Mehandjiev, 2009] | globale | hill climbing | Tr , C |
| [Klein et al., 2011] | globale | hill climbing | Tr, C, D, F |
| [Moustafa and Zhang, 2013] | globale | Apprentissage par renforcement | D, C, Tr |
| [Feng et al., 2013] | globale | Apprentissage par renforcement | D, C, Tr |
| [Li and Wen, 2012] | globale | Algorithme Gagnant-gagnant | Tr, F, D |

***Tableau 3.3:*** *Quelques travaux utilisant les approches heuristiques (approximatives) avec la stratégie, l'algorithme utilisé et QoS traités (Temps de réponse(Tr), Coût (c), réputation (R), Disponibilité (D), Fiabilité(F))*

### 3.5.4 Approches basées sur les techniques de bases de données

Cette catégorie traite la sélection comme un problème multi-objectif qu'il soit traité en local ou en global (sans le réduire à une fonction mono-objectif). Elle repose surtout sur la notion de Pareto dominance et sur le principe des requêtes Skyline ou Top-k. Elle utilise des algorithmes comme celui de Divide and conquer ou autres. Le chapitre 5 sera consacré à cette classe en mettant l'accent sur les concepts de bases, tout en présentant un état des travaux actuels utilisant ces concepts pour la résolution du problème QoSWSC.

## 3.6 Synthèse

Nous avons vu que le problème QoSWSC peut être considéré comme un problème d'optimisation. Différentes approches sont utilisées pour la résolution de ce dernier. Certains travaux utilisent les méthodes exactes (Branch and Bound, programmation

dynamique, etc..), ces méthodes permettent de trouver la solution optimale. Malheureusement, ces algorithmes de type énumératif, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grande taille (même si en pratique la taille n'est pas le seul critère limitant). Par conséquent, elles ne peuvent correspondre au caractère NP-difficile du problème QoSWSC où la taille de l'espace de recherche ne cesse de croitre. Dans ce cas, il est nécessaire de faire appel à des heuristiques permettant de trouver de bonnes solutions approchées.

A partir des tableaux comparatifs que nous avons présenté ci-dessus, nous avons constaté que les paramètres QoS les plus considérés dans les travaux sont le temps de réponse, le coût, la fiabilité et la disponibilité. En effet, ce sont les qualités les plus sensibles et les plus exigées par les utilisateurs et les plus importantes dans la phase de sélection pour la réussite d'une composition. Cette constatation sera aussi confirmée pour les travaux se basant sur les méta-heuristiques et techniques orientées bases de données que nous présenterons dans le chapitre 4 et 5. D'autres paramètres peuvent être considérés mais la priorité est donnée à ces derniers en premier.

Nous avons établi une classification des travaux en 4 catégories parmi lesquelles nous trouvons les méta-heuristiques. Ces dernières fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes. Les méta-heuristiques sont très utilisées dans la résolution du problème de sélection des services web. Un récent état de l'art [Jatoth et al., 2015], qui est en cours de publication dans le journal 'IEEE Transactions on Services Computing', montre à travers une riche étude de 85 papiers sur le problème QoSWSC que les approches les plus utilisées pour résoudre ce problème sont les méta-heuristiques (50%) contre une utilisation de 30% d'approches heuristiques et seulement 20% d'approches exactes. Ceci peut être justifié par le fait que QoSWSC appartient aux problèmes NP-difficile et nécessite par conséquent une résolution dans un temps acceptable. Partant de cette constatation, on a dédié un chapitre complet pour les travaux QoSWSC basés sur les méta-heuristiques qui est l'axe le plus actif des recherches dans la résolution du problème QoSWSC et où s'insère aussi une partie de nos contributions.

Enfin, une nouvelle catégorie a récemment émergé, se basant sur les techniques venant des bases de données qui reposent sur la notion de Pareto dominance comme le Skyline et Top-k dominance. Nous verrons dans le chapitre 5 que cette dernière permet de réduire l'espace de recherche et donne de meilleurs résultats comparée aux autres méthodes. D'où l'idée de contribuer avec ces techniques.

## 3.7 Conclusion

Nous avons présenté, analysé et formulé, dans ce chapitre, le problème de sélection des services web à base de QoS 'QoSWSC'. Qu'il soit considéré comme mono-objectif ou multi-objectif, local ou global, nous avons regroupé les travaux de résolution de ce problème en quatre familles d'approches. Nous avons consacré ce chapitre pour présenter un état de l'art des deux premières catégories d'approches utilisant respectivement les méthodes de résolution exactes et les méthodes heuristiques (approximatives). Ces dernières sont peu utilisées comparées à la famille des méta-heuristiques auxquelles sera consacré le chapitre suivant.

Chapitre 4

# La sélection des services Web basée sur les méta-heuristiques

## Sommaire

## 4.1 Introduction

Les méta-heuristiques construisent une solution moins exigeante que les solutions exactes. Nous avons vu dans le chapitre précédent que les solutions exactes ne sont pas vraiment adaptées au problème QoSWSC quand la taille de ce dernier est relativement grande. Les méthodes approchées dont les heuristiques et méta-heuristiques constituent une alternative aux méthodes exactes ; elles permettent de fournir des solutions de très bonne qualité en un temps de calcul raisonnable. Contrairement aux heuristiques qui sont des méthodes spécifiques à un problème particulier, les méta-heuristiques sont applicables pratiquement sur une grande variété de problèmes d'optimisation de différentes complexités.

Les méta-heuristiques sont les méthodes d'optimisations les plus utilisées pour la résolution des problèmes combinatoires NP-difficiles. Par conséquent, elles constituent la solution à adopter pour la résolution du problème QoSWSC. Effectivement, un récent état de l'art [Jatoth et al., 2015] montre à travers une riche étude de 85 papiers sur le problème QoSWSC que les méta-heuristiques sont les approches les plus utilisées pour résoudre le problème QoSWSC.

Ce chapitre sera dédié à la classe des méta-heuristiques et leur utilisation dans l'optimisation du problème QoSWSC. Il ne s'agit pas de décrire en détail chacune de ces méta-heuristiques mais simplement de les classer et de recenser celles utilisées pour la sélection des services web. Nous donnerons une classification des méta-heuristiques selon leur nature de fonctionnement et leur chronologie d'apparition. Ce qui nous amènera à dresser un constat sur l'utilisation de ce que nous appelons la nouvelle génération des méta-heuristiques dans l'optimisation des QoSWSC. En effet, ces dernières années de récentes méta-heuristiques ont été définies ; elles s'avèrent être plus performantes que leurs prédécesseurs. Ces dernières restent encore peu ou pas exploitées pour solutionner le problème QoSWSC en dépit de leurs performances qui sont démontrées dans plusieurs autres domaines comme la résolution des problèmes de Job-shop et flow shop, sac à dos, ou encore d'ordonnancement.

Nous définissons, dans le début de ce chapitre, le principe des méta-heuristiques et donnons une classification de leurs algorithmes. Nous dressons ensuite, un état de l'art comparatif des travaux QoSWSC basés sur les méta-heuristiques. Enfin, nous terminerons par une synthèse qui présentera l'efficacité et le degré d'exploitation des différentes méta-heuristiques dans la résolution du problème QoSWSC.

## 4.2 Principe et classification des méta-heuristiques

Les méthodes dites méta-heuristiques sont des méthodes générales, des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une alternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné. Selon Osman et Laporte [Osman and Laporte, 1996], une méta-heuristique est définie comme suit :

**Définition 4.1 - méta-heuristique.**
Une méta-heuristique est un processus itératif qui subordonne et guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque optimales.

Les méta-heuristiques fonctionnent en deux temps : une phase de diversification ou exploration, où on explore l'espace de recherche ; une phase d'intensification ou exploitation, où l'algorithme va se concentrer sur les solutions trouvés. L'efficacité d'une méta-heuristique dans la résolution d'un problème d'optimisation est liée à sa capacité à établir un certain équilibre entre ces deux phases.

Les méta-heuristiques sont généralement des algorithmes stochastiques itératifs inspirées de mécanismes d'optimisation rencontrés dans la nature (physiques ou biologiques), nous pouvons citer à titre d'exemple : la recherche Tabou qui s'inspire de la mémoire des êtres humains, le recuit simulé qui est inspiré d'un processus métallurgique, les algorithmes basés sur l'intelligence d'essaim comme l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis, l'algorithme de colonies d'abeilles,..,etc. qui s'inspirent du comportement social de certaines espèces évoluant en groupe.

La nature ne cesse d'inspirer la recherche et nous assistons chaque année à l'apparition de nouvelles méta-heuristiques très prometteuses utilisées pour l'optimisation. La Figure 4.1 dresse un panorama des principales méta-heuristiques par ordre chronologique de leur apparition. Chaque méta-heuristique est accompagnée de la référence au premier papier qui définit son fonctionnement de base.

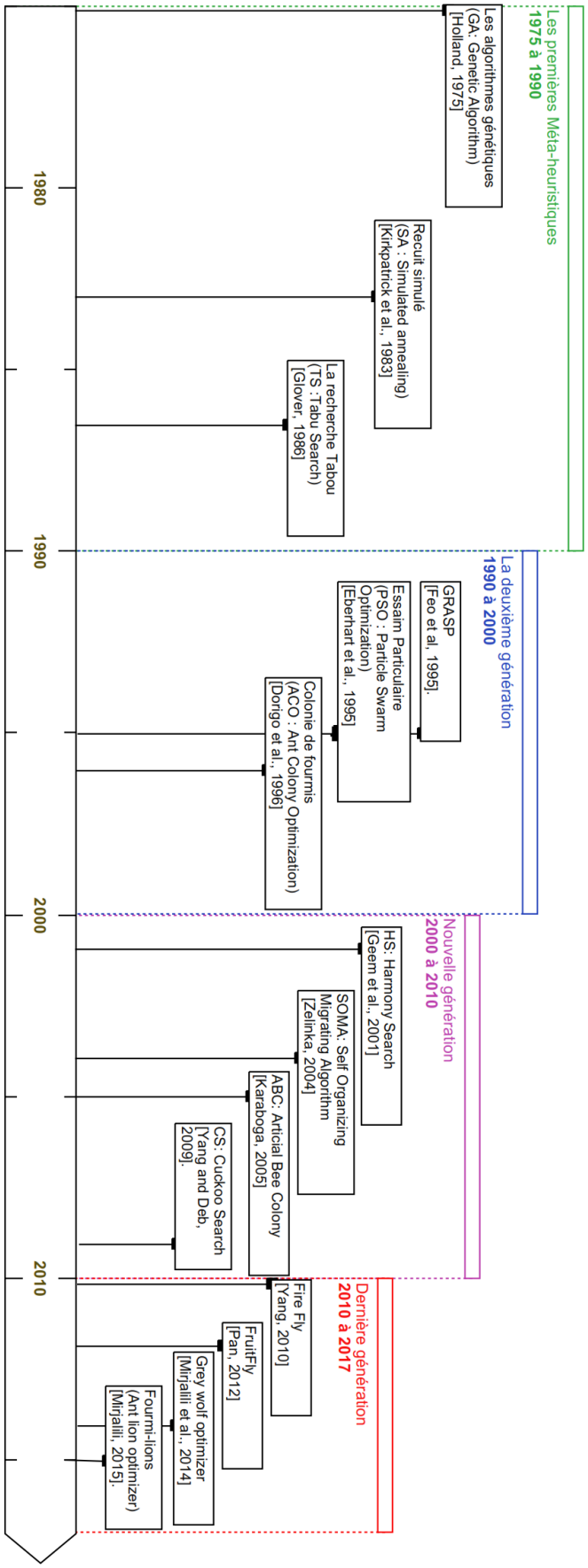



**Figure *4.1:*** *La chronologie des principales Méta-heuristiques.*

Ces méta-heuristiques sont généralement classées dans la littérature en deux catégories : (i) les méta-heuristiques à base de solution unique et, (ii) les méta-heuristiques à base d'une population de solutions. Dernièrement, une troisième catégorie est apparue, elle regroupe un ensemble de méta-heuristiques hybridées qui combinent deux ou plusieurs autres méta-heuristiques.

### 4.2.1 Les méta-heuristiques à base de solution unique

Les algorithmes de cette classe, aussi appelés les méthodes trajectoires (décrivent une trajectoire en l'espace de recherche au cours du processus de recherche), elles travaillent sur un seul point de l'espace de recherche à un instant donné en commençant avec une solution initiale. Cette solution est ensuite améliorée itérativement en choisissant une nouvelle solution dans son voisinage.

Nous trouvons dans cette classe deux familles d'algorithmes : d'une part, les algorithmes de recherche locale pour une optimisation locale comme les algorithmes de 'Descente' (DM : Descent method) aussi appelée hill climbing [Papadimitriou and Steiglitz, 1982]. D'autre part, les algorithmes de recherche locale pour une optimisation globale comme l'algorithme de 'Recuit simulé' (SA : Simulated annealing) [Kirkpatrick et al., 1983], 'Recherche par Tabous' appelée aussi, de manière simplifiée, 'Recherche Tabou'(TS :Tabu Search) [Glover, 1986], 'Recherche à voisinage variable' ou encore 'La procédure de recherche gloutonne aléatoire adaptative' (GRASP : Greedy Randomized Adaptive Search Procedure) [Feo and Resende, 1995].

### 4.2.2 Les méta-heuristiques à base de population de solutions

Contrairement aux méta-heuristiques à base de solution unique, les méta-heuristiques à base de population de solutions commencent la recherche avec un groupe de solutions appelées population. Cette population de solutions est améliorée au fur et à mesure des itérations. L'utilisation d'un ensemble de solutions au lieu d'une seule permet de renforcer la diversité de la recherche, ce qui augmente les chances de rencontrer des solutions de bonne qualité. Nous trouvons dans cette classe d'algorithmes, les algorithmes évolutionnaires et les algorithmes d'intelligence par essaim.

#### 4.2.2.1 Les algorithmes évolutionnaires

Appelés aussi les algorithmes évolutionnistes (EC : Evolutionary Computation), sont une famille d'algorithmes qui s'inspirent de la théorie du naturaliste Charles Darwin. Le terme Evolutionary Computation englobe une classe de méta-heuristiques telles que la programmation évolutive [Fogel et al., 1966], les stratégies d'évolution [Huning et al., 1976], les algorithmes génétiques (GA : Genetic Algorithm) [Holland, 1975], et la programmation génétique [Koza, 1992].

#### 4.2.2.2 Les algorithmes basés sur l'intelligence par essaim (ou l'intelligence collective : Swarm Intelligence)

Les algorithmes basés sur l'intelligence par essaim forment une branche d'algorithmes inspirés de l'intelligence collective de communautés. Selon la communauté, et ses membres, cette forme d'intelligence est différente. L'intelligence collective désigne les capacités cognitives d'une communauté résultant des interactions multiples entre les membres (ou agents) de la communauté. La collaboration entre les membres de la communauté, fait émerger des possibilités de représentation, de création et d'apprentissage supérieures à celles des individus isolés. Le mot « essaim » est généralement utilisé pour désigner un ensemble fini de particules ou d'agents interactifs.

Les algorithmes basés sur l'intelligence par essaim sont inspirés des phénomènes naturels comme l'intelligence des animaux ou l'intelligence collective observée notamment chez les insectes sociaux (comme les fourmis, les abeilles les animaux évoluant en groupe comme les oiseaux migrateurs, les bancs de poissons. D'autres intelligences collectives sont observées dans certains phénomènes biologiques comme les systèmes immunitaires qui sont des exemples d'essaim.

Les oiseaux évoluant en groupe forment des essaims dont les particules sont des oiseaux, les bancs de poissons forment des essaims dont les particules sont des poissons, les colonies de fourmis forment des essaims dont les particules sont des fourmis, les colonies d'abeilles forment des essaims dont les particules sont des abeilles, le système immunitaire forme un essaim de particules représenté par des cellules de reconnaissance et de protection. Ainsi, en imitant le comportement social des particules formant des essaims capables de s'auto-organiser, plusieurs algorithmes ont été proposés ces dernières décennies parmi elles, on trouve l'optimisation par colonie de fourmis (ACO : Ant Colony Optimization) [Dorigo et al., 1996], l'optimisation par essaim particulaire (PSO : Particle Swarm Optimization) [Eberhart et al., 1995].

Une liste non exhaustif de ces différentes méta-heuristiques classées par catégorie est donnée dans la figure 4.2.

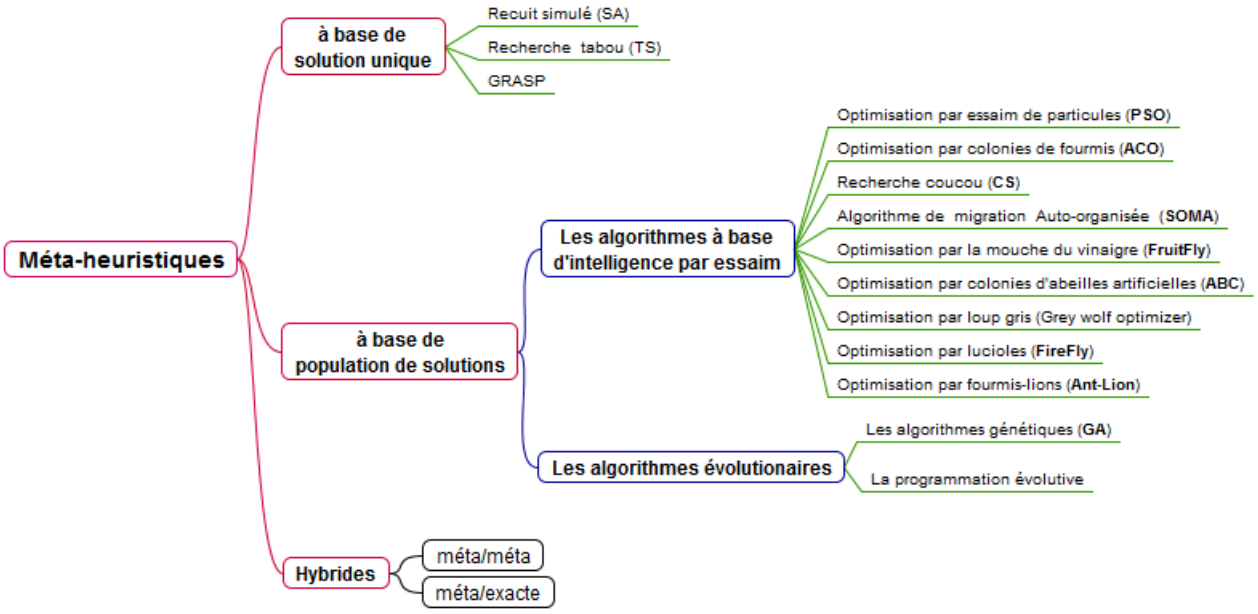



**Figure *4.2:*** *La classification des méta-heuristiques*

Généralement, et de manière simplifiée, les systèmes à base de l'intelligence par essaim reposent sur les principales caractéristiques suivantes :

1. l'information locale : chaque individu ne possède qu'une connaissance partielle de l'environnement et n'a pas conscience de la totalité des éléments qui influencent le groupe,
2. L'ensemble de règles : chaque individu obéit à un ensemble restreint de règles simples par rapport au comportement du système global,
3. Les interactions multiples : chaque individu est en relation avec un ou plusieurs autres individus du groupe,
4. La collectivité : les individus trouvent un bénéfice à collaborer (parfois instinctivement) et leur performance est meilleure que s'ils avaient été seuls.

### 4.2.3 Les méta-heuristiques hybrides

L'idée de ces approches est de faire coopérer différents types de méta-heuristiques ou méta-heuristiques et approches exactes. L'idée n'est pas nouvelle, car très vite, il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés et les chercheurs ont combiné les méta-heuristiques de manière à profiter des avantages des différentes méthodes.

Nous trouvons une coopération **méta/méta** qui essayent d'hybrider différents algorithmes méta-heuristiques. Les méta-heuristiques hybrides sont devenues maintenant assez classiques dans le domaine de l'optimisation [Basseur, 2005]

Nous trouvons aussi un autre type de coopération **méta/exacte** qui est entre algorithme exacte et méta-heuristique. Nous avons vu que les méthodes exactes correspondaient à la résolution des petits problèmes tandis que les méta-heuristiques sont capables d'appréhender de grands problèmes sans pouvoir donner la solution optimale. Cependant les méthodes exactes peuvent, plus au moins, être utilisées pour traiter des sous-problèmes extraits du problème global. En effet, La résolution de ces sous problèmes permet de contribuer à la recherche de la solution globale dans les méta-heuristiques, soit en combinant judicieusement différents sous-problèmes, soit en hybridant la résolution exacte de sous-problèmes et la résolution heuristique du problème complet. Cette hybridation des deux approches, permet de conserver aux mieux les avantages de chacune des approches

## 4.3 État de l'art des méta-heuristiques appliquées au problème QoSWSC

Il y' a peu de travaux qui utilisent les méta-heuristiques à base de solution unique à elles seules, généralement nous les trouvons, hybridées ou combinées avec d'autres méta-heuristiques. Elles sont surtout utilisées dans la phase locale de recherche où la notion de voisinage est très importante.

• **La recherche Tabou, Recuit simulé et GRASP**

Parmi les travaux qui utilisent les algorithmes à base de solution unique pour l'optimisation des QoSWSC nous trouvons celui de [Pop et al., 2011b] où l'auteur présente une méthode à base de recherche Tabou pour choisir la solution quasi optimale dans la composition de services Web sémantique. La méthode proposée est appliquée sur une structure de graphe de planification amélioré. Ce graphe code toutes les solutions des compositions permettant de satisfaire une requête de l'utilisateur. Les critères pour choisir la solution optimale incluent les attributs de QoS et la similarité sémantique entre les services impliqués dans la composition. La méthode à base de recherche Tabou a été évaluée sur des scénarios de services web de réservation et planification de voyage.

Dans Hadjila et. all [Hadjila and Chikh, 2012], la composition des services est

modélisée sous forme d'un workflow. Les auteurs ne considèrent que la structure séquentielle et utilisent la recherche Tabou pour optimiser la fonction mono-objectif qui considère 5 paramètres QoS.

[Ko et al., 2008] proposent un algorithme de composition de service Web basé sur la satisfaction de contraintes qui combine la recherche Tabou et la méta-heuristique recuit simulé. [Rosenberg et al., 2010] quant à eux utilisent la méta-heuristique recuit simulé.

[Parejo et al., 2014] proposent une méthode qui utilise la méta-heuristique GRASP pour la résolution du problème QOSWSC, ils définissent l'algorithme QoS-Grasp. Les auteurs comparent l'algorithme proposé avec celui de [Ko et al., 2008] et l'algorithme génétique. Les expérimentations montrent que QoS-Grasp s'avère plus performant, en terme de coût d'exécution et de la qualité de la solution, que l'algorithme de recherche Tabou combiné avec Récuit simulé et l'algorithme génétique. Par la suite, les auteurs améliorent l'algorithme QoS-Grasp en utilisant l'algorithme évolutionniste 'la recomposition de chemin' (Path relinking).

• **L'algorithme génétique et ses variantes**

Le premier travail, qui a appliqué l'algorithme génétique pour solutionner le problème QoSWSC, a été proposé par Canfora et al [Canfora et al., 2005], les auteurs adaptent l'algorithme génétique au problème de sélection des SW où Les points de mutation sont choisis aléatoirement . Le codage des chromosomes utilise la représentation entière. Les auteurs adoptent une fonction objectif dynamique qui pénalise les chromosomes lorsqu'ils violent les contraintes globales, et de ce fait l'approche favorise la convergence rapide vers un optimum local. Les auteurs comparent leur algorithme aux approches utilisant la programmation linéaire entière. Évidement, les résultats de l'expérimentation confirment, que la programmation linéaire, est préférable quand la taille du workflow est limitée.

Dans [Wang and Hou, 2008] les auteurs proposent un algorithme génétique en utilisant une optimisation multi-objectif, ils définissent une fonction objectif pour chaque paramètre QoS considéré qui sont au nombre de 3. Contrairement au travail de Canfora et al, les auteurs codent les chromosomes en utilisant une représentation binaire.

Dans [Jaeger and Mühl, 2007], les auteurs implémentent un algorithme génétique et se sont focalisés sur la capacité d'optimisation de l'algorithme génétique en étudiant l'impact d'influence des différents paramètres de l'algorithme proposé comme le taux de mutation. Ils comparent l'algorithme génétique avec celui de 'branch and bound'.

Les résultats ont montré que les solutions obtenues dépendent des paramètres de configuration de l'algorithme qui influencent ses performances. Une des principales différences parmi les algorithmes génétiques existants pour la composition de service Web est la méthode de codage du problème en lui même. Certains, adoptent une méthode de codage de chromosome sous une seule dimension comme [Canfora et al., 2005] et [Jaeger and Mühl, 2007].

D'autres [Zhang et al., 2006] ont présenté l'algorithme génétique basé sur une matrice de relation . Alors que [Gao et al., 2007] ont proposé un algorithme génétique qui supporte une représentation du problème par un arbre. Dans [Xiangbing et al., 2012] les auteurs modélisent le problème des services web en utilisant l'ontologie(WSMO) et appliquent ensuite l'algorithme génétique.

L'algorithme génétique peut être sujet au problème de prématurité, qui empêche l'algorithme de faire de nouvelles améliorations en stagnant dans un optimum local. Pour remédier à cette faille, plusieurs travaux ont essayé de contourner le problème en améliorant la recherche locale en le combinant avec d'autres heuristiques ou méta-heuristiques. Certains utilisent la recherche Tabou combinée aux algorithmes génétiques comme [Bahadori et al., 2009], [Parejo et al., 2008] ou encore [Yu et al., 2014], la recherche Tabou intervient après le croisement et la mutation, elle sert à choisir le meilleur chromosome (ayant la meilleure fitness) issu du voisinage des enfants, en outre elle permet d'éviter la stagnation dans des optimums locaux. De la même manière mais avec l'utilisation de l'algorithme recuit simulé, le travail de [Gao et al., 2009] a amélioré le fonctionnement, en évitant la prématuration de l'algorithme génétique appliqué au problème de QoSWSC.

• **L'algorithme PSO et ses variantes**

[Ming and Zhen-wu, 2007] montrent que l'optimisation en essaims particulaires PSO est plus rapide et plus scalable par rapport aux algorithmes génétiques (version standard sans hybridation), 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. Plusieurs autres travaux ont utilisé PSO comme [Liu and Yin, 2009], [Long and Gui, 2009], [Amiri and Serajzadeh, 2012], [Hadjila et al., 2012b], [Ludwig, 2012]. Mais nous notons aussi que, PSO ne peut éviter le problème des optimums locaux. Donc, plusieurs versions de PSO ont été proposées pour améliorer la version standard PSO. [Liu et al., 2007] proposent l'hybridation des PSO avec les algorithmes génétiques afin de marier les opérateurs locaux avec les opérateurs globaux, ils ajoutent l'opérateur de croisement à la version initiale de PSO, en

vu d'explorer de nouvelles portions d'espace de recherche. Les travaux qu'on vient de citer se focalisent sur la découverte d'une solution. Par contre, Les algorithmes évolutionnistes qui s'appuient sur des méthodes de nichage permettent de trouver des solutions multiples pour des problèmes multimodales [Brits et al., 2007]. Dans ce sens, [Liao et al., 2011] présentent un l'algorithme NichePSO et considèrent le problème QoSWSC comme un problème multimodale, l'algorithme proposé permet de localiser et de raffiner plusieurs solutions en même temps en considérant de multiples contraintes globales. Liu. et all [Liu et al., 2011] ont présenté un algorithme d'optimisation d'essaim de particule 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 pétri coloré (CPN) pour résoudre le problème QoSWSC. [Zhao et al., 2012b] adoptent la version discrète de PSO qui est basée sur les systèmes immunitaires artificiels.

• **Les systèmes immunitaires**

D'autre travaux utilisent les systèmes immunitaires artificiels (SIA) comme la sélection clonale [De Castro and Von Zuben, 2002] ou la sélection négative pour l'optimisation de QoSWSC. [Hadjila et al., 2012a] présentent l'algorithme de sélection clonale pour la résolution du problème QoSWSC. Ils considèrent une fonction monoobjectif qui manipule 5 paramètres QoS. Les auteurs montrent que la sélection clonale s'avère plus intéressante que l'algorithme génétique générale. [Pop et al., 2009] et [Xu and Reiff-Marganiec, 2008] proposent un algorithme immunitaire artificiel qui combinent la sélection clonale et d'autres heuristiques. [Zhao et al., 2012a] proposent eux aussi une hybridation de l'algorithme clonale, mais cette fois ci, avec l'algorithme PSO. Récemment [Zhao et al., 2014] proposent une première adaptation de la sélection négative au problème QoSWSC et présentent l'algorithme 'NSA'. Ils proposent ensuite une amélioration de l'algorithme initiale et présentent l'algorithme 'NSA++'. Ce dernier s'avère plus performant que PSO.

• **Les algorithmes de Colonie de fourmis et ses variantes**

Plusieurs travaux adoptent une optimisation par colonie de fourmis ACO pour résoudre le problème QoSWSC. La différence entre eux concerne l'ajustement des paramètres qu'utilise ACO comme la population initiale, la mise à jour et la stratégie d'évaporation de phéromone, les informations heuristiques et la manière dont la phéromone dénote des informations. L'analyse de performance de ces paramètres dans l'optimisation de QoSSWSC par les colonies de fourmis est donnée dans [Wang

et al., 2010].

En résolvant le problème QoSWSC par l'optimisation de colonie de fourmis, plusieurs travaux [Pop et al., 2010], [Xia et al., 2008], [Li and Yan-Xiang, 2010] modélisent le problème comme un graphe direct acyclique pondéré ou le point de départ dénote le nid de fourmis, le point cible dénote des sources alimentaires et les contraintes QoS dénotent les poids des arcs. Par contre les travaux ont des représentations de graphes différentes.

[Xia et al., 2008] modélisent la composition en termes de graphes tels que les nœuds sont des structures d'orchestration et les arcs sont des services. Par contre dans [Li and Yan-Xiang, 2010] le noeud dans le graphe dénote le service Web et les arcs dénotent l'interaction entre des services.

Les auteurs dans [Xia et al., 2008] ont indiqué que la présence d'une seule sorte de phéromone dans l'algorithme de colonie de fourmi ne pouvait pas traiter plusieurs attributs de QoS , donc les auteurs ont présenté une méthode d'optimisation d'algorithme de colonie de fourmis dynamique basée sur de multiples phéromones.

[Pop et al., 2010] traitent les services web sémantiques qui reposent sur l'utilisation des ontologies, les auteurs représentent la composition par un graphe où le nœud représente un cluster de services qui sont fonctionnellement similaires et l'arc représente le score d'une fonction qui prend en considération l'aspect fonctionnel (la qualité sémantique) et les paramètres QoS.

[Yunwu, 2009] propose une nouvelle version de l'algorithme ACO en utilisant l'opérateur de chaos. L'objectif du nouvel algorithme proposé est d'améliorer l'efficacité de ACO en lui évitant l'éventualité d'être piégé dans un optimum local. [Li and Yan-Xiang, 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 multiobjectif du problème QoSWSC.

[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.

- **Quelques méta-heuristiques récentes**

D'autres méta-heuristiques ont été élaborées ces dernières années ; elles sont relativement récentes par rapport à celles que nous venons de citer (voire le schéma de chronologie, Figure 4.1, page 57). Ces dernières s'avèrent plus intéressantes que celles

que nous venons de mentionner. A titre d'exemple, nous citons : Harmony Search (HS) [Geem et al., 2001] , Artificial Bee Colony (ABC) [Karaboga, 2005] ou Self Organizing Migrating Algorithm (SOMA) [Zelinka, 2004] ou encore Cuckoo Search (CS) [Yang and Deb, 2009].

**- La recherche harmonique**

plusieurs travaux [Jafarpour and Khayyambashi, 2009], [Jafarpour and Khayyambashi, 2010], [Esfahani et al., 2012], [Merzoug et al., 2014] utilisent l'optimisation par harmonie pour résoudre le problème QoSWSC. La majorité de ces travaux montre que l'optimisation par harmonie donne de meilleurs résultats que les algorithmes génétiques.

**- La recherche par colonies d'abeilles artificielles**

L'optimisation par colonies d'abeilles artificielles (ABC) a été adoptée elle aussi par quelques chercheurs pour résoudre le problème QoSWSC, nous citons les travaux de [Hadjila et al., 2013] , [Kousalya et al., 2011], [Chifu et al., 2010]. Une amélioration de l'algorithme ABC par la recherche glouton a été proposée par [Wang et al., 2013b] où les auteurs proposent deux algorithmes basés sur la recherche glouton pour calculer le voisinage des services dans la composition. Une très récente amélioration de l'algorithme ABC a été proposée dans les travaux de [Zhou et al., 2013] où les auteurs utilisent la théorie du chaos et les systèmes immunitaires pour booster l'algorithme ABC, mais ce travail n'a pas été appliqué dans l'optimisation des QoSWSC. Tous ces travaux modélisent le problème QoSWC en utilisant une fonction multi-objectif mis à part celui de [Hadjila et al., 2013] où les auteurs adoptent une fonction mono-objectif.

**- La recherche par l'auto migration organisée**

L'algorithme Self Organizing Migrating Algorithm (SOMA) est une méta-heuristique [Zelinka, 2004] intéressante qui montre une efficacité dans la résolution de problèmes d'optimisation connus comme le voyageur de commerce et Flow shop. Cependant, elle reste encore peu exploitée dans l'optimisation de QoSWSC. Un seul travail [Krithiga, 2012], à notre connaissance, a appliqué l'algorithme SOMA pour l'optimisation de QOSWSC ; où les auteurs utilisent une fonction mono-objectif pour agréger les paramètres QoS et montrent à travers les expérimentations que SOMA est supérieur aux algorithmes GA et PSO.

**- La recherche par coucou**

L'algorithme Cuckoo Search (CS) a été appliqué pour l'optimisation de QoSWSC par les travaux de [Chifu et al., 2011] où l'algorithme a été appliqué pour l'optimisation de la composition des services Web sémantiques. Les auteurs expérimentent

l'algorithme en ajustant ses différents paramètres, par contre aucune comparaison avec d'autres méta-heuristiques n'a été faite.

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.

[Boussalia and Chaoui, 2014] proposent un algorithme qui combine les principes de la recherche coucou et ceux de la mécanique quantique.

Un récapitulatif de l'ensemble des travaux utilisant les méta-heuristiques pour l'optimisation du problème QoSWSC est listé dans le tableau 4.1, le tableau 4.2 et le tableau 4.3.

Le tableau 4.1 regroupe les papiers utilisant les méta-heuristiques à base de solution unique. Le tableau 4.2 présente les papiers utilisant les méta-heuristiques à base de population de solution. Quant au tableau 4.3, il recense les papiers utilisant des versions de méta-heuristiques améliorées par d'autres heuristiques ou hybridées avec d'autres.

Nous présentons, dans les trois tableaux et pour chaque papier, l'algorithme utilisé et l'ensemble de paramètres QoS considérés dans l'étude.

| **Approches** | **Algorithmes** | **QoS traités** |
|---|---|---|
| [Pop et al., 2011b] | recherche Tabou | C, Tr, D |
| [Hadjila and Chikh, 2012] | recherche Tabou | C, Tr, D |
| [Ko et al., 2008] | recherche Tabou et récuit simulé | Tr, C, D, F |
| [Rosenberg et al., 2010] | récuit simulé | D, C, Tr |
| [Parejo et al., 2014] | GRASP et Path relinking | D, C, Tr |

***Tableau 4.1:*** *Travaux utilisant les méta-heuristiques à base d'une seule solution et QoS traités (Temps de réponse(Tr), Coût (c), réputation (R), Disponibilité (D), Fiabilité(F))*

| **Approches** | **Algorithmes** | **QoS traités** |
|---|---|---|
| [Canfora et al., 2005] | Algorithme génétique GA | C, Tr,D, F |
| [Jaeger and Mühl, 2007] | | C, Tr, F |
| [Zhang et al., 2006] | | Pas mentionnées |
| [Gao et al., 2007] | | C, Tr |
| [Xiangbing et al., 2012] | | C, Tr, D, F |
| [Rodriguez-Mier et al., 2010] | | C |
| [Wang and Hou, 2008] | Multi-Objectif GA | C, Tr, F |
| [Ming and Zhen-wu, 2007] | Algorithme d'optimisation par essaime particulaire PSO | C, Tr |
| [Long and Gui, 2009] | | C, Tr |
| [Amiri and Serajzadeh, 2012] | | C, Tr,D, R |
| [Hadjila et al., 2012b] | | C, Tr, F, D, R |
| [Ludwig, 2012] | | F,D, R, Tr, C |
| [Liu and Yin, 2009] | version Discrète de PSO | Pas mentionnées |
| [Liao et al., 2011] | NichePSO | C, Tr |
| [Hadjila et al., 2012a] | sélection clonale | C, Tr, F, D, R |
| [Zhao et al., 2014] | la sélection négative | C, Tr |
| [Wang et al., 2010] | colonies de fourmis ACO | C, Tr |
| [Pop et al., 2010] | | D, F, Tr, C |
| [Xia et al., 2008] | ACO (multi-phéromones) | Pas mentionnées |
| [Jafarpour and Khayyambashi, 2009] | l'optimisation par harmonie HS | C, Tr,D, F |
| [Jafarpour and Khayyambashi, 2010] | | C, Tr,D, F |
| [Esfahani et al., 2012] | | Pas mentionnées |
| [Merzoug et al., 2014] | | C, Tr, F, D, R |
| [Hadjila et al., 2013] | Colonie d'abeilles artificielles ABC | C, Tr, F, D, R |
| [Chifu et al., 2010] | | D, F, Tr, C |
| [Kousalya et al., 2011] | Multi-objectif ABC | D, F, Tr, C |
| [Krithiga, 2012] | L'algorithme SOMA | Tr, C, D |
| [Chifu et al., 2011] | L'algorithme coucou | D, F, Tr, C |

***Tableau 4.2:*** *Travaux utilisant les méta-heuristiques à base de population de solutions, avec QoS traités (Temps de réponse(Tr), Coût (C), réputation (R), Disponibilité (D), Fiabilité(F))*

| **Approches** | **Algorithmes** | **QoS traités** |
|---|---|---|
| [Parejo et al., 2008] | recherche Tabou et AG | C, Tr, F, Taux de succès |
| [Yu et al., 2014] | | C, Tr, F, D |
| [Bahadori et al., 2009] | | Tr , C |
| [Gao et al., 2009] | récuit simulé et GA | D, C, Tr |
| [Liu et al., 2007] | PSO et GA | C, Tr |
| [Zhao et al., 2012b] | PSO et systèmes immunitaires | pas mentionnées |
| [Liu et al., 2011] | hqpso | C, Tr,F |
| [Pop et al., 2009] | algorithme immunitaire (Clonal et heuristique) | pas mentionné |
| [Yunwu, 2009] | colonies de fourmis et l'opérateur Chaos | C, Tr |
| [Li and Yan-Xiang, 2010] | Multi objectif chaos ant coloni (MOCACO) | Tr,D, R, C |
| [Xu and Reiff-Marganiec, 2008] | algorithme immunitaire (Clonal et heuristique) | Tr, C,D, F |
| [Yang et al., 2010] | ant coloni et GA | Tr |
| [Wang et al., 2013b] | ABC amélioré par la recherche Glouton | C |
| [Wang et al., 2013a] | GA amélioré par la recherche Glouton | C |
| [Zhao et al., 2012a] | l'algorithme clonale et PSO | C, Tr,D, F |
| [Pop et al., 2011a] | L'algorithme Coucou et la recherche Tabou avec l'apprentissage par renforcement | D, F, Tr, C |
| [Salomie et al., 2014] | L'algorithme cuckoo et l'algorithme Firefly | C, Tr |
| [Boussalia and Chaoui, 2014] | principes de la recherche cuckoo et ceux de la mécanique quantique | D, F, Tr, C |

***Tableau 4.3:*** *Travaux utilisant les méta-heuristiques hybrides ou améliorées et QoS traités (Temps de réponse(Tr), Coût (c), réputation (R), Disponibilité (D), Fiabilité(F))*

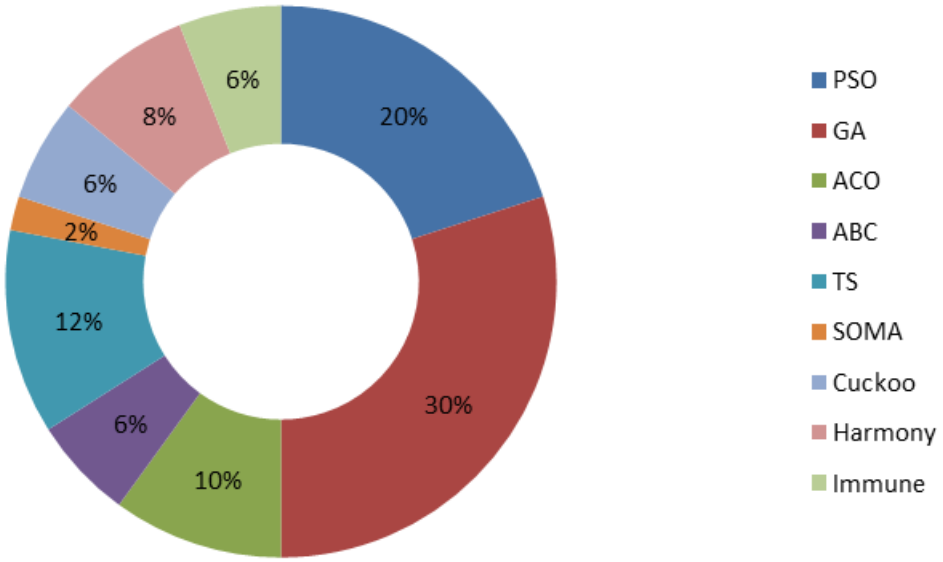



**Figure** ***4.3:*** *La fréquence d'utilisation des méta-heuristiques dans la résolution de QoSWSC*

## 4.4 Synthèse

Nous avons présenté, de manière succincte, dans cette section, une cinquantaine de papiers qui utilisent les méta-heuristiques dans la résolution de la problématique QoSWSC. A partir de l'analyse des papiers présentés, nous pouvons aboutir aux constations suivantes :

Il est clair, que l'importance de certaines méta-heuristiques dépassent de loin les autres. Nous constatons que certaines méta-heuristiques sont plus utilisées que d'autres dans la résolution du problème QoSWSC. En effet, nous pouvons remarquer, à partir de la figure 4.3, qu'un nombre élevé de papiers utilisent les algorithmes génétiques, du fait de leurs popularité. Se positionnent juste après, les algorithmes PSO qui sont suivis par les autres méta-heuristiques avec un pourcentage plus au moins équivalent.

Nous remarquons aussi que d'autres méta-heuristiques sont peu utilisées du fait de leurs récentes apparitions comme les algorithmes de recherches par coucou CS ou par harmonie HS ou encore SOMA. Cependant, l'algorithme SOMA , bien qu'il soit apparu dans la même période que la recherche par harmonie, n'est pratiquement pas utilisé dans la résolution du problème de QoSWSC. Cependant, l'utilisation intensive d'une méta-heuristique ne reflète pas son efficacité par rapport aux autres. La preuve, l'algorithme SOMA est très performant dans la résolution des problèmes académiques comme les problèmes de Flow shop [Davendra and Bialic-Davendra, 2013] ou encore

le voyageur de commerce [Čičková et al., 2008]. Par contre, il demeure encore peu connu et peu exploité par la communauté des services web.

D'autres méta-heuristiques sont très jeunes et appartiennent à la dernière génération des méta-heuristiques comme l'optimisation par les lucioles Fire Fly (FF) [Yang, 2010], par la mouche des fruits (FruitFly) [Pan, 2012], par loup gris (Grey wolf optimizer) [Mirjalili et al., 2014] , ou par mécanisme de chasse de fourmi-lions (Ant lion optimizer) [Mirjalili, 2015]. Elles restent encore non exploitées dans l'optimisation du problème de sélection des services web.

Si nous considérons les méta-heuristiques standards (sans amélioration, ni hybridation), il s'avère que la majorité des papiers s'accorde sur le fait que l'algorithme PSO donne de meilleurs résultats que les autres algorithmes. Cependant le papier [Krithiga, 2012], trouve via une expérimentation que SOMA est meilleur que PSO. Par ailleurs, nous pouvons remarquer qu'un nombre assez important de travaux combinent l'utilisation de plusieurs méta-heuristiques. Effectivement, si nous comparons les méta-heuristiques standards, il est clair que chaque méta-heuristique présente des avantages que nous cherchons à maximiser et des lacunes à combler. Partant de ce principe, beaucoup de papiers (regroupés dans le tableau 4.3) proposent la combinaison des méthodes de résolution des problèmes afin de tirer profit des points forts de chacune et de proposer des alternatives plus efficaces et plus performantes.

Enfin, nous pouvons confirmer l'observation faite pour l'analyse des QoS des travaux présentés dans le chapitre 3. Effectivement, dans ces papiers aussi, nous constatons que les paramètres QoS les plus considérés sont le temps de réponse, le coût, la fiabilité et la disponibilité.

## 4.5 Conclusion

Nous avons présenté dans ce chapitre un état de l'art des méta-heuristiques utilisées dans la résolution du problème de QoSWSC. Après avoir rappelé quelques notions préliminaires, nous avons présenté une classification des méta-heuristiques en deux classes : les méthodes qui font évoluer une seule solution et les méthodes à base de population de solutions. Nous avons, par la suite, passé en revue les principaux algorithmes de l'état de l'art utilisés pour solutionner le problème de sélection des services web composites. Nous avons constaté que les jeunes méthodes de résolution comme SOMA, FireFly ou encore FruitFly sont peu exploitées dans la résolution du problème QoSWSC. Un intérêt particulier sera porté à la méta-heuristique SOMA qui sera étudiée dans le chapitre 6.

Chapitre 5

# La sélection des Services Web basée sur les techniques orientées bases de données

*"La marque d'une domination est qu'elle déplace les problèmes et parvient à empêcher qu'on pose ceux qu'elle est incapable de résoudre."*

Jean-Marie Domenach

## Sommaire

## 5.1 Introduction

La majorité des travaux, cités dans les approches basées sur les méthodes exactes et/ou heuristiques (cf.chapitre 3), est plus appropriée pour un nombre limité de services et de paramètres QoS. Par contre, le processus de sélection est devenu un problème de plus en plus complexe avec l'augmentation croissante des services dans le web. Ces approches utilisent le plus souvent une fonction de score SAW qui oblige les utilisateurs à exprimer leurs préférences sous forme d'une valeur numérique, car ils doivent assigner des poids pour chaque paramètre QoS, ceci conduit à deux inconvénients majeurs :

- Le premier est relatif aux choix et à l'attribution des poids. En effet, les utilisateurs ne savent pas très bien comment balancer leurs préférences entre les différents critères de qualités en utilisant des valeurs numériques ; ils peuvent ainsi perdre en flexibilité pour le choix de leurs services Web désirés.
- Le deuxième est relatif aux changements de ces poids. Ce point est lié au premier, dans le sens où les utilisateurs doivent passer par plusieurs essais d'exécution (changement de valeurs de poids) afin de trouver les poids ajustés à leurs préférences. Chaque changement de poids nécessite d'exécuter une nouvelle recherche, ce qui augmente le temps d'attente pour le retour de la requête, et ce temps augmente aussi considérablement avec l'augmentation du nombre des critères QoS à prendre en considération.

Pour pallier ces limites, de nouveaux travaux sur la sélection des services web ont adopté des approches qui reposent sur les techniques avancées des bases de données pour le traitement de requêtes complexes. Effectivement, deux grands domaines ont reçu une importance croissante de la part des chercheurs et ont fait l'objet de plusieurs travaux : les requêtes Top-k et les requêtes Skyline. Récemment, ces deux domaines ont été combinés en un troisième appelé requêtes Top-k dominating. Le domaine des requêtes Top-k dominating tire les avantages des requêtes Top-k et ceux des Skyline en évitant ainsi les inconvénients des deux.

Les requêtes Skyline et les requêtes Top-k dominating constituent un outil puissant d'analyse de données multicritères en vue de prendre des décisions intelligentes face à des données à grande échelle, où l'utilisateur a du mal à formuler ses préférences. Elles permettent d'extraire l'ensemble des points les plus intéressants quand différents critères, souvent conflictuels, sont pris en compte. Elles s'appuient sur le principe de dominance de Pareto. Ce chapitre sera consacré à ces deux axes de recherche et leurs

applications dans le domaine de la sélection des services web. Dans un premier temps, nous allons donner quelques définitions des concepts bases de ces deux techniques à travers des exemples illustratifs, ensuite un état de l'art des travaux sera établi. Enfin, nous terminerons avec une étude comparative des différents travaux actuels dans la communauté des services web qui se basent sur ces deux domaines pour résoudre le problème QoSWSC.

## 5.2 Concepts de bases et définitions

### 5.2.1 Exemple introductif

Considérons une personne qui cherche à se faire une coloration de cheveux dans un salon de coiffure. La personne en question sera intéressée par le salon de coiffure le plus proche possible de son domicile et ayant le tarif le plus faible. Chaque salon est représenté par deux dimensions : la distance du salon et le prix de la prestation (voir tableau 5.1). Nous allons définir le principe de requêtes Skyline et Top-k dominating à travers cet exemple.

| **Salons** | **p1** | **p2** | **p3** | **p4** | **p5** | **p6** | **p7** | **p8** | **p9** | **p10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Distance (m)** | 100 | 250 | 400 | 700 | 300 | 480 | 400 | 600 | 850 | 950 |
| **Prix (€)** | 50 | 20 | 40 | 15 | 65 | 55 | 80 | 70 | 30 | 60 |

***Tableau 5.1:*** *Exemple de salons de coiffures (dimensions : distances et prix)*

### 5.2.2 Le concept Skyline

Reprenons l'exemple introductif du tableau 5.1, et comparons les deux points p3 et p7. On remarque que p3 est égal à p7 dans la distance mais est inférieur sur le prix, donc p3 domine p7. De la même façon le point p1 domine p3 car il est inférieur sur les deux dimensions que p3.

**Définition 5.1 - La relation de Pareto dominance.**
Étant donné un espace $D = d1, .., dn$ à n dimensions et un ensemble de points E, on dit qu'un point $p \in E$ domine un autre point $q \in E$ au sens de Pareto selon D, noté par $p \succ q$, si $p$ est meilleur ou égal à $q$ dans toutes les dimensions et strictement meilleur dans au moins une dimension.

$$p \succ q \Leftrightarrow \forall di \in D, p[i] \geq q[i] \wedge \exists j \in D, p[j] > q[j] \tag{5.1}$$

Trois cas de comparaisons entre deux points p et v se présentent selon la notion de Pareto :

- le point p **domine** le point v ou bien ;
- le point p **est dominé par** v ou bien ;
- le point p et v sont **incomparables** : le point p ne domine pas v et le point v ne domine pas p.

Les points qui ne sont dominés par aucun autre point de l'espace appartiennent au Skyline, ces derniers sont incomparables entre eux.

**Définition 5.2 - Skyline.**
L'ensemble des points Skyline de E selon D est l'ensemble des points de E qui ne sont dominés (selon D) par aucun autre point de E :

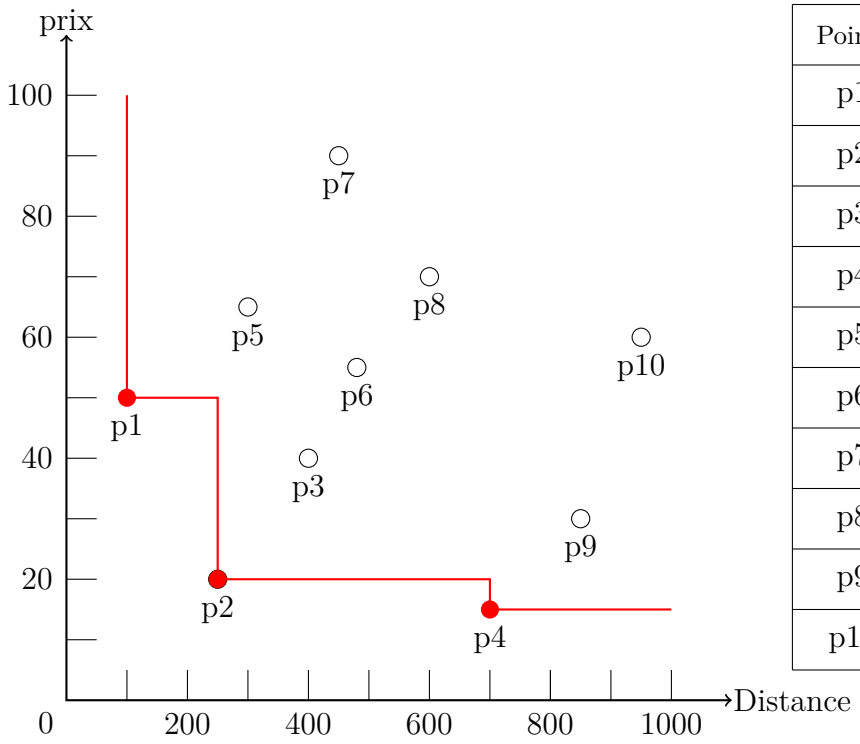
$$Skyline(D, E) = \{u \in E | \ \nexists p \in E; p \prec u\}. \tag{5.2}$$

Reprenons l'exemple introductif (voir Tableau 5.1), le Skyline de l'exemple est Skyline(D,E)=p1, p2, p4. En effet, les salons p3 et p5,..,p10 sont dominés par au moins un autre salon. Donc ils ne peuvent être dans le Skyline. Par contre les points p1, p2, P4 ne sont dominés par aucun autre point. La figure 5.1 montre une représentation graphique du Skyline.

Les requêtes Skyline retournent un ensemble de données qui n'est pas trié. Le concept n'offre pas de fonction de score contrairement aux requêtes Top-k qui, quant à elles, donnent la possibilité à l'utilisateur de spécifier le nombre k de réponses les plus pertinentes que le système doit lui retourner. Le degré de pertinence (score) des réponses par rapport à la requête est déterminé par une fonction de score qui est souvent linéaire. Cependant, Les requêtes Top-k sont parfois difficiles à définir par l'utilisateur, spécialement si plusieurs attributs doivent être optimisés (difficulté de définition de la fonction de score). Les deux domaines ont tiré avantages l'un de l'autre en fusionnant leurs principales notions donnant naissance à une nouvelle forme de requête : le Top-k dominating queries ou le Top-k Skyline.

### 5.2.3 Concept de Top-k dominating

L'idée principale de requêtes Top-k dominating est de calculer et assigner à chaque objet de la base un score de manière intuitive. Ce score, basé sur la relation de dominance, reflète l'importance de chaque object du dataset d'une manière naturelle




| Points | Coordonnées |
|---|---|
| p1 | (100 , 50) |
| p2 | (250,20) |
| p3 | (400, 40) |
| p4 | (700,15) |
| p5 | (300,65) |
| p6 | (480,55) |
| p7 | (400, 80) |
| p8 | (600, 70) |
| p9 | (850, 30) |
| p10 | (950, 60) |

**Figure *5.1:*** *Exemple illustratif de points Skyline*

et permet de retourner à l'utilisateur les k point les plus pertinents. Ainsi, le Top-k dominating permet de retourner les Top-k points selon leurs forces de dominance. Ces k points sont déterminés grâce à leurs puissances de domination (domination power). La puissance de domination d'un point p dans un ensemble S est le nombre de points que p domine dans S.

$$dom(p) = |\{q \in E : q \prec p\}| \tag{5.3}$$

Autrement dit, Top-k dominating permet de retourner l'ensemble de k points qui dominent le plus large nombre de points de l'ensemble total des points de l'espace de recherche. Ceci veut dire qu'un point p est préféré à un autre point q, si la force de dominance (domination power) du point p est plus grande que celle du point q.

Reprenons notre exemple illustratif (voir Tableau 5.1), et calculons la force de dominance des points de notre exemple selon 5.3. La force de dominance d'un point p peut être aussi calculée à partir de la définition de la région de dominance de ce dernier. La région de dominance d'un point p est la zone rectangulaire qui a des valeurs de x et y plus grandes au coordonnées du point p, alors la région de dominance contient tous les points que p domine. La figure 5.2 représente la région de dominance du point p3 qui contient 4 points, $dom(p3) = 4$ et la région de dominance de p2,

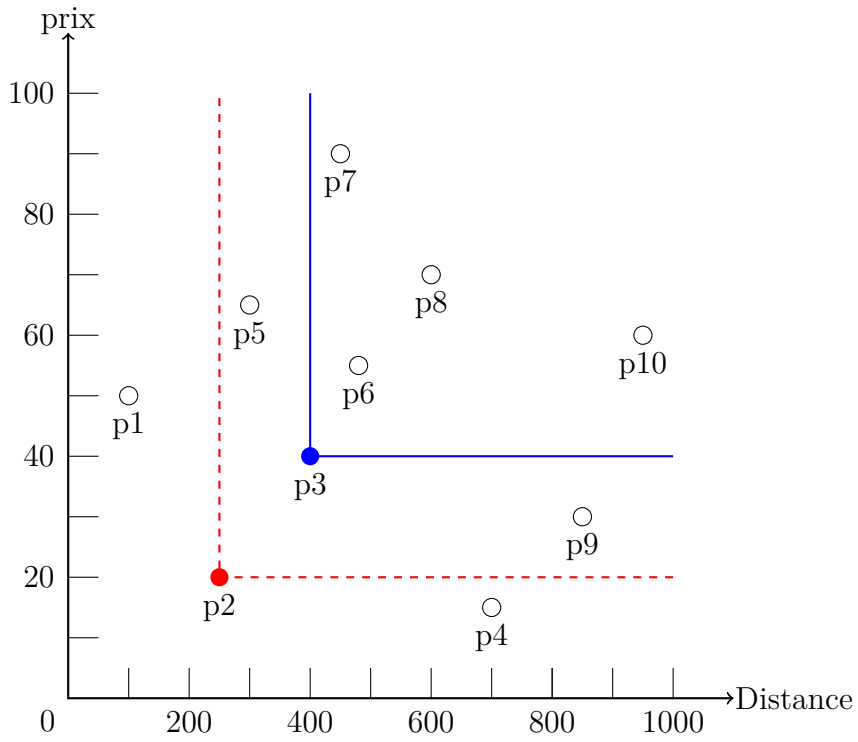



**Figure *5.2:*** *Exemple de Top-k dominating*

$dom(p2) = 7$. De ce fait, le point p2 est préféré à p3 car sa force de dominance est supérieure à ce dernier. De la même manière, la force de dominance de p1 = 5 ; la force de dominance des points p4, P5 et P6 est égale à 2 ; et la force de dominance des points p7, P8, p10 est égale à 0. Par conséquent le top-3 dominating de l'exemple retourne les points p2, p1 et p3. Notons que p2 et p1 sont des points Skyline alors que p3 ne l'est pas. Le point Skyline p4 ne figure pas dans le top-3 dominating car $dom(p3) > dom(p4)$. Nous constatons que les deux concepts ne retournent pas le même résultat, du fait qu'ils ne fonctionnent pas de la même manière. Nous constatons aussi que le Top-1 dominating est un point Skyline car c'est le point qui a le plus grand nombre de points dominés donc, il n'est dominé par aucun autre point.

Nous avons vu que le Top-k dominating permet de classer les points de l'espace de recherche selon une relation de dominance. Il n'existe pas une fonction standard pour calculer la force de dominance et le score de dominance qui servira à classer et retourner les k premiers points. Une des manières les plus simples et intuitives et de calculer ce score comme nous l'avons défini dans l'équation 5.3 et de classer les points par ordre décroissant, mais nous verrons par la suite, dans les travaux proposés (cf section 5.3.2), que certains proposent de classer les points de la base selon des fonctions plus élaborées en considérant par exemple le nombre de points dominés, le nombre de points dominants et/ou une combinaison des deux en introduisant un facteur de

pénalisation ou autre. Cette notion est importante, pour nous, dans notre travail car nous verrons, plus tard, qu'elle déterminera la qualité du classement retourné. En effet, cette notion qui, associée à l'un des algorithmes de classements Top-k ou de Skyline, est le cœur du classement et définit, selon nous , la qualité de l'approche proposée. Nous nous appuierons ainsi sur la fonction de score de dominance pour classer les travaux de la partie (cf section 5.3.2).

La majorité des recherches sur le Top-k dominating et/ou le Skyline concentre leurs travaux sur les algorithmes de calcul qui retournent de manière efficace ces points. La section suivante présentera l'essentiel de ces travaux et leurs utilisations dans le domaine de sélection des services web.

## 5.3 Etat de l'art

### 5.3.1 Travaux sur Skyline et Top-k dominating

L'idée de Skyline vient des anciens travaux sur le problème de contours, le vecteur maximal et enveloppe convexe (Convex hull) et a été présentée pour la première fois dans des bases de données par Borzsonyi [Borzsony et al., 2001] comme étant une clause SQL étendue. Borzsonyi a développé trois algorithmes : Block Nested Loop(BNL), Divide & Conquer (DC) and B-tree. BNL permet d'itérer un ensemble de comparaison de chaque point de l'ensemble de données avec les autres et ne rapporte que ceux qui ne sont pas dominés. L'idée principale de l'algorithme DC est de diviser l'ensemble de données en plusieurs partitions et de calculer pour chaque partition le Skyline partiel, le Skyline final est calculé en fusionnant l'ensemble des Skyline partiels.

L'algorithme BNL et DC génèrent un temps de calcul assez important et ne sont adaptés que pour des ensembles de données réduits, de ce fait, plusieurs améliorations ont été proposées comme l'algorithme SFS (Sort First Skyline) [Chomicki et al., 2005] où l'auteur propose d'ordonner toutes les données en entrée à l'aide d'une fonction de score monotone, et permet ainsi d'extraire les points Skyline de manière progressive en diminuant le nombre de comparaisons de dominance entre les points. [Bartolini et al., 2008] propose quant à lui une amélioration de l'algorithme SFS en proposant l'algorithme 'Linear Elimination Sort for Skyline (LESS)'. [Bartolini et al., 2008] rajoute une autre amélioration à SFS qui permet de stopper le calcul du Skyline avant le balayage de l'ensemble entier des données.

D'autres améliorations reposent sur l'exploitation des structures d'index pour accélérer le processus de calcul du Skyline comme les travaux de [Lee et al., 2007].

Nearest Neighbor algorithm (NN) est proposé par [Chan et al., 2006a]. Il permet de retourner les points Skyline de manière progressive en utilisant une indexation basée sur l'arbre R*-tree. Cette méthode utilise la méthode du plus proche voisin pour partitionner les données de manière récursive. [Kossmann et al., 2002] ont proposé une extension de l'algorithme NN pour supporter les données multidimensionnelles.

Papadias and al [Papadias et al., 2005] ont proposé l'algorithme 'the Branch And Bound Skyline Algorithm' (BBS) qui tout comme l'algorithme NN, utilise un arbre R*-tree et est basé sur la recherche des plus proches voisins. Par contre, contrairement à NN, il n'explore qu'une seule fois l'arbre. Il part de la racine du R-Tree et l'explore en descendant seulement dans les branches qui lui sont utiles.

Les algorithmes que nous venons de citer permettent de calculer le Skyline sur tout l'ensemble des dimensions. Lorsque le nombre de dimensions augmente, alors les requêtes Skyline commencent à perdre leur pouvoir discriminant en renvoyant une grande partie des données. Pour pallier cette contrainte, des travaux se sont alors intéressés à l'extraction de points Skyline dans des sous-espaces de dimensions et non sur la totalité. Parmi ces travaux nous citons celui de Tao and al [Tao et al., 2006] où les auteurs ont proposé l'algorithme 'Subsky'. Subsky permet le calcul de requêtes Skyline dans des sous-espaces arbitraires. Il convertit les données multidimensionnelles en valeurs à une dimension qui sont indexées dans un B-Tree.

D'autre travaux proposent de nouveaux algorithmes de calcul de Skyline et présentent d'autres opérateurs Skyline. Dans [Chan et al., 2006a], les auteurs proposent le concept de fréquence de Skyline 'Skyline frequency' qui est le nombre de sous ensemble où un point p est Skyline , cependant ceci amène à calculer le Skyline de tous les ensembles et conduit à un très grand temps de calcul, plus tard les auteurs introduisent un algorithme approximatif pour réduire le temps de calcul. Dans [Chan et al., 2006b], Chan et al. présentent la notion de k-dominance qui relaxe la notion de Pareto dominance à un sous ensemble de k paramètres, cependant cette approche rend la relation de dominance cyclique (CDR) qui conduit parfois à éliminer certains éléments du Skyline et parfois retourne un ensemble vide. Dans [Lin et al., 2007], les auteurs proposent l'opérateur 'Top-k representatif Skyline' qui permet de sélectionner un ensemble de points, pour qui le nombre de points dominés, par au moins un d'entre eux, soit maximisé. Mais cette approche est plus appropriée pour les données anti-correlées, de plus le k-representatif Skyline est considéré comme étant NP-difficile

pour un ensemble de données à 3 dimensions. Dans [Vlachou and Vazirgiannis, 2010], l'auteur présente un graphe de Skyline qui transforme la dominance des différents sous espaces du Skyline en un graphe direct pondéré et utilise link-based techniques pour classer le Skyline.

Ces algorithmes de calcul de Skyline que nous venons de citer, ne sont pas directement applicables à notre problème de sélection de services Web car :

- Ils ne traitent pas la question de classement. En effet, les algorithmes de Skyline n'utilisent pas de fonction pour classer les objets retournés (ce n'est pas leur but de toute façon), ils retournent un ensemble d'objets incomparables entre eux.
- Nous ne connaissons pas la taille des objets retournés à priori. Car, selon la dimension de données ainsi que leur distribution (indépendante, corrélée ou anti-corrélée)(voir image 5.3), la taille peut souvent être trop grande ou trop petite. En effet, les objets retournés par le Skyline augmentent considérablement avec l'augmentation de la taille des dimensions. Ce qui peut engendrer un temps d'attente important pour l'utilisateur et une difficulté de choix entre le grand nombre de résultats retournés.

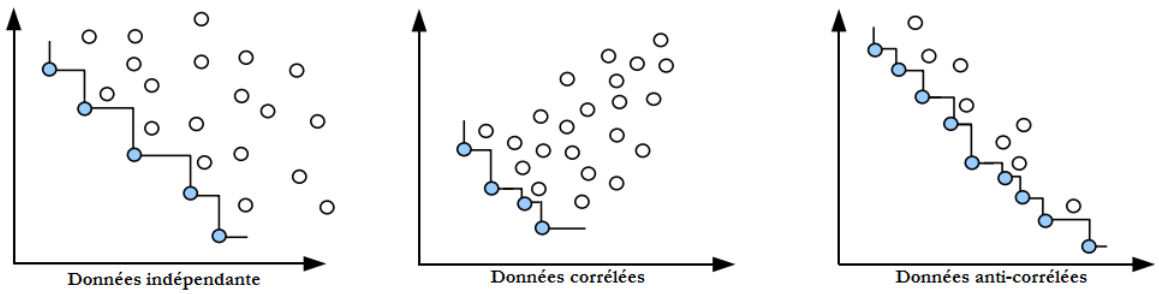



**Figure *5.3:*** *Taille de Skyline selon différentes distributions de données*

Nous verrons par la suite que ces algorithmes sont généralement utilisés seuls, dans le but de réduire le nombre de services candidats. Cette utilisation correspond à une sélection locale. Autrement, ils sont combinés à d'autres algorithmes pour traiter les contraintes globales de la composition.

Pour pallier aux problèmes que nous venons d'énoncer, de très récents travaux ont combiné les avantages des requêtes Top-k et des requêtes Skyline en donnant naissance au concept Top-K dominanting (voir définition section 5.2.3). Nous présentons l'essentiel de ces travaux dans ce qui suit.

Les travaux sur les requêtes Top-k dominating ont eux aussi joué un rôle significatif

dans les applications de décisions multi-critères. La première approche pour calculer et trouver les points Top-k dominating a été présentée dans [Papadias et al., 2005] où le concept de Top-k dominating a été défini comme une variation du Skyline. Notons que la notion de Top-k dominating repose elle aussi sur la notion de Pareto dominance, quoiqu'elle présente quelques différences par rapport au Skyline (cf section 5.2.3). Papadias a défini un algorithme nommé 'Skyline based Top-k dominating algorithme' (STD) qui utilise le Branch and Bound Skyline Algorithme (BBS).

Entre temps, d'autres travaux sur les requêtes Top-k dominating [Soliman et al., 2008] ont montré l'importance de combiner les deux concepts : les requêtes Top-k avec celles du Skyline et ont concentré leur efforts pour adresser comment retourner les données du Top-k dominating [Lian and Chen, 2009], [Yiu and Mamoulis, 2007], [Zhang et al., 2010]. Un très bon état de l'art sur les travaux actuels pour améliorer les algorithmes de calcul de Top-k dominating peut être trouvé dans l'article [Tiakas et al., 2007]

### 5.3.2 Travaux sur la sélection des services web utilisant Skyline et/ou Top-k dominating

L'utilisation des Skyline avec ses variantes et/ou Top-k dominating concept n'ont été introduits pour des services web que récemment. Ils ont été exploités aussi bien dans la phase de découverte des services que dans la phase de sélection. Ces techniques permettent, entre autres, de présenter des listes de services web réduites pour chaque classe participante à la composition. Avant de présenter les travaux se basant sur le Skyline dans le processus de sélection , nous commençons dans un premier temps par présenter l'un des travaux importants reposant sur le principe du Top-k dominating qui est celui de Skoutas [Skoutas et al., 2009].

Quoique [Skoutas et al., 2009] traitent les contraintes fonctionnelles et non celles des paramètres de QoS, le principe reste le même et applicable sur les critères de QoS. Les auteurs utilisent les principes de Probabilistic Skyline 'p-Skyline' [Pei et al., 2007] et rajoutent par rapport à ce dernier un classement pour les données (principes de Top-k dominating). En effet, Skoutas et al. ont défini trois façons de calculer le score de dominance ('the dominated score', 'the dominating score' et 'the dominance score) pour classer les services selon leurs vecteurs de données.

- le premier score est appelé **'the dominated score'** ou **'le score dominé'** : il est basé sur la relation de Pareto 'est dominé par' et considère le nombre de

services qui dominent u dans sa classe. Il est défini comme suit : Étant donné un vecteur u dans l'ensemble de vecteur V, le score dominé de u dans V dénoté par *dds* est calculé par :

$$u.dds = \sum_{v \neq u} (\frac{|\{v \in V, v \succ u\}|}{|V|}) \tag{5.4}$$

Le score $u.dds$ considère le nombre moyen de vecteurs qui dominent $u$. Par conséquent, plus le score u.dds est petit, meilleur est le vecteur de critères.

- le deuxième score est appelé **'the dominating score'** ou **'le score dominant'** : il est basé sur la relation de Pareto 'domine', Étant donné un vecteur u dans l'ensemble de vecteur V, le score dominant de u dans V dénoté par *dgs* est calculé par :

$$u.dgs = \sum_{v \neq u} (\frac{|\{v \in V, u \succ v\}|}{|V|}) \tag{5.5}$$

Le score $u.dgs$ considère le nombre moyen de vecteurs que $u$ domine. Par conséquent, plus le score $u.dgs$ est élevé, meilleur est le vecteur de critères.

- Le troisième score est appelé **'the dominance score'** ou **'le score de dominance'** il est noté par *dgs* et est calculé à partir des deux scores précédents par :

$$u.ds = \sum_{u \in V} \frac{(u.dgs - \lambda.u.dds)}{|V|} \tag{5.6}$$

Le score de dominance de $u.ds$ favorise $u$ pour chaque vecteur que $u$ domine, et le pénalise pour chaque vecteur qui domine $u$ avec un paramètre $\lambda$ qui est fixé empiriquement par les auteurs.

Les auteurs proposent ensuite 3 algorithmes TKDD, TKTD, et TKM dont chacun présente un classement de services qui utilisent respectivement les trois scores Dominated score, Dominating score et Dominance score. Les expérimentations menées sont très intéressantes, dans le sens où elles montrent que l'algorithme le plus efficace est celui qui utilise le score dominé 'the dominating score' TKDD. Ce dernier retourne en haut de la liste les services de meilleur qualité que ceux retournés par TKTD et TKM. Basés sur ces résultats, le score dominé (the dominated score) sera redéfini selon notre contexte de sélection basée sur QoS et utilisé dans la proposition de nos approches.

La notion de Skyline est utilisée dans la sélection des services web basée sur QoS beaucoup plus pour permettre la réduction de l'espace de recherche dans la phase

locale, et par la suite, assurer un gain de temps considérable lors de l'optimisation globale. Parmi les travaux utilisant le Skyline et/ ou top-dominance pour classer les services selon QoS, on cite :

1- Alrifai et. al [Alrifai et al., 2010] proposent une approche qui permet de sélectionner les services web d'une composition en ne considérant que les services Skyline de chaque classe, ce qui permet de réduire considérablement l'espace de recherche. Pour ne pas affecter le temps de calcul, la phase de calcul des Skyline des classes est faite en off-line. Faire participer les Skylines trouvés de chaque classe, tels quels, ne permet pas de garantir les critères globaux. Pour cela, les auteurs proposent une approche qui repose sur un clustering et un choix de services représentatifs pour trouver par la suite la composition proche de l'optimale qui satisfait les contraintes QoS globales. Les auteurs proposent trois algorithmes :

- L'algorithme 1 'Exact Skyline', permet d'extraire les services Skyline de chaque classe.
- L'algorithme 2 'representative- Skyline', permet de réaliser un clustering hiérarchique descendant sur les services de chaque Skyline trouvé par l'agorithme 1. Chaque cluster possède un service représentant qui a le plus grand score (calculé par une fonction mono-objectif).
- L'algorithme 3 'Hybride-Skyline' adapte la recherche hybride [Alrifai and Risse, 2009] pour l'ensemble des Skyline extraits de chaque classe. Il utilise MIP pour résoudre l'optimisation globale. Les entrées de cet algorithme sont les services représentatifs de chaque niveau du Skyline.

Les auteurs ont testé leur approche sur la base de données réelles [Al-Masri and Mahmoud, 2008] et 03 autres bases synthétiques : données indépendantes, données positivement corrélées et données anti-correlées car la taille de Skyline diffère selon la nature des données. La taille du Skyline augmente plus dans la base anti corrélée. Ils comparent ensuite leur proposition avec leur première approche [Alrifai and Risse, 2009] qui utilise une approche de résolution exacte. L'approche proposée améliore nettement le temps d'exécution par rapport à l'approche exacte.

2- Yu et. al [Yu and Bouguettaya, 2013] proposent une nouvelle méthode Bottom-Up Computation framework (BUC) qui permet d'extraire les compositions Skyline C-Sky. Yu et. al, utilisent, dans un premier temps, une stratégie locale qui permet de calculer l'ensemble des Skyline S-Skies, qui sont ensuite utilisées pour déterminer les C-Sky. L'approche BUC permet d'intégrer une stratégie de composition linéaire

pour booster la scalabilité de ces algorithmes. Elle repose sur deux algorithmes :

- le premier est appelé 'One Pass Algorithm' (OPA) qui permet de scanner une seule fois la liste triée des S-Skies pour générer les C-Sky, alors que d'autres algorithmes de Skyline ont besoin de faire plusieurs passages.
- le deuxième algorithme appelé 'Dual Progressive Algorithm' DPA qui repose sur une pile et une comparaison entre Skyline pour énumérer progressivement les C-Sky.

Les expérimentations sont réalisées sur des collections de données synthétiques corrélées, anti-corrélées et indépendantes. Les expérimentations montrent l'efficacité à calculer l'ensemble des Skyline pour les compositions sous un nombre important de QoS, mais l'approche génère l'ensemble de compositions qui ne sont pas dominées sans traiter les contraintes globales QoS.

3- Tang et. al [Tang et al., 2011] proposent une approche qui se base sur le principe Top-k dominating pour retourner les Top-k services d'une classe. Les auteurs définissent un premier algorithme 'Dominating Score Algorithm DSA' qui permet de calculer le score de dominance et d'attribuer un score à chaque service d'une classe. Le score est calculé par la fonction $\psi(s)$ qui repose sur le principe de 'The dominating score', $\psi(s) = |\{s'|s \succ s'\}|$. L'algorithme DSA repose sur une recherche avec un index multidimentionnel 'agreggated index' aR-tree [Papadias et al., 2005], [Yiu and Mamoulis, 2007] pour représenter les services, ce qui permet d'affecter le score à un service en un seul parcours de l'arbre en évitant de faire des comparaisons de chaque service avec l'ensemble de tous les autres services dans la même classe. Notons que l'algorithme DSA peut suffire à lui seul pour retourner les Top-k services, il suffit de trier les services selon leur score $\psi(s)$. Néanmoins, les auteurs définissent un deuxième algorithme nommé 'Dept-first Retrieving' (DFR) qui se base sur DSA. Ce dernier s'avère être plus performant que l'utilisation de DSA seul, car il permet de trier les services en utilisant un tas ce qui diminue le temps de calcul considérablement. Nous notons que aR-tree utilisé dans l'approche proposée est dynamique [Guttman, 1984], ce qui signifie que l'action d'insertion ou de suppression des services peut être entremêlée avec la recherche sans aucune réorganisation périodique et convient à l'aspect dynamique des services web où les services peuvent quitter ou rejoindre une classe dynamiquement. Par contre, les auteurs ne traitent pas l'aspect des contraintes globales d'une composition. Il génèrent les Top-k dominating services pour chaque classe du workflow, ensuite, il choisissent aléatoirement un service de chaque classe pour générer la composition.

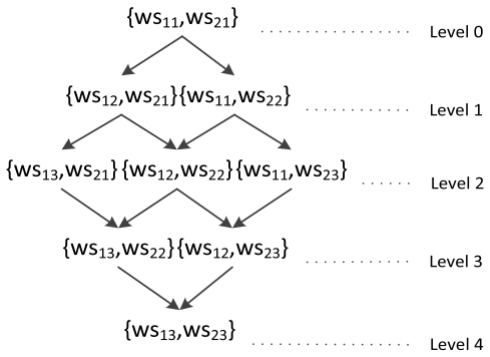



**Figure *5.4:*** *Un exemple d'un treillis de services qui énumère toutes les compositions*

4- zhang et. al [Zhang et al., 2013] proposent une approche qui permet de sélectionner les Top-k compositions. Cette approche se déroule en deux phases de sélection :

- La phase locale : les auteurs calculent le Skyline de chaque classe nommé UPS(User Préférence-aware service Skyline) en ne considérant qu'un sous ensemble des QoS (UP), qui représente les préférences des utilisateurs (c-à-d les QoS à prendre en considération dans la sélection). Ils proposent un algorithme de calcul des Skyline UPS appelé 'Multi-Index-Alogorithm'. L'algorithme permet de générer dynamiquement un index pour chaque UPS différent. Les différents Skyline UPS, quant à eux, sont générés par une version améliorée de l'algorithme 'Index' [Tan et al., 2001]. Les services de chaque UPS sont classés dans un ordre ascendant selon leurs scores. Le score d'un service est calculé à partir d'une équation de sommation de tous les QoS appartenant au UP.
- La phase globale : les auteurs proposent un algorithme d'optimisation globale qui permet de sélectionner les Top-k compositions, pour cela, l'algorithme reçoit en entrée les Top-k services de chaque classe. L'algorithme est combiné avec l'utilisation d'une structure de données en treillis (voir figure 5.4) qui énumère toutes les compositions (Chaque noeud du treillis est une composition) et une structure en tas ordonnée par maximum qui permet de sauvegarder les nœuds de treillis lors de son parcours. La structure en tas ordonnée par maximum repose sur un arbre binaire parfait où tout nœud est plus grand que ses fils. Donc, la composition qui a le score maximum est insérée dans la racine du tas. L'algorithme permet de parcourir le tas, de le classer et de retourner les Top-k compositions. Le classement dans cette phase est différent de celui utilisé pour classer les services, les compositions sont classées selon le principe 'the dominating score' où le score d'une composition est égal à la somme du nombre

dominant de chaque service de la composition. Le nombre dominant d'un service S est égal au nombre des services S' dominés par S dans son UPS.

5- Hadjila et al. [Hadjila, 2014] présentent une approche de sélection des services proche de celle proposée par [Alrifai et al., 2010] dans le sens où, Alrifai et al. utilisent eux aussi le Skyline dans la sélection locale associée à un algorithme de classification, pour réduire l'ensemble des services dans chaque classe. L'approche proposée par Hadjila et al. est définie en trois étapes :

- La première étape, extrait les services Skyline de chaque classe en utilisant l'algorithme SFS 'Sort First Skyline'. [Chomicki et al., 2005]
- La deuxième étape, permet le clustering hiérarchique ascendant des services Skyline, de chaque classe abstraite, en utilisant l'algorithme de ward [Ward, 1963]. L'avantage de cette méthode par rapport à celle utilisée dans [Alrifai et al., 2010], est qu'elle donne un résultat plus stable (la méthode K-means est toujours dépendante de la phase d'initialisation aléatoire, et de ce fait, elle n'est pas stable). Un service représentatif est choisi de chaque cluster pour participer à l'optimisation globale.
- La troisième étape est la phase globale. Elle repose sur une fonction objectif qui permet de vérifier si la composition des services représentatifs de chaque classe respecte les contraintes globales. La composition qui maximise la fonction objectif et vérifie les contraintes est choisie.

Tous les travaux que nous venons de citer reposent sur la notion de Pareto dominance au sens strict (sans extension) que ce soit pour :

- le calcul du score utilisé pour le classement : tous les algorithmes Top-k utilisent une fonction qui repose sur la relation de Pareto dominance au sens strict.
- le calcul du Skyline : tous les algorithmes proposés dans ces travaux pour le calcul de Skyline reposent sur le principe de Pareto dominance au sens strict.

La relation de Pareto dominance présente la propriété d'incomparabilité entre certains services, ce qui est un inconvénient quand on veut classer les services. Certains travaux ont utilisé l'extension de Pareto dominance ou carrément une définition d'une nouvelle relation de préférence. Les travaux que nous allons présenter, reposent sur une extension ou une nouvelle relation de dominance .

6- Benouaret et. al [Benouaret et al., 2011b] proposent une variante de Skyline qui repose sur une extension de Pareto dominance, ils proposent une fuzzification de Pareto dominance appelée '$\alpha$-dominance'. Cette relation permet la comparaison des

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

7- Dans [Nacer et al., 2015], les auteurs définissent une nouvelle approche de classement des services en se basant sur deux tests : (i) concordance et (ii) discordance entre les différents critères QoS.

- Le test de concordance utilise la règle majoritaire pour décider entre chaque paire de services $s_i$ et $s_j$. Plus précisément, ce test représente la force de la coalition de critères étant en faveur de $si \succ_{CND} sj$.
- Les indices de discordance mesurant le degré avec lequel un critère cj est fortement opposé à la déclaration '$s_i$ est meilleur que $s_j$'. Autrement dit, le test de discordance présente une notion complémentaire pour prendre en compte fortement le conflit Minorité. Les auteurs montrent l'efficacité de leur approche en la comparant avec le classement basé sur la notion de dominance au sens de Pareto.

Nous proposons le tableau comparatif des approches de sélection (voir tableau 5.2) en considérant pour chaque approche les critères de comparaison suivants :

- l'algorithme utilisé : Skyline ou Top-k dominating ou les deux ;
- la stratégie de sélection locale ou globale ou les deux : la phase d'optimisation dans laquelle l'algorithme est utilisé ;
- La relation de comparaison utilisée : la relation de dominance de Pareto strict, une extension de la relation de Pareto, ou une autre relation de dominance.

| Aproches | Relation de dominance | | | Stratégie | | Algorithme utilisé |
|---|---|---|---|---|---|---|
| | Pareto Strict | Extension Pareto | Non Pareto | Locale | Globale | |
| [Alrifai et al., 2010] | + | - | - | + | + | Skyline + clustering |
| [Yu and Bouguettaya, 2013] | + | - | - | + | + | Skyline (algorithme OPA + BUA) |
| [Hadjila, 2014] | + | - | - | + | + | Skyline (algorithme SFS) |
| [Tang et al., 2011] | + | - | - | + | - | 'Top-k Dominating' (Algorithme DSA) |
| [Zhang et al., 2013] | + | - | - | + | + | 'Top-k Dominating' (Algorithme multi index) |
| [Benouaret et al., 2011b] | - | + | - | + | - | Skyline (Algorithme basé sur BBS) |
| [Nacer et al., 2015] | - | - | + | + | - | Algorithme basé sur la relation de concordance |

***Tableau 5.2:*** *Comparaison des approches de sélection selon la stratégie et la relation de dominance utilisées*

## 5.4 Synthèse

L'utilisation de Skyline pour la résolution du problème QoSWSC est très récente. La preuve, l'ensemble des travaux que nous avons pu recenser remonte à 2010. Nous remarquons que la majorité des travaux ne traite pas l'aspect de la composition. En effet, le Skyline est surtout utilisé pour extraire les services Skyline dans une classe. Le but de calculer les services Skyline pour chaque classe est de réduire l'espace de recherche de sélection qui ne cesse d'augmenter. En fait, les algorithmes de Skyline n'utilisent pas de fonction pour classer les objets retournés, ils retournent un ensemble d'objets incomparables entre eux ; d'où la nécessité de définir une fonction objectif comme le cas des travaux [Hadjila, 2014], [Alrifai et al., 2010] pour supporter la composition des services et satisfaire les contraintes globales. A cet effet, [Yu and Bouguettaya, 2013] proposent une approche efficace pour composer les services Skyline de chaque classe et extraire les compositions Skyline à l'utilisateur. Cependant, le nombre de compositions Skyline retournées peut être grand. Dans ce cas, le choix de la composition optimale peut s'avérer contraignant pour l'utilisateur.

Seuls les deux travaux de [Tang et al., 2011] et [Zhang et al., 2013] utilisent le principe de Top-k dominating où le premier l'utilise pour classer les services dans une seule classe et ne traite donc pas le problème de sélection dans une composi-

tion. Par contre, le deuxième travail propose une approche complète pour traiter le problème QoSWSC. Néanmoins, les deux travaux classent les services selon 'le score dominant'. Alors que, [Skoutas et al., 2009] ont démontré que le classement selon 'le score dominant' présente des services de moindre qualité que celui fait par 'le score dominé'.

La majorité des travaux repose sur la dominance de Pareto qui, nous le verrons par la suite (dans la partie contribution), présente parfois des services qui ne sont pas équilibrés dans leurs valeurs QoS. Cette observation est faite par [Benouaret et al., 2011b] et [Nacer et al., 2015]. Pour remédier à ce problème, [Benouaret et al., 2011b] utilisent une extension de la relation de Pareto ; alors que [Nacer et al., 2015] définissent carrément une nouvelle relation de préférence.

## 5.5 Conclusion

Nous avons consacré ce chapitre à la présentation des notions de Skyline et Top-k dominating qui sont des techniques avancées de bases de données pour traiter les requêtes complexes. Les requêtes Skyline et Top-k sont récemment utilisées par la communauté des services web pour la sélection des services web dans une composition. A travers les travaux présentés tout au long de ce chapitre, nous avons constaté que le Skyline est beaucoup plus utilisé pour son pouvoir discriminant, ce qui permet de réduire l'espace de recherche. Il est surtout utilisé dans la phase locale et ne traite pas la composition. Le Top-k dominating profite des avantages des algorithmes de calcul de Skyline et Top-k. Il permet de retourner les services classés sans que nous ayons besoin de définir une fonction de classement. Le Top-k dominating est encore très peu utilisé pour traiter le problème de QoSWSC comparé aux algorithmes de Skyline. Nous nous intéressons dans nos travaux beaucoup plus aux algorithmes de calcul de Skyline. Tout comme dans les travaux de [Alrifai et al., 2010] et [Hadjila, 2014], nous utiliserons le Skyline dans le but de réduire l'espace de recherche dans la phase de sélection de QoSWSC. Par contre, à la différence de ces travaux, nous proposerons un algorithme de calcul de Skyline qui repose sur une extension de la relation de Pareto dominance. Alors que, ces travaux utilisent des algorithmes de calcul de Skyline qui reposent sur la notion de dominance de Pareto au sens strict. Cette proposition sera développée en détail dans le prochain chapitre.

# Partie II :
# Contributions

Chapitre 6

# Approches proposées

## Sommaire

## 6.1 Introduction

Dans ce chapitre nous décrivons nos approches proposées pour résoudre le problème de QoSWSC. Nous commençons dans un premier temps par présenter en détail deux contributions pour la résolution de QoSWSC en utilisant les techniques de Skyline et Top-k. Nous présentons d'abord nos motivations à travers un exemple, ce dernier sera déroulé tout au long des définitions et explications des concepts et algorithmes de ces deux contributions. Nous présentons, ensuite trois autres contributions pour la résolution de QoSWSC en s'appuyant sur les méta-heuristiques. Nous terminons le chapitre par une conclusion qui résume l'essentiel de nos contributions.

## 6.2 La sélection locale des services web basée sur les techniques orientées bases de données

Nous avons présenté un ensemble de travaux dans le chapitre 5 qui utilisent les techniques des bases de données qui reposent sur le concept Skyline et/ou Top-k dominance pour la sélection des services web. En effet, le Skyline a été récemment adopté par la communauté de chercheurs dans la sélection des services web pour résoudre la nature multicritères du problème QoSWSC et pour sélectionner un ensemble de services d'élite (qui ne sont dominés par aucun autre service).

Le Skyline permet de réduire efficacement le nombre de services Web retournés. Néanmoins, en pratique, le calcul du Skyline peut conduire à deux scénarios :

1. soit un nombre assez important de services sont retournés (cas où le nombre de paramètres QoS est grand), ce qui complique davantage la tâche de sélection en augmentant le choix de l'utilisateur ;
2. soit un nombre assez faible de services sont retournés, ce qui restreint la prise de décisions des utilisateurs.

D'autre part, le Skyline élimine certains services qui peuvent être intéressants dans quelques paramètres QoS. Le Skyline ne permet pas aussi d'ordonner les services sélectionnés car les objets appartenant au Skyline sont incomparables les uns aux autres.

D'un autre côté, les travaux sur le QoSWSC reposant sur les requêtes Top-k dominating utilisent tous un classement basé sur le score de dominance de paréto. Deux manières sont présentées pour calculer ce score [Skoutas et al., 2009] selon la

considération de la relation *'domine'* ou la relation *'est dominé par'* (voir chapitre 5). *'Le score dominant'*, qui s'appuie sur la relation *'domine'*, associe à chaque service s, le nombre de services que s domine. Cette façon de faire s'avère moins efficace qu'un classement basé sur *'Le score dominé'* qui s'appuie sur la relation *'est dominé par'* ; en associant à chaque service s, le nombre de services qui dominent s. Notre proposition effectue un classement de services qui s'appuie sur la relation *'est dominé par'*. Cette dernière s'avère donner un meilleur classement des services web que la première [Skoutas et al., 2009].

Finalement, la majorité des travaux que nous avons présentée dans le chapitre 5, repose sur la relation de dominance de Pareto au sens strict qui, en dépit du problème d'incomparabilité, retourne des services avec un mauvais compromis entre les valeurs QoS. Notre proposition repose sur une extension de la relation de dominance. Afin de mieux argumenter et expliquer nos motivations, nous présentons dans la section qui suit un exemple illustratif.

### 6.2.1 Exemple illustratif

Il existe un grand nombre de fournisseurs de services concurrents qui offrent la même fonctionnalité, mais avec différentes qualités de services (QoS) telles que : le temps de réponse, le prix, etc. Par conséquent, les paramètres QoS sont donc des critères essentiels pour choisir parmi les services Web fonctionnellement similaires.

Considérons un service Web pour envoyer des SMS, il y a beaucoup de services Web qui fournissent cette fonctionnalité (par exemple, ClicSend, Inteltech, etc.), mais avec différents QoS. Le tableau 6.1 donne un ensemble de fournisseurs offrant ce service avec leurs paramètres QoS. Ces paramètres de qualités ont été tirés de la base de données ouverte [1] qui contient des données réelles. La base contient plus de 2500 services web et 9 paramètres QoS. Les services Web de l'exemple ont été obtenus en utilisant le mot clé SMS qui représente le tag associé aux services représentant cette fonctionnalité.

Dans notre exemple, chaque service Web contient quatre paramètres QoS, $q_1, q_2, q_3$ et $q_4$, qui représentent respectivement le temps de réponse, débit (ie, Nombre total d'invocations / période de temps) Fiabilité (Ratio : nombre de messages d'erreur / messages au total) et les meilleures pratiques (c.à.d , l'étendue qu'un WSDL suit une spécification).

1. http ://www.uoguelph.ca/ qmahmoud/qws

| **Service** | **Fournisseur** | **opération** | $q1(ms)$ | $q2$(hits/sec) | $q3(\%)$ | $q4(\%)$ |
|---|---|---|---|---|---|---|
| $S_1$ | acrosscommunications.com | SMS | 113.8 | 5.2 | 81 | 84 |
| $S_2$ | sjmillerconsultants.com | SMS | 179.2 | 0.7 | 65 | 69 |
| $S_3$ | webservicex.net | SendSMS | 1308 | 6.3 | 67 | 84 |
| $S_4$ | webservicex.net | SendSMSWorld | 3103 | 5.3 | 79.3 | 91 |
| $S_5$ | smsinter.sina.com.cn | SMSWS | 751 | 6.8 | 64.3 | 87 |
| $S_6$ | sms.mio.it | SendMessages | 291.07 | 5.2 | 53.6 | 84 |
| $S_7$ | www.barnaland.is | SMS | 436.5 | 4.5 | 43.2 | 84 |
| $S_8$ | emsoap.net | emSoapService | 424.54 | 4.3 | 11.9 | 80 |

***Tableau 6.1:*** *Un ensemble de services Web pour envoyer des SMS*

Pour sélectionner un service Web adéquat, les utilisateurs ont besoin d'examiner tous les services manuellement. L'utilisateur peut également trouver des difficultés à équilibrer entre les différentes métriques de qualité. Le skyline contient l'ensemble des services web qui ne sont pas Pareto dominés par les autres. Le Skyline présente une bonne solution pour réduire le nombre de services Web candidats ; et permet de simplifier le processus de sélection car, il surmonte la limitation majeure des approches actuelles qui nécessitent que l'utilisateur attribue des poids aux différents QoS.

Selon notre exemple (Tableau 6.1), nous avons $S_1$ domine $S_6$, $S_7$ et $S_8$. Les services $S_1$,$S_3$, $S_4$ et $S_5$ appartiennent au Skyline et ne sont pas comparables entre eux. Nous pouvons remarquer que le Skyline réduit considérablement le nombre de services candidats, dans notre exemple, nous avons éliminé 50 % des services candidats. Cependant, le nombre de services appartenant au Skyline augmente avec l'augmentation du nombre des paramètres Qos, il devient donc difficile de calculer le Skyline dans un ensemble multidimensionnel très grand [Kossmann et al., 2002], [Papadias et al., 2005]. Ajouté à cela , dans le rapport [Benouaret et al., 2011b] les auteurs montrent que le Skyline peut écarter certains services Web intéressants. En effet, le service $S_6$ est éliminé du Skyline, car il est dominé par $S_4$ alors que $S_4$ est le pire service en terme de temps de réponse, cependant $S_6$ présente un temps de réponse satisfaisant et est plus proche de $S_4$ sur les autres paramètres de qualité de service.

Motivés par ces observations, nous proposons dans cette contribution [Halfaoui et al., 2015] une extension de la relation de Pareto dominance nommée Averaged-Fuzzy-Dominated-score $AFDedS()$ qui associe un score à chaque service dans le but de les classer et sélectionner les Top-k meilleurs d'entre eux, nous proposons aussi une comparaison entre un classement par 'the dominated-score', $AFDedS$ (Le score dominé flou) et un classement par 'the Dominating score' $AFDing$ (le score

dominant flou) proposé par [Benouaret et al., 2011a]. Nous confirmerons à travers des exemples démonstratifs que l'utilisation de $AFDedS$ est plus intéressante que $AFDing$ et délivre ainsi, de meilleurs résultats pour le classement des services. En effet, cette observation est aussi confirmée par les travaux de [Skoutas et al., 2010], alors que la majorité des travaux s'appuie sur 'Dominating score' dans leur classement (voir chapitre 5, section état de l'art).

### 6.2.2 Proposition 1 : La séléction des Top-k Services basée sur la dominance floue

#### 6.2.2.1 Formalisation du problème

Pour uniformiser les valeurs des QoS des services web, nous commençons dans un premier temps par normaliser les différentes valeurs de QoS dans l'intervalle [0,1].

**a) Normalisation des paramètres QoS**

Soit $S$ un ensemble de services qui offrent la même fonctionnalité $S = S_1, .., S_n$. Supposons que nous avons $r$ valeurs quantitatives QoS pour un service $S_i$. Nous utilisons le vecteur $Q(S_i) = \{Nq_1(S_i), .., Nq_r(Si)\}$ pour représenter les attributs QoS d'un service $S_i$ où la fonction $Nq_k(S_i)$ représente le kième attribut de qualité normalisée de $S_i$. On convertit les attributs négatifs (temps, coût) en des attributs positifs en multipliant leurs valeurs par $-1$ de telle sorte que la valeur la plus grande représente la meilleure qualité. On normalise les différentes valeurs des attributs QoS dans l'intervalle [0, 1], comme suit :

$$Nq_k(S_i) = \frac{q_k(S_i) - Qmin(q_k)}{Qmax(q_k) - Qmin(q_k)} \tag{6.1}$$

où :

- $Nq_k(S_i)$ est la valeur normalisée de l'attribut $q_k$ du service web $S_i$.
- $Qmin(q_k)$, (resp.$Qmax(q_k)$ représente le minimum (resp. maximum) des valeurs des attributs $q_k$,$(Qmax(q_k) \neq Qmin(q_k))$.

Le Tableau 6.2 montre les valeurs des attributs QoS des services web de l'exemple (voir Tableau 6.1) après normalisation. Dans la section suivante, nous présentons la fuzzification de la relation de Pareto dominance, et nous montrons son application à travers notre exemple ( voir Tableau 6.1). Nous présentons aussi l'algorithme de classement des services.

| Web service | $Nq1$ | $Nq2$ | $Nq3$ | $Nq4$ |
|---|---|---|---|---|
| $S_1$ | 1 | 0.74 | 1 | 0.68 |
| $S_2$ | 0.98 | 0 | 0.77 | 0 |
| $S_3$ | 0.60 | 0.92 | 0.80 | 0.68 |
| $S_4$ | 0 | 0.75 | 0.98 | 1 |
| $S_5$ | 0.79 | 1 | 0.76 | 0.82 |
| $S_6$ | 0.94 | 0.74 | 0.60 | 0.68 |
| $S_7$ | 0.89 | 0.62 | 0.45 | 0.68 |
| $S_8$ | 0.90 | 0.59 | 0 | 0.50 |

***Tableau 6.2:*** *Les valeurs des QoS des services web après normalisation*

**b) Fuzzification de la relation de Pareto dominance**

Les services web appartenant à la même classe sont différents dans leurs attributs QoS. Comme nous l'avons mentionné avant, les services peuvent être comparés par la relation de Pareto dominance sur leurs vecteurs QoS. Nous définissons, dans ce qui suit la relation de dominance selon Pareto entre deux services comme suit :

**Définition 6.1 - Pareto Dominance entre deux services.**
Soient $S_i$ et $S_j$ deux services web, étant donné un ensemble de d $QoS$ attributs $Q = \{q_1, ..., q_d\}$, On dit que $S_i$ domine $S_j$ dénoté par $S_i \succ S_j$ selon $Q$, si $\forall q_k \in Q$, $q_k(S_i) \geq q_k(S_j)\, and\, \exists q_t \in Q,\ q_t(S_i) > q_t(S_j)$.

Pareto dominance ne différencie pas entre les services Web qui ont un bon compromis et ceux avec un mauvais compromis. Nous notons que l'expression *'bon compromis'* est utilisée pour décrire les services qui sont équilibrés dans les valeurs de leurs attributs QoS c-à-d les services qui ont des valeurs satisfaisantes dans tous leurs attributs. Par opposition, nous employons l'expression *'mauvais compromis'* pour décrire les services qui ne sont pas équilibrés dans les valeurs de leurs attributs c-à-d des services qui présentent des attributs avec des valeurs nulles et d'autres avec des valeurs élevées.

Reprenons notre exemple (Tableau 6.2) et considérons $S_4$ et $S_5$. En effet, ni $S_4$ domine $S_5$, ni $S_5$ domine $S_4$, les deux services sont incomparables et appartiennent au Skyline car, $S_4$ est meilleur que $S_5$ dans $q_3$ et $q_4$, et $S_5$ est meilleur que $S_4$ dans $q_1$ et $q_2$. Toutefois, nous pouvons considérer que $S_5$ est meilleur que $S_4$ car $q_1(S_5) = 0.79$ est nettement meilleur que $q_1(S_4) = 0$. En plus, $q_3(S_5) = 0.76$ et $q_4(S_5) = 0.82$ sont proches des valeurs respectives $q_3(S_4) = 0.98$ et $q_4(S_4) = 1$. Notons aussi que,

contrairement à $S_4$, le service $S_5$ présente un bon compromis entre ses valeurs QoS. Pour toutes ces raisons, il est intéressant de fuzzifier la relation de Pareto dominance.

L'objectif principal derrière la fuzzification de la relation de Pareto dominance est de permettre une comparaison numérique entre des services Web qui étaient incomparables au sens strict de Pareto. Cette comparaison exprime l'extension par laquelle un service est (plus ou moins) dominé par un autre.

Pour calculer le degré de dominance floue entre deux services, il est important d'abord de distinguer entre les mesures retournées par deux relations : le score retourné par la relation 'domine' (The dominanting score) que nous appelons le *'score dominant flou'* et le score retourné par la relation 'est dominé par' (The dominated score) que nous appelons le *'score dominé flou'*.
Si nous prenons deux services $S_i$ et $S_j$, le *'score dominant flou'* exprime le degré avec lequel $S_i$ domine $S_j$, tandis que le *'score dominé flou'* exprime le degré avec lequel $S_i$ est dominé par $S_j$.
Nous notons aussi que les deux mesures ne sont pas symétriques. De plus, comme nous l'avons mentionné dans le chapitre 5, section 5.3.2, la mesure selon la relation 'est dominé par' s'avère plus performante que la mesure selon la relation inverse, pour le classement des vecteurs avec prise en compte des compromis.

Dans notre travail, nous utilisons la relation 'est dominé par' et nous définissons dans ce qui suit la Fuzzification de cette relation.

**Définition 6.2 - Le score dominé flou (Fuzzy-Dominated Score).**
Supposons que $S$ représente un ensemble de services ayant la même fonctionnalité , $S_i$ et $S_j \in S$. $Q = \{q_1, ..., q_d\}$ représente un vecteur de $d$ paramètres de $QoS$.

Nous définissons, dans un premier temps, une fonction de comparaison monotone $\mu_{\varepsilon,\lambda}$ pour exprimer le degré avec lequel $u$ est dominé par $v$, où $u$ représente $q_k(s_i)$ et $v$ représente $q_k(s_j)$ comme suit :

$$\mu_{\varepsilon,\lambda}(u,v) = \begin{cases} 0 & \text{if } (u-v) \geq \varepsilon \\ |u-v-\varepsilon| \,/\, |\lambda+\varepsilon| & \text{if } \lambda+\varepsilon \leq (u-v) < \varepsilon \\ 1 & \text{if } (u-v) < \lambda+\varepsilon \end{cases} \tag{6.2}$$

où $\varepsilon, \lambda \in [-1, 0], \varepsilon + \lambda \geq -1$ et $(\lambda + \varepsilon) \neq 0$

Nous définissons, ensuite, 'the **F**uzzy-**D**ominat**ed** score' $FDed(S_i, S_j)$ pour expri-

mer le degré avec lequel $S_i$ est dominé par $S_j$ comme suit :

$$FDed(s_i, s_j) = \frac{1}{d} \sum_{k=1}^{d} \mu_{\lambda,\varepsilon}(q_k(s_i), q_k(s_j))) \tag{6.3}$$

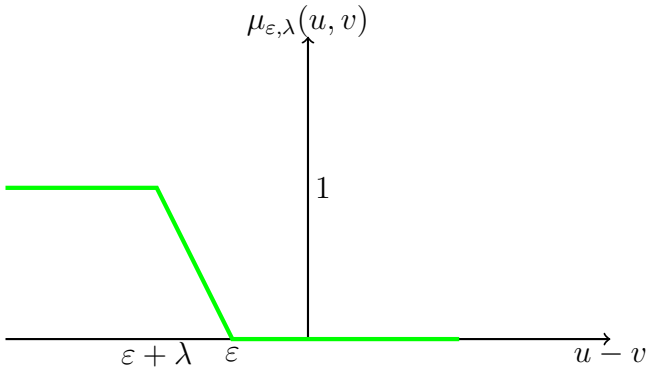



**Figure *6.1:*** *Représentation graphique de la fonction* $\mu_{\varepsilon,\lambda}(u, v)$

Reconsidérons maintenant notre exemple et comparons les Services Web $S_4$ et $S_5$ en utilisant $FDed()$, avec $\varepsilon = -0.1$ et $\lambda = -0.2$. Nous avons $FDed(S_4, S_5) = 0.5$ et $FDed(S_5, S_4) = 0$, ceci signifie que $S_5$ est meilleur que $S_4$ car $S_5$ est dominé avec un degré de 0.4 par $S_5$ alors que $S_5$ ne l'est pas. Le service $S_5$ est légèrement meilleur que $S_4$ au sens de la dominance floue. En effet, ceci est plus expressif que $S_4$ et $S_5$ sont incomparables par Pareto dominance. Dans ce qui suit, nous utilisons $FDed()$ pour classer les services Web.

**Définition 6.3 - Averaged-Fuzzy-Dominated-Score.**
Pour pouvoir classer un service web $S_i$ dans sa classe $S$, nous devons d'abord faire des paires de comparaisons avec les autres services de la même classe et associer un score moyen par :

$$AFDedS(S_i) = \frac{1}{|S| - 1} \sum_{j=1, i \neq j}^{n} FDed(S_i, S_j) \tag{6.4}$$

avec $S$ contient au moins 02 services.

#### 6.2.2.2 Algorithme de Top-k services basé sur la dominance floue

Le classement des services est établi par un ordre croissant des scores $AFDedS(S_i)$ associés aux services. Le service qui a le score minimal est le meilleur service selon ses critères QoS. Le processus global de sélection des k meilleurs services est résumé dans L'algorithme 1 : $FTop\text{-}KS$.

**Algorithme 1** '**FTop-KS**' : la sélection des K Meilleurs services basée sur la dominance floue

**Entrée(s):** **R** : requête de l'utilisateur,

**S** : Une liste de n services web,

**k** : le nombre de services à retourner (Top-k services)

$\alpha, \varepsilon$ : les paramètres d'ajustement de la fonction de comparaison $\mu_{\varepsilon,\lambda}()$

**Sortie(s):** $C_L$ : les k meilleurs services (Top-K)

1: Top-$K \leftarrow \{\}$ ▷ ensemble des k meilleurs services .
2: $Resultat = \emptyset$
3: **pour tout** $si$ dans $S$ **faire**
4: $Score_i = 0$
5: **fin pour**
6: **pour tout** $si$ dans $S$ **faire**
7: **pour tout** $sj$ dans $S$ et $si \neq sj$ **faire**
8: $Score_{ij} = FDed(si, sj)$
9: $Score_i = Score_i + Score_{ij}$
10: **fin pour**
11: $AScore_i = Score_i/(|S| - 1)$
12: $Resultat = Resultat \cup \{(Si, Ascore_i)\}$
13: **fin pour**
14: $TrierAsc(Resultat)$ ▷ trier en ascendant la liste $Resultat$ selon $AScorei$
15: **pour** $i = 1..k$ **faire**
16: $Scourant \rightarrow ExtrairePosition(i, Resultat)$
17: $insérer(Top\text{-}K, Scourant)$
18: **fin pour**
19: **retourner** $Top\text{-}k$

Il est clair que la complexité temporelle de l'algorithme 1 est $O(n+n^2.d+nlog_2n+k)$, où $n^2.d$ correspond à la complexité temporelle de la boucle imbriquée (lignes 6-13) qui permet de calculer le score $AFDedS()$ de chaque service. La complexité $nlog_2n$ correspond à la fonction de trie (ligne 14). Enfin, $K$ correspond à la boucle qui permet de retourner les $K$ premiers services.

Le tableau 6.3 montre le classement des services de l'exemple du Tableau 6.1 après le calcul de score $AFDedS()$ avec $\varepsilon = 0$ and $\lambda = -0.2$

Nous pouvons observer que le meilleur service (Top-1) de la classe SMS est $S_1$. Ce dernier est meilleur que les autres dans $q_1$, $q_2$ et présente de bonnes valeurs dans les autres paramètres de qualités. Nous remarquons que les services qui ont l'une de leurs qualités égales à 0 sont en bas du classement. Considérons maintenant les deux services $S_6$ et $S_4$, selon les résultats de comparaison par Pareto dominance, le service

| service Web | AFDedS() | Nq1 | Nq2 | Nq3 | Nq4 |
|---|---|---|---|---|---|
| s1 | 0,071 | 1 | 0,74 | 1 | 0,68 |
| s5 | 0,107 | 0,79 | 1 | 0,76 | 0.82 |
| s6 | 0,143 | 0,94 | 0,74 | 0,60 | 0,68 |
| s3 | 0,25 | 0,60 | 0,92 | 0,80 | 0,68 |
| s7 | 0,286 | 0,89 | 0,62 | 0,45 | 0,68 |
| s4 | 0,312 | 0 | 0,75 | 0,98 | 1 |
| s8 | 0,393 | 0,90 | 0,59 | 0 | 0,50 |
| s2 | 0,571 | 0,98 | 0 | 0,77 | 0 |

***Tableau 6.3:*** *le classement des services selon AFDedS()*

$S_4$ appartient au Skyline alors que $S_6$ ne l'est pas. Cependant, $S_4$ présente la plus mauvaise qualité temps de réponse ($q_1$) alors que $S_6$ offre un bon compromis entre les valeurs des QoS. En effet, selon le classement (voir Tableau 6.3) le service $S_4$ a été déclassé à la position 7. En revanche, le service $S_6$ qui offre un bon compromis entre ses valeurs $QoS$ a été reclassé à la troisième place.

A travers ces résultats, nous pouvons confirmer que $Fed()$ peut donner de très intéressants résultats qui offrent un bon compromis entres les critères QoS à optimiser.

### 6.2.3 Proposition 2 : la sélection des services Skyline basée sur la dominance floue

La majorité des algorithmes de calcul de Skyline repose sur la notion de dominance selon Pareto, nous définissons dans ce qui suit un algorithme de calcul de Skyline qui repose sur la notion de dominance floue. Nous utilisons plus précisément, la fonction de fuzziffication de Pareto dominance que nous avons définie dans la section 6.3 et nous définissons un algorithme de calcul de Skyline des services web qui se base sur l'algorithme SFS [Chomicki et al., 2005].

#### 6.2.3.1 Skyline vs. $\alpha$-*dominated*-*Skyline* d'une classe de services

Le calcul de Skyline des services d'une classe retourne l'ensemble des services qui ne sont pas Pareto dominés. Nous définissons le Skyline d'une classe de services comme suit :

**Définition 6.4 - Skyline d'une classe de Services .**
Le Skyline d'une classe S de n services $S = S_1, .., S_n$ dénoté par $SkyS$, comprend

l'ensemble des services appartenant à la classe des services S qui ne sont pas dominés par les autres sur l'ensemble de leurs critères de qualités $Q$

$$SkyS(Q,S) = \{S_i \in S |\ \nexists S_j \in S; S_j \succ S_i\}. \tag{6.5}$$

Nous reprenons le concept utilisé dans [Benouaret et al., 2011b] pour définir $\alpha$-*dominance* et nous l'utilisons pour définir la notion de $\alpha$-*dominance* qui repose sur la fonction $FDed()$ comme suit :

**Définition 6.5 - Le concept $\alpha$-*dominated* .**
Prenons deux services $S_i$, $S_j \in S$, un paramètre $\alpha \in [0,1]$ et la fonction $FDed()$ définie dans l'équation 6.3. On dit que le service $S_i$ est $\alpha$-*dominated* par le service $S_j$ noté par $\prec^{\alpha}_{\mu_{\varepsilon,\lambda}}$ si $FDed(S_i,S_j) \geq \alpha$.

Si nous reprenons l'exemple précédent du Tableau 6.2), nous avons, dans le contexte $\varepsilon = -0.1$, $\lambda = -0.2$, $S_2$ est 0,5-*dominated* par $S1$ ($S_2 \prec^{0.5}_{\mu_{-0.2,-O.1}} S_1$), car $FDed(S_1,S_2) = 0.61$. Par contre $S_7$ n'est pas 0,5-*dominated* par $S3$ car $FDed(S_3,S_7) = 0.16$. Par contraste, avec le concept de Skyline qui utilise la notion de Pareto dominance, $\alpha$-*dominated* Skyline s'appuie sur le concept $\alpha$-*dominated*. Nous définissons $\alpha$-*dominated* Skyline d'une classe de service comme suit :

**Définition 6.6 - $\alpha$-*dominated* Skyline d'une classe de Services .**
Le $\alpha$-*dominated* Skyline d'une classe S de n services $S = S_1,..,S_n$ dénoté par $\alpha$-$DSkyS$, comprend l'ensemble des services appartenant à la classe des services S qui ne sont pas $\alpha$-*dominated* par les autres sur l'ensemble de leurs critères de qualités $QoS$

$$\alpha - DSkyS(Q,S) = \left\{S_i \in S |\ \nexists S_j \in S; S_i \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_j\right\}. \tag{6.6}$$

En considérant toujours notre exemple, pour $\varepsilon = -0.1$, $\lambda = -0.2$ et $\alpha = 0.7$, nous avons $0.7\text{-}DSkyS(Q,S) = \{S_1, S_3, S_4, S_5, S_6\}$. Nous remarquons que le service $S_6$ a été rajouté à $0.7\text{-}DSkyS(Q,S)$ alors qu'il n'appartenait pas au Skyline, car il était dominé par $S_4$. Nous notons que le service $S_6$ présente un bon compromis entre ses valeurs QoS.

Nous pouvons conclure que $\alpha$-$DSkyS$ permet d'étendre le Skyline et prendre en considération les services offrant un bon compromis.

#### 6.2.3.2 Algorithme de calcul des services Skyline $\alpha$-$DSkyS$

**Algorithme 2** '**$\alpha$-DSkyS-SFS**' : Algorithme de calcul des Skylines basé sur $FDed()$ et $SFS$.

**Entrée(s):** **S** : Une liste de **n** service web ;
**$\alpha$** : le paramètre de degré de dominance ;
**$\lambda, \varepsilon$** : les paramètres d'ajustement de la fonction de comparaison $\mu_{\varepsilon,\lambda}()$

**Sortie(s):** $\alpha$-$DSkyS$ ;

1: $\alpha$-$DSkyS \leftarrow \emptyset$ ;
2: **pour tout** $s \in S$ **faire**
3: calculer $f(s)$ ; ▷ (f() est une fonction objectif monotone qui calcule la moyenne des R valeurs normalisées de QoS de $s$)
4: **fin pour**
5: $L \leftarrow TrierDesc(S)$ ; ▷ Trier la liste $S$ du plus grand au plus petit score selon f() et l'affecter à L ;
6: **tant que** $L \neq \emptyset$ **faire**
7: extraire le service s à la position L[i] et le supprimer de L ;
8: **si** $\exists s' \in \alpha$-$DSkyS$ tel que s est $\alpha$-*dominated* par s' **alors**
9: négliger s et passer au service suivant ; (i.e. i=i+1).
10: **sinon**
11: **si** $\exists s' \in \alpha$-$DSkyS$ tel que s' est $\alpha$-*dominated* par s **alors**
12: remplacer s' par s et passer au suivant ;
13: **sinon**
14: $\alpha$-$DSky = \alpha$-$DSky \cup s$ ▷ ajouter s à $\alpha$-$DSkyS$ et passer au suivant ;(i=i+1).
15: **fin si**
16: **fin si**
17: **fin tant que**
18: **retourner** $\alpha$-$SkyS$ ;

L'algorithme 2 repose sur les principes de l'algorithme SFS pour le calcul de Skyline. En utilisant SFS, nous pouvons extraire les services Skyline de manière progressive ; tout en diminuant le nombre de comparaisons de dominance floue $FDed()$ entre les services grâce à l'utilisation d'une fonction de score monotone utilisée pour l'ordonnancement des services en entrée.

Il est évident que la complexité temporelle de l'algorithme 2 : $\alpha$-$DSkyS$-$SFS$ est $O(n.d + n\log n + n^2.d + n)$ où $n$ correspond aux instructions de la boucle (ligne 2-4), $n\log n$ correspond à la fonction de trie (ligne 5) et $n^2.d$ correspond aux instructions des lignes (6-17)

## 6.3 La sélection des services web basée sur les méta-heuristiques

### 6.3.1 Motivations

Ces dernières années, de nouveaux algorithmes d'optimisation sont apparus dans le domaine de l'optimisation comme Self-Organising Migrating Algorithm (SOMA) [Zelinka, 2004], Fire Fly (FF) [Yang, 2009], Cuckoo Search (CS) [Yang and Deb, 2009] 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. Plus précisément, nous avons vu dans le chapitre 4, qu'ils ont été aussi appliqués avec efficacité dans le domaine de QoSWC. Cependant, pour autant que nous sachions, SOMA n'a pas été très exploité dans le domaine du problème de sélection de service Web, alors que , certains récents travaux démontrent sa performance comme celui de [Davendra and Bialic-Davendra, 2013] ou encore [Singh and Agrawal, 2015]. Nous n'avons recensé qu'un seul papier [Krithiga, 2012] appliquant SOMA au domaine QoSWSC, cependant, l'auteur utilise la version de SOMA à variables continues. Or, vu que nous considérons le problème QoSWSC comme un problème combinatoire, la version discrète de SOMA est plus adaptée à notre cas. Bien que [Krithiga, 2012] ait utilisé SOMA, L'auteur n'a cependant mené aucune expérimentation pour ajuster les paramètres de SOMA dans le cas de QoSWSC.

Motivés par ces observations, nous proposons [Halfaoui et al., 2017] dans cette partie deux premières contributions :

- Une adaptation de l'algorithme SOMA pour la résolution du problème QoSWC (cf. section 6.3.3).
- Une amélioration de l'algorithme SOMA en utilisant la notion de dominance floue (définie et formulée dans la section 6.2.2.1) dans la phase de la sélection locale (cf. section 6.3.4).

Parmi les observations faites dans la synthèse du chapitre 4, celle que les méta-heuristiques à base de solution unique sont peu utilisées par rapport à celles de population de solutions. Ceci est justifié par le fait que les algorithmes à base de population de solutions permettent, contrairement aux autres, d'explorer un très vaste espace de recherche et d'utiliser la population comme facteur de diversité, de plus elles sont très adaptées et très largement utilisées pour l'optimisation multi-objectif. Partant de ce constat, nous avons choisi d'utiliser un algorithme à base de

solution unique et de contourner la lacune de l'exploration de l'espace de recherche par sa réduction en utilisant le calcul de Skyline. Dans ce sens, nous proposons [Halfaoui et al., 2014] dans la section 6.3.5, la dernière contribution qui consiste en la sélection des services web basée sur la recherche Tabou et le calcul de Skyline.

### 6.3.2 Formalisation du problème

Dans cette section, nous définissons les préliminaires du problème de QoSWSC.

#### 6.3.2.1 Composition abstraite vs. concrète

- Une composition abstraite de services est la définition abstraite d'une représentation de la composition (le workflow). Elle spécifie les classes de services exigées (par exemple le service de traduction), sans se référer à n'importe quel service Web concret (e.g. google traduction or reverso).

  **Définition 6.7 - Composition abstraite.**
  Étant donné un ensemble de classes de services $m$, $S = \{S_1, .., S_m\}$ représente une composition abstraite de m services, où la classe $S_i$ comprend les services qui ont la même fonctionnalité mais diffèrent dans les contraintes $QoS$.

  Dans cette thèse, nous assumons que le flux de contrôle de notre workfow est séquentiel. Les autres flux de contrôles peuvent facilement être convertis en un workflow séquentiel.

- Une composition concrète de service est définie comme l'instanciation de la composition abstraite.

  **Définition 6.8 - Composition concrète.**
  Étant donné un ensemble de services concret $m$, $C = \{S_{1j_1}, .., S_{mj_l}\}$ représente une composition concrète avec $(i = 1, .., m; j = 1, .., l)$. Où $S_{ij}$ est le $j$ ème service concret de la classe $i$ parmi $l$ services découverts.

#### 6.3.2.2 Les propriétés QoS d'une composition

Les propriétés quantitatives non fonctionnelles servent à décrire les propriétés de qualité d'un service Web. Ces propriétés peuvent inclure le temps d'attente, le coût, etc. Comme nous l'avons souligné dans la section 6.2.2.1, nous utilisons le vecteur $Q(S_{ij}) = (Nq_1(S_{ij}), .., Nq_d(S_{ij}))$ pour représenter les propriétés $QoS$ d'un service $S_{ij}$

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

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

où la fonction $Nq_k(S_{ij})$ représente la valeur normalisée de la qualité k du service $S_{ij}$. La normalisation a été définie dans la section 6.2.2.1 avec l'équation 6.1.

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

Le vecteur $QoS$ d'un service composite $C$ est défini par $Q(C) = (Q_1(C), .., Q_d(C))$, où $Q_k(c)$ représente la valeur du k ième attribut $QoS$ de $C$ $(k = 1, .., d)$. Il peut être calculé en appliquant de façon récursive une fonction d'agrégation des QoS selon les blocs de composantes qui définissent la structure de la composition. Le Tableau 6.4 montre les fonctions d'agrégations appliquées à 4 propriétés $QoS$.

#### 6.3.2.3 Les contraintes de qualité globales

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

**Définition 6.9 - Composition faisable.**
soit $gc = (gc_1, .., gc_n)$ un vecteur de contraintes globales $QoS$, $1 < n < d$ (d est la taille du vecteur des qualités $QoS$) et soit $Q(C) = (Q_1, ..Q_d)$ le vecteur des valeurs de qualités agrégées $QoS$ de la composition $C$. On dit qu'une composition $C$ est faisable si toutes les contraintes globales sont satisfaites, ceci signifie que : $Q_k \geq gc_k, \forall k \in \{1..n\}$.

#### 6.3.2.4 La fonction objectif

Différentes compositions peuvent être générées à partir du workflow abstrait. Pour évaluer ces différentes compositions et ne retenir qu'une, nous avons besoin d'une fonction objectif (fonction d'utilité) $U(c)$ qui associe une valeur à chaque composition. Cette fonction est sous forme d'une somme pondérée des différents $QoS$ [Yoon and Hwang, 1995] additionnée a une fonction de pénalité. La fonction de pénalité $p(C)$ exprime le degré avec lequel la composition $C$ viole les contraintes globales.

$$U(C) = \sum_{k=1}^{d} w_k . \frac{QMax(k) - Q_k(C)}{QMax(k) - QMin(k)} + p(C) \tag{6.7}$$

avec $w_k \in \mathbb{R}_0^+$, $QMax(k) ne QMin(k)$ et $\sum_{k=1}^{d} w_k = 1$ .
$w_k$ représente le poids associé à $Q_k(C)$ (l'importance de $Q_k(C)$). Nous avons considéré des poids équitables pour tous les critères.
Notons que la complexité de la fonction $U()$ est $O(d.m)$.

$$QMax(k) = \sum_{i=1}^{m} qmax(i,k) \tag{6.8}$$

avec $qmax(i,k) = max(q_k(S_{ij})), \forall S_{ij} \in S_i$

$$QMin(k) = \sum_{i=1}^{m} qmin(i,k) \tag{6.9}$$

avec $qmin(i,k) = min(q_k(S_{ij})), \forall S_{ij} \in S_i$.
L'objectif de $P(c)$, est de décroitre le score de $U(c)$ lorsqu'une ou plusieurs contraintes globales sont violées. Plusieurs fonctions de pénalité sont proposées dans la littérature, [Yeniay, 2005], [Adeli and Cheng, 1994], (des fonctions statiques, dynamiques, adaptatives..). Pour des raisons de simplicité nous avons opté pour une fonction statique [Hadjila, 2014]. La fonction $P(C)$ est définie comme suit :

$$p(C) = -\sum_{k=1}^{d} D_k(C) \tag{6.10}$$

où $D_k = \begin{cases} 0 & \text{if } Q_k(C) \leq gc_k \\ |Q_k(C) - gc_k| & \text{otherwise} \end{cases}$

Une composition $C$ est dite optimale si :

- elle est faisable selon la définition 6.9.

- elle maximise la fonction $U$

Notre but, est de chercher la solution quasi-optimale $C$ qui maximise au mieux la fonction objectif $U$ (nommée aussi fitness). Rappelons que, la recherche de la solution optimale exige l'énumération de toutes les combinaisons possibles des services candidats. Comme mentionné dans la problématique P2, ce problème est NP-difficile. Si la requête d'un utilisateur concerne $m$ classes de services ($m$ tâche) et chaque classe contient $l$ services candidats, il y a $l^m$ combinaisons possibles à examiner. Dans la section suivante, nous proposons une variante de l'algorithme d'optimisation SOMA. Comme nous l'avons mentionné avant, beaucoup de travaux [Singh and Agrawal, 2015], [Davendra and Bialic-Davendra, 2013] ont prêté attention à cet algorithme mais il n'a pas été très examiné dans le contexte de composition de services Web.

### 6.3.3 Proposition 1 : Adaptation de la métaheuristique SOMA au problème QoSWSC

Self Organizing Migrating Algorithm [Zelinka, 2004] est une technique de recherche stochastique basée sur une population d'individus. Elle est basée sur l'auto organisation du comportement des groupes d'individus dans un environnement social (par exemple un groupe d'animaux cherchant la nourriture). A chaque génération, l'individu avec la meilleure valeur de la fonction objectif appelé le **'Leader'** est trouvé. Plutôt que de rivaliser les uns avec les autres, les autres individus passent dans la direction du leader en effectuant des sauts **'Jumps'**. Cet algorithme fonctionne en effectuant un nombre de migrations **'Migration'**. Pour chaque migration, chaque individu, (appelé l'individu actif) voyage avec une certaine distance vers le leader par un nombre de pas **'Step'** de longueur définie **'Pathlength'**. Ce chemin est perturbé aléatoirement par le paramètre de perturbation **'PRT'**. Un vecteur de perturbation **'PRTVector'** est créé avant qu'un individu ne se dirige vers le leader. Il est défini dans l'intervalle [0,1]. L'équation d'un mouvement d'un individu est donnée comme suit :

$$x_{i,j}^{MLnew} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML}) tPRTVector_j \tag{6.11}$$

où :

$ML$ : est la migration actuelle.

$x_{i,j}^{MLnew}$ : est la position du nouvel individu.

$x_{i,j,start}$ : est la position de l'individu actif (celui qui va effectuer le saut vers le leader).

$x_{L,j}^{ML}$ : est la position du leader.

$t$ : est la taille du saut. ce paramètre définit la granularité avec laquelle l'espace de recherche est échantillonné. $t \in < 0, pathLenght >$

$PRTVector_j$ : est la matrice de perturbation.

#### 6.3.3.1 L'algorithme QoS-SOMA

SOMA est généralement utilisé pour résoudre des problèmes à variables continues et non des problèmes discrets. Plusieurs travaux adaptent l'approche pour traiter des problèmes discrets comme la résolution du problème de voyageur de commerce [Čičková et al., 2008] et le problème de scheduling flow shops [Davendra and Bialic-Davendra, 2013]. Dans ce qui suit, nous définissons la version discrète de SOMA adaptée à la composition de service web à base de QoS. Nous présentons aussi l'algorithme QoS-SOMA (QoS-aware web service composition based on SOMA algorithm), nous expliquons aussi son déroulement à travers un exemple illustratif (voir section 6.3.3.2) .

Prenons $t$ comme étant la variable d'itération pour les migrations ($t = 1, 2, ..., MIG$), $k$ comme l'index des compositions dans la population ($k = 1, .., n$), $n$ comme le nombre de compositions ou la taille de la population, l'algorithme 3 peut être décrit par les étapes suivantes :

**a) Initialisation**

Pour la population initiale, on génère aléatoirement $n$ compositions. Pour chaque composition, on choisit aléatoirement un service de chaque classe. Le service est représenté par sa position dans la liste des services candidats. $C_k^t$ représente la $k^{ème}$ composition, t représente le compteur de migration, n représente la taille de la population. $P = C_1^t, C_2^t, .., C_n^t$,
où chaque $C_k^t = p_{k,1}^t, p_{k,2}^t, ..p_{k,i}^t$ et $(i = 1..m)$ représente le nombre de tâches.
$p_{k,i}$ représente la position $j$ du service appartenant à la composition $k$ selon la tâche $i$. Par exemple, si on a $p_{3,2} = 4$, ceci veut dire que nous utilisons le service $S_{24}$ pour construire la composition $C_3$. Ce service est à la position 4 dans la liste des services candidats de la tâche 2.

**b) Création du vecteur de perturbation**

Cette étape correspond à la fonction CreatePRTVector() (ligne 15). Dans ce qui suit, nous allons détailler comment les éléments de ce vecteur sont créés.

**Algorithme 3** '**QoS-SOMA**' : la sélection des services web composés basée sur SOMA

**Entrée(s):** un workflow qui contient **m** tâches et **l** services,
**R** : requête de l'utilisateur,
**n** : Taille de la population initiale,
$MIG$ : nombre de migrations,
$PRT$ : valeur de perturbation,
$NbJump$ : le nombre de sauts.

**Sortie(s):** $C_L$ : la composition optimale

1: $P \leftarrow \{\}$ ▷ ensemble de n compositions initiales.
2: $G \leftarrow \{\}$ ▷ matrice des positions de sauts.
3: $T \leftarrow \{\}$ ▷ ensemble de compositions d'essais
4: **pour** $k = 1..n$ **faire**
5: **pour** $i = 1, ..m$ **faire**
6: $p_{k,i} = rndint[1, l]$
7: insert $p_{k,i} \rightarrow C_k(p_{k,i})$
8: insert $C_k \rightarrow P(p_{k,i})$
9: $Score_k = U(C_k)$
10: **fin pour**
11: **fin pour**
12: $C_L = GetBestC(P)$ ▷ $C_k$ avec le meilleur score
13: **tant que** $t < MIG$ **faire**
14: **pour tout** $C_k$ in $P$ **faire**
15: $A = CreatePRTVector(C_k)$
16: $G = CreateJumpSequence(C_k, A, NbJump)$
17: $T = CreateTrialSol(G)$
18: $C_{best} = GetBestC(T)$
19: $C_k \leftarrow C_{best}$
20: $A = \emptyset, G = \emptyset, T = \emptyset$ ▷ initialiser les vecteurs pour la prochaine composition
21: **fin pour**
22: $t = t + 1$ ▷ la prochaine migration
23: $C_L = UpdateLeader(P)$
24: **fin tant que**
25: **retourner** $C_L$

Avant qu'une composition ne commence à effectuer une séquence de sauts vers le leader, une valeur aléatoire est générée et associée à chaque service de la composition en question. Cette valeur est comparée au paramètre PRT pour créer une perturbation. PRT est définie dans le rang [0,1]. Pour chaque service se trouvant à la position $p^t_{k,i}$ dans la composition $C^t_k$ , Les éléments de perturbation $a_i$ du vecteur A sont crées par :

$$a_i = \begin{cases} 1 & \text{if } rand() < prt \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

avec $i = 1..m$.

La valeur binaire qui est générée contrôle les sauts d'un service dans une composition $C^t_k$. Elle donne la permission à ce service d'effectuer un saut. Notons que nous désignons par un service actif, le service qui s'apprête à effectuer le saut vers le leader.

Nous considérons un saut dans notre cas comme étant le remplacement du service actif par le service se trouvant à la position du saut. Ces positions sont données par la matrice des sauts qui sera définie dans la section suivante.

Si l'élément de perturbation $a_i$ est égal à 1, alors le service actif peut sauter vers la leader, sinon le service est gelé et le saut n'est pas autorisé.

**c) La création de la séquence des sauts et la construction des solutions d'essais**

La création de la matrice G qui contient toutes les séquences de sauts possibles pour les services d'une composition (fonction CreateJumpSequence() à ligne 16), consiste à créer un nombre de positions de sauts pour chaque service qui est autorisé à effectuer des saut. Dans QoS-SOMA, nous avons défini le paramètre *NbJump* (nombre de sauts). Dans la version initiale de SOMA, *PathLenght* et *Step* doivent être initialisés. Dans QoS-SOMA, ces deux paramètres ne sont pas initialisés, ils ne peuvent pas être connus à l'avance car ils sont calculés automatiquement par l'algorithme en utilisant le paramètre NbJump et les positions des services adjacents se trouvant entre le service actif et le leader.

Prenons deux compositions dans la population, une comme étant la composition active ($C^t_k$) et l'autre le leader $C^t_L$, le nombre de sauts possibles d'un service dans la composition ($C^t_k$) est :

$$J^t_i = |p^t_{k,i} - p^t_{L,i}| \tag{6.13}$$

La taille du pas $s_i^t$ est calculée par :

$$s_i^t = \begin{cases} \lfloor J_i^t / NbJump \rfloor & \text{if } a_i = 1 \\ 0 & \text{if } a_i = 0 \end{cases} \tag{6.14}$$

où la notation $\lfloor \ \rfloor$ représente la partie entière par défaut (sans arrondir à la plus proche valeur). Elle est généralement représentée par la fonction mathématique $floor()$.

Les éléments de la matrice des sauts G, qui contient toutes les positions des sauts possibles, peuvent être calculés par :

$$G_{i,r}^t = \begin{cases} p_{k,i}^t + s'_i.r & \text{if } p_{L,i}^t < p_{k,i}^t \\ p_{k,i}^t - s'_i.r & \text{if } p_{L,i}^t > p_{k,i}^t \end{cases} \tag{6.15}$$

pour $i = 1, ..m$, $r = 1..NbJump$

Après, nous construisons les compositions d'essais $T_r$ à partir des positions des sauts (Ligne 17). Chaque élément de la composition d'essai $T_{r,i}^t$ est construit par :

$$T_{r,i}^t = G_{i,r}^t \tag{6.16}$$

avec $i = 1, ..m$, $r = 1..NbJump$

Si $NbJump = 1$, ceci signifie que les services de la composition active vont sauter directement au leader ( en d'autres termes, nous aurons tout simplement un seul saut à effectuer. Les services qui sont autorisés à se déplacer, seront remplacés par les services du leader).

**d) Mise à jour de la population**

Chaque composition d'essai $T_r^t$ est évaluée par la fonction objectif $U$ (ligne 18). La composition d'essai avec le meilleur score $T_{Best}^t$ est choisie pour remplacer la composition active ($C_k^t$) (ligne 19).

**e) Itération**

Nous traitons la prochaine composition ($C_{k+1}^t$) de la population. La composition ($C_{k+1}^t$) commence son propre déplacement vers le leader $C_L^t$. Nous notons que le leader ne change pas durant l'évaluation d'une seule migration.

**f) Migration**

Quand toutes les compositions exécutent leurs déplacements vers l'actuel leader, le leader est mis à jour (ligne 23) et le compteur de migration est incrémenté $t = t + 1$.

**• Complexité temporelle de QoS-SOMA**

Notons que la complexité temporelle de l'algorithme 3 est $O((n.m).(m + d.m) + n.d.m + MIG(n[m + 2NbJump.m + NbJump.d.m + m] + n.d.m + m)$. Le détail de la complexité de l'algorithme par rapport à ses différentes instructions est donné dans le tableau 6.5.

| **N° de la ligne de l'instruction** | **Complexité de l'instruction** | **Explication** |
|---|---|---|
| Lignes 6-9 | $O(m + d.m)$ | boucle (la sélection d'un service pour **m** tâches avec **d** QoS) |
| Ligne 12 | $O(n.d.m)$ | sélection de la meilleure composition parmi n candidats |
| ligne 15 | $O(m)$ | boucle (création du vecteur de perturbation pour chaque service (taille de la composition=m)) |
| lignes 16 ou 17 | $O(NbJump.m)$ | boucle (création de la matrice des sauts et des compositions d'essais) |
| ligne 18 | $O(NbJump.d.m)$ | sélectionner la meilleure trial solution parmi $NbJump$ candidats |

***Tableau 6.5:*** *Complexité temporelle de quelques instructions de l'algorithme 3*

#### 6.3.3.2 Exemple illustratif

Un bref exemple qui illustre le déroulement de l'algorithme QoS-SOMA est présenté dans la Figure 6.2. Il montre le déplacement d'une composition vers le leader. Considérons une base qui contient 4 tâches, $m = 4$ et 8 services candidats par tâche, $l = 8$. Nous générons dans un premier temps une population aléatoire qui contient 4 compositions ($n = 4$), $P = (C_1, C_2, C_3, C_4)$, $C_3$. $C_3$ est la composition qui détient le meilleur score retourné par la fonction objectif, donc le leader est la composition 3, $C_L = C_3$ (voir Figure 6.2).

Supposons que NbJump = 2, toutes les compositions vont se déplacer vers le leader. Nous prenons $C_1$ comme étant la composition active. En premier, nous générons

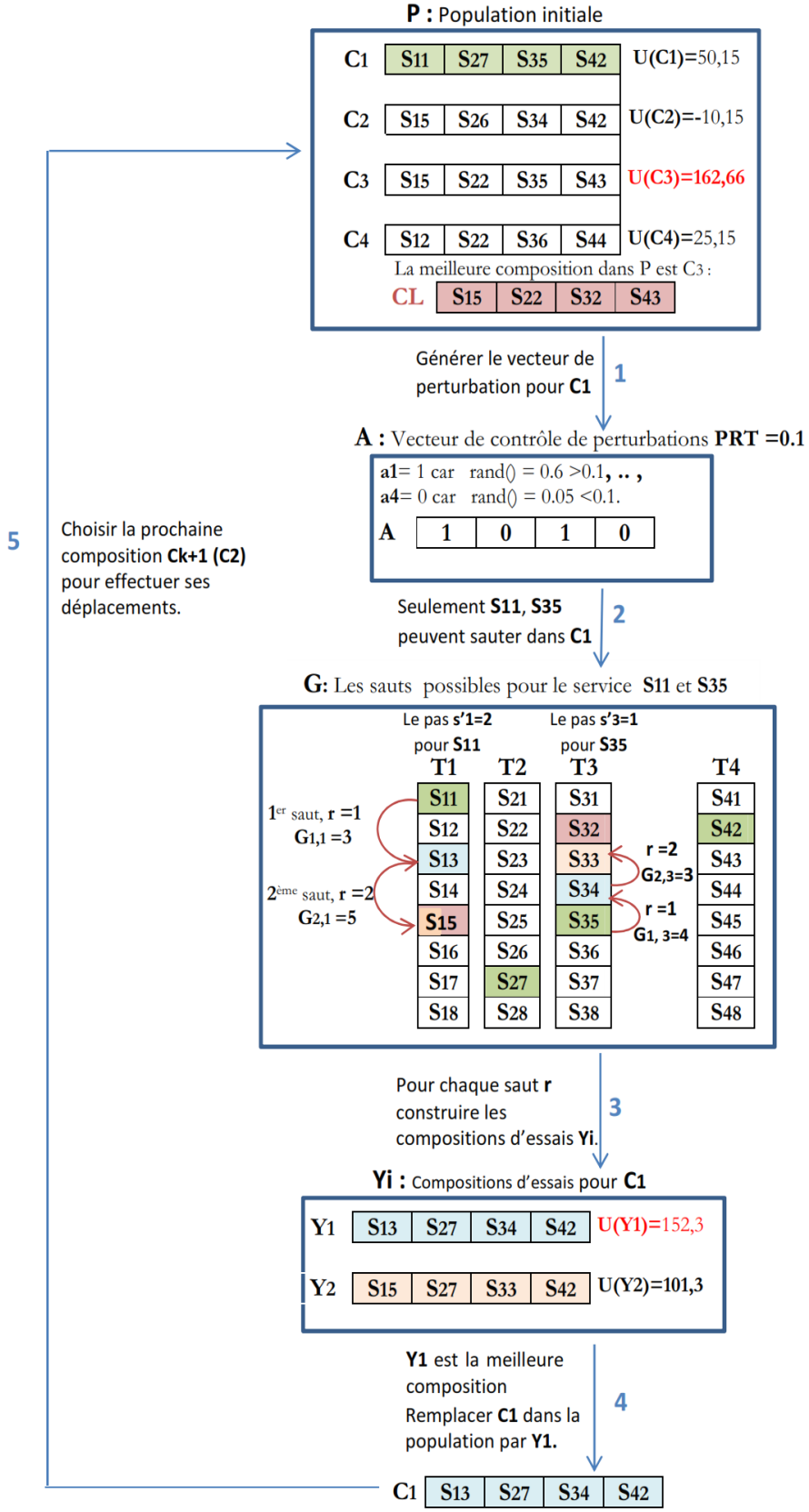



**Figure *6.2:*** *Un exemple illustratif : le déplacement de $C_1$ vers le leader $C_L$ dans QoS-SOMA*

le vecteur de perturbation $A = \{1, 0, 1, 0\}$. Dès lors, la première composition effectue son déplacement vers le leader.

Selon les éléments $a_1 = 1$ et $a_3 = 1$ dans le vecteur de perturbation $A$, seulement les services $S_{11}$, $(p_{1,1} = 1)$ et $S_{35}$, $(p_{1,3} = 5)$ sont autorisés à se déplacer respectivement vers les services correspondants du leader $S_{15}$, $(p_{L,1} = 5)$ et $S_{32}$, $(p_{L,3} = 2)$.
En utilisant les équations 6.13 et 6.14, les pas des sauts pour chaque service sont : pour $S_{11}$ le pas $s'_1 = \lfloor 4/2 \rfloor = 2$ et pour $S_{35}$ le pas est $s'_3 = \lfloor 3/2 \rfloor = 1$. $s'_2 = 0$ et $s'_4 = 0$ car les services se trouvant aux positions 2 et 4 ne sont pas autorisés à se déplacer.

En utilisant l'équation 6.15 , la matrice des sauts $G$ est générée dans le Tableau 6.6. En utilisant $G$ et 6.16, nous construisons pour chaque saut $(r = 1, r = 2)$, une composition d'essai : $T_1 = \{3, 7, 4, 2\}$, $T_2 = \{5, 7, 3, 2\}$. La composition d'essai avec la meilleure valeur de la fonction objectif remplace la composition $C_1$.

| $\mathbf{p_{1,i}}$ | $\mathbf{p_{L,i}}$ | $\mathbf{a_i}$ | $\mathbf{s'}_i$ | $\mathbf{G_i}$ |
|---|---|---|---|---|
| 1 | 5 | 1 | 2 | $\{3, 5\}$ |
| 7 | 2 | 0 | 0 | $\{7, 7\}$ |
| 5 | 2 | 1 | 1 | $\{4, 3\}$ |
| 2 | 3 | 0 | 0 | $\{2, 2\}$ |

***Tableau 6.6:*** *La matrice $G$ entre $C_1$ et $C_L$.*

Itérativement, chaque composition $(C_2, C_3, C_4)$ se déplace vers le leader. Quand toutes les compositions tour à tour effectuent leurs séries de sauts, nous mettons à jour le leader et nous passons à la prochaine migration.

### 6.3.4 Proposition 2 : Optimisation locale de l'algorithme QoS-SOMA par la dominance floue

Dans cette proposition, nous avons effectué une amélioration de l'algorithme de QoS-SOMA. Cette amélioration est faite au niveau locale du processus de sélection des services composés. L'optimisation locale est réalisée en utilisant la dominance floue (voir proposition 1 section 6.3). La version améliorée de QoS-SOMA nommée FQoS-SOMA rajoute un test entre les compositions dans la phase locale. L'algorithme FQoS-SOMA contient exactement les mêmes étapes que QoS-SOMA, à part la fonction CreateJumpSequence() de l'algorithme 3 (ligne 17) qui sera modifiée.

La modification de la fonction CreateJumpSequence() est faite en ajoutant une comparaison par la dominance floue entre les vecteurs des propriétés QoS des services de la composition active et les services se trouvant aux positions des sauts à effectuer.

Si le saut est autorisé, nous remplaçons le service de la composition active, seulement si ce dernier est dominé au sens de 'fuzzy dominated'$FDed()$ (en utilisant l'équation 6.3) par le service se trouvant à la position du saut. Ceci signifie que ce déplacement assure une amélioration du score de la fonction objectif de la composition.

FQoS-SOMA remplace la création de la matrice des sauts $G$ (équation 6.15) par les deux équations suivantes :

$$j_{i,r}^t = \begin{cases} p_{k,i}^t + s'_i.r & \text{if } p_{L,i}^t < p_{k,i}^t \\ p_{k,i}^t - s'_i.r & \text{if } p_{L,i}^t > p_{k,i}^t \end{cases} \tag{6.17}$$

$$G_{i,r}^t = \begin{cases} j_{i,r}^t & \text{if FDed}(S_{ij_{i,r}^t}, S_{ip_{k,i}^t}) < \text{FDed}(S_{ip_{k,i}^t}, S_{ij_{i,r}^t}) \\ p_{k,i}^t & \text{otherwise} \end{cases} \tag{6.18}$$

avec $i = 1, ..m, r = 1..NbJump$.

Reprenons notre exemple illustratif (voir Figure 6.2, et Tableau 6.6) et générons à nouveau la matrice $G$ en appliquant FQoS-SOMA. Les positions des prochains sauts sont définies en utilisant les équations 6.17 et 6.18. La matrice G est présentée dans le Tableau 6.7. Les compositions d'essais sont : $T_1 = \{1, 7, 4, 2\}$, $T_2 = \{5, 7, 5, 2\}$.

### 6.3.5 Proposition 3 : La sélection des services web basée sur la recherche Tabou et calcul de Skyline

L'approche proposée [Halfaoui et al., 2015] dans cette contribution est une approche hybride qui contient deux phases. Une phase de sélection locale et une phase d'optimisation globale.

#### 6.3.5.1 La phase 1 : la sélection locale

Nous calculons le Skyline des services et nous réduisons l'espace de recherche en utilisant l'algorithme $\alpha$-$DSkyS$ défini dans la contribution 6.2.3 (voir l'algorithme 2. Pour chaque classe $S_i$, nous calculons son Skyline $\alpha$-$DSkyS_i$ , $i = 1..m$. Nous pouvons aussi réduire l'espace de recherche par le calcul des Top-k services de chaque classe

| $\mathbf{p_{1,i}}$ | **j** | **FDed**() | $\mathbf{G_i}$ |
|---|---|---|---|
| 1 | $\{3,5\}$ | $FDed(S_{11},S_{13})=0.2$<br>$FDed(S_{13},S_{11})=0.5$<br>($S_{13}$ est F-dominé par $S_{11}$)<br>$FDed(S_{11},S_{15})=0.4$<br>$FDed(S_{15},S_{11})=0.1$<br>($S_{11}$ est F-dominé par $S_{15}$) | $\{1,5\}$ |
| 7 | $\{7,7\}$ | le saut n'est pas autorisé | $\{7,7\}$ |
| 5 | $\{4,3\}$ | $FDed(S_{35},S_{34})=0.5$<br>$FDed(S_{34},S_{35})=0.2$<br>($S_{35}$ est F-dominé par $S_{34}$)<br>$FDed(S_{35},S_{33})=0.1$<br>$FDed(S_{33},S_{35})=0.6$<br>($S_{33}$ est F-dominé par $S_{35}$) | $\{4,5\}$ |
| 2 | $\{2,2\}$ | le saut n'est pas autorisé | $\{2,2\}$ |

***Tableau 6.7:*** *La matrice des sauts G entre $C_1$ et $C_L$ dans FQoS-SOMA.*

en utilisant l'algorithme 1 .

#### 6.3.5.2 La Phase 2 : l'optimisation globale par la recherche Tabou

A partir des $m$ Skylines $\alpha$-$DSkyS_i$, nous cherchons la composition proche de l'optimale qui satisfait les contraintes globales en utilisant la recherche Tabou.

1. **L'optimisation par la recherche Tabou**

   Nous présentons dans cette partie les principes de notre approche d'optimisation qui utilise les concepts de base de l'algorithme tabou qui sont : la solution initiale, voisinage d'une solution, la liste taboue et la stratégie de diversification.

   (a) **La solution initiale**

   La recherche Tabou [Glover, 1989] est une méta-heuristique à base d'une solution unique. De ce fait, la solution initiale $C_0$ est une composition aléatoire. Pour cela, nous choisissons un service de manière aléatoire de chaque classe.

   Dans le but d'avoir des compositions différentes les uns des autre lors de la génération des compositions voisines d'une solution, il faudra générer une solution initiale efficace.

**Définition 6.10 - Solution initiale efficace.**

Une solution initiale est considérée efficace, si les services de ses différentes classes n'ont pas le même rang (position du service dans la classe).

Soit $C_0 = \{S_{1j_1}, .., S_{mjl}\}$ une solution initiale, La composition $C_0$ est une solution initiale efficace si : $\forall S_{ij_k} \in C_0, \forall S_{i'j_p} \in C_0, j_k \neq j_p$ avec $i \neq i'$.

(b) **Le voisinage d'une composition**

L'espace de recherche de notre algorithme est l'ensemble de compositions obtenues par des mouvements appliqués à la solution initiale. Un mouvement peut être considéré comme une permutation entre les rangs de deux services appartenant à deux classes différentes. Soit la composition $C \in X$. La solution $C' \in X$ est une solution voisine de $C$ si elle est obtenue en appliquant un mouvement à $C$.

Par exemple, considérons les deux compositions $C = \{S_{11}, S_{23}, S_{34}\}$ et $C' = \{S_{13}, S_{21}, S_{34}\}$. La solution $C'$ est la solution voisine de $C$ obtenue par le remplacement du service 3 par le service 1 dans la classe 1 et le changement du service 1 par le service 3 dans la classe 2. Nous avons appliqué le mouvement suivant : permuter les rang 1 et 3 entre les deux classes 1 et 2.

(c) **La liste taboue**

La liste taboue dans notre approche est la structure qui contient les compositions utilisées dans les itérations précédentes. Nous utilisons la fonction objectif $U(c)$ (c.f section 6.3.2.4) pour évaluer chaque solution voisine.

Pour éviter la stagnation dans un optimum local, nous utilisons comme stratégie de diversification, le redémarrage de l'algorithme (réinitialisation des paramètres avec une nouvelle exécution) après un certain nombre d'itérations quand la solution locale optimale stagne.

2. **L'algorithme QoS-SkyTabou**

Nous présentons, dans ce qui suit, l'algorithme d'optimisation du problème QoSWSC qui repose sur la recherche Tabou. L'algorithme 4 procède comme suit :

(a) **Étape 1** : Le voisinage d'une solution (lignes 8-16).

Pour chaque solution, nous produisons toutes les solutions voisines, en utilisant la fonction $SwapMoves()$.

Pour cela, nous générons tous les mouvements possibles, et nous calculons

**Algorithme 4** '**QoS-SkyTabou**' : l'optimisation de QoSWSC par la recherche Tabou

**Entrée(s):** un workflow qui contient **m** tâches et **l** services par tâche,
**R** : requête de l'utilisateur,
**$\alpha$-DSkyS**=$\{\alpha\text{-}DSkyS_1, .., \alpha\text{-}DSkyS_m\}$ ▷ un ensemble de services Skyline pour chaque tache
**t** : la longueur de la liste taboue,
**it** : l'itération courante,
**MaxIt** : le nombre d'itérations maximales,
**p** : la période de stagnation,
**seuil** : le seuil de stagnation,
**Sortie(s):** $C^*$ : La solution quasi-optimale.

1: $C_0 \leftarrow ComposeServices(random(S_{ij}), ..random(Sij))$, ▷ la solution initiale
2: $C^* \leftarrow C_0$
3: $Ls \leftarrow \{C^*\}$ ▷ L'ensemble de solutions
4: $LTabu \leftarrow \{\}$ ▷ L'ensemble de compositions taboues
5: $Scores[\,]$ ▷ Vecteur des scores de p solutions initialisées à 0
6: $Stagnation[\,]$ ▷ Vecteur de p stagnations initialisées à 0
7: **tant que** $it < MaxIt$ **faire**
8: $Ls \leftarrow SwapMoves(C^*)$ ▷ Générer les solutions voisines de $C^*$
9: **pour tout** $Ci$ in $Ls$ **faire**
10: $score_i = U(C_i)$
11: **fin pour**
12: $Sort(Ls)$, ▷ Trier les solutions voisines selon le score $U(C_i)$
13: $C_0 \leftarrow MaxScoreComposition(Ls)$
14: **tant que** $C_0$ est taboue **faire**
15: $C_0 \leftarrow NextMaxScoreComposition(Ls)$
16: **fin tant que**
17: $LTabu \leftarrow$ Update$(LTabu, C_0)$
18: $ScoreC_0 = U(C_0),\ \ ScoreC^* = U(C^*)$
19: **si** $ScoreC_0 > ScoreC^*$ **alors**
20: $C^* \leftarrow C_0$
21: $Scores[1] = ScoreC_0$
22: $debut = 2,\ k = 1$
23: **sinon**
24: **si** $debut > 1$ **alors**
25: $Scores[debut] = ScoreC_0$
26: $Stagnation[k] = |scores[debut] - scores[debut - 1]|$
27: $debut = debut + 1,\ k = k + 1$
28: **fin si**
29: **fin si**
30: **si** $(debut == p)$ et (toutes $Stagnation[k] < seuil$) **alors**
31: Réinitialiser tous les paramètres et aller à la ligne 7
32: **fin si**
33: $it = it + 1$
34: **fin tant que**
35: **retourner** $C^*$

le score de toutes les solutions voisines (ligne 10). Enfin, nous choisissons la meilleure solution voisine pour la prochaine itération. Cette dernière est celle qui a le score le plus élevé (ligne 13). Si la meilleure solution voisine est dans la liste taboue, nous choisissons la deuxième meilleure solution qui n'appartient pas à la liste taboue (ligne 15).

(b) **Étape 2** : Mettre à jour la liste taboue (ligne 17).

Nous ajoutons la meilleure solution voisine à la liste taboue (pour éviter un cycle).

(c) **Étape 3** : Mettre à jour la solution optimale et éventuellement redémarrage de l'algorithme (lignes 19-32).

Si la solution courante $C_0$ est meilleure que la solution quasi-optimale $C^*$, nous mettons à jour $C^*$ (ligne 20) et nous initialisons le début de la période de stagnation (lignes 21 et 22). Pour les prochaines itérations, nous calculons la stagnation par la différence du score entre la solution courante et celle de l'itération précédente. Si nous exécutons $p$ itérations et nous n'avons aucune amélioration de l'optimalité (i.e toutes les stagnations sont inférieures au *seuil*), alors on redémarre l'algorithme. Cette amélioration permet d'éviter les régions non prometteuses, surtout lorsque nous dépassons un certain nombre d'itérations.

L'algorithme se termine quand le nombre maximal d'itérations est atteint. Le paramétrage des valeurs des variables d'entrées de l'algorithme, à savoir, la taille de la liste taboue, le nombre d'itérations et les paramètres de stagnation seront discutés dans le chapitre expérimentation.

• **Complexité temporelle de QoS-SkyTabou**

Soit $v$ le nombre de voisins d'une solution, qui correspond à la taille de la liste $Ls$ dans l'algorithme 4, la complexité de l'algorithme 4 est : $O(4m + 2p + MaxIt(5m + 3p + v.m + v.d.m + v \log v + v.m.t + t.m + 2d.m))$. Le tableau 6.8 présente la complexité temporelle de certaines instructions.

## 6.4 Conclusion

Nous avons présenté dans ce chapitre les approches proposées pour résoudre la problématique QoSWSC. Nous avons, dans un premier temps, formalisé le concept de dominance floue $FDed()$ et par la suite montré son utilisation à travers les deux

| $N^o$ **de la ligne de l'instruction** | **Complexité de l'instruction** | **Explication** |
|---|---|---|
| Lignes 1, 2, 3 ou 20 | $O(m)$ | boucle de chacune de ces instructions de taille **m** : les services de la composition) |
| Ligne 8 | $O(v.m)$ | boucle de taille **v** : générer les voisins |
| Lignes (14-16) | $O(v.m.t)$ | tester les $v$ voisins s'ils ne sont pas tabous $t$ |
| lignes (9-11) | $O(v.d.m)$ | boucle qui calcule le score $U()$ des **v** voisins |
| ligne 12 | $O(v \log v)$ | le tri de la liste **Ls** de taille **v** |
| ligne 17 | $O(t.m)$ | boucle de tri de la liste taboue de taille **t** |
| lignes 5, 6 ou 30 | $O(p)$ | boucle de parcours des vecteurs : Stagnation et Scores, de taille **P** |
| Ligne 31 | $O(3m + 2p)$ | Initialisation des paramètres exp : $Ls, C_0, C^*, Stagnation$ |

***Tableau 6.8:*** *Complexité temporelle de quelques instructions de l'algorithme 4*

algorithmes *FTop-KS* et *α-DSkyS-SFS* qui permettent de retourner respectivement les K- meilleurs services et les services Skyline.

Dans un second temps, nous avons présenté les algorithmes *QoS-SOMA*, *FQoS-SOMA* qui reposent sur la méta-heuristique SOMA et l'algorithme *QoS-SkyTabou* qui repose sur la méta-heuristique Tabou. Ces algorithmes seront validés dans le prochain chapitre qui présente l'architecture du système et le prototype qui implémente l'ensemble de nos propositions.

Chapitre 7

# Implémentation et expérimentations

## Sommaire

## 7.1 Introduction

Dans ce chapitre, nous décrivons un prototype implémentant l'environnement de sélection des services Web, cette sélection peut être locale, globale ou hybride. Nous décrivons, en premier lieu, l'architecture générale du système nommé SoS-QoS (environnement de **S**élection et d'**O**ptimisation de **S**ervices web basée **QoS**) en présentant les différents modules ainsi que leurs rôles. Ensuite, nous présentons les corpus (bases de test : réelle et synthétique) utilisés pour évaluer l'environnement. Enfin, nous montrons les différentes expérimentations menées tout en discutant les résultats obtenus.

## 7.2 Présentation du système de sélection des services web : SoS-QoS

Dans le but d'expérimenter les approches proposées dans le chapitre 6, nous avons développé l'environnement SoS-QoS. L'architecture de SoS-QoS est présentée dans la figure 7.1.

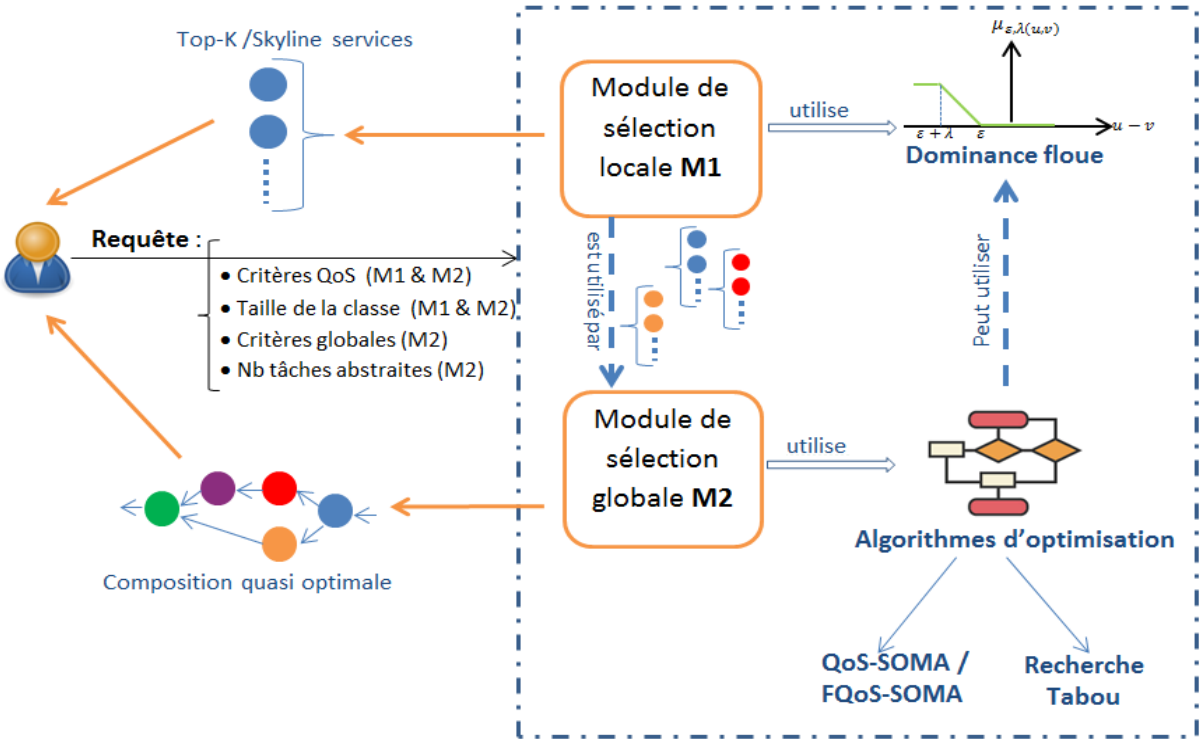



**Figure 7.1:** *L'architecture du système SoS-QoS*

Le système se compose principalement de deux modules M1 et M2 qui interagissent avec la requête de l'utilisateur.

### 7.2.1 Module de sélection locale (M1)

Ce module implémente les propositions 1 et 2 présentées dans le chapitre 6 respectivement dans la section 6.2.2 et la section 6.2.3, il permet de sélectionner les Top-K services web ou bien de calculer les services Skyline. Cette sélection est locale et repose sur l'extension de la relation de Pareto dominance (voir chapitre 5 section 1.2) où les paramètres $\varepsilon$ et $\lambda$ doivent être initialisés.

Le module implémente l'algorithme 2 : '*FTop-KS*' et l'algorithme 2 : '*$\alpha$-DSkyS-SFS*'. Les deux algorithmes ont besoin d'une requête qui a comme paramètres d'entrées : la taille de la classe, le nombre de critères de qualités de services à prendre en considération avec leurs bornes respectives. Dans le cas de Top-k nous avons besoin d'initialiser le paramètre k qui définit le nombre de premiers services à retourner. Dans le cas de Skyline, nous avons besoin d'initialiser le paramètre $\alpha$. Nous avons aussi implémenté la fuzzification de la relation 'est dominant' proposée par [Benouaret et al., 2011b] pour la comparer à notre proposition.

### 7.2.2 Module de sélection globale (M2)

Ce module implémente les propositions 3, 4 et 5 présentées dans le chapitre 6 respectivement dans la section 6.3.3, section 6.3.4 et section 6.3.5. Il permet de trouver la composition quasi optimale en utilisant les méta-heuristiques. Plus précisément, nous avons implémenté l'algorithme 3 : 'QoS-SOMA' et sa version améliorée 'FQoS-SOMA'. Les deux versions d'algorithmes proposés 'QoS-SOMA' et 'FQoS-SOMA' sont une adaptation de l'algorithme initial de la méta-heuristique SOMA. Nous avons aussi implémenté l'algorithme 4 : 'QoS-SkyTabou' qui implémente les principes de la recherche Tabou appliqués sur un ensemble de services Skyline.

L'algorithme 'FQoS-SOMA' et 'QoS-SkyTabou' sont des algorithmes de sélection hybride ; car les deux utilisent une phase de sélection locale qui repose sur le principe de la dominance floue pour 'FQoS-SOMA', et l'extraction de services Skyline pour 'QoS-SkyTabou'.

Les paramètres d'entrée de ce module sont ceux utilisés dans le module M1 plus ceux qui sont propres à la phase de sélection globale à savoir : Le nombre de tâches abstraites de la composition, les critères de QoS globales (les bornes QoS de la composition qui ne doivent pas être violées). Les paramètres d'ajustement de chaque algorithme doivent être initialisés aussi.

## 7.3 Implémentation du système SoS-QoS

L'implémentation de l'environnent à été faite en utilisant java (JDK 1.7.0) et l'environnement de développement intégré netbeans 8.1. Le prototype dont l'interface principale est représentée dans la Figure 7.2 contient les modules définis dans l'architecture du système. L'interface principale permet de paramétrer les variables de la requête et les différents algorithmes.

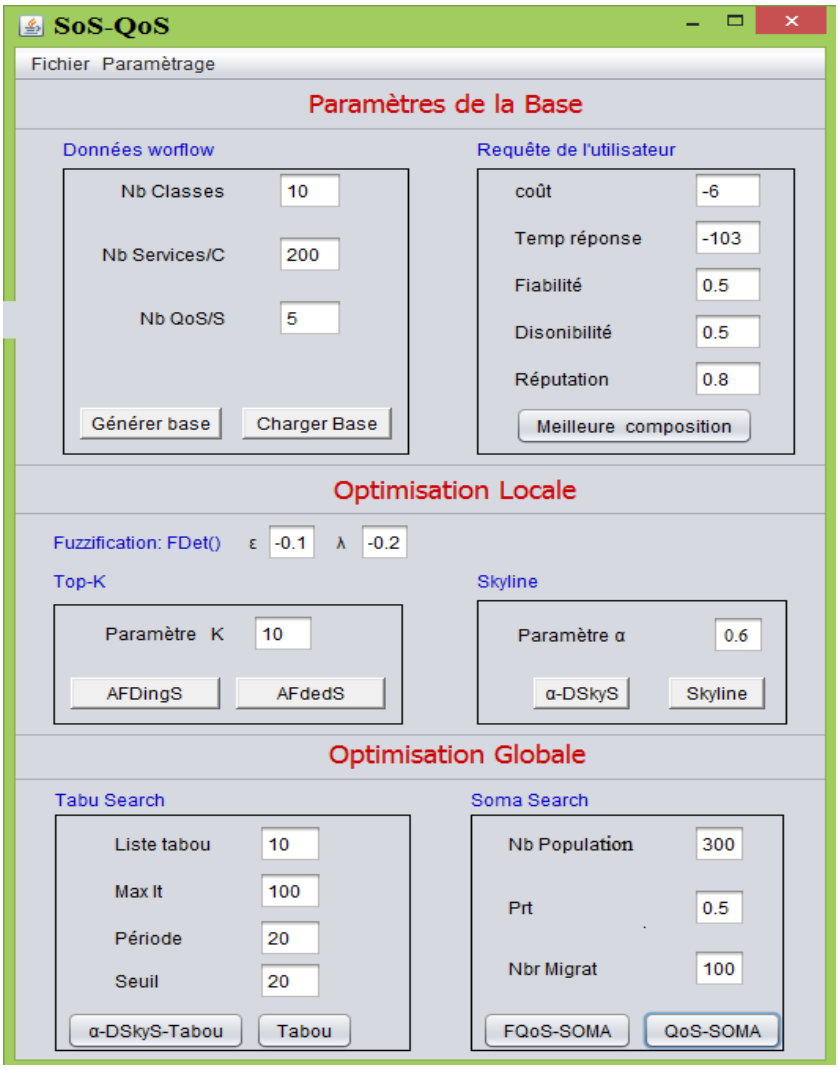



**Figure 7.2:** *Interface principale de SoS-QoS*

Le prototype permet aussi de générer une base de services synthétiques dans un format XML et de manipuler aussi une base réelle. Les deux bases sont définies dans la section suivante.

## 7.4 Corpus utilisés

Pour évaluer les performances de SoS-QoS, nous avons utilisé deux collections de données

### 7.4.1 Base de test 1 : La base réelle 'QWSDataset'

La première est la base 'QWS Dataset' [Al-Masri and Mahmoud, 2008]. Les services Web contenus dans QWS Dataset ne sont pas publiés par les fournisseurs mais récupérés par un moteur d'aspiration de services Web (Web Service Crawler Engine) [Al-Masri and Mahmoud, 2007] à travers différentes sources telles que les registres UDDI, les moteurs de recherche ou les portails spécifiques aux services Web. Les services Web aspirés sont ensuite centralisés dans le QWS Dataset. Le dataset présente un ensemble de 9 paramètres de QoS par service. Nous avons utilisé la version (2.0) [1] qui contient 2507 services Web. La base est récupérée sous forme d'un fichier texte. Chaque ligne du fichier contient un ensemble de 11 paramètres séparés par des virgules. Les paramètres représentent les 9 QoS, le nom du service et L'URL de son fichier WSWL. Le Tableau 7.1 donne la description de ces paramètres. Cette base est surtout utilisée pour expérimenter le module M1.

### 7.4.2 Base de test 2 : La base synthétique

Cette base est utilisée, beaucoup plus, pour tester les contributions implémentées dans le module M2. Dans cette base, nous générons un ensemble d'instances de problèmes en spécifiant un nombre de paramètres qui sont définis dans le tableau 7.2

Plusieurs instances de problèmes ont été générées en utilisant l'algorithme [Parejo et al., 2014]. Chaque instance de problème est définie par un nom $P_i$ où $i$ représente l'identifiant du problème. Pour générer un problème $P_i$, nous devons d'abord définir la structure utilisée en précisant le nombre de tâches à générer pour la composition. Pour des raisons de simplicité, nous nous sommes contentés de générer des workflow à structure séquentielle ; nous rappelons que les workflow à structures complexes (boucles, parallèles) peuvent être facilement réduits à des workflow séquentiels [Cardoso et al., 2004].

Le nombre de tâches du workflow varie entre 1 et 100 tâches. Pour chaque tâche, un ensemble de services est aléatoirement généré. Le nombre de services pour une

1. http ://www.uoguelph.ca/ qmahmoud/qws

| ID paramètre QoS | Nom du paramètre | Description | Unité |
|---|---|---|---|
| 1 | Temps de réponse | Le temps pris pour envoyer une requête et recevoir une réponse | ms |
| 2 | Disponibilité | Nombre d'invocations réussies/total d'invocations | % |
| 3 | Débit (Throughput) | Nombre total d'invocations pendant une période de temps donnée | i/s |
| 4 | Successability | Nombre de réponses/nombre de messages de demande | % |
| 5 | Fiabilité | Nombre de messages d'erreurs / le total des messages | % |
| 6 | Conformité | La mesure à laquelle un document WSDL suit la spécification WSDL | % |
| 7 | Meilleures pratiques | Le degré avec lequel le service web suit WS-I Basic Profile | % |
| 8 | Latence | Le temps pris par le serveur pour traiter une requête | ms |
| 9 | Documentation | Mesure de documentation (i.e tags de description) dans WSDL | % |
| 10 | Nom du service | Le nom du service web | rien |
| 11 | Location du service | Emplacement du document de définition du service WSDL dans le web | rien |

***Tableau 7.1:*** *Description des paramètres QoS du QWSdataset*

| **Paramètres du problème $p_i$** | **Domaine de définition du paramètre** |
|---|---|
| Nombre de tâches $m$ | $m \in$[1-100] |
| Nombre de services candidats par tâche $l$ | $l \in$[1-500] |
| Critères $QoS$ | $Tempsderéponse \in [0, 300ms]$ |
| | $Réputation \in [0, 5]$ |
| | $Coût \in [0, 30€]$ |
| | $Fiabilité \in [0.5 - 1.0]$ |
| | $Disponibilité \in [0.7 - 1.0]$ |
| Fonction objectif $U()$ | l'importance de chaque critère $QoS$ est égale à $w_{qi} = 0.2$ |

***Tableau 7.2:*** *Description des paramètres d'une instance de Problème QoSWSC.*

tâche peut varier entre 1 et 500. Chaque service contient un ensemble de 5 paramètres QoS. Les intervalles de valeurs des 5 paramètres QoS ont été inspirés de [Yu and Bouguettaya, 2009]. Notons que, nous avons utilisé les valeurs log(fiabilité) et log (disponibilité) au lieu des valeurs originales de ces paramètres. Ceci permet l'uniformisation des fonctions d'agrégations, i.e. que tous les critères de QOS vont utiliser la somme comme fonction d'agrégation, au lieu du produit (log (a.b)=log a+ log b).

## 7.5 Expérimentations

Dans cette section nous allons mener une série de tests réalisée avec le prototype SoS-QoS.

Toutes les expérimentations ont été menées sur la même machine dotée d'un processeur Intel i3-2365M CPU @ 1.40 GHz 4 processeurs, une RAM de 4.0 GB et un système d'exploitation Ubuntu 13.10. Dans un premier temps, nous présentons les résultats d'expérimentations relatives au module de sélection locale M1.

### 7.5.1 Performances de l'approche de sélection locale des Top-k services

L'objectif de cette expérimentation [Halfaoui et al., 2015] est de prouver l'efficacité de l'utilisation de la fonction $AFDedS()$ dans la sélection des Top-k services et le calcul de Skyline. Nous avons mentionné auparavant (Chapitre 3 et Chapitre 5)

qu'il existe deux mesures de dominance 'est dominé' et 'est dominant'. Nous avons utilisé, dans notre proposition, le concept 'est dominé' à travers sa fuzzification par la fonction $FDed()$ . Pour renforcer la justification de ce choix , nous allons comparer les résultats de l'utilisation de $AFDedS()$ avec la fonction inverse, pour cela nous avons implémenté la fonction définie dans [Benouaret et al., 2011b] et nous l'avons nommée $AFDingS()$.

Plusieurs simulations ont été faites sur la base de test 1 en variant les paramètres :

- $\varepsilon, \lambda$ : paramètres de la fonction $AFDedS()$ ;
- $d$ :nombre de paramètre $QoS$ ;
- $n$ :nombre de services dans la même classe $S$.

Pour chaque simulation, nous comparons les $Top$-5 services retournés par les algorithmes $AFDedS$ et $AFDingS$. La base de test 1 contient 2507 services web appartenant à différentes classes. Nous avons d'abord groupé les services fonctionnellement semblables dans des classes en leur associant des étiquettes ; par exemple la classe "SMS" (l'envoi du SMS) contient 30 services réels. La classe "recherche" (c'est-à-dire. Les services Web de moteur de recherche comme Google, Amazone, etc) contient 92 services.

#### 7.5.1.1 Simulation 1 : en variant $\varepsilon$ and $\lambda$

Dans cette simulation, nous présentons deux cas (Tableau 7.3) et (Tableau 7.4) en variant $\varepsilon$ et $\lambda$ sur un ensemble de 30 services appartenant à la classe SMS. Chaque service contient 4 paramètres $QoS$.

Nous pouvons observer à partir des résultats présentés dans le Tableau 7.3 et le Tableau 7.4 que le classement donné par $AFDedS$ est plus intéressant que celui donné par $AFDingS$ ; même si nous varions $\varepsilon$ et $\lambda$ le Top-1 (c.à.d le premier service) est toujours meilleur selon $AFDedS$.

En effet, le service S4 (Tableau 7.4) est le Top-1 selon $AFDing$ tandis qu'il n'appartient pas aux premiers 5 services selon $AFDedS$ à cause de sa mauvaise valeur du paramètre $q_1$. Nous pouvons dire que $AFDedS$ favorise les services avec une bonne valeur dans tous les paramètres QoS (services qui sont équilibrés dans leurs paramètres QoS et offrent un bon compromis entre ces derniers) et rejette les services avec les pires valeurs dans certains paramètres $QoS$ même si les autres sont bons.

| **Top-5 AFDingS** | | | | | | |
|---|---|---|---|---|---|---|
| Top-K | $Si$ | $AFDingS$ | $QoS$ | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ |
| 1 | S5 | 0.566 | 0.787 | 1.0 | 0.758 | 0.818 |
| 2 | S4 | 0.551 | 0.0 | 0.754 | 0.975 | 1.0 |
| 3 | S12 | 0.529 | 1.0 | 0.738 | 1.0 | 0.682 |
| 4 | S30 | 0.423 | 0.6 | 0.918 | 0.797 | 0.682 |
| 5 | S6 | 0.329 | 0.941 | 0.738 | 0.603 | 0.682 |
| **Top-5 AFDedS** | | | | | | |
| Top-K | $Si$ | $AFDedS$ | $QoS$ | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ |
| 1 | S12 | 0.071 | 1.0 | 0.738 | 1.0 | 0.682 |
| 2 | S5 | 0.107 | 0.787 | 1.0 | 0.758 | 0.818 |
| 3 | S6 | 0.143 | 0.941 | 0.738 | 0.603 | 0.682 |
| 4 | S30 | 0.25 | 0.6 | 0.918 | 0.797 | 0.682 |
| 5 | S7 | 0.286 | 0.0 | 0.754 | 0.975 | 1.0 |

***Tableau 7.3:*** *Top-5 Services de la classe SMS : classement $AFDingS$ vs. $AFDedS()$, avec $\varepsilon = 0, \lambda = -0.2$*

| **Top-5 AFDingS** | | | | | | |
|---|---|---|---|---|---|---|
| Top-K | $Si$ | $AFDingS$ | $QoS$ | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ |
| 1 | S4 | 0.443 | 0.0 | 0.754 | 0.975 | 1.0 |
| 2 | S5 | 0.421 | 0.787 | 1.0 | 0.758 | 0.818 |
| 3 | S12 | 0.036 | 1.0 | 0.738 | 1.0 | 0.682 |
| 4 | S30 | 0.321 | 0.6 | 0.918 | 0.797 | 0.682 |
| 5 | S6 | 0.223 | 0.941 | 0.738 | 0.603 | 0.682 |
| **Top-5 AFDedS** | | | | | | |
| Top-K | $Si$ | $AFDedS$ | $QoS$ | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ |
| 1 | S5 | 0.0 | 0.787 | 1.0 | 0.758 | 0.818 |
| 2 | S12 | 0.036 | 1.0 | 0.738 | 1.0 | 0.682 |
| 3 | S6 | 0.107 | 0.941 | 0.738 | 0.603 | 0.682 |
| 4 | S30 | 0.143 | 0.6 | 0.918 | 0.797 | 0.682 |
| 5 | S7 | 0.25 | 0.892 | 0.623 | 0.453 | 0.682 |

***Tableau 7.4:*** *Top-5 services de la classe 'SMS' : classement $AFDingS()$ vs. $AFDedS()$, avec $\varepsilon = -0.1, \lambda = -0.2$*

#### 7.5.1.2 Simulation 2 : en variant $d$ and $n$ :

Nous présentons, dans cette simulation, deux scénarios en variant $d$ de 7 à 9 QoS sur un ensemble de 92 services appartenant à la classe 'recherche'. Nous avons fixé $\varepsilon = -0.1$ et $\lambda = -0.2$. Le résultat des 5 premiers services fournis par le classement $AFDtingS$ et $AFDedS$ est montré dans le Tableau 7.5 et le Tableau 7.6.

| **Top-5 AFDingS** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Top-K | $Si$ | $AFDingS$ | $QoS$ | | | | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ |
| 1 | S70 | 0.409 | 0.183 | 0.904 | 0.618 | 0.964 | 0.767 | 1 | 0.815 |
| 2 | S30 | 0.388 | 0.164 | 0.904 | 1 | 0.964 | 0.767 | 1 | 0.815 |
| 3 | S24 | 0.385 | 0.005 | 1 | 0.829 | 1 | 0.767 | 1 | 0.667 |
| 4 | S72 | 0.381 | 0.474 | 0.795 | 0.260 | 0.807 | 0.767 | 0.667 | 0.815 |
| 5 | S16 | 0.365 | 0.003 | 1 | 0.419 | 1 | 1 | 0.667 | 0.111 |
| **Top-5 AFDedS** | | | | | | | | | |
| Top-K | $Si$ | $AFDedS$ | $QoS$ | | | | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ |
| 1 | S30 | 0.005 | 0.164 | 0.904 | 1 | 0.964 | 0.767 | 1 | 0.815 |
| 2 | S52 | 0.006 | 0.016 | 0.819 | 0.955 | 0.94 | 0.767 | 1 | 0.667 |
| 3 | S24 | 0.006 | 0.005 | 1 | 0.829 | 1 | 0.767 | 1 | 0.667 |
| 4 | S70 | 0.008 | 0.183 | 0.904 | 0.618 | 0.964 | 0.767 | 1 | 0.815 |
| 5 | S45 | 0.022 | 0.042 | 0.831 | 0.382 | 0.940 | 0.767 | 1 | 0.667 |

***Tableau 7.5:*** *Top-5 services de la classe 'recherche' ($AFDingS()$ vs. $AFDedS()$) avec $d = 7$*

En observant les résultats donnés dans le tableau 7.5, nous pouvons remarquer que le classement retourné par $AFDedS$ est plus intéressant que celui retourné par $AFDingS$. Le Top-1($AFDedS$) est le service S30. Ce dernier a la meilleure valeur que le Top-1($AFDingS$) sur la qualité $q_3$. De plus, le service S30 est proche la valeur du service S70 sur la qualité $q_1$. Nous pouvons aussi remarquer que le service S16 est inclus dans le top-5($AFDingS$) tandis qu'il n'appartient pas au top-5($AFDedS$) à cause de ses mauvaises valeurs sur les paramètres $q_3$ et $q_7$. En fait, il est remplacé par le service S45 qui a un bon compromis entre ses paramètres $QoS$.

| **Top-5 AFDingS** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Top-K | $Si$ | Score | $QoS$ | | | | | | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $q_8$ | $q_9$ |
| 1 | S24 | 0.397 | 0.050 | 1 | 0.829 | 1 | 0.767 | 1 | 0.667 | 0.004 | 0.958 |
| 2 | S16 | 0.366 | 0.003 | 1 | 0.419 | 1 | 1 | 0.667 | 0.111 | 0.030 | 0.358 |
| 3 | S60 | 0.344 | 0.016 | 0.988 | 0.955 | 1 | 0.333 | 1 | 0.259 | 0.008 | 0.337 |
| 4 | S55 | 0.328 | 0.179 | 0.916 | 0.244 | 0.976 | 0.767 | 1 | 0.815 | 0.066 | 0.800 |
| 5 | S70 | 0.318 | 0.183 | 0.904 | 0.618 | 0.964 | 0.767 | 1 | 0.815 | 0 | 0.021 |
| **Top-5 AFDedS** | | | | | | | | | | | |
| Top-K | $Si$ | Score | $QoS$ | | | | | | | | |
| | | | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $q_8$ | $q_9$ |
| 1 | S24 | 0.006 | 0.050 | 1 | 0.829 | 1 | 0.767 | 1 | 0.667 | 0.004 | 0.958 |
| 2 | S45 | 0.018 | 0.042 | 0.831 | 0.382 | 0.940 | 0.767 | 1 | 0.667 | 0.030 | 0.937 |
| 3 | S55 | 0.018 | 0.179 | 0.916 | 0.244 | 0.976 | 0.767 | 1 | 0.815 | 0.066 | 0.800 |
| 4 | S52 | 0.024 | 0.016 | 0.819 | 0.955 | 0.940 | 0.767 | 1 | 0.667 | 0.017 | 0.105 |
| 5 | S30 | 0.027 | 0.064 | 0.904 | 1 | 0.964 | 0.767 | 1 | 0.815 | 0.092 | 0.053 |

***Tableau 7.6:*** *Top-5 services de la classe 'recherche' ($AFDingS()$ vs. $AFDedS()$) avec $d = 9$*

Les mêmes résultats sont confirmés pour un plus grand nombre de paramètres QoS (voir Tableau 7.6). Par exemple, si nous considérons le Top-2($AFDingS$) vs. Top-2($AFDedS$) ou encore, le Top-3($AFDingS$) vs. Top-3($AFDedS$) ; nous remarquons que les services S45 et S55 offrent un meilleur compromis que le service S16 respectivement S60.

### 7.5.2 Performances des approches de sélection globale

Dans cette section, nous testons les performances du module de sélection globale (M2) du prototype SoS-QoS, nous allons évaluer les performances des trois algorithmes proposés l'algorithme QoS-SOMA, la version améliorée FQoS-SOMA et l'algorithme QoS-SkyTabou. Dans la première section, nous comparons les deux algorithmes QoS-SOMA et FQoS-SOMA ; ensuite, nous comparons les deux algorithmes avec l'algorithme le plus utilisé actuellement qui est celui de PSO (ou Optimisation d'Essaim de Particule) nous avons utilisé la version standard décrite dans [Hadjila, 2014]. Dans la dernière section, nous comparons l'algorithme QoS-SkyTabou avec l'approche standard qui utilise juste l'algorithme Tabou sans le calcul de Skyline.

Toutes les expérimentations menées dans cette section sont faites en utilisant la base de test 2 7.4.2. Afin de ne pas biaiser les résultats des expérimentations, toutes les mesures des simulations représentent les résultats moyens pris de 30 exécutions successives.

#### 7.5.2.1 Comparaison de QoS-SOMA avec FQoS-SOMA

Pour évaluer l'influence des différents paramètres d'ajustement de SOMA , les expérimentations sont conduites avec une taille modérée du problème de sélection de service Web ; incluant 10 tâches et 100 services candidats par tâche. Nous nommons cette instance $P_1$. Nous comparons le taux d'optimalité pour chaque simulation. Le taux d'optimalité représente le rapport entre la valeur de la fonction objectif de la composition courante i ($C_i$), et la valeur de la fonction objectif de la composition optimale notée ($C_{opt}$). La solution optimale de l'instance $P_1$ est $U(C_{opt}) = 0,77$, donc le taux d'optimalité d'une solution ($C_i$) est : $(U(C_i)/0.77) * 100$. Plusieurs expériences ont été d'abord faites avec des combinaisons différentes pour choisir le paramètre PRT et comme mentionné dans tous les travaux qui utilisent l'algorithme SOMA, la meilleure valeur pour PRT est 0.1, ainsi pour toutes les prochaines simulations le PRT est fixé à la valeur 0.1.

**- Optimalité vs. nombre de sauts et nombre de migrations**

Dans cette simulation, nous calculons la performance moyenne de QoS-SOMA et FQoS-SOMA pour 30 exécutions indépendantes. Plus spécifiquement, nous mesurons le taux d'optimalité et le temps d'exécution. Nous utilisons l'instance de problème P1 mentionné ci-dessus P1 (10 tâches x 100 Services). La taille de la population initiale est n = 150 et le nombre de sauts est fixé à 1 (NbJump = 1).

La Figure 7.3 montre les valeurs de taux d'optimalité des solutions trouvées par rapport au nombre de migrations.

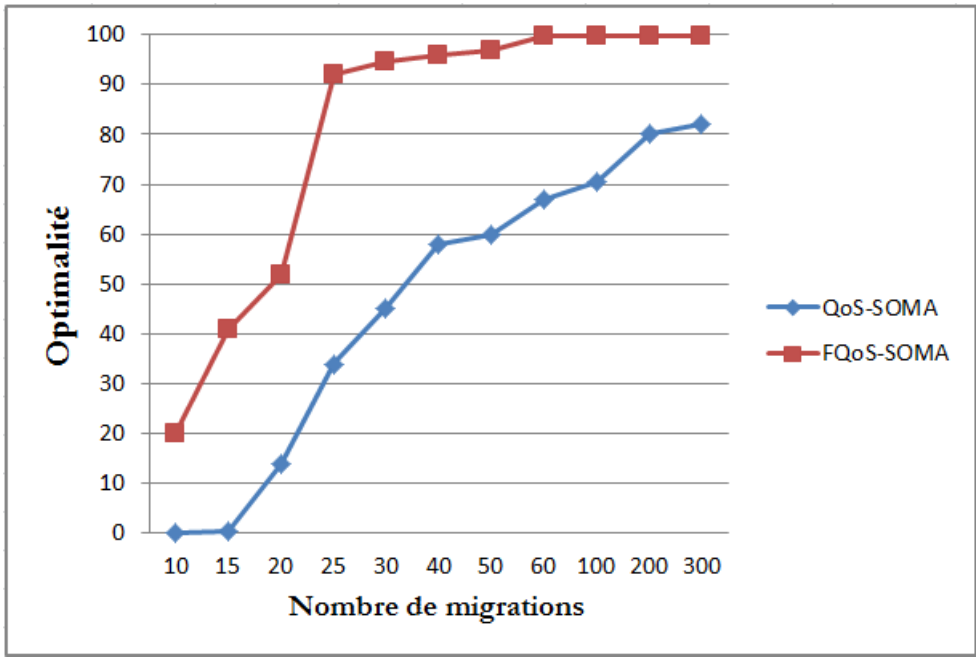



**Figure** ***7.3:*** *Optimalité vs. nombre de migrations avec NbJump=1*

Nous remarquons que l'algorithme FQoS-SOMA montre une convergence rapide en comparaison avec QoS-SOMA. FQoS-SOMA peut atteindre la meilleure solution pour $P_1$, il commence à trouver une solution proche de l'optimale à la migration = 25 (optimality = 91.56 %) et la meilleure solution est trouvée à la migration = 60 (optimality = 100 %). QoS-SOMA a une convergence lente avec NbJump = 1. Après plusieurs simulations, nous avons constaté que QoS-SOMA commence à trouver une solution proche de l'optimale (optimality = 85 %) avec NbJump = 1 seulement si nous augmentons le nombre de migrations à 300. Nous observons aussi que QoS-SOMA n'arrive jamais à atteindre la meilleure solution avec les paramètres : NbJump = 1 et une taille de population initiale = 150.

Nous examinons maintenant le paramètre NbJump. La figure 7.4 montre que QoS-SOMA commence à converger vers une bonne solution quand $NbJump = 3$. Cependant, FQoS-SOMA peut converger avec seulement un seul saut, $NbJump = 1$. Dans la simulation suivante, nous considérons la même instance de problème P1 et le même nombre de migrations = 100.

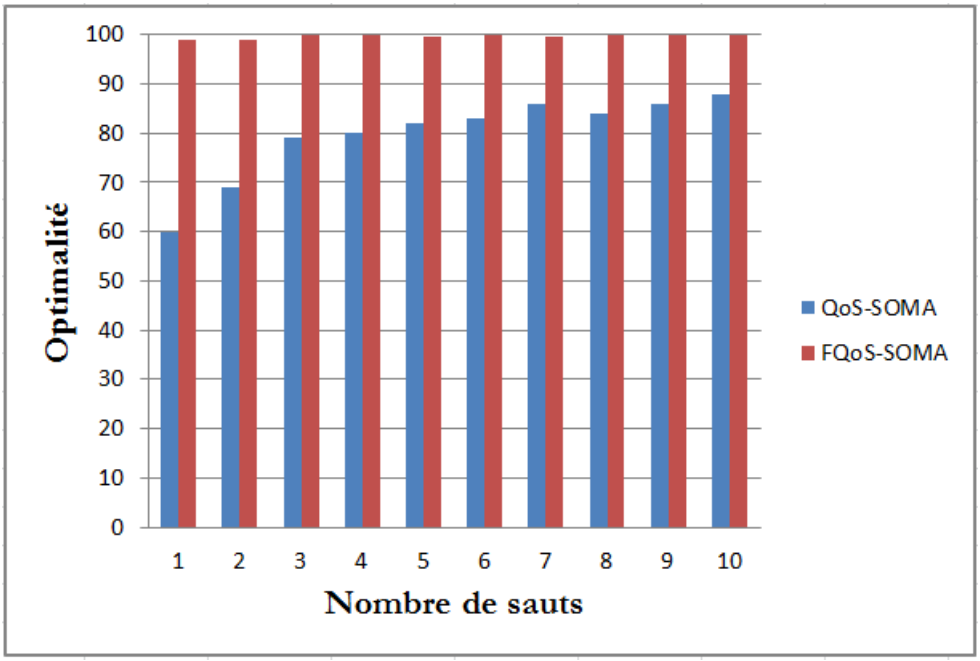



**Figure 7.4:** *Optimalité vs. nombre de sauts avec Migrations=150*

**- Temps d'exécution vs. nombre de sauts et nombre de migrations**

Comme prévu, le temps d'exécution augmente avec l'augmentation du nombre de sauts (voir Figure 7.5) et/ou le nombre de migrations (voir Figure 7.6).

Nous remarquons que FQOS-SOMA prend un temps additionnel minime (peu de millisecondes) pour calculer les compositions en comparaison avec QOS-SOMA. Ce fait est causé par l'optimisation locale qui utilise la fonction $FDedS()$, mais malgré cela, FQoS-SOMA converge rapidement avec un seul saut, et reste donc plus efficace que QoS-SOMA. FQoS-SOMA peut trouver une solution proche de l'optimale avec seulement un saut et 30 migrations. La meilleure solution est trouvée après 60 migrations.

En résumé, FQoS-SOMA donne de meilleurs résultats en termes d'efficacité et optimalité en comparaison avec QoS-SOMA. QoS-SOMA converge lentement ; cependant, il peut fournir des solutions compétitives si on augmente le nombres de migrations et la taille de population initiale. Nous faisons remarquer que, si nous mettons les paramètres de QoS-SOMA à : la population initiale = 280, la migration = 300 et NbJump = 3, il peut donner de très bon résultats et atteindre 90 % de taux d'optimalité dans 21 secondes. Cependant, il n'est pas pratique et souhaitable d'obtenir une solution satisfaisante en consommant un temps d'exécution élevé (car le temps d'exécution augmente avec l'augmentation du nombre de migrations et de sauts).

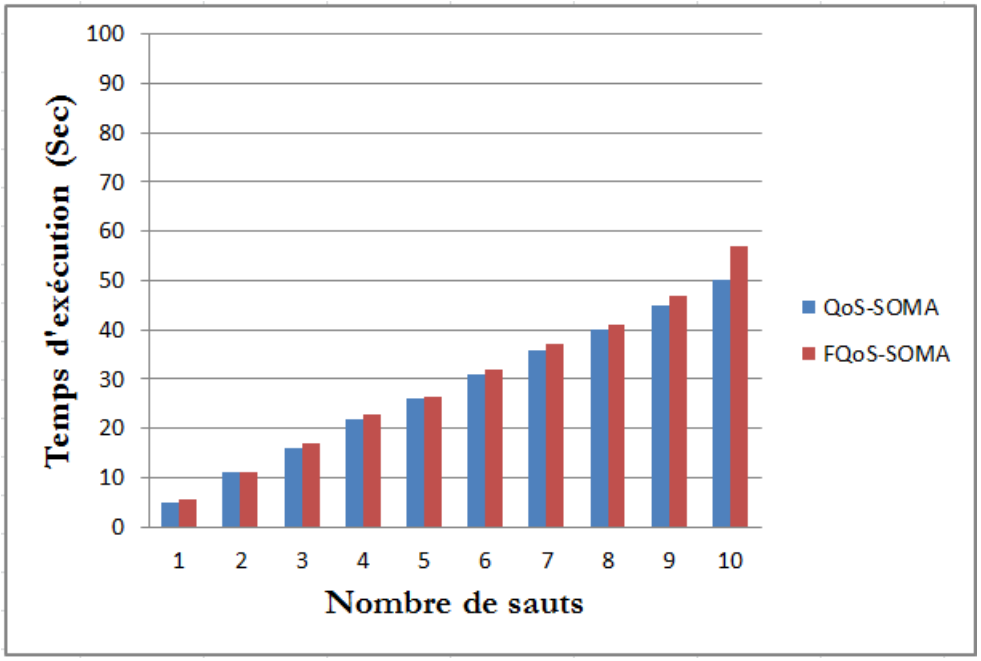



**Figure *7.5:*** *Temps d'exécution vs. nombre de sauts avec Migrations=100*

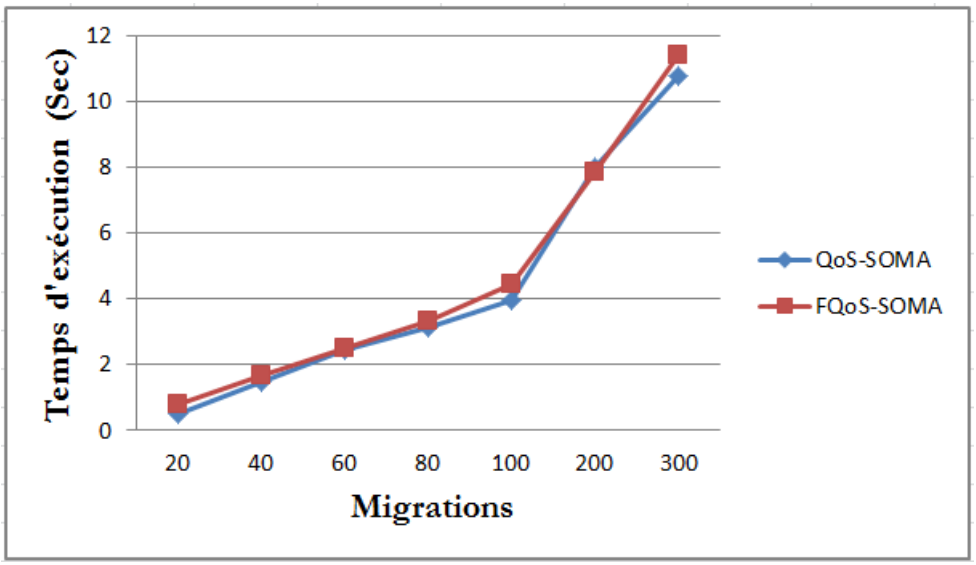



**Figure *7.6:*** *Temps d'exécution vs. nombre de migrations avec NbJump=1*

A partir de ces résultats, on peut conclure que FQoS-SOMA surpasse QoS-SOMA. Ce fait peut être expliqué par la convergence rapide de FQoS-SOMA grâce à l'optimisation locale du fait de l'utilisation de la dominance floue.

En se basant sur les résultats de ces expérimentations, nous pouvons constater que la meilleure configuration de FQoS-SOMA est obtenue en fixant les paramètres comme suit :

- Taille de la population initiale = 150,
- PRT = 0.1,
- NbJump = 1.

#### 7.5.2.2 Comparaison de QoS-SOMA, FQoS-SOMA avec PSO

Dans cette section, nous comparons QoS-SOMA, FQoS-SOMA avec l'algorithme le plus utilisé PSO. En effet, PSO est l'algorithme qui a reçu le plus d'attention ces dernières années par la communauté de chercheurs. De plus, la raison principale de notre choix de comparaison de SOMA avec PSO est que les deux algorithmes appartiennent à la même classe des algorithmes méta-heuristiques à base de populations en déduisant leurs règles du comportement animal dans la nature. Ils obtiennent la solution quasi optimale en se basant sur les changements de positions suivant la position de l'individu qui évolue vers la solution optimale.

Nous avons fixé les paramètres des trois algorithmes comme suit :

- Pour l'algorithme PSO, la meilleure configuration est :
  - nombre de particules=140,
  - voisinage complètement maillé (l'essaim entier).
- Pour FQoS-SOMA, la meilleure configuration est :
  - Taille de population = 150,
  - Nombre de sauts : NbJump = 1,
  - Paramètre de perturbation : PRT = 0.1
- Pour QoS-SOMA, la meilleure configuration est :
  - Taille de population = 300,
  - Nombre de sauts : NbJump = 3,
  - Paramètre de perturbation : PRT = 0.1

Pour les simulations de cette partie, nous avons utilisé une autre instance de problème nommée $P_2$ qui inclut 15 tâches et 250 services par tâche. Afin de comparer l'efficacité des trois algorithmes, nous avons comparé leurs vitesses de convergence et leurs taux d'optimalités à travers plusieurs itérations Comme le montre la Figure 7.7, QoS-SOMA montre une convergence rapide avec un petit nombre d'itérations, QoS-SOMA présente une solution avec une optimalité de 92% pour l'itération 60. Pour la même itération, PSO a juste 86%. Cependant, les deux algorithmes convergent plus

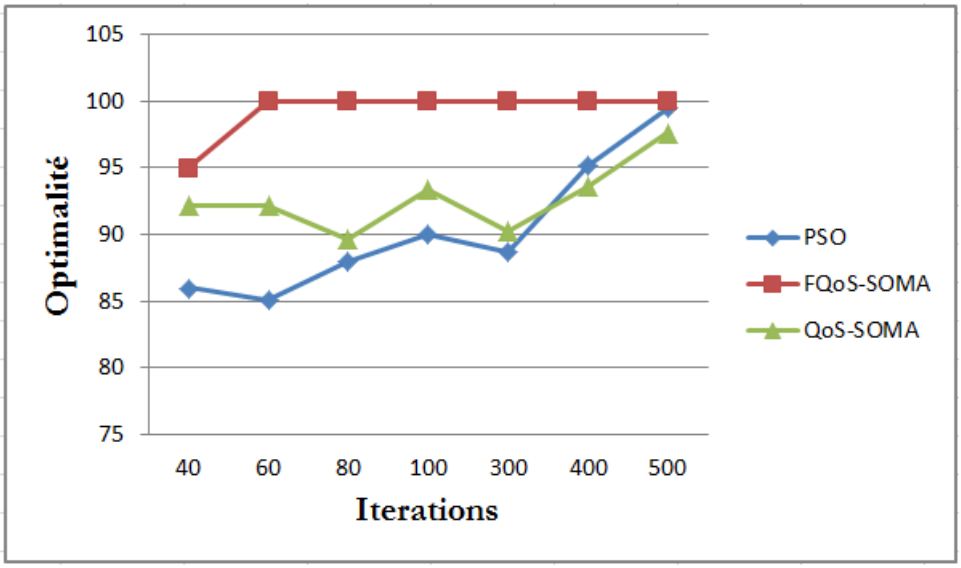



**Figure** ***7.7:*** *Optimalité vs. nombre d'itérations pour PSO, QoS-SOMA et FQoS-SOMA (instance du problème* $P_2$*)*

tard approximativement vers des taux d'optimalité comparables (avec une légère supériorité pour PSO). Ceci peut être expliqué par le fait que les deux algorithmes opèrent de la même façon en faisant des changements de positions de particules pour se déplacer vers le leader, mais dans des chemins d'optimisation différents. Quand une particule trouve une meilleure position dans SOMA, la population entière se déplace vers cet emplacement ; tandis que dans l'algorithme PSO chaque particule change dynamiquement sa position en se basant sur la meilleure position de particule locale et la meilleure position globale.

Nous observons aussi que FQOS-SOMA surpasse les deux autres algorithmes car il converge très vite (à partir de l'itération 40) et atteint la solution optimale 100% à l'itération 60. Nous remarquons aussi que FQOS-SOMA peut trouver la solution optimale en un temps acceptable (voir Figure 7.7, 7.8). En général, la vitesse de convergence de FQOS-SOMA est beaucoup plus rapide que PSO et QoS-SOMA. Il présente une excellente performance en termes de vitesse de convergence et de qualité de solution.

#### 7.5.2.3 Comparaison de QoS-SkyTabou avec QoS-Tabou

L'objectif de cette expérimentation est de prouver l'efficacité de l'utilisation de Skyline ou Top-k dominating pour la réduction de l'espace de recherche et de présenter des services de bonne qualité pour participer à l'optimisation globale. Ainsi, certaines méta-heuristiques à base de population unique peuvent rapidement converger vers

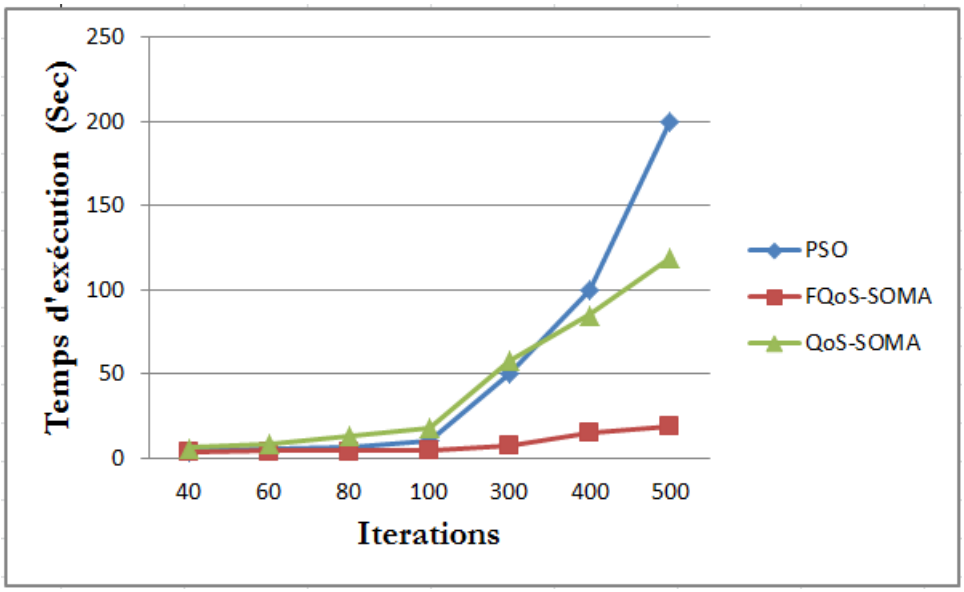



**Figure** ***7.8:*** *Temps d'exécution vs. nombre d'itération pour PSO, QoS-SOMA et FQoS-SOMA (instance du problème $P_2$)*

une solution proche de l'optimale. Pour cela, nous comparons notre approche avec l'approche standard (sans Skyline, ni Top-k dominating) que nous avons nommée 'QoS-Tabou'. Nous notons que le calcul de Skyline ou Top-k est fait hors-ligne (off-line).

Nous avons utilisé dans cette simulation l'instance $P_2$ qui inclut 15 tâches et 250 services par tâche. Plusieurs simulations ont été faites pour trouver la configuration optimale des paramètres de l'algorithme Tabou. Pour cela nous avons varié

- le nombre maximal d'itération : $Maxit \in \{100, 10000\}$ ;
- la période de stagnation : $p \in \{10, 35\}$ ;
- la taille de la liste tabou : $t \in \{7, 20\}$ ;
- le seuil pour le redémarrage de l'algorithme : $seuil \in [0.2, 0.6]$.

La meilleure configuration des paramètres d'ajustement de l'algorithme Tabou a été fixée comme suit :

- $Maxit = 500$ ;
- $t = 10$ ;
- $p = 21$ ;
- $seuil = 0.2$

Afin de comparer notre approche (QoS-SkyTabou) avec l'approche standard (QoS-Tabou), Nous avons comparé le taux d'optimalité des deux approches en variant le

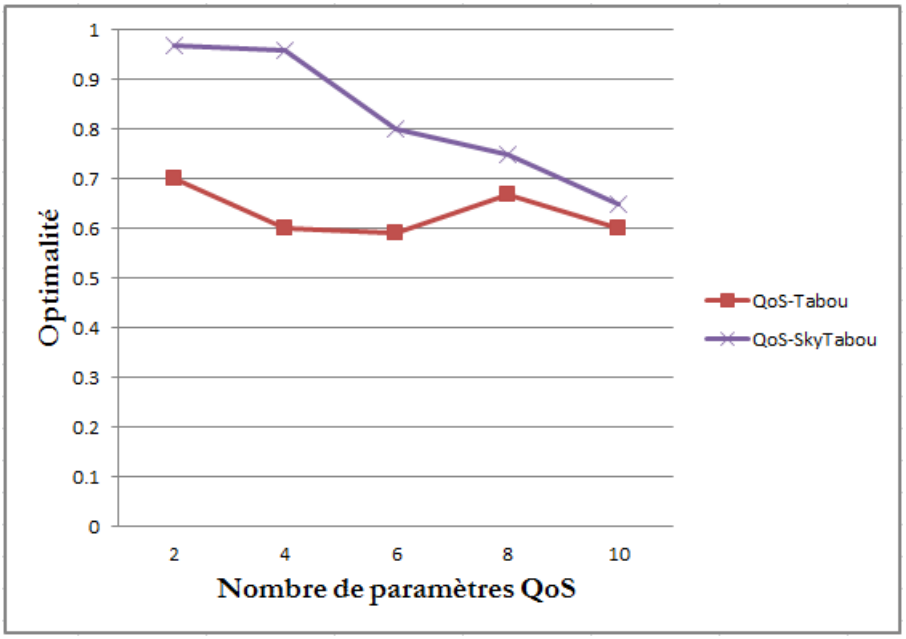



**Figure** ***7.9:*** *Optimalité vs. nombre QoS pour QoS-Tabou, QoS-SkyTabou avec (it=500, t=10, p=0.4 et seuil=0.2)*

nombre de paramètre QoS.

Comme le montre la Figure 7.9, Notre approche retourne une composition de meilleure optimalité que l'approche standard. Cependant, le taux d'optimalité se dégrade de manière significative quand le nombre de QoS augmente. Ceci est justifié par le fait que le nombre de services Skyline augmente avec l'augmentation de nombre de paramètres QoS. Si le nombre de QoS est très grand, le nombre de services dans chaque classe ne change pas trop ce qui ramène l'algorithme QoS-SkyTabou à sa version standard.

Dans [Halfaoui et al., 2014], nous avons aussi remplacé l'utilisation de Skyline par Top-k dominating où la réduction de la taille des services pour chaque classe est faite par l'algorithme 6.2.2.2. Dans ce cas, les meilleurs K services de chaque classe sont utilisés par la recherche Tabou. Pour cela, nous considérons plusieurs Top-k (top-100, top-50, top-20 and top-10) et nous les comparons avec QoS-Tabou. Les résultats de cette simulation sont donnés dans la figure 7.10

A partir de la simulation présentée dans la figure 7.10, nous remarquons que, plus le nombre de services est réduit, plus l'optimalité est amélioré ; ce qui est évident car plus le nombre est réduit, meilleure est la qualité des services qui sont choisis. Par contre, le problème qui se pose est comment fixer la valeur de K (valeur minimale) de façon à satisfaire les contraintes globales. En effet, si nous ignorons les contraintes globales, le Top-1 de chaque classe constituerait la composition optimale.

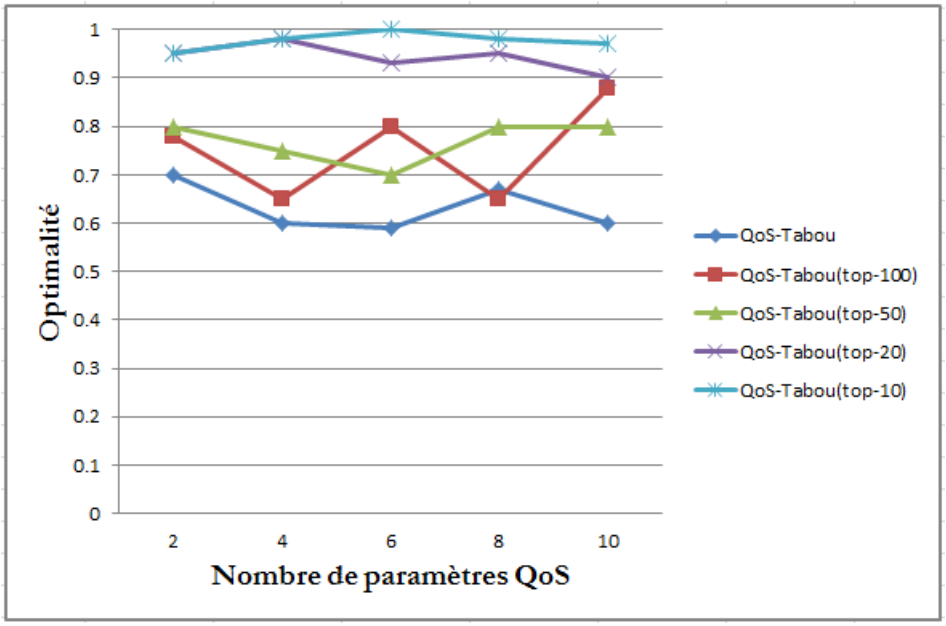



**Figure** ***7.10:*** *Optimalité vs. nombre QoS pour QoS-Tabou et QoS-Tabou(Top-K) avec (it=500, t=10, p=0.4 et seuil=0.2)*

En résumé, l'utilisation de Skyline et/ ou de Top-k dominating combinée à une méta-heuristique (dans notre cas, la recherche Tabou) réduit considérablement l'espace de recherche et surpasse de loin la version standard de l'algorithme. Néanmoins, le Skyline est préconisé dans le cas où le nombre de QoS est réduit. Contrairement au top-k qui n'est pas sensible au nombre de QoS. Néanmoins, il faudra rajouter à l'algorithme un calcul dynamique de la valeur minimale de k qui permet de réduire au maximum le nombre de compositions qui violent les contraintes globales.

## 7.6 Conclusion

Dans ce chapitre, nous avons présenté notre prototype dénommé SoS-QoS, ce dernier permet de vérifier nos propositions de sélection des services web basée QoS, en utilisant les techniques orientées bases de données (Skyline et Top-k) et/ou les méta-heuristiques.

L'architecture proposée se compose de deux modules : l'un est pour la sélection locale, et l'autre pour la sélection globale ; les deux coopèrent en cas de sélection hybride.

La sélection locale repose sur un classement (Top-k) ou un calcul de Skyline qui, tous deux, utilisent une extension de la relation de dominance $AFDedS()$. Les expérimentations menées ont démontré que les approches qui reposent sur la relation de

pareto 'est dominé' $AFDedS()$ proposent des services de meilleure qualité que ceux utilisant la relation 'est dominant'. En effet, les résultats expérimentaux confirment que l'approche proposée est efficace en comparaison avec le classement fait par l'approche 'est-dominant' $AfdtingS()$.

La sélection globale repose sur une adaptation et une amélioration de l'algorithme SOMA ainsi qu'une hybridation de l'algorithme Tabou avec le Skyline ou Top-k pour optimiser le problème QoSWSC. Les comparaisons avec l'algorithme PSO indiquent que la performance de l'algorithme QoS-SOMA est légèrement plus faible à celle de PSO, mais, notons que QoS-SOMA converge plus rapidement (avec une configuration particulière) que PSO. La version améliorée de QoS-SOMA qui intègre une optimisation locale en utilisant $FDed()$ surpasse l'algorithme QoS-SOMA et PSO sans présenter de surcoût de calcul significatif. FQoS-SOMA montre une performance même plus améliorée en termes d'efficacité de calcul et de qualité de solution retournée.

L'utilisation combinée de l'algorithme Tabou avec le calcul de Skyline ou Top-k permet de retourner des compositions de meilleure qualité que l'utilisation seule de l'algorithme Tabou. Ceci est du au fait de la réduction de l'espace de recherche et à la participation que des meilleurs services dans le processus de composition.

Chapitre 8

# Conclusion et perspectives

La sélection des services web est l'une des problématiques les plus importantes de l'architecture orientée service. Elle constitue aussi l'une des étapes les plus importantes dans le processus de composition. Au cours de cette thèse, nous nous sommes intéressés au problème de sélection des services web dans une composition sur la base des besoins non fonctionnels (QoS), communément connu sous le nom QoSWSC. L'objectif de cette thèse est de proposer des solutions qui aident à choisir les services de meilleures qualités pour participer dans le processus de composition.

## 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, et mis en évidence la phase de sélection dans le cycle de vie de la composition.

Nous avons aussi, analysé, modélisé et formulé le problème QoSWSC. Nous avons présenté les approches d'optimisation dédiées à la résolution de ce problème que nous avons regroupées en 4 catégories qui sont : les approches de résolutions exactes, heuristiques approximatives, méta-heuristiques et celles utilisant les techniques orientées bases de données. Nous nous sommes particulièrement intéressés aux deux dernières catégories qui constituent deux axes majeurs de recherches pour la résolution du problème QoSWSC.

Nous avons étudié les travaux qui utilisent les techniques de bases de données qui reposent sur la dominance de Pareto comme les requêtes Skyline et Top-k, pour classer et filtrer les services web. Nous avons aussi présenté et classé les travaux utilisant les méta-heuristiques, qui sont connues pour leurs performances dans la résolution des problèmes Np-difficiles.

La deuxième partie est consacrée aux contributions, où nous avons décrit l'ensemble de nos propositions. Nous avons proposé dans la catégorie d'approches basées

sur les requêtes Skyline et Top-k, deux contributions qui reposent sur la définition de la relation de dominance floue $FDet()$. Dans la première contribution, nous avons présenté l'algorithme '$FTop\text{-}K$ services' qui permet la sélection des Top-k services web basée sur la dominance floue. Dans la deuxième contribution, nous avons présenté l'algorithme '$\alpha\text{-}DSkyS$' qui permet de calculer les services Skyline sur la base de la dominance floue.

Nous avons proposé, dans la catégorie d'approches basées sur les méta-heuristiques, trois contributions. Dans la première contribution, nous avons présenté l'algorithme '$QoS$-SOMA' qui est une adaptation de la récente méta-heuristique SOMA (self-organizing migrating algorithm) au problème QoSWSC. L'algorithme '$QoS$-SOMA utilise une version discrète de SOMA qui est mieux adaptée au problème de sélection des services web composés. Nous avons proposé, dans la deuxième contribution, L'algorithme '$FQoS$-SOMA' qui est une amélioration de l'algorithme QoS-SOMA en utilisant la notion de dominance floue dans la phase de la sélection locale. Dans la troisième contribution, Nous avons proposé l'algorithme '$QoS\text{-}SkyTabou$' qui est basé sur la recherche Tabou et le calcul des Skyline. L'algorithme de Skyline est utilisé dans le but de réduire l'espace de recherche de la sélection.

Nous avons aussi validé et mis en œuvre nos propositions à travers le développement de l'outil de sélection nommé 'SoS-QoS'. Les expérimentations réalisées à travers cet outil nous ont permis de constater d'une part, que la sélection basée sur la relation $FDed()$ retourne des services qui sont équilibrés dans leurs valeurs QoS et sont, de ce fait, de meilleure qualité. D'autre part, Les comparaisons avec l'algorithme PSO indiquent que l'algorithme PSO présente une infine légère supériorité que QoS-SOMA, néanmoins QoS-SOMA trouve les solutions quasi optimales bien avant PSO (avec une configuration particulière). Par contre la version améliorée '$FQoS$-SOMA' montre une performance satisfaisante en termes d'efficacité de calcul et de qualité de solutions retournées. Enfin„ l'algorithme '$QoS - SkyTabou$' permet de trouver des compositions de meilleure qualité que l'utilisation seule de l'algorithme Tabou. Ceci est du à la réduction de l'espace de recherche et à la participation des meilleurs services dans le processus de composition.

## 8.2 Perspectives

Plusieurs améliorations et extensions peuvent être envisagées pour enrichir les approches proposées :

- La prise en considération de l'aspect de dépendances et de conflits entre les services : pendant la modélisation d'une composition de services, la sélection d'un service est fortement dépendante de contraintes comme la date de l'utilisation des services et le lieu. Ce problème a été d'abord identifié par les auteurs Ai et la Saveur en 2008. Par contre, la considération des dépendances de services et des conflits reste toujours peu explorée.
- La prise en considération des aspects et des critères relatifs au cloud : définir la sélection dans le contexte du cloud computing où d'autres critères de sélection de services sont à prendre en considération comme la distance entre services dans les data centers dans les clouds. Dans ce contexte, l'utilisation de l'algorithme SOMA deviendra plus intéressante où le calcul de distance entre le service leader et les autres services s'effectuera aussi sur la base de la position des services web dans les data centers dans le cloud.
- Intégrer l'incertitude dans la QoS : nous constatons que les valeurs des paramètres QoS ne sont pas déterministes, mais elles évoluent et changent en fonction de l'environnement du Service web. De ce fait, il serait judicieux de développer des algorithmes qui solutionnent la problématique QoSWSC qui gère la QoS incertaine.
- Implémenter des algorithmes de calcul de Skyline plus performants : pour l'instant, nous avons adapté l'algorithme de calcul de Skyline SFS pour générer les services Skyline sur la base de la dominance floue. Il serait plus intéressant d'utiliser des algorithmes de type index comme l'algorithme BBS pour accélérer la recherche.
- Enfin, nous sommes en train de tester et d'adapter d'autres nouvelles méta-heuristiques pour la résolution du problème QoSWSC comme la méta-heuristique FruitFly [Pan, 2012].

# Liste des publications

## - Journaux internationaux avec comité de lecture

Halfaoui Amal, Hadjila Fethallah, and Didi Fedoua (2017, in press). 'Qos-aware web service selection based on self-organising migrating algorithm and fuzzy dominance'. International Journal of Computational Science and Engineering. Special Issue on : Advanced Information Processing in Communication. Inderscience.

## - Chapitre d'un livre

Halfaoui Amal, Hadjila Fethallah, and Didi Fedoua Fedoua (2015). 'QoS-Aware Web Services Selection Based on Fuzzy Dominance', IFIP Advances in Information and Communication Technology, pages 291–300, volume 456, URL="http://dx.doi.org/10.1007/978-3-319-19578-0_24", Springer International Publishing.

Hadjila Fethallah, Belabed Amine, and Halfaoui Amal, (2015). 'Hybrid Web Service Discovery Based on Fuzzy Condorcet Aggregation'. Advances in Databases and Information Systems, pages 415-427, URL="http://dx.doi.org/10.1007/978-3-319-23135-8_28", Springer International Publishing.

## - Conférences internationales avec comité de lecture

Halfaoui Amal, Hadjila Fethallah, and Didi Fedoua (2014, November). Hybrid web service selection based on functional and non-functional properties. In The International Conference on Advanced Aspects of Software Engineering (ICAASE'14), Constantine, Algeria, November 2-4.

Halfaoui Amal, Hadjila Fethallah, and Didi Fedoua (2015, May). QoS-Aware Web Services Selection Based on Fuzzy Dominance. In Computer Science and Its Applications : 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings (Vol. 456, p. 291). Springer.

# Bibliographie

[Adeli and Cheng, 1994] Adeli, H. and Cheng, N.-T. (1994). Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1) :104–118.

[Agarwal and Jalote, 2010] Agarwal, V. and Jalote, P. (2010). From specification to adaptation : an integrated qos-driven approach for dynamic adaptation of web service compositions. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 275–282. IEEE.

[Akbar et al., 2001] Akbar, M. M., Manning, E. G., Shoja, G. C., and Khan, S. (2001). Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In *International Conference on Computational Science*, pages 659–668. Springer.

[Akkiraju et al., 2005] Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics-wsdl-s.

[Al-Masri and Mahmoud, 2007] Al-Masri, E. and Mahmoud, Q. H. (2007). Crawling multiple uddi business registries. In *Proceedings of the 16th international conference on World Wide Web*, pages 1255–1256. ACM.

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

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

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

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

[Alves et al., 2006] Alves, A., Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., König, D., Mehta, V., Thatte, S., et al. (2006). Web services business process execution language version 2.0.

[Amiri and Serajzadeh, 2012] Amiri, M. A. and Serajzadeh, H. (2012). Effective web service composition using particle swarm optimization algorithm. In *Telecommunications (IST), 2012 Sixth International Symposium on*, pages 1190–1194. IEEE.

[Andrews et al., 2003] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., et al. (2003). Business process execution language for web services.

[Andrieux et al., 2007] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007). Web services agreement specification (ws-agreement). In *Open Grid Forum*, volume 128, page 216.

[Araban and Sterling, 2004] Araban, S. and Sterling, L. (2004). Measuring quality of service for contract aware web-services. In *First Australian Workshop on Engineering Service-Oriented Systems*, pages 54–56.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6) :369–384.

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

[Bartolini et al., 2008] Bartolini, I., Ciaccia, P., and Patella, M. (2008). Efficient sort-based skyline evaluation. *ACM Transactions on Database Systems (TODS)*, 33(4) :31.

[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.

[Benouaret et al., 2011a] Benouaret, K., Benslimane, D., and Hadjali, A. (2011a). A fuzzy framework for selecting top-k web service compositions. *ACM SIGAPP Applied Computing Review*, 11(3) :32–40.

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

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

[Borzsony et al., 2001] Borzsony, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE.

[Boussalia and Chaoui, 2014] Boussalia, S. R. and Chaoui, A. (2014). Optimizing qos-based web services composition by using quantum inspired cuckoo search algorithm. In *International Conference on Mobile Web and Information Systems*, pages 41–55. Springer.

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

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

[Cardoso et al., 2004] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of service for workflows and web service processes. *Web Semantics : Science, Services and Agents on the World Wide Web*, 1(3) :281–308.

[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.

[Chan et al., 2006a] Chan, C.-Y., Jagadish, H., Tan, K.-L., Tung, A. K., and Zhang, Z. (2006a). Finding k-dominant skylines in high dimensional space. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 503–514. ACM.

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

[Chen et al., 2004] Chen, Z., Liang-Tien, C., and Bu-Sung, L. (2004). Qos-aware and federated enhancement for uddi. *International Journal of Web Services Research*, 1(2) :58.

[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.

[Chinnici et al., 2007] Chinnici, R., Haas, H., Lewis, A. A., Moreau, J.-J., Orchard, D., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 2 : Adjuncts. *W3C Recommendation*, 6.

[Chomicki et al., 2005] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2005). Skyline with presorting : Theory and optimizations. In *Intelligent Information Processing and Web Mining*, pages 595–604. Springer.

[Čičková et al., 2008] Čičková, Z., Brezina, I., and Pekár, J. (2008). Alternative method for solving traveling salesman problem by evolutionary algorithm. *Management information systems*, 3(1) :17–22.

[Clement et al., 2004] Clement, L., Hately, A., Von Riegen, C., Rogers, T., Bellwood, T., Capell, S., Colgrave, J., Dovey, M., Feygin, D., Kochman, R., et al. (2004). Uddi version 3.0. 2, 2004. *URL http ://uddi. org/pubs/uddi v3. htm*.

[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.

[D'Ambrogio and Bocciarelli, 2007] D'Ambrogio, A. and Bocciarelli, P. (2007). A model-driven approach to describe and predict the performance of composite services. In *Proceedings of the 6th international workshop on Software and performance*, pages 78–89. ACM.

[Davendra and Bialic-Davendra, 2013] Davendra, D. and Bialic-Davendra, M. (2013). Scheduling flow shops with blocking using a discrete self-organising migrating algorithm. *International Journal of Production Research*, 51(8) :2200–2218.

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

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

[Eberhart et al., 1995] Eberhart, R. C., Kennedy, J., et al. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY.

[El Hadad et al., 2010] El Hadad, J., Manouvrier, M., and Rukoz, M. (2010). Tqos : Transactional and qos-aware selection algorithm for automatic web service composition. *IEEE Transactions on Services Computing*, 3(1) :73–85.

[Erl, 2004] Erl, T. (2004). *Service-oriented architecture : a field guide to integrating XML and web services*. Prentice Hall PTR.

[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.

[Feng et al., 2013] Feng, L.-i., Obayashi, M., Kuremoto, T., Kobayashi, K., and Watanabe, S. (2013). Qos optimization for web services composition based on reinforcement learning. *International Journal of Innovative Computing, Information and Control*, 9(6) :2361–2376.

[Feo and Resende, 1995] Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2) :109–133.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

[Fogel et al., 1966] Fogel, L., Owens, A., and Walsh, M. (1966). Artificial intelligence through simulated evolution john wiley. *New York*.

[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., 2007] Gao, C., Cai, M., and Chen, H. (2007). Qos-aware service composition based on tree-coded genetic algorithm. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, pages 361–367. IEEE.

[Gao et al., 2006] Gao, Y., Na, J., Zhang, B., Yang, L., and Gong, Q. (2006). Optimal web services selection using dynamic programming. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pages 365–370. IEEE.

[Gao et al., 2009] Gao, Z.-p., Jian, C., Qiu, X.-s., and Meng, L.-m. (2009). Qoe/qos driven simulated annealing-based genetic algorithm for web services selection. *The Journal of China Universities of Posts and Telecommunications*, 16 :102–107.

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

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

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

[Group et al., 2006] Group, O. et al. (2006). Business process modeling notation (bpmn) version 1.0. omg final adopted specification. *Object Management Group*.

[Gudgin et al., 2003] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., and Lafon, Y. (2003). Simple object access protocol (soap) 1.2. *World Wide Web Consortium*.

[Guttman, 1984] Guttman, A. (1984). *R-trees : a dynamic index structure for spatial searching*, volume 14. ACM.

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

[Hadjila and Chikh, 2012] Hadjila, F. and Chikh, A. (2012). Qos-aware service selection based on tabu serach. In *JEESI '12. Alger. Algerie*.

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

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

[Hadjila et al., 2012b] Hadjila, F., Chikh, M. A., Mohammed, M., and Zineb, K. (2012b). Qos-aware service selection based on swarm particle optimization. In *International Conference on Information Technology and e-Services (ICITeS)*, pages 1–6. IEEE.

[Halfaoui et al., 2014] Halfaoui, A., Hadjila, F., and Didi, F. (2014). Hybrid web service selection based on functional and non-functional properties. In *The International Conference on Advanced Aspects of Software Engineering (ICAASE'14)*, pages 20–26. Constantine, Algeria.

[Halfaoui et al., 2015] Halfaoui, A., Hadjila, F., and Didi, F. (2015). *QoS-Aware Web Services Selection Based on Fuzzy Dominance*, pages 291–300. Springer International Publishing, Cham.

[Halfaoui et al., 2017] Halfaoui, A., Hadjila, F., and Didi, F. (2017). Qos-aware web service selection based on self-organising migrating algorithm and fuzzy dominance. *International Journal of Computational Science and Engineering.*

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

[Horstmann and Kirtland, 1997] Horstmann, M. and Kirtland, M. (1997). Dcom architecture. *Microsoft Corporation, July.*

[Huang et al., 2009] Huang, Z., Jiang, W., Hu, S., and Liu, Z. (2009). Effective pruning algorithm for qos-aware service composition. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 519–522. IEEE.

[Huning et al., 1976] Huning, A., Rechenberg, I., and Eigen, M. (1976). Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution.

[Jaeger and Muhl, 2006] Jaeger, M. C. and Muhl, G. (2006). Soft real-time aspects for service-oriented architectures. In *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, pages 5–5. IEEE.

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

[Jaeger et al., 2004] Jaeger, M. C., Rojec-Goldmann, G., and Muhl, G. (2004). Qos aggregation for web service composition using workflow patterns. In *Enterprise distributed object computing conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, pages 149–159. IEEE.

[Jafarpour and Khayyambashi, 2009] Jafarpour, N. and Khayyambashi, M. R. (2009). A new approach for qos-aware web service composition based on harmony search algorithm. In *Web Systems Evolution (WSE), 2009 11th IEEE International Symposium on*, pages 75–78. IEEE.

[Jafarpour and Khayyambashi, 2010] Jafarpour, N. and Khayyambashi, M. R. (2010). Qos-aware selection of web service composition based on harmony search algorithm. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1345–1350. IEEE.

[Jatoth et al., 2015] Jatoth, C., Gangadharan, G., and Buyya, R. (2015). Computational intelligence based qos-aware web service composition : A systematic literature review. *IEEE Transactions on Services Computing*, PP(99) :1–1.

[Jordan et al., 2007] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al. (2007). Web services business process execution language version 2.0. *OASIS standard*, 11(120) :5.

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

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

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

[Kavantzas, 2004] Kavantzas, N. (2004). Web services choreography description language version 1.0, w3c. *http ://www. w3. org/TR/ws-cdl-10/*.

[Keller and Ludwig, 2003] Keller, A. and Ludwig, H. (2003). The wsla framework : Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1) :57–81.

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

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

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

[Ko et al., 2008] Ko, J. M., Kim, C. O., and Kwon, I.-H. (2008). Quality-of-service oriented web service composition algorithm and planning architecture. *Journal of Systems and Software*, 81(11) :2079–2090.

[Kossmann et al., 2002] Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting stars in the sky : An online algorithm for skyline queries. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 275–286. VLDB Endowment.

[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.

[Koza, 1992] Koza, J. R. (1992). *Genetic programming : on the programming of computers by means of natural selection*, volume 1. MIT press.

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

[Kritikos and Plexousakis, 2009] Kritikos, K. and Plexousakis, D. (2009). Mixed-integer programming for qos-based web service matchmaking. *IEEE Transactions on Services Computing*, 2(2) :122–139.

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

[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 et al., 2003] Lee, K., Jeon, J., Lee, W., Jeong, S., and Park, S. (2003). Qos for web services : Requirements and possible approaches. w3c working group note. *World Wide Web Consortium (W3C)*.

[Lee et al., 2007] Lee, K. C., Zheng, B., Li, H., and Lee, W.-C. (2007). Approaching the skyline in z order. In *Proceedings of the 33rd international conference on Very large data bases*, pages 279–290. VLDB Endowment.

[Li et al., 2010] Li, J., Zhao, Y., Liu, M., Sun, H., and Ma, D. (2010). An adaptive heuristic approach for distributed qos-based service composition. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 687–694. IEEE.

[Li and Yan-Xiang, 2010] Li, W. and Yan-Xiang, H. (2010). A web service composition algorithm based on global qos optimizing with mocaco. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 218–224. Springer.

[Li and Wen, 2012] Li, Y.-Q. and Wen, T. (2012). An approach of qos-guaranteed web service composition based on a win-win strategy. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 628–630. IEEE.

[Lian and Chen, 2009] Lian, X. and Chen, L. (2009). Top-k dominating queries in uncertain databases. In *Proceedings of the 12th international conference on extending database technology : advances in database technology*, pages 660–671. ACM.

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

[Lin et al., 2007] Lin, X., Yuan, Y., Zhang, Q., and Zhang, Y. (2007). Selecting stars : The k most representative skyline operator. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 86–95. IEEE.

[Liu et al., 2009] Liu, D., Shao, Z., Yu, C., and Fan, G. (2009). A heuristic qos-aware service selection approach to web service composition. In *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, pages 1184–1189. IEEE.

[Liu et al., 2007] Liu, J., Li, J., Liu, K., and Wei, W. (2007). A hybrid genetic and particle swarm algorithm for service composition. In *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on*, pages 564–567. IEEE.

[Liu 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.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73. ACM.

[Long and Gui, 2009] Long, J. and Gui, W. (2009). An environment-aware particle swarm optimization algorithm for services composition. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4. IEEE.

[Ludwig, 2012] Ludwig, S. A. (2012). Applying particle swarm optimization to quality-of-service-driven web service composition. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 613–620. IEEE.

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

[Martin et al., 2004] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., et al. (2004). Bringing semantics to web services : The owl-s approach. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 26–42. Springer.

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

[Menascé, 2002] Menascé, D. A. (2002). Qos issues in web services. *IEEE internet computing*, 6(6) :72–75.

[Merzoug et al., 2014] Merzoug, M., Chikh, M. A., and Hadjila, F. (2014). Qos-aware web service selection based on harmony search. In *ISKO-Maghreb : Concepts*

*and Tools for knowledge Management (ISKO-Maghreb), 2014 4th International Symposium*, pages 1–6. IEEE.

[Ming and Zhen-wu, 2007] Ming, C. and Zhen-wu, W. (2007). An approach for web services composition based on qos and discrete particle swarm optimization. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*.

[Mirjalili, 2015] Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software*, 83 :80–98.

[Mirjalili et al., 2014] Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69 :46–61.

[Moghaddam and Davis, 2014] Moghaddam, M. and Davis, J. G. (2014). Service selection in web service composition : A comparative review of existing approaches. In *Web Services Foundations*, pages 321–346. Springer.

[Moustafa and Zhang, 2013] Moustafa, A. and Zhang, M. (2013). Multi-objective service composition using reinforcement learning. In *International Conference on Service-Oriented Computing*, pages 298–312. Springer.

[Nacer et al., 2015] Nacer, A. A., Bessai, K., Youcef, S., and Godart, C. (2015). A multi-criteria based approach for web service selection using quality of service (qos). In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 570–577. IEEE.

[Nam et al., 2009] Nam, W., Kil, H., and Lee, J. (2009). Qos-driven web service composition using learning-based depth first search. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 507–510. IEEE.

[Osman and Laporte, 1996] Osman, I. H. and Laporte, G. (1996). Metaheuristics : A bibliography. *Annals of Operations research*, 63(5) :511–623.

[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.

[Papadias et al., 2005] Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2005). Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)*, 30(1) :41–82.

[Papadimitriou and Steiglitz, 1982] Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial optimization : algorithms and complexity*. Courier Corporation.

[Parejo et al., 2008] Parejo, J. A., Fernandez, P., and Cortés, A. R. (2008). Qos-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, 2(1) :55–66.

[Parejo et al., 2014] Parejo, J. A., Segura, S., Fernandez, P., and Ruiz-Cortés, A. (2014). Qos-aware web services composition using grasp with path relinking. *Expert Systems with Applications*, 41(9) :4211–4223.

[Parra-Hernandez and Dimopoulos, 2005] Parra-Hernandez, R. and Dimopoulos, N. J. (2005). A new heuristic for solving the multichoice multidimensional knapsack problem. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 35(5) :708–717.

[Pei et al., 2007] Pei, J., Jiang, B., Lin, X., and Yuan, Y. (2007). Probabilistic skylines on uncertain data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 15–26. VLDB Endowment.

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

[Pop et al., 2009] Pop, C. B., Chifu, V. R., Salomie, I., and Dinsoreanu, M. (2009). Optimal web service composition method based on an enhanced planning graph and using an immune-inspired algorithm. In *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pages 291–298. IEEE.

[Pop et al., 2010] Pop, C. B., Chifu, V. R., Salomie, I., Dinsoreanu, M., David, T., and Acretoaie, V. (2010). Ant-inspired technique for automatic web service composition and selection. In *SYNASC*, pages 449–455.

[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., Vlad, M., Chifu, V. R., Salomie, I., and Dinsoreanu, M. (2011b). A tabu search optimization approach for semantic web service composition. In *2011 10th International Symposium on Parallel and Distributed Computing*, pages 274–277. IEEE.

[Portal, 2009] Portal, O. (2009). Object management group : Common object request broker. *OMG. org. Retrieved from omg. org August*, 22 :2009.

[Protocol, 2003] Protocol, S. S. O. A. (2003). World wide web consortium http ://www. w3c. org. Technical report, TR/soap.

[Ran, 2003] Ran, S. (2003). A model for web services discovery with qos. *ACM Sigecom exchanges*, 4(1) :1–10.

[Rao and Su, 2004] Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer.

[Rodriguez-Mier et al., 2011] Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2011). Automatic web service composition with a heuristic-based search algorithm. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 81–88. IEEE.

[Rodriguez-Mier et al., 2010] Rodriguez-Mier, P., Mucientes, M., Lama, M., and Couto, M. I. (2010). Composition of web services through genetic programming. *Evolutionary Intelligence*, 3(3-4) :171–186.

[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.

[Rosenberg et al., 2010] Rosenberg, F., Müller, M. B., Leitner, P., Michlmayr, A., Bouguettaya, A., and Dustdar, S. (2010). Metaheuristic optimization of large-scale qos-aware service compositions. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 97–104. IEEE.

[Russell et al., 2003] Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2003). *Artificial intelligence : a modern approach*, volume 2. Prentice hall Upper Saddle River.

[Sadiq and Racca, 2003] Sadiq, W. and Racca, F. (2003). *Business services orchestration : The hypertier of information technology*. Cambridge University Press.

[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.

[Sen et al., 1988] Sen, T., Raiszadeh, F. M., and Dileepan, P. (1988). Note—a branch-and-bound approach to the bicriterion scheduling problem involving total flowtime and range of lateness. *Management Science*, 34(2) :254–260.

[Singh and Agrawal, 2015] Singh, D. and Agrawal, S. (2015). Hybridization of self organizing migrating algorithm with quadratic approximation and non uniform mutation for function optimization. In *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*, pages 373–387. Springer.

[Skoutas et al., 2009] Skoutas, D., Sacharidis, D., Simitsis, A., Kantere, V., and Sellis, T. (2009). Top-k dominant web services under multi-criteria matching. In *Proceedings of the 12th international conference on extending database technology : advances in database technology*, pages 898–909. ACM.

[Skoutas et al., 2010] Skoutas, D., Sacharidis, D., Simitsis, A., and Sellis, T. (2010). Ranking and clustering web services using multicriteria dominance relationships. *IEEE Transactions on Services Computing*, 3(3) :163–177.

[Soliman et al., 2008] Soliman, M. A., Ilyas, I. F., and Chang, K. C.-C. (2008). Probabilistic top-k and ranking-aggregate queries. *ACM Transactions on Database Systems (TODS)*, 33(3) :13.

[Stewart and White III, 1991] Stewart, B. S. and White III, C. C. (1991). Multiobjective a. *Journal of the ACM (JACM)*, 38(4) :775–814.

[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.

[Tan et al., 2001] Tan, K.-L., Eng, P.-K., Ooi, B. C., et al. (2001). Efficient progressive skyline computation. In *VLDB*, volume 1, pages 301–310.

[Tang et al., 2011] Tang, C., Li, Q., Xiong, Y., Liu, A., and Wen, S. (2011). A dominance-based approach for selection in service composition. In *2011 IEEE 8th International Conference on e-Business Engineering*.

[Tao et al., 2006] Tao, Y., Xiao, X., and Pei, J. (2006). Subsky : Efficient computation of skylines in subspaces. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 65–65. IEEE.

[Tiakas et al., 2007] Tiakas, E., Papadopoulos, A. N., and Manolopoulos, Y. (2007). Top-k dominating queries.

[Vlachou and Vazirgiannis, 2010] Vlachou, A. and Vazirgiannis, M. (2010). Ranking the sky : Discovering the importance of skyline points through subspace dominance relationships. *Data & Knowledge Engineering*, 69(9) :943–964.

[Wang et al., 2013a] Wang, A., Ma, H., and Zhang, M. (2013a). Genetic programming with greedy search for web service composition. In *International Conference on Database and Expert Systems Applications*, pages 9–17. 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 et al., 2013b] Wang, X., Wang, Z., and Xu, X. (2013b). An improved artificial bee colony approach to qos-aware service selection. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 395–402. IEEE.

[Wu et al., 2012] Wu, J., Chen, L., Xie, Y., and Zheng, Z. (2012). Titan : a system for effective web service discovery. In *Proceedings of the 21st International Conference on World Wide Web*, pages 441–444. ACM.

[Xia et al., 2011] Xia, Y., Chen, P., Bao, L., Wang, M., and Yang, J. (2011). A qos-aware web service selection algorithm based on clustering. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 428–435. IEEE.

[Xia et al., 2008] Xia, Y.-m., Chen, J.-l., and Meng, X.-w. (2008). On the dynamic ant colony algorithm optimization based on multi-pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 630–635. IEEE.

[Xiangbing et al., 2012] Xiangbing, Z., Hongjiang, M., and Fang, M. (2012). An optimal approach to the qos-based wsmo web service composition using genetic algorithm. In *International Conference on Service-Oriented Computing*, pages 127–139. Springer.

[Xu and Reiff-Marganiec, 2008] Xu, J. and Reiff-Marganiec, S. (2008). Towards heuristic web services composition using immune algorithm. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 238–245. IEEE.

[Yang, 2009] Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms*, pages 169–178. Springer.

[Yang, 2010] Yang, X.-S. (2010). Firefly algorithm, levy flights and global optimization. In *Research and development in intelligent systems XXVI*, pages 209–218. Springer.

[Yang and Deb, 2009] Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE.

[Yang et al., 2010] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. *Journal of Computational Information Systems*, 6(8) :2617–2622.

[Yeniay, 2005] Yeniay, Ö. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1) :45–56.

[Yiu and Mamoulis, 2007] Yiu, M. L. and Mamoulis, N. (2007). Efficient processing of top-k dominating queries on multi-dimensional data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 483–494. VLDB Endowment.

[Yoon and Hwang, 1995] Yoon, K. P. and Hwang, C.-L. (1995). *Multiple attribute decision making : an introduction*, volume 104. Sage publications.

[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., 2008] Yu, Q., Liu, X., Bouguettaya, A., and Medjahed, B. (2008). Deploying and managing web services : issues, solutions, and directions. *The VLDB Journal—The International Journal on Very Large Data Bases*, 17(3) :537–572.

[Yu and Lin, 2005a] Yu, T. and Lin, K.-J. (2005a). Service selection algorithms for composing complex services with multiple qos constraints. In *International Conference on Service-Oriented Computing*, pages 130–143. Springer.

[Yu and Lin, 2005b] Yu, T. and Lin, K.-J. (2005b). Service selection algorithms for web services with end-to-end qos constraints. *Information systems and e-business management*, 3(2) :103–126.

[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., 2014] Yu, Y., Ma, H., and Zhang, M. (2014). A hybrid gp-tabu approach to qos-aware data intensive web service composition. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 106–118. Springer.

[Yunwu, 2009] Yunwu, W. (2009). Application of chaos ant colony algorithm in web service composition based on qos. In *Information Technology and Applications, 2009. IFITA'09. International Forum on*, volume 2, pages 225–227. IEEE.

[Zelinka, 2004] Zelinka, I. (2004). Soma—self-organizing migrating algorithm. In *New optimization techniques in engineering*, pages 167–217. Springer Berlin Heidelberg.

[Zeng et al., 2003] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421. ACM.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5) :311–327.

[Zhang et al., 2006] Zhang, C.-w., Su, S., and Chen, J.-L. (2006). Genetic algorithm on web services selection supporting qos. *Jisuanji Xuebao(Chinese Journal of Computers)*, 29(7) :1029–1037.

[Zhang et al., 2013] Zhang, S., Dou, W., and Chen, J. (2013). Selecting top-k composite web services using preference-aware dominance relationship. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 75–82. IEEE.

[Zhang et al., 2010] Zhang, W., Lin, X., Zhang, Y., Pei, J., and Wang, W. (2010). Threshold-based probabilistic top-k dominating queries. *The VLDB Journal—The International Journal on Very Large Data Bases*, 19(2) :283–305.

[Zhao et al., 2012a] Zhao, X., Huang, P., Liu, T., and Li, X. (2012a). A hybrid clonal selection algorithm for quality of service-aware web service selection problem. *Int J Innov Comput Inf Control*, 8(12) :8527–8544.

[Zhao et al., 2012b] Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., and Fan, Y. (2012b). An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing*, 12(8) :2208–2216.

[Zhao et al., 2014] Zhao, X., Wen, Z., and Li, X. (2014). Qos-aware web service selection with negative selection algorithm. *Knowledge and Information Systems*, 40(2) :349–373.

[Zhou et al., 2013] Zhou, X., Shen, J., and Li, Y. (2013). Immune based chaotic artificial bee colony multiobjective optimization algorithm. In *International Conference in Swarm Intelligence*, pages 387–395. Springer.

Ce document a été préparé à l'aide de l'éditeur Texmaker 4.4 : un logiciel
libre sous la licence open source GNU GPL, qui se présente sous la forme d'un
environnement de développement intégré pour le langage LaTeX sous Windows.
Il est basé sur les distributions TeXLive 2015

**Résumé**

Le marché de commercialisation des services web sur internet ne cesse d'augmenter, ce qui résulte en un nombre de plus en plus croissant de services offrant des fonctionnalités équivalentes. De ce fait, la sélection d'un service web approprié pour une tâche particulière est devenue un défi difficile pour l'utilisateur.

Nous nous intéressons, dans cette thèse, à la problématique de sélection des services web dans une composition sur la base des besoins non fonctionnels (QoS), communément connue sous le nom QoSWSC. La problématique QoSSWC est considérée comme un problème d'optimisation multi-objectif. Nous nous intéressons, plus particulièrement, dans nos travaux, à deux catégories d'approches d'optimisations, à savoir, la première qui se base sur les techniques de bases de données qui utilisent la dominance de Pareto et la deuxième qui repose sur les méta-heuristiques.

Nous proposons dans la première catégorie d'approches deux contributions qui reposent sur la fuzzification de la relation de dominance selon Pareto. La première permet la sélection des Top-k services web basée sur la dominance floue. La deuxième permet de calculer les services Skylines sur la base de la dominance floue.

Nous proposons, dans la deuxième catégorie, trois contributions. Dans la première contribution, nous adaptons la récente méta-heuristique SOMA (Self-Organizing Migrating Algorithm) et nous présentons l'algorithme QoS-SOMA qui est la version discrète de SOMA. Cette dernière est mieux adaptée au problème de sélection des services web composés. Nous proposons, dans la deuxième contribution, une amélioration de l'algorithme QoS-SOMA en utilisant la notion de dominance floue dans la phase de la sélection locale. Dans la troisième contribution, Nous proposons une approche de sélection des services web composés, basée sur la recherche Tabou et le calcul des Skylines. L'algorithme de Skyline est utilisé dans le but de réduire l'espace de recherche de la sélection.

Les résultats expérimentaux montrent que les méta-heuristiques SOMA et Tabou améliorées par la dominance floue et le Skyline sont plus prometteuses que les versions standards.

**Mots clés**: La sélection des services web, Skyline, Optimisation combinatoire, Dominance floue, Méta-heuristiques, Qualité de service, SOMA.

**Abstract**

The Web services market on the Internet does not stop growing, what results in a more and more increasing number of functionally similar services. Therefore, the selection of an appropriate web service for a particular task has become a difficult challenge for the user.

In this thesis, we address the composite web service selection problem based on QoS known as QoS-aware web service selection 'QoSWSC'. This problem is considered as a multiobjective optimization problem. We mainly focus our research on two categories for solving the QoSWSC problem: the first one is based on the databases techniques that use the Pareto dominance, and the second is based on the metaheuristics algorithms.

We propose in the first category, two major contributions that are based on the fuzzification of Pareto dominance to compare functionally similar services. The first contribution selects the top-k web services. The second one computes the service skylines.

We propose in the second category, three contributions. In the first one, we adapt a recent stochastic optimization algorithm called Self Organizing Migrating Algorithm (SOMA) and present QoS-SOMA algorithm which is the discrete version of SOMA. In the second contribution, we improve the current version by using the fuzzy dominance comparison in the step of local search. In the third contribution, we propose an approach that is based on Tabu search and the skyline. The skyline is used to reduce the search space of the QoSWSC problem.

The experimental results show that the hybridization of metaheuristics with the fuzzy domination and the skyline outperform the standard version.

**Key words**: Web service selection, Skyline, Combinatorial optimization, Fuzzy dominance, Metaheuristics, Quality of service, SOMA.

**ملخص**

ان تسويق خدمات الويب على الانترنت في زيادة مستمرة مما يؤدى الى تزايد عدد الخدمات التي تقدم نفس الوظيفة. ولذلك، فان إنتقاء خدمة الويب الملائمة لوظيفة معينة ،أصبح تحديا صعبا بالنسبة للمستخدم.

نتطرق في هذه الأطروحة إلى إشكالية اختيار خدمات الويب لتكوين خدمة ويب مركبة بناء على متطلبات الجودة غير الوظيفية، المعروفة باسم (QoSWSC). تنتمي الاشكالية إلى فئة مشاكل الأمثلة متعددة الأهداف (المعروف أيضا باسم البرمجة متعددة الاهداف او الأمثلة متعددة المعايير (Optimisation multi-objectifs).نتطرق في عملنا، بالأخص إلى فئتين لحل إشكاليتنا، الأولى تعتمد على تقنيات قواعد البيانات التي تستخدم هيمنة باريتو (Dominance de Pareto) والثانية على أساس خوارزميات الادلة العليا(Méta-heuristiques). نقترح فى الفئة الاولى مساهمتين **تعتمدان على علاقة** fuzzification لهيمنة باريتو. الاولى تسمح باختيار Top_k خدمات الويب. اما الثانية تمكن من حساب Skyline.

نقترح، في الفئة الثانية، ثلاث مساهمات. في الأولى ، نقوم بتعديل الخوارزمية الحديثة في مجال الادلة العليا المسماة SOMA (self-organizing migrating algorithm) ونقدم الخوارزمية QoS-SOMA **المناسبة لحل مشكلة** QoSWSC **نقترح في** المساهمة الثانية، تحسين الخوارزمية **بلستعمل**(FuzzyDominance). **في** المساهمة الثالثة، نقدم منهجية تعتمد على خوارزمية 'Tabu Search' **وحساب** Skyline. **تستخدم** Skyline من أجل تقليص منطقة البحث.

**كلمات البحث**: اختيار خدمات الويب، **حساب** Skyline ، البرمجة متعددة الاهداف المعايير، النوعية غير الوظيفية(QoS)، خوارزميات الادلة العليا (Méta-heuristiques) , SOMA