Évaluation de performances de solutions pour la découverte et la composition des services web

Walid Serrai

UNIVERSITÉ PARIS EST
UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

Thèse en cotutelle de doctorat

Pour l'obtention du grade de
DOCTEUR DE L'UNIVERSITÉ PARIS EST

Champ disciplinaire : Informatique

Présentée et soutenue publiquement par:
Walid Salah Eddine SERRAI

Évaluation de performances de solutions pour la découverte et la composition des services web

Thèse dirigée par Pr Lynda MOKDAD / Pr Abdelkrim ABDELLI

Soutenue le 22/01/2020

JURY :

Président :	Pr Abdelmadjid BOUKRA
Rapporteurs :	Pr Amar BALLA Pr Salima BENBERNOU
Examinateurs :	Pr Jalel BEN-OTHMAN Pr Mahmoud BOUFAIDA MC Nouria HARBI
Directeurs :	Pr Lynda MOKDAD Pr Abdelkrim ABDELLI

Thèse préparée au LACL
(Laboratoire d'Algorithmique, Complexité et Logique)
https://lacl.fr LACL

Faculté des Sciences et Technologie
61 avenue du Général De Gaulle
94 010 Créteil

Aussi préparée au LSI
(Laboratoire des Systèmes Informatiques)
www.lsi.usthb.dz LSI

Faculté Électronique et Informatique
BP 32 Bab Ezzouar
16 111 ALGER

Résumé

Évaluation de performances de solutions pour la découverte et la composition des services web.

Les systèmes logiciels accessibles via le web sont construits en utilisant des services web existants et distribués qui s'interagissent par envoi de messages. Le service web expose ses fonctionnalités à travers une interface décrite dans un format manipulable par ordinateur. Les autres systèmes interagissent, sans intervention humaine, avec le service web selon une procédure prescrite en utilisant les messages d'un protocole. Les services web peuvent être déployés sur des plateformes cloud. Ce type de déploiement occasionne un grand nombre de services à gérer au niveau de mêmes répertoires soulevant différents problèmes : Comment gérer efficacement ces services afin de faciliter leur découverte pour une éventuelle composition. En effet, étant donné un répertoire, comment définir une architecture voire une structure de données permettant d'optimiser la découverte des services, leur composition et leur gestion. La découverte de services consiste à trouver un ou plusieurs services satisfaisant les critères du client. La composition de services consiste quant à elle à trouver un nombre de services pouvant être exécutés selon un schéma et satisfaisant les contraintes du client. Comme le nombre de services augmente sans cesse, la demande pour la conception d'architectures permettant d'offrir non seulement un service de qualité mais aussi un temps de réponse rapide pour la découverte, la sélection et la composition, est de plus en plus intense. Ces architectures doivent de plus être facilement gérables et maintenables dans le temps. L'exploration de communautés et de structures d'index corrélée avec l'utilisation de mesures multi critères pourrait offrir une solution efficace à condition de bien choisir les structures de données, les types de mesures, et les techniques appropriées. Dans cette thèse, des solutions sont proposées pour la découverte, la sélection de services et leur composition de telle façon à optimiser la recherche en termes de temps de réponse et de pertinence des résultats. L'évaluation des performances des solutions proposées est conduite en utilisant des plateformes de simulations.

Mots-clés : Services web • sélection de services • découverte de service • services composites • qualité de services • simulation • index • méthodes multicritères.

Abstract

Solution Performance Evaluation for the Discovery and Composition of Web Services.

Software systems accessible via the web are built using existing and distributed web services that interact by sending messages. The web service displays its functionalities through an interface described in a computer-readable format. Other systems interact, without human intervention, with the web service according to a prescribed procedure using the messages of a protocol. Web services can be deployed on cloud platforms. This type of deployment causes a large number of services to be managed at the level of the same directories raising different problems: How to manage these services effectively to facilitate their discovery for a possible composition. Indeed, given a directory, how to define an architecture or even a data structure to optimize the discovery of services, their composition, and their management. Service discovery involves finding one or more services that meet the client's criteria. The service composition consists of finding many services that can be executed according to a scheme and that satisfy the client's constraints. As the number of services is constantly increasing, the demand for the design of architectures to provide not only quality service but also rapid response time for discovery, selection, and composition, is getting more intense. These architectures must also be easily manageable and maintainable over time. The exploration of communities and index structures correlated with the use of multi-criteria measures could offer an effective solution provided that the data structures, the types of measures, are chosen correctly, and the appropriate techniques. In this thesis, solutions are proposed for the discovery, the selection of services and their composition in such a way as to optimize the search in terms of response time and the relevance of the results. The performance evaluation of the proposed solutions is carried out using simulation platforms.

Keywords: *Web services • Service selection • Service discovery • Composite services • Quality of service • Simulation • Index • Multi-criteria methods.*

Remerciements

Je voudrais tout d'abord remercier profondément mes deux directeurs de thèse, les Pr Abdelli Abdelkrim et Pr Mokdad lynda, pour toute leur aide et surtout leur confiance. Je suis ravi d'avoir travaillé en leur compagnie car outre leur appui scientifique, je leur dois beaucoup de cette thèse : sujet, financement, encadrement, débats et coopération. Ils ont toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse.

La recherche m'a permis de rencontrer et travailler avec plusieurs personnes. À Créteil comme à Fontainebleau, j'ai découvert et pris gout à l'enseignement ; j'ai certes croisé beaucoup d'élèves, mais aussi rencontré des collègues avec qui l'organisation des cours est un plaisir.

Ma reconnaissance va à ceux qui ont plus particulièrement assuré le soutien affectif de ce travail doctoral : Ma famille et mes proches, qui m'ont soutenu et parfois supporté pendant ces années, se retrouvent lésés : car je ne pourrai jamais les remercier assez.

Table des matières

Table des figures

Liste des tableaux

1

Introduction

1 Contexte

Les services web sont des programmes informatiques qui sont généralement conçus pour effectuer une tâche spécifique à la demande. Le rôle d'un service web est de produire un élément livrable, par exemple, consulter un compte bancaire, réserver un hôtel etc., par le biais de requêtes ou de messages. Ils ont comme avantages d'avoir un couplage faible, d'être réutilisable dans de nombreuses applications différentes, d'avoir une architecture distribuée... . Un fournisseur de services publie ses services web dans un registre que le client (entreprise ou programme) consulte pour invoquer les services dont il a besoin. Son concept innovant a fait que plusieurs très grandes entreprises s'y sont intéressées tel que Amazon, Disney, Microsoft, SNCF, Air France,... . En conséquence de l'engouement suscité par les services web, un nombre toujours croissant de services web devient disponible sur des réseaux. La découverte et la sélection de services web deviennent des problématiques préoccupantes pour le client de services. En effet, comme il existe un grand nombre de services web proposant des fonctionnalités similaires, le choix du meilleur service devient une tâche ardue. Les paramètres non fonctionnels, en particulier la qualité de service (QdS), peuvent être vus comme une solution utilisée par le client de services pour affiner la sélection de son service web suivant ses besoins. Cependant, une question reste en suspens qui est comment comparer des services web quand les critères sont opposés. Les critères de QdS sont divisées en deux groupes opposés nommés critères positifs tels que le débit, la fiabilité, la disponibilité,... et critères négatifs tels que le coût, le temps de réponse, ... Les services web simples sont très souvent combinés, afin d'offrir de nouvelles fonctionnalités spécifiques que les services web simples ne pouvaient pas offrir. Les services résultants de ces combinaisons sont

appelés services web composites ou composés. Dans la littérature, le problème de sélection de services web composites pour un seul client est souvent vu comme un problème de sac à dos multidimensionnel qui est un problème NP difficile. De multiples solutions ont été proposées par la communauté de chercheurs dans le domaine des services web pour optimiser la résolution de ce problème. Cependant, d'autre types de sélection de services web composites n'ont pas été suffisamment étudiés comme par exemple la sélection d'un groupe de services web composites pour un groupe de clients qui peut être vu comme un problème de sac à dos multiple à choix multiple.

Aussi, les problèmes se posent ainsi : comment sélectionner le meilleur service web avec ou sans contrainte ? quelle méthode choisir pour avoir des résultats satisfaisants ? comment sélectionner un bon groupe de services web composites pour un groupe de clients dans un délai acceptable et avec des résultats satisfaisants ? aussi toujours dans l'optique de gagner du temps pour la sélection de services, comment accélérer la découverte des services web candidats ?

Cette thèse s'appuie sur l'étude des processus de découverte, sélection et composition, décrivant les différentes étapes par lesquelles un service web transite et aussi les différentes approches proposées dans la littérature. Plus spécifiquement, c'est le processus de sélection de services web qui fait l'objet d'une étude approfondie. Plusieurs solutions sont proposées :

- Une sélection basée sur les votes des méthodes multicritères les plus utilisées, permettant d'obtenir de bons résultats avec un temps de réponse rapide.
- Une sélection basée sur une nouvelle distance combinée avec des méthodes multicritères adaptées, dont l'avantage est de prendre en compte les contraintes du client, offrant ainsi au client les services candidats qui satisfont ses contraintes ou à défaut (aucun service ne satisfait ses contraintes) un classement des meilleurs services qui s'en rapprochent.
- Une sélection basée sur la théorie des jeux et les communautés dont l'idée est d'offrir un groupe de services web composites avec une fonctionnalité similaire mais avec des services web simples distincts, à un groupe de clients prenant en compte ces différentes contraintes dans un environnement dynamique ;
- Enfin un nouveau processus de découverte, basé sur des index à plusieurs niveaux, est proposé pour améliorer la rapidité du processus global (découverte et sélection de services).

Mais en sus des algorithmes concrets, il est utile de pouvoir valider les approches proposées par des simulations. En fin d'étude, le processus global est modélisé pour décrire les différents acteurs et les différentes opérations proposés dans cette thèse. Le but des travaux présentés dans cette thèse est donc de proposer un ensemble de méthodes de sélection et de découverte pour soumettre des solutions efficaces aux clients de services web, tout en réduisant la consommation du temps de traitement des opérations afin d'augmenter leur satisfaction. Les simulations fournies permettent à la fois d'évaluer et de valider ces approches. Ces nouvelles approches serviront de base, dans le futur, pour la conception de nouvelles approches ou méthodes toujours plus performantes.

2 Organisation de la thèse

Chapitre 2 : Technologie des services web Les services web reposent sur des architectures orientées service(AOS) et exposent leurs fonctionnalités à travers des interfaces décrites dans un format manipulable par ordinateur (spécifiquement WSDL décrit avec XML). Ces interfaces présentent quatre niveaux d'interopérabilité, à savoir, les signatures des messages, les protocoles d'interaction, la qualité des services et la sémantique. De cette manière, d'autres systèmes peuvent interagir avec ces services web sans nécessité d'intervention humaine. Ce chapitre présente plus en détails les principes de l'architecture des services web et du cycle de vie des services web ainsi que les acteurs, les actions et les standards qui sont en relation avec la technologie des services web puis introduit les problématiques principales liées à cette technologie : la sélection de services web, la découverte de services web et la composition de services web en forment les grands axes.

Chapitre 3 : Sélection de services web La sélection de services consiste à choisir les services web les plus appropriés parmi les services web découverts qui répondent le mieux aux exigences non fonctionnelles et les contraintes spécifiées par le client. Les besoins non fonctionnels des services web généralement utilisés par le client sont les critères de QdS. Les approches de sélection à base de QdS sont divisées en deux équipes : la sélection de services web simples et la sélection de services web composites. Dans ce chapitre sont proposées et développées les principales étapes d'une sélection de services web simples ou composites à base de QdS : traitement des contraintes fixées par un client, réduction de l'espace des services web candidats, traitement des préférences du client sur les critères QdS, normalisation des valeurs des services web candidats, classement des services web simples ou composites sont parmi les plus importantes problématiques et ont fait l'objet, dans le contexte des services web, de propositions spécifiques.

Chapitre 4 : Sélection de services web basée sur des méthodes AMCD Une méthode de sélection de service web consiste à sélectionner le meilleur service web parmi les services candidats (services retournés lors de l'étape de découverte) en tenant compte des préférences du client sur la qualité de service. Le processus de sélection peut être optimisé en combinant deux ou plusieurs techniques. Les méthodes multicritères sont les méthodes de sélection les plus utilisées dans le domaine des services web. Généralement, chaque méthode multicritères offre un classement différent des services web candidats. Se pose alors la question sur le choix de la méthode à utiliser. Une méthode de vote simple à mettre en œuvre est proposée dans ce chapitre pour choisir un compromis entre les différents classements des méthodes multicritères. Des simulations sont réalisées pour évaluer puis comparer le classement de chaque méthode multicritères avec le classement de compromis.

Chapitre 5 : Sélection de services web basée sur les contraintes de QdS Une deuxième façon de sélectionner le meilleur service web simple consiste à considérer les contraintes d'un client où le meilleur service répond aux attentes du client de sorte que chaque contrainte sur chaque critère est respectée. Il est possible dans certains cas qu'aucun

service web ne réponde aux contraintes du client. Afin de pouvoir proposer des services web au client dans ce cas précis, des approches à base de méthodes multicritères sont utilisées pour sélectionner le meilleur service web en fonction de la QdS des services web candidats et les contraintes du client. La distance entre la QdS de chaque service candidat et les contraintes du client est calculée. Des méthodes multicritères sont adaptées pour prendre en compte cette distance et ainsi sélectionner le meilleur service web. Mais il ne faut pas oublier que chaque méthode multicritères propose son propre classement des services web et comme il n'est pas possible de dire quelle est la meilleure méthode, l'approche à base de vote est utilisée par chaque méthode multicritères pour élire le meilleur service web à retourner au client. Des simulations sont réalisées pour évaluer puis comparer les approches utilisées pour mettre en avant l'efficacité de ce type d'approche pour le cas précis ou aucun service web ne répond aux contraintes du client.

Chapitre 6 : Sélection de services web composites pour un groupe de clients Plutôt que de se concentrer uniquement sur la sélection optimale de services web composites pour un seul client à la fois, il est possible d'envisager une approche cherchant à maximiser la satisfaction d'un groupe de clients pour la sélection d'un groupe similaire de services web composites. Le troisième type d'approche proposé pour la sélection de services web consiste à proposer un service web composite unique à chaque client en fonction de ses contraintes établies, de telle façon à ce que chaque client soit satisfait du service composite *unique* retourné, c'est-à-dire que les services proposés au groupe de client sont tous distincts en terme de services simples. La satisfaction du groupe de clients est définie à l'aide d'un score calculé à partir de la QdS des services web candidats et les contraintes clients. Les contraintes clients sont utilisées pour trouver les groupes de services composites qui satisfont le groupe de clients. La qualité de service est utilisée pour sélectionner le meilleur groupe parmi les groupes de service composites candidats. Le système évolue à travers le temps c'est-à-dire des services web peuvent être supprimés ou ajoutés à travers le temps en adéquation avec le monde réel. Des techniques à base de théorie des jeux sont utilisées pour proposer de meilleurs services composites dans le cas de nouveaux services ajoutés ou trouver des services de substitution dans le cas de services supprimés. Ce troisième type d'approche est évalué, puis comparé à la méthode optimale à travers des simulations. Cette comparaison permet d'établir, en fonction du critère à favoriser (solution optimale ou solution approchée, temps de réponse, espace mémoire…), quelle est la meilleure solution à déployer pour la sélection des web composites dans un environnement dynamique.

Chapitre 7 : Découverte de services web basée sur l'indexation Le dernier chapitre des contributions présente des nouvelles techniques d'indexation se basant sur une indexation à plusieurs niveaux. La représentation ne porte pas uniquement sur le processus d'indexation des paramètres d'entrées et sorties des services web, mais aussi sur la représentation du processus d'indexation des services web des services composites. Les techniques d'indexation de services web simples (respectivement services web composites) sont utilisées pour optimiser le temps de recherche de services dans le registre (respectivement le registre de composition). Ces techniques sont modélisées sous forme d'arbres à plusieurs niveaux qui tiennent compte du nombre de paramètres et du nombre de services ; les niveaux détaillés des index et les opérations sur ces mêmes index sont fournis et un exemple est développé pour chacun des deux types d'indexation (indexation des services web simples et indexation des services web

composites). Les approches d'indexation de services web simples sont évaluées puis comparées avec deux méthodes existantes dans le domaine des service web à travers plusieurs simulations. Ces simulations permettent d'établir, en fonction des performances recherchées (temps de traitement, ressources mémoires… etc) et en fonction du nombre de paramètres des services web, quelles sont les meilleures techniques à utiliser pour optimiser la découverte de services web dans le registre.

Chapitre 8 : Conclusion et Persepectives Une synthèse des principales idées de nos propositions est présentée. Nous reprenons certains objectifs dans le but de mettre en avant, les principales contributions, d'identifier les perspectives de ce travail.

2

Technologie des services web

Pour établir le contexte de cette thèse, nous fournissons une introduction aux services web. Ce chapitre décrit donc le fonctionnement basique des services web, leur architecture et fournit des exemples d'applications. Dans un second temps, ce sont certains des principaux problèmes soulevés par le déploiement des services web pour la composition de services dans leurs cycles de vie qui sont brièvement introduits, notamment les problèmes de découverte de services web, les problèmes de sélection de services web ou encore des problèmes de composition de services web. Finalement, nous terminons par une synthèse de ce chapitre.

1 Service web

1.1 Définition

La technologie des services web est un moyen rapide de distribution de l'information entre clients, fournisseurs et leurs différentes plates-formes. L'objectif des services web est d'obtenir une interopérabilité entre les applications à l'aide de standards du web tel que XML, HTTP, etc.[9]. Les solutions proposées par les services web, utilisent un modèle d'intégration faiblement couplé pour permettre une intégration flexible de systèmes hétérogènes dans divers domaines[115]. Les servies web décrivent un ensemble de protocoles standards pour faciliter le développement d'applications réparties très faiblement couplées sur un réseau[73]. L'intégration est le facteur essentiel qui favorise l'utilisation des services web. Selon la définition du consortium du World Wide Web (W3C) : "un service web est un composant logiciel conçu pour prendre en charge une interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format pouvant être traité par une machine (en particulier WSDL). Les requêtes et les réponses utilisées lors de l'interaction des systèmes avec le service web sont soumises à des standards et normalisées à chacun de leur échange" [114]. Les services web permettent aux applications développées dans différentes technologies de communiquer entre elles par le biais de différends standards (WSDL, HTTP, SOAP, XML, etc.) [20]. Les services web ne sont liés à aucun système d'exploitation ou langage de programmation[65]. C'est à dire qu'une application développée en java peut communiquer avec une application développée en C. La mise en œuvre des services web repose sur une architecture décentralisée appelée Architecture Orientée Services AOS (Service Oriented Architecture - SOA).

1.2 Les avantages des services web

Les services web présentent plusieurs avantages tels que :

- **La disponibilité d'un nombre de plus en plus grand de services** :

  Le nombre de services web disponible sur les réseaux est en constante d'augmentation et leur fonctionnalité est très variée, ce qui offre aux clients une multitude de solutions pour le développement de nouvelles applications personnalisées à leur besoin. De plus, en cas panne de service, il est facilement remplacé par un autre avec la même qualité de service sinon meilleure que celle du premier[55, 3].

- **L'utilisation de standards XML et d'une architecture qui adopte les avantages des paradigmes orientés objets et orientés composants** :

  Les standards XML liés aux services web tel que WSDL, SOAP, UDDI, HTTP, BPEL, etc. facilitent grandement l'utilisation et la compréhension des services web. L'utilisation des standards permet l'intégration de services de manière flexible (BPEL), en représentant chaque service et ses fonctionnalités sous la forme d'une description standardisée (WSDL), permettant à un service d'échanger des informations structurées tels que des messages, document, etc. (SOAP/HTTP), en organisant les services afin qu'ils puissent être invoqués, utilisés efficacement (UDDI) [24].

- **La capacité d'avoir un couplage faible, une indépendance par rapport aux plateformes et une facilité d'intégration et de réutilisation** :

  Un couplage faible et une indépendance par rapport aux plateformes signifie que la dépendance entre les services est très réduite et ceci permet une large capacité de réutilisation et d'intégration de ces services afin de proposer de nouveaux services web qui offrent de nouvelles fonctionnalités aux clients [63].

- **La réduction du temps du développement d'application** :

  Avant la technologie des services web, un client pour développer son logiciel d'entreprise avait deux solutions : la première solution lui permet d'effectuer une commande auprès d'une entreprise spécialisée, cependant elle occasionne des coûts et une durée de développement élevés ; la deuxième solution lui permet d'acheter un logiciel mais elle n'est pas forcement bien adaptée à son besoin. Maintenant avec la technologie des services web, le client sélectionne directement chaque fonctionnalité de son logiciel à travers le registre des services, ce qui lui permet de gagner du temps et de l'argent, en plus d'avoir un produit fini totalement adapté à ses besoins[63].

1.3 Domaine d'utilisation

Les services web sont devenus incontournables dans notre monde moderne axé sur les appareils. Les protocoles, interfaces et normes de communication qui ont évolué au cours des 20 dernières années nous permettent de développer des applications web pour tablettes, PC et mobiles, capables de commander des repas, de trouver le cadeau idéal rapidement, d'effectuer des opérations boursières de manière fiable et sécurisée. Nous présentons ci-dessous quelques exemples applications des services web parmi les nombreuses qui existent.

- **La grande distribution** : Les services web peuvent être utilisés pour générer un numéro de colis comme celui proposé par Colissimo [108].

- **La logistique** : Les services web peuvent être utilisés pour gérer la logistique d'une entreprise e-commerce comme celui proposé par Endurancelogistique [109].

- **Le e-commerce** : Les services web peuvent être utilisés pour la gestion des données commerciales d'une entreprise e-commerce comme celui proposé par Oxatis [**oxatis**], pour la gestion de panier (la réservation du stock, la valorisation du panier, toutes les adresses de livraison et de facturation, le choix de mode de livraison, le mode de paiement, la transformation du panier en commande...) comme celui proposé par SRD [113].

- **Les sites de réservations en ligne** : Les services web peuvent être utilisés pour réaliser des fonctionnalités avancées et automatiser le système de réservations comme celui proposé par Planyo [111].

- **Le domaine médical** : Les services web peuvent être utilisés pour la gestion d'un cabinet médical (gérer les plages horaires, consulter des dossiers médicaux, gérer des bilans d'activités,...) comme celui proposé par Ferkal et Chaibi [40].

  Amazon Web Services (AWS) est l'un des plus grands registres concrets de services web au monde. Il propose plus de 90 services, comprenant le calcul, le

stockage, le réseau, la base de données, l'analyse de données, la gestion de système, la gestion d'applications mobiles, des outils pour les développeurs et pour l'internet des objets,... [4]. Il a comme client plusieurs grandes entreprises tel que Disney, Expedia, la NFL, Whirlpool, Honeywell,...[22].

Le champ d'application des services web est très grand, et ne cesse de s'agrandir au fil du temps et des avancées technologiques.

1.4 Architecture des services web

L'architecture des services web est basée sur une Architecture Orientée Service (AOS). Selon la définition du consortium d'Organization for the Advancement of Structured Information Standards (OASIS) : "l'architecture orientée service est un paradigme pour l'organisation et l'utilisation de fonctionnalités distribuées pouvant être sous le contrôle de propriétaires de différents domaines" [79]. AOS a été conçue pour créer un environnement de composition et d'interopérabilité de services distribués. Ceci a comme avantage de créer des applications dynamiques [74]. Les services web, dans une architecture orientée service, doivent avoir comme propriétés d'être [12] :

- **Découvrable** : leur permettant d'être facilement trouvé par un client.
- **Adressable** : leur permettant d'exister dans un emplacement défini.
- **Dynamique** : leur permettant d'évoluer avec le temps.
- **Faiblement couplé** : leur permettant de n'avoir presque aucune dépendance.
- **Interopérable** : leur permettant d'appartenir à des propriétaires divers.
- **Composables** : leur permettant de proposer de nouvelles fonctionnalités.

Il existe trois principaux acteurs dans une architecture orientée service : le fournisseur, l'annuaire et le client[79].

Un fournisseur de services développe et héberge des services web. Le fournisseur décrit et publie ses services dans des annuaires. Un annuaire fournit des informations sur les services web et leurs fournisseurs; On peut classer ces informations en quatre catégories : les informations générales relatives à la nature du service (nom, description), les informations techniques relatives à la façon de l'utiliser, les informations fonctionnelles qui décrivent les fonctionnalités du service et les informations non fonctionnelles qui concernent la qualité de service. Un annuaire est accessible via un réseau pour les clients. Un client cherche, découvre, sélectionne des services dans les annuaires. Un client communique avec le fournisseur pour invoquer et utiliser un service web.

1.4.1 ▶ Description d'un service web

La description d'un service web est une étape essentielle pour la découverte de services web. Une description précise la fonction d'un service web, ses contraintes de fonctionnement et la façon d'interagir avec lui. Il existe trois types de description de services web:

- **Description syntaxique:** le standard WSDL (Web Service Description Language) est un langage recommandé par la W3C basé sur XML généralement utilisé pour

la description syntaxique d'un service web. Selon la définition du consortium du W3C : "le langage WSDL fournit un modèle et un format XML pour décrire les services web. WSDL permet de séparer la description abstraite de la fonctionnalité offerte par un service de la description concrète des détails du service, telle que "comment" et "où" cette fonctionnalité est offerte" [117].

WSDL spécifie les fonctionnalités d'un service web en définissant des messages et des opérations. Les messages fournissent une définition abstraite des données qui doivent être échangées. Les opérations sont fournies par les services pour transformer les messages. Chaque message contient un ou plusieurs paramètres. WSDL permet une description centrée sur la fonction du service web qui est représentée par les paramètres en entrée et en sortie des opérations [70].

Un fichier WSDL contient au minimum 8 types de balise XML [116].

- La balise <types> définit tous les types de données utilisés pour décrire les messages échangés entre le client et le serveur. les types peuvent être prédéfinis tels que le type integer, string, boolean... ou plus complexes tels que le type structure, tableau, liste...
- La balise <message> définit de manière abstraite les données en cours de transmission (requête ou réponse).
- La balise <part> définit le nom et le type d'un paramètre de l'élément <message>.
- la balise <portType> définit d'une manière abstraite l'ensemble des opérations pouvant être effectuées par un service web.
- La balise <operation> décrit les données que le service s'attend à recevoir (avec la balise <input>) et/ou les données que le service prévoit d'envoyer en réponse (avec la balise <output>) ; dans certains cas, elle décrit aussi les messages d'erreurs qui peuvent être retournés par le service (avec la balise <fault>). Dans un document WSDL, il existe 4 types d'opérations :
  * Requête/Réponse: le client envoie la demande, et le service répond.
  * Sollicitation/Réponse : un service web envoie un message au client, et le client répond.
  * Sens unique : un client envoie un message au service web, mais ne s'attend à aucune réponse.
  * Notification : un service web envoie un message au client, mais n'attend pas de réponse.
- La balise <binding> (liaison) définit les protocoles de communication à utiliser pour l'appel des opérations du service. La balise peut contenir autant de descriptions de protocole que d'opérations, dans la mesure ou chaque opération peut utiliser un protocole de communication différent.
- La balise <service> décrit la collection des points d'accès au service web. Un service web peut avoir plusieurs différents points d'accès.
- La balise <port> décrit la localisation du service défini par une adresse réseau et une liaison.

- **Description sémantique**: les standards OWL-S (Ontology Web Language for Services)[57] et WSMO (Web Service Modeling Ontology))[110] sont généralement utilisés pour la description sémantique. Ils associent aux paramètres d'en-

trée/sortie des concepts ontologiques. L'objectif de la description sémantique est d'automatiser les activités du cycle de vie des services web.

- **Description hybride:** le standard SAWSDL (Semantic Annotation for WSDL)[88] et WSDL-S (WSDL-Semantic)[112] sont généralement utilisés pour une description hybride(syntaxique et sémantique). Ils enrichissent les descriptions syntaxiques des services web en utilisant des ontologies. Ces deux standards établissent une correspondance entre certains éléments WSDL existants et des concepts ontologiques. Les opérations et les messages peuvent être annotés dans WSDL-S. Seuls les paramètres peuvent être annotés dans SAWSDL.

1.4.2 ▶ Communication

Le standard SOAP (Simple Object Access Protocol) est un protocole recommandé par le W3C basé sur XML généralement utilisé pour assurer la communication entre machines. Selon la définition du W3C " SOAP est un protocole léger destiné à l'échange d'informations structurées dans un environnement décentralisé et distribué. Il utilise les technologies XML pour définir une infrastructure de messagerie extensible fournissant une structure de message pouvant être échangée via divers protocoles sous-jacents. Le cadre a été conçu pour être indépendant de tout modèle de programmation particulier et de toute autre sémantique spécifique à la mise en œuvre"[101]. C'est à dire que SOAP est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité (client, fournisseur, annuaire). Il peut s'appuyer sur n'importe quel protocole de communication (HTTP, SMTP, FTP ...) pour transmettre les messages. SOAP a pour avantage d'être aisément porté sur toutes les plates-formes et les technologies existantes.

Un fichier SOAP contient au minimum deux types de balise XML [100].

- La balise obligatoire <SOAP-ENV :Envelope> contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Elle indique le début et la fin du message afin que le destinataire sache quand un message est entièrement reçu.
- L'attribut optionnel encodingStyle décrit une URL qui définit les types applicables au message SOAP. Les types sont divisés en deux catégories : les types scalaires et les types composés.
  - Les types scalaires contiennent exactement une valeur telle que le nom d'un étudiant, la note ou son année d'inscription. Ils peuvent avoir des types basiques tels que entier, réel, booléen, chaine de caractère...etc.
  - Les types composés contiennent plusieurs valeurs telles qu'un relevé de notes d'un étudiant ou une liste d'étudiants. Ils peuvent être divisés en tableaux et en structures.
- La balise optionnelle <header> contient des informations d'en-têtes sur le message. Elle peut aussi permettre de spécifier des exigences supplémentaires au niveau de l'application telles qu'une signature numérique pour les services protégés par mot de passe.
- La balise obligatoire <body> contient les données XML définies par la description d'un service et échangées dans le message SOAP. Elle doit suivre toutes les balises <header> pouvant être définies pour le message. Elle contient des informations obligatoires destinées au destinataire final du message.

- La balise optionnelle <fault> rapporte les erreurs lors du traitement du message, ou lors de son transport.

1.4.3 ▶ Stockage

UDDI (Universal description, discovery, and integration) est un standard basé sur XML utilisé pour décrire, publier et rechercher des informations sur des services web[67]. Selon la définition fournie par le consortium OASIS "UDDI définit une méthode standard permettant aux entreprises de découvrir et d'invoquer de manière dynamique des services web"[68]. Trois types d'éléments d'informations (White paper – pages blanches, Yellow paper – pages jaunes, Green paper – pages vertes) peuvent être consultés pour rechercher des services web dans un registre UDDI[98].

- Les pages blanches contiennent des informations basiques sur l'entreprise telles que son identifiant unique son nom, son adresse, numéro de téléphone, son activité, etc. Ces informations permettent aux clients de découvrir le service web de l'entreprise en fonction de son identification d'entreprise.
- Les pages jaunes contiennent plus de détails non techniques sur les services web offerts par l'entreprise. Elles décrivent la qualité de service des services web. Elles utilisent des schémas de classification industrielle standards tels que des codes du secteur d'activités, des codes de produits, etc., afin de permettre aux entreprises du même secteur d'activité de rechercher plus facilement dans l'annuaire et trouver précisément ce qu'elles veulent.
- Les pages vertes contiennent des informations techniques sur un service web. Une page verte permet à une entreprise de consulter la description d'un service web (fichier WSDL) pour pouvoir exécuter un service web.

1.5 Qualité de Service (QdS)

Selon la définition de la norme ISO 9000 :2000, la qualité est l'aptitude d'un ensemble de caractéristiques intrinsèques à satisfaire des exigences. Selon RAN [78], la qualité de services (QdS) est un ensemble de paramètres non fonctionnels susceptibles d'avoir une incidence sur la qualité du service offert par un fournisseur. Selon Papazoglou[63], la qualité de services désigne la capacité d'un service web à répondre aux attentes mutuelles de son fournisseur et ses clients. Plusieurs critères de qualité influent sur la satisfaction du client, tels que la fiabilité et la disponibilité et sont devenus essentiels pour les fournisseurs de services pour mettre en avant la qualité de leurs services web offerts. La qualité de services devient alors un critère important afin de déterminer et sélectionner les meilleurs services pour accomplir une tache donnée. Les critères QdS peuvent être classés en cinq catégories[90].

- **Critères orientés exécution** :
  - Évolutivité : Aptitude d'un service web à traiter plus d'opérations ou de transactions sur une période donnée.
  - Capacité : Limite des requêtes simultanées d'un service web pour des performances garanties.
  - Performance : La rapidité avec laquelle une requête est traitée par un service web. On trouve dans ce critère plusieurs sous critères :

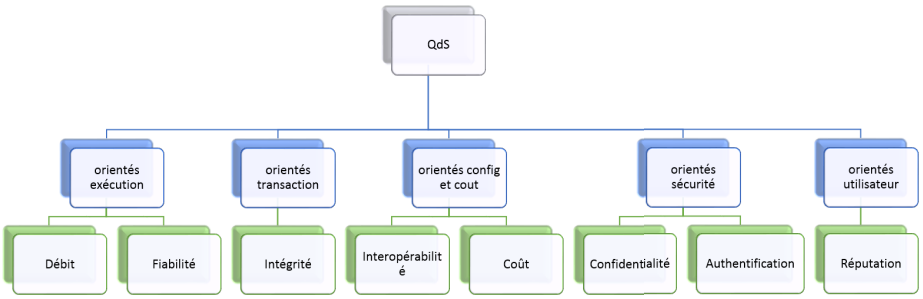

FIG. 2.1 : Qualité de service.

  * Temps de réponse : Le temps maximum garanti (ou moyen ou minimum) requis par un service web pour satisfaire une requête.
  * Latence : Le temps écoulé entre l'arrivée d'une requête et le retour d'une réponse.
  * Débit : Nombre de requêtes traitées par un service web sur une période donnée.

- Fiabilité : Aptitude du service web à accomplir une fonction requise, dans des conditions données d'utilisation et de maintenance, pendant une durée donnée.
- Disponibilité : La probabilité pour qu'un service web ne tombe pas en panne lors d'une requête.
- Robustesse / Flexibilité : C'est le degré auquel un service web peut fonctionner correctement dans le cas de paramètres d'entrées non valides, incomplets ou contradictoires.
- Traitement des exceptions : Aptitude d'un service web à gérer des exceptions.
- Précision : Définit le taux d'erreurs généré par un service web.

- **Critères orientés transaction** :

  - Intégrité : Détermine si les données n'ont pas été altérées durant une requête(de manière fortuite ou intentionnelle). Les transactions peuvent être regroupées dans une unité afin de garantir l'intégrité des données exploitées par ces transactions. L'unité peut être soit réussie où toutes les transactions sont dans l'unité « commit », soit échec de la transaction où toutes les transactions qui sont dans l'unité « rollback » sont remises à leur état initial.

- **Critères orientés configuration et coûts** :
  - Réglementation : Aptitude d'un service web à respecter la réglementation.
  - Norme prise en charge : Indique si un service web est conforme aux normes (par exemple, normes spécifiques au tourisme).
  - Interopérabilité : Aptitude d'un service d'interagir avec d'autres services en respectant des standards spécifiques.
  - Cycle de stabilité / changement : Mesure la fréquence d'un service web à changer son interface ou sa mise en œuvre.
  - Besoins de messagerie garantis : Aptitude d'un service web à garantir l'ordre et la persistance des messages.
  - Coût : Une mesure du prix impliqué dans la demande du service.
  - Complétude : Mesure la différence entre l'ensemble des fonctionnalités spécifiées et l'ensemble des fonctionnalités disponibles.
- **Critères orientés sécurité** :
  - Authentification : Comment un service web authentifie-t-il les administrateurs (clients ou autres services) ? qui peut accéder au service web et aux données ?
  - Autorisation : Comment un service web autorise-t-il les administrateurs afin que seuls eux puissent accéder aux services protégés ?
  - Confidentialité : Comment un service web traite-t-il les données, de sorte que seuls les administrateurs autorisés peuvent accéder ou modifier les données ?
  - Responsabilité : Le fournisseur peut-il être tenu responsable de ses services web?
  - Traçabilité : Est-il possible de retracer l'historique d'un service lorsqu'une demande a été traitée.
  - Chiffrement des données : Comment le service web chiffre-t-il les données ?
  - Non-répudiation : Comment le fournisseur de service web répond-il à ces exigences de sécurité ?
- **Critères orientés client** :
  - Réputation (confiance) : Mesure la satisfaction des clients pour un service web. Un exemple de mesure est "la moyenne des notes" donnée par des clients sur un intervalle $[0, 5]$. Ce critère peut refléter une variété d'aspects, tels que l'évaluation du vecteur qualité de service, l'évaluation de la qualité de soutien technique et l'évaluation subjective des résultats fournis,... . C'est le critère le plus important et le plus étudié dans la communauté de chercheurs dans le domaine des services web.

Remarque :

- Un critère est dit positif si sa valeur maximale représente sa meilleure valeur (le débit est un critère positif). Un critère est dit négatif si sa valeur minimale représente sa meilleure valeur (le temps de réponse est un critère négatif).

- Un critère est dit qualitatif s'il est décrit à l'aide d'une échelle ordinale consistant en un ensemble de valeurs qualitatives prédéfinies que le critère peut avoir (la réputation est un critère qualitatif). Un critère est dit quantitatif si sa valeur est mesurable ; sa valeur peut être continue ou discrète (le débit est un critère quantitatif).

2 Cycle de vie des services web dans une composition

2.1 Découverte de services

La découverte de services web est une étape essentielle. Avant toute interaction entre un client de services et un fournisseur de services, le client doit trouver le service web ou le processus métier qui répond le mieux à son besoin. Les recherches se sont principalement concentrées sur les approches de découverte de services pour une tâche unique [64]. Selon la définition du consortium du W3C : "la découverte de services web est l'acte de localiser une description de service web qui répond à certains critères fonctionnels. La découverte peut être (semi)automatique dans le cas d'un client de type machine, ou manuelle pour le cas d'un client de type humain. Il existe quatre types de support d'approches pour la découverte : approches à base d'annuaire (système centralisé géré par un administrateur qui contrôle l'ajout, mise à jour et suppression de service web), approches à base d'index (système décentralisé sans aucune autorité), approches à base de pair à pair (système dynamique) et approches à base de compromis (système hybride qui utilise les avantages des trois premiers systèmes selon le contexte)"[114].

2.2 Sélection de services

La sélection de services web est une autre étape essentielle. Les chercheurs divisent la sélection de services en deux catégories : la première se concentre uniquement à sélectionner le meilleur service web unique suivant les besoins et les contraintes client. La deuxième se concentre à sélectionner le meilleur service composite suivant les besoins et les contraintes du client.

Selon la définition de Moghaddam et Davis [64], la sélection de service est l'acte de sélectionner les services web les plus appropriés parmi les services web disponibles qui répondent le mieux aux exigences non fonctionnelles(QdS) et les contraintes spécifiées par le client.

2.3 Composition de services

La composition de services web est considérée comme l'une des principales motivations de l'utilisation des services web. Selon la définition de Martin et al [58], la composition de services est l'acte de sélectionner, combiner, exécuter et maintenir des services avec d'autre services en vue d'accomplir une tache complexe donnée. La combinaison de services web est appelé processus métier. Le standard WS-BPEL (Web Services Business Process Execution Language) est un langage recommandé par le OASIS basé sur XML généralement utilisé pour permettre au client de décrire les activités du processus métier en tant que services web et de définir comment ils

peuvent être connectés pour accomplir des tâches spécifiques[69]. Selon la définition du consortium OASIS " WS-BPEL définit un modèle et une grammaire décrivant le comportement d'un processus métier en fonction des interactions entre le processus et ses partenaires. L'interaction avec chaque partenaire s'effectue via les interfaces des services web et la structure de la relation au niveau de l'interface est encapsulée dans ce qu'on appelle un partnerLink. Le processus WS-BPEL définit la manière dont plusieurs interactions de service avec ces partenaires sont coordonnées pour atteindre un objectif commercial, ainsi que l'état et la logique nécessaire à cette coordination. WS-BPEL introduit également des mécanismes systématiques pour traiter les exceptions métier et traiter les erreurs. En outre, WS-BPEL introduit un mécanisme permettant de définir le mode de compensation des activités individuelles ou composites d'une unité de travail en cas d'exception ou de renversement de la demande d'un partenaire" [115].

2.4 Cycle de vie

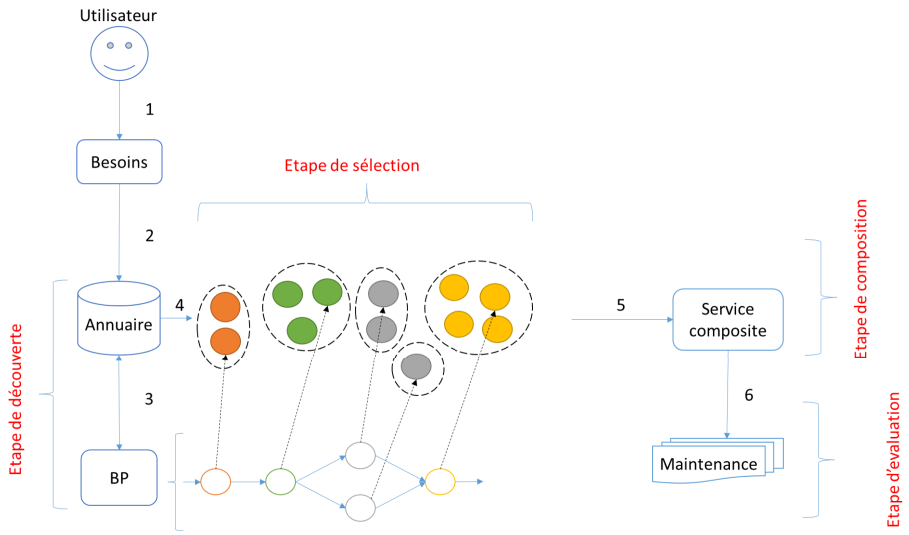

FIG. 2.2 : Cycle de vie des services web dans une composition.

Le cycle de vie d'une composition de services web est illustré par la figure 2.2[64]. Dans ce cycle de vie, la première étape est la spécification des besoins du client, dans laquelle les paramètres fonctionnels (fonctionnalités d'un service web composite) et non fonctionnels (QdS) sont définis. Suite à cela, les paramètres fonctionnels sont recherchés d'une manière (semi) automatique dans l'annuaire en utilisant des algorithmes de recherches[5, 65]. Comme résultat, nous aurons un processus métiers (Business process - BP) comprenant un ensemble de tâches abstraites. L'étape de découverte recherche une correspondance entre une tache abstraite et des services web concrets. L'étape de sélection utilise des algorithmes d'optimisation pour sélectionner la meilleure composition possible qui peut satisfaire le client [29]. Durant l'étape de composition, une instance du processus métiers est créée en exécutant le service composite. Enfin, la maintenance permet de surveiller continuellement l'instance du

processus pour résoudre des problèmes de changement de statut liés à des pannes ou à l'évolution d'un seul service web ou plusieurs services web[64].

3 Conclusion

Dans ce chapitre, nous avons présenté les principales spécifications des services web. Nous avons montré l'architecture des services, la description d'un service, la qualité de service. Nous avons présenté le cycle de vie des services web dans une composition. Nous avons montré aussi la découverte et la sélection de services web. Nous notons que la découverte et la sélection de services sont primordiaux pour la composition de services web. Le chapitre suivant présente plus en détails la sélection de services web.

3

Sélection de services web

De nos jours, en raison de la croissance rapide du nombre de services web proposant les mêmes fonctionnalités, la sélection des meilleurs services web devient une tache délicate. Par conséquent, pour proposer un service de qualité, les fournisseurs de services web se concentrent maintenant sur les aspects non fonctionnels des services web plus spécifiquement la Qualité de Service (QdS) . En effet, la solution basée sur la QdS des services web aide à promouvoir les meilleurs services web candidats parmi ceux qui répondent aux mêmes exigences fonctionnelles. Jusqu'à présent, diverses approches de sélection de services web basées sur la qualité de service ont été explorées dans la littérature. La sélection de services web simples et la sélection de services web composites sont les deux principales problématiques de la sélection de services web . Dans ce chapitre, nous présentons les principales approches de sélection de services web simples et composites proposées dans la littérature (voir Figure 3.1) et les principales approches de découverte de services à base d'index.

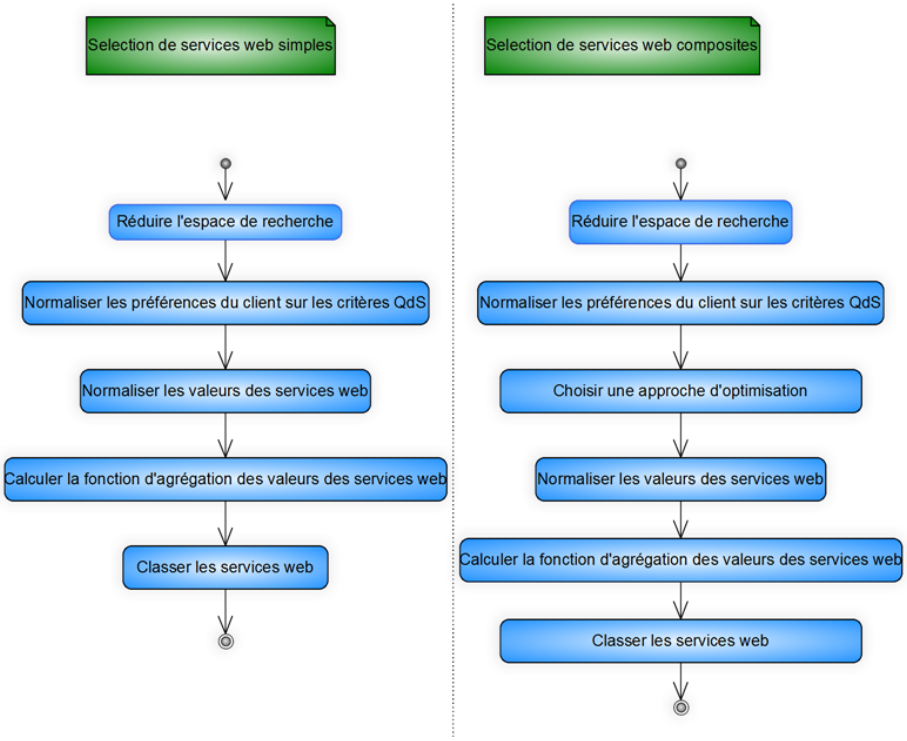

FIG. 3.1 : Étapes de sélection de services web.

1 Réduction de l'espace de recherche

1.1 Réduction manuelle de l'espace de recherche

1.1.1 ▶ Traitement des contraintes

Le traitement des contraintes représente la première étape dans le cas d'une sélection d'un seul service simple comme dans le cas d'une sélection d'un service composite.

Liu *et al* [50] et Raj *et al* [77] proposent deux modèles similaires de AOS pour prendre en compte les paramètres QdS appelés modèle exSOA pour le premier et QoSDB pour le second. Dans leur modèle, ils ajoutent un acteur appelé gestionnaire de services web . Son rôle est de gérer toute les opérations dans l'architecture (publier les services web des fournisseurs après vérification, rechercher les paramètres fonctionnels et non fonctionnels pour répondre aux requêtes clients ...). Son but est d'assurer des services web fiables aux clients. L'algorithme proposé par les auteurs élimine les services web qui ne répondent pas aux contraintes client puis classe les services web restants à l'aide d'une méthode de classement. Song *et al* [102] proposent un modèle de gestion des services web basé sur la QdS, appelé MQM (QoS-based multi-dimensionalmanagement model). L'approche est basée sur un modèle de

données multidimensionnelles afin de mieux gérer les services web. Les critères QdS sont extraits de l'annuaire UDDI et stockés dans le cube MOLAP (Multidimensional OnLine Analytical Processing). MOLAP signifie que les données QdS sont stockées dans une base de données multidimensionnelles appelé cube. L'objectif de ce modèle est la restitution instantanée des données aux requêtes OLAP du client.

1.1.2 ▸ ELECTRE

La méthode ELECTRE *(ELimination et Choix Traduisant la REalité)* est une méthode AMCD proposée par Roy dans [81]. ELECTRE est une méthode dite de surclassement qui effectue des comparaisons par paires des éléments en fonction de deux indices appelés l'indice de concordance et l'indice de discordance qui sont déterminés par le client. ELECTRE traite simultanément des critères qualitatifs et quantitatifs et permet de réduire l'espace de recherche lors de la sélection de services. Cependant, la méthode ELECTRE est une méthode de prise de décision plutôt complexe qui nécessite une variété de données primaires techniques à fournir par le client pour réduire l'espace de recherche.

Chakhar *et al* [16] proposent une extension d'ELECTRE, appelée ELECTRE-TRI, pour classer les services web composites lorsque les critères QdS prennent des valeurs qualitatives.

1.2 Réduction automatique de l'espace de recherche

1.2.1 ▸ Skyline

Skyline est une solution basée sur le front de Pareto introduite dans [99] pour la sélection de services web. Skyline permet d'extraire un groupe de services web qui dominent le reste des autres services web candidats(appelés dominés), quel que soit le profil d'exigence de QdS du client. Skyline bénéficie d'une très faible complexité ce qui favorise son utilisation en tant que solution de base pour diminuer le nombre de services web candidats avec un coût minime. Alrifai *et al* [1], Bouanaka *et al* [8] et Chen *et al* [19] utilisent la méthode Skyline pour sélectionner les services web dominant en fonction de leurs valeurs QdS dans le but de réduire l'espace de recherche des meilleurs services web composites. Ces auteurs utilisent ensuite différentes méthodes d'optimisation pour classer les services web composites.

L'utilisation de Skyline en amont réduit considérablement le temps d'exécution global comparé à l'application d'une technique d'aide multicritères à la décision (AMCD) directement sur tout l'espace des services candidats [72]. C'est la raison pour laquelle Skyline est principalement combinée avec d'autres approches AMCD. Étant donné le temps de traitement très rapide de Skyline, elle est utilisée pour réduire l'espace de recherche afin de ne garder que les services pertinents (éléments dominants) qui seront traités à une autre étape en utilisant une méthode AMCD plus précise.

2 Normalisation des préférences du client

Le client dans cette étape va exprimer ses préférences sur l'importance relative ou le poids des critères QdS. Ensuite les poids vont être calculés par une technique spécifique de telle sorte que la somme des poids des critères soit égale à 1. Les méthodes que

nous allons décrire ci-après utilisent le principe de comparaison par paires de critères pour calculer le poids des critères. En plus ces méthodes facilitent la décision du client dans ses choix de préférences en offrant une meilleure visibilité des critères traduite par une matrice de comparaison par paire et ainsi éviter que les choix de préférences du client ne soient contradictoires.

2.1 AHP and ANP

La méthode AHP *(Analytical Hierarchy Process)* est une méthode AMCD proposée par *Saaty* dans [86]. AHP effectue des comparaisons par paires basées sur une échelle de comparaison standardisée de neuf niveaux (1 à 9). La méthode ANP *(Analytical Network Process)* est une autre methode AMCD proposée par *Saaty* dans [85]. La principale différence entre AHP et ANP est que AHP utilise une hiérarchie lorsque ANP utilise un réseau. En d'autres termes, ANP est une forme généralisée d'AHP. AHP peut être utilisée dans des modèles hiérarchiques unidirectionnels sous forme d'arbre, tandis que ANP peut être utilisée pour des problèmes de décisions plus complexes principalement en raison de sa capacité à considérer des relations de facteurs interdépendants [27]. AHP organise les critères d'une manière structurée tout en donnant une solution relativement simple pour les problèmes de prise de décision. Elle permet de classer les critères d' une manière logique et hiérarchique en passant d'un niveau supérieur à un niveau inférieur jusqu'à parvenir à une comparaison simple pour chaque paire de critères, par la suite on peut remonter au niveau supérieur pour la prise de décision.

En effectuant des comparaisons par paires, les deux méthodes sont bien adaptées pour la normalisation du poids des critères. Elles sont jugées fiables pour assigner des poids représentant l'importance relative des critères suivant le jugement du client [124]. Cependant, les comparaisons par paires souffrent de trois insuffisances majeures. Tout d'abord, un très grand nombre de critères QdS pris en compte dans la formulation du problème de sélection ne permet pas au client de transmettre explicitement un sens à son choix, puisque les clients sont généralement invités à simplement comparer les critères QdS par paire. Deuxièmement, la complexité de la comparaison des éléments par paires peut être assez élevée pour un grand nombre de critères QdS , ce qui entraîne généralement des choix conflictuels. Le dernier et non des moindres, en raison de l'intervention fréquente du client, le traitement d'un grand nombre de critères QdS et de services web devient une tache ardue (risque d'incohérence accru) et couteuse en terme de temps. Par conséquent, elles ne peuvent être appliquées qu'à des problèmes simples [124].

Dans le contexte de la sélection des services web, AHP a été principalement utilisée pour l'attribution et la normalisation des poids des critères QdS sélectionnés par le client. Tran *et al*[104] utilisent AHP pour construire une structure hiérarchique qui permet de normaliser les critères. Les niveaux hiérarchiques de la structure contiennent : les objectifs de classement, les classes QdS, les critères QdS et les services web. Godse *et al* [26] utilisent ANP pour trier les services web. Pour ce faire, les critères QdS sont regroupés en trois catégories principales : critères orientés exécution, critères orientés sécurité et critères orientés configuration. Les poids sont calculés en fonction du réseau ANP pour les critères QdS et les scores finaux sont obtenus par la suite.

3 Normalisation des valeurs des services web

Les procédures de normalisation sont nécessaires lors de l'utilisation de méthodes AMCD pour adapter les valeurs QdS des services web[106]. La normalisation ou la standardisation des valeurs de performance des services web suivant une échelle entre 0 et 1 permet une distribution unifiée des valeurs des critères QdS afin d'éviter le biais dû à l'effet unité des différents critères QdS[76].

La normalisation joue un rôle crucial dans la majorité des techniques AMCD. Un grand nombre de méthodes de normalisation a été développé. Nous présentons dans la suite les plus populaires parmi celles utilisées dans les approches de sélection de services web. Définissons tout d'abord quelques paramètres.

Soit la matrice de décision de dimensions , qui indique valeurs de performance des services web qui sont évalués en fonction des critères de QdS

Soit la matrice de décision normalisée de dimensions , qui dénote les valeurs de performance normalisées .

Soit and respectivement les cotes de performance minimale et maximale pour le critère .

3.1 Normalisation basée sur la valeur maximale

Cette normalisation notée divise la valeur de performance par la valeur maximale de performance pour ce critère.

La normalisation basée sur la valeur maximale a été utilisée avec la méthode SAW et la méthode WPM. Son principal avantage est que si nous modifions les valeurs d'un service web ou si nous ajoutons ou supprimons des services web (sauf si c'est la valeur de performance maximale), nous n'avons pas besoin de recalculer la normalisation des services web déjà calculée.

3.2 Normalisation basée sur la somme des valeurs

Cette normalisation notée divise le valeur de performance par la somme des valeurs de performance pour ce critère.

La normalisation basée sur la somme des valeurs est utilisée avec AHP et ANP pour normaliser les poids des critères. Son principal inconvénient est que si nous mettons à jour les données d'un service web ou si nous ajoutons ou supprimons des services web alors il sera nécessaire de recalculer à nouveau la normalisation.

3.3 Normalisation basée sur la valeur maximale et la valeur minimale

Cette normalisation notée divise la distance entre la valeur de performance et la valeur de performance minimale pour ce critère par la distance entre la valeur de performance maximale et la valeur de performance minimale pour ce critère.

La normalisation basée sur la valeur maximale et la valeur minimale est l'une des techniques de normalisation les plus populaires. Elle est utilisée dans MAUT et SAW dans la composition du service [64], ainsi qu'avec VIKOR [71]. L'avantage de cette normalisation est qu'il n'est pas nécessaire de normaliser à nouveau les valeurs de performances déjà calculées lorsqu'un service web est mis à jour (ajouté/ supprimé) sauf dans le cas d'une mise à jour (ajout/suppression) des valeurs de performances maximales ou minimales.

3.4 Normalisation vectorielle

La normalisation vectorielle notée divise la racine de la valeur de performance par la racine de la somme des puissances des valeurs de performance pour ce critère.

La normalisation vectorielle est utilisée dans TOPSIS [34]. Son inconvénient est que cette approche est sensible à la mise à jour des services web (ajout/suppression).

3.5 Comparaison entre les techniques de normalisation

Nous présentons dans la table 3.1, une comparaison des différentes techniques de normalisation des valeurs de performance utilisées dans les approches AMCD dans le contexte de la sélection de services web. La normalisation basée sur la somme des valeurs est jusqu'à présent la technique la plus populaire quand il s'agit de normaliser les poids des critères de QdS, comme par exemple avec la méthode AHP. D'autre part, en raison de leur pertinence, la normalisation basée sur la valeur maximale et la valeur minimale et la normalisation vectorielle ont souvent été considérées dans la normalisation des valeurs de performances au niveau du classement des services web.

TAB. 3.1 : Comparaison des approches de normalisation pour la sélection de services web.

Méthode de normalisation	Formule		Utilisée dans	Intervalle
	critère positif	critère négatif		
Norm basée max			SAW	
Norm basée somme			AHP, ANP, SAW, COPRAS	
Norm basée max-min			SAW, VIKOR	
Norm vectorielle			TOPSIS, SAW	

La table 3.2 est une synthèse des méthodes de normalisation utilisées dans le processus de sélection des services web. L'intervalle de valeurs des services web candidats diffère suivant la technique de normalisation utilisée . Dans le cas de la normalisation basée sur la valeur maximale, la plage des valeurs s'étend sur l'intervalle . Pour

la normalisation basée sur la somme des valeurs et la normalisation vectorielle, elle est limitée à l'intervalle . Pour la normalisation basée sur la valeur maximale et la valeur minimale, elle couvre l'intervalle entier .

Vafaei *et al* [106] ont conclu que la combinaison de la normalisation basée sur la valeur maximale avec la normalisation basée sur la somme des valeurs est la technique de normalisation la plus appropriée à utiliser dans AHP. Chakraborty *et al* [17] ont conclu que la normalisation vectorielle est la plus appropriée pour SAW en la comparant avec quatre autres normalisations différentes. Les mêmes auteurs ont conclu que la normalisation vectorielle est la plus appropriée à utiliser avec TOPSIS [18].

4 Calcul de la fonction d'agrégation des valeurs des services web

Les fonctions d'agrégation permettent d'agréger et de modéliser les préférences du client en tenant compte des poids des critères QdS exprimés par le client. Le résultat est un classement des services web pour le client, selon ses préférences. Nous passons en revue ci-dessous les plus populaires parmi celles utilisées dans les approches de sélection de services web.

4.1 SAW

SAW *(Simple Additive Weight)* est une méthode AMCD introduite dans [123]. SAW, également appelée méthode des sommes pondérées, est l'une des techniques AMCD les plus simples et les plus utilisées dans la littérature. La méthode SAW utilise une fonction d'agrégation simple qui consiste à intégrer les valeurs de performances et les poids de critères dans une seule valeur. La fonction d'agrégation est utilisée uniquement pour les critères positifs. Par conséquent, les critères négatifs peuvent facilement être transformés en critères positifs lors de la normalisation des valeurs de performances (voir la section précédente). SAW possède une faible complexité et possède la capacité d'agréger les valeurs de performances en étant tout aussi intuitif pour les décideurs. Cependant, différentes expériences menées ont montré que la méthode peut fournir des résultats incohérents dans certains cas [107]. Dans [77], les auteurs ont discuté à travers des exemples des limites de la méthode SAW en termes de précision lorsque les performances des services web sont très proches.

SAW a été utilisée dans la sélection de services web dans de nombreux travaux. Cependant, le gain de popularité de certaines méthodes AMCD récentes comme TOPSIS et PROMETHEE a réduit son utilisation. Néanmoins, en raison de sa simplicité et de son efficacité, SAW continue d'être développée et utilisée dans la composition de services web. Par exemple, Jaeger *et al* [36], Lo *et al* [53] et Raj *et al* [77] utilisent SAW avec la normalisation basée sur la valeur maximale et la valeur minimale pour classer les services web en fonction des exigences QdS du client.

4.2 TOPSIS

TOPSIS *(The Technique for Order of Preference by Similarity to Ideal Solution)* est une méthode AMCD proposée par Huang *et al* dans [34]. Cette méthode est basée sur la distance euclidienne des solutions idéales et des solutions négatives-idéales à

un élément donné. Les valeurs des solutions idéales correspondent aux valeurs maximales des critères QdS; les valeurs des solutions négatives-idéales correspondent aux valeurs minimales des critères QdS (dans le cas de critères positifs et inversement dans le cas de critères négatifs). TOPSIS considère simultanément les distances des solutions idéales et négatives-idéales en prenant une proximité relative avec la solution idéale pour élaborer le classement final [6].

TOPSIS a l'avantage d'être peu complexe et facile à utiliser ; le nombre d'étapes reste le même quel que soit le nombre de services web. Cependant, l'utilisation de la distance euclidienne ne tient pas compte de la corrélation des critères; elle a aussi comme inconvénient de ne pas pondérer les critères QdS et à garantir la cohérence dans le processus de décision [107].

TOPSIS a été exploité dans de nombreux travaux pour sélectionner des services web. Zou *et al* [126] et Zhang *et al* [54] proposent une extension de l'algorithme TOPSIS pour classer les services web composites dans le but de supporter plusieurs décideurs (clients) à la fois, appelée permettant la spécification et l'agrégation des critères QdS suivant le type de valeurs des critères :

- variables discrètes : une valeur est discrète si elle ne prend que des valeurs isolées.
- variables continues : une valeur est continue si elle peut prendre toutes les valeurs comprises entre 2 nombres.
- nombre triangulaires flous : est un nombre flou à trois valeurs possibles qui a une fonction d'appartenance[2].
  - nombre flou : est une généralisation d'une valeur discrète. Il fait référence à un ensemble de valeurs discrètes possibles, où chaque valeur possible a son propre poids[2].

où les poids des critères QdS sont supposés être préalablement fixés et normalisés. Feng *et al*[25] proposent un modèle étendu de l'architecture AOS pour prendre en compte les QdS appelé modèle symétrique basé QdS. Dans son modèle, ils ajoutent un acteur appelé gestionnaire de la qualité de services. Son rôle est de vérifier la fiabilité de la qualité de service transmise par le fournisseur de services ou le retour d'information du client à propos d'un service. Un module ontologie représentant un ensemble de concepts dans le domaine QdS (cout, débit, réputation, critère, valeurvaleur , critère négatif , critère positif,...) est utilisé pour publier les informations QdS d'un service (éviter la redondance) et rechercher des critères qualité de service (correspondance QdS). Un algorithme étendu de TOPSIS est introduit pour prendre en compte des valeurs indéfinies représentées sous forme d'intervalle dans le but de sélectionner les meilleures services web composites parmi les services web composites candidats.

4.3 VIKOR

La méthode VIKOR *(le nom serbe est : VIse Kriterijumska Optimizacija kompromisno Resenje)* est une méthode AMCD proposée par Opricovi ć *et al* dans [71]. VIKOR fait référence à la solution de compromis ; elle a été développée pour résoudre des problèmes de décision multicritères avec des critères contradictoires et non-proportionnés. En supposant que le compromis peut être acceptable pour la résolution de conflit, VIKOR est approprié lorsque le décideur veut une solution réalisable qui soit la plus

proche de la solution idéale [28]. Cependant, dans VIKOR, l'évaluation des performances est quantifiée sous forme de valeurs discrètes. Dans de nombreuses circonstances, les valeurs discrètes sont inadéquates pour modéliser la situation réelle. En outre, en cas de conflit de critères, un décideur doit également prendre en compte des données imprécises ou ambiguës (flous, intervalle). A notre connaissance, VIKOR n'a jamais été utilisé seul pour sélectionner des services web, mais combiné avec d'autres méthodes AMCD pour plus d'efficacité dans la sélection de services web.

4.4 MAUT

MAUT *(Multi Attribute Utility Theory)* est une méthode AMCD proposée par Keeney *et al* dans [42]. Elle est basée sur les préférences du client pour déterminer diverses fonctions d'utilité spécifique pour chaque critère. La courbe d'utilité pour chaque critère est rendue disponible. Les valeurs des critères QdS peuvent être des valeurs catégorielles, ordinales ou de type intervalle. Chaque service web peut être évalué sur chaque critère. L'extraction des courbes d'utilité et leur addition permettent de dériver un score global pour sélectionner le meilleur service. MAUT est utilisé pour aider le client à mieux comprendre et analyser l'objectif du problème, les sous-objectifs, les poids et les scores. Cependant, la méthode est considérée comme une méthode plutôt complexe. La méthode SAW est considérée comme une simplification de MAUT pour résoudre des problèmes multi-critères. Seo *et al* [89] utilisent MAUT pour la sélection de services web composites.

4.5 PROMETHEE

La méthode PROMETHEE *(The Preference Ranking Organization METHod for Enrichment of Evaluations)* est une méthode AMCD proposée par Brans [11]. PROMETHEE est une méthode dite de surclassement qui effectue des comparaisons par paires afin de classer les services web par rapport à un certain nombre de critères. Elle permet une modélisation fonctionnelle des préférences en offrant un choix de six types de critères généralisés (fonctions d'utilité spécifique). PROMETHEE a l'avantage de traiter simultanément des critères qualitatifs et quantitatifs. Elle fournit un classement général des différents services web avec respectivement des flux de surclassement positifs (de combien un service surclasse tous les autres services) et négatifs (de combien un service est surclassé par tous les autres services). Cependant, il ne fournit pas de méthode spécifique selon laquelle les poids peuvent être déterminés. Ainsi, dans le cas d'un nombre important de critères (plus de sept), il devient très difficile pour le décideur de concevoir le problème et d'évaluer la pertinence des résultats obtenus. Herssens *et al* [32] sont les premiers à utiliser PROMETHEE pour la sélection de services web.

4.6 Composition de plusieurs méthodes AMCD dans la sélection de services web

Par souci de pertinence et d'efficacité dans le processus de sélection des services web, de nombreux travaux ont été explorés pour composer plusieurs méthodes AMCD entre elles ou avec d'autres techniques. Choi *et al* [21] ont converti les valeurs qualitatives en valeurs quantitatives, et les valeurs quantitatives avec différentes plages de valeurs sont converties en échelle standard. Ils proposent aussi un algorithme inspiré

de AHP pour fixer un poids prioritaire afin d'être applicable sur un système réel. Enfin, la méthode SAW est utilisée pour classer les services web. Seo *et al* [90] proposent une solution générique compatible avec toute les approches de sélection existantes. La méthode PROMETHEE est utilisée pour définir un système de contrainte de priorité globale plutôt que de fournir un classement final. En ce qui concerne la pondération des poids, une extension de la méthode Simos est proposée et appliquée. Jian *et al* [38], les auteurs ont proposé un modèle d'intervalle QdS pour mesurer les valeurs incertaines des critères QdS. Cette approche utilise une méthode floue, PROMETHEE et un algorithme génétique pour la sélection de services web composites. Zhang *et al* [125] proposent une approche appelée , basée sur une méthode floue et TOPSIS. Elle est appliquée pour évaluer les informations de QdS dans le cas de plusieurs registres de QdS afin de réduire la charge et le taux d'échec de l'utilisation d'un seul registre de QdS. La méthode TOPSIS combinée à une approche floue permet de classer les services web dont leur valeurs peuvent être exprimées en intervalle. Lo *et al* [52] appliquent des nombres flous triangulaires pour fixer le poids de chaque critère suivant les préférences du client. TOPSIS est ensuite exploité pour classer les services web. Raed *et al* [41] proposent de combiner deux méthodes AMCD, à savoir ANP avec PROMETHEE. ANP est utilisée pour calculer les poids des critères QdS, puis PROMETHEE est exécutée pour classer et trier les services web. Dans la même thématique, Negi *et al* [66] proposent d'utiliser AHP comme solution pour calculer les poids des différents critères QdS et TOPSIS pour classer les services web. Zhang *et al* [48] proposent un algorithme nommé qui combine une approche floue, AHP et TOPSIS. Ils convertissent les valeurs hybrides des services web en intervalles avec la méthode floue. Ensuite, ils utilisent AHP pour définir la matrice de comparaison par paire en fonction du profil de préférences du client. Enfin, ils classent les services web candidats pour la composition de services en utilisant TOPSIS. Setiawan *et al* [95] combinent les mêmes méthodes pour trouver les solutions optimales qui reflètent les besoins et les préférences du client. La méthode AHP floue est utilisée pour exprimer les besoins et les préférences du client tandis que TOPSIS est appliquée pour sélectionner la meilleure solution parmi les services web sémantiques. Maheswari *et al* [56] proposent un framework composé du client, du fournisseur de services, du registre et de l'algorithme TOPSIS floue. Un fournisseur de services en plus de publier le WSDL d'un service web, publie aussi le SLA (Service Level Agreements) du service web. SLA contient la description QdS d'un service web. L'étape de découverte de services est effectuée en convertissant le fichier WSDL en fichier OWL-S puis un algorithme de correspondance(matching) est utilisé pour trouver les services web candidats. La méthode TOPSIS floue est appliquée pour classer les services web candidats. un algorithme de réplication est fourni pour trouver des substituts au meilleur service web en cas de panne. Khezrian *et al* [44] proposent de combiner les méthodes AMCD : VIKOR et AHP. AHP est utilisée pour normaliser les préférences client sur les critères QdS. VIKOR est utilisée pour sélectionner le meilleur service web. Cependant, aucune simulation n'est fournie pour évaluer l'efficacité de ce système. Khezrian *et al*[43] proposent une approche basée sur la QdS et la méthode VIKOR. Dans leur approche, il existe deux acteurs importants que sont le client et l'expert. Le client peut exprimer ses préférences sur les critères QdS d'une manière linguistique pour ensuite être converties en poids normalisés ou il peut fournir directement les poids normalisés des critères QdS . De même, l'expert peut exprimer ses évaluations sur les valeurs des services web d'une manière linguistique pour ensuite être converties en valeurs normalisées ou il peut fournir directement les valeurs normalisées des services web. La méthode VIKOR est ensuite utilisée pour classer les services web. Des tests simples

ont été effectués en comparant leur approche avec TOPSIS floue [56].Plus récemment, Ouadah *et al* [72] proposent de combiner trois méthodes. Dans un premier temps, la méthode skyline est utilisée pour éliminer les services web non dominants, puis AHP pour attribuer des poids aux critères QdS et finalement la méthode PROMETHEE est utilisée pour classer les services skyline.

La table 3.2 est une synthèse des méthodes AMCD utilisées dans le processus de sélection des services web. En raison de leur avantage à classer les critères et à faciliter la prise de décision, AHP et ANP sont les approches les plus utilisées jusqu'ici dans la normalisation des poids des critères QdS. D'autre part, SAW, MAUT et TOPSIS ont été largement utilisés pour classer les services en raison de leur faible complexité, comparativement à ELECTRE et PROMETHEE.

TAB. 3.2 : Comparaison des approches à base de AMCD utilisées pour la sélection de services web.

Méthode AMCD	Complexité	Utilisée pour	Travaux de normalisation des poids	Travaux de classement des services
AHP		normalisation et classsement	[44, 66, 21, 95, 48, 104]	[104]
ANP		normalisation ET classement	[41, 26]	[26]
SAW		classement		[33, 21, 72, 77, 36]
TOPSIS		classement		[126, 66, 52, 125, 95, 48, 54, 25]
VIKOR		classement		[45, 43]
PROMETHEE		classement	[32]	[32, 41, 72]
MAUT		classement		[89]
ELECTRE		classement		[16]

: le nombre de services web. : le nombre de critères QdS.

5 Sélection de services web composite

La majorité des approches de sélection de services composites peut être classée en deux groupes suivant le profil du client appelés approches de sélection à base de négociation et approches de sélection à base d'optimisation.

5.1 Negociation

L'approche basée sur la négociation a été proposée pour répondre aux contraintes des clients en concevant un cadre de négociation. Ceci est réalisé en rendant les exigences de QdS des clients flexibles et négociables entre le client et le fournisseur de services via un contrat afin de parvenir à un accord [30]. Ce contrat est appelé SLA (*Service Level Agreement*) qui contient les termes de la négociation, tels que les attributs QdS, récompenses, pénalités, etc [64]. L'avantage de ce type d'approche est la

flexibilité du profil d'exigence du client pour arriver à un compromis.

Comuzzi *et al* [23] proposent une structure de courtier de négociation basée sur WS-Policy(recommandation du W3C[118]) pour effectuer une négociation partiellement ou entièrement automatisée des critères QdS entre le client de services et le fournisseur de services. Ils ont défini une extension de l'architecture AOS dans laquelle le fournisseur de services publie les informations sur QdS de son service en utilisant WSQL pour spécifier différentes contraintes QdS possibles. Patankar *et al* [75] proposent une approche de négociation automatisée basée sur le compromis pour l'achat de services web en utilisant un protocole bilatéral pour gérer les interactions entre les parties prenantes (client/fournisseur) à la négociation. Ils ont utilisé un mécanisme de compromis itératif pour évaluer les offres des adversaires (client/fournisseur) et générer des contre-offres de gain mutuel sur la base des critères QdS sélectionnés. Sathya *et al* [87] proposent un modèle de négociation égalitaire pour sélectionner le meilleur service qui répond aux exigences du client en utilisant le principe égalitaire selon lequel le fournisseur de services et le client de services reçoivent des récompenses égales.

L'inconvénient principal des approches de négociation est le nombre peu élevé de critères QdS pris en compte pour proposer une solution réalisable. Généralement, pas plus de deux, en raison de la complexité d'une telle approche [64].

5.2 Optimisation

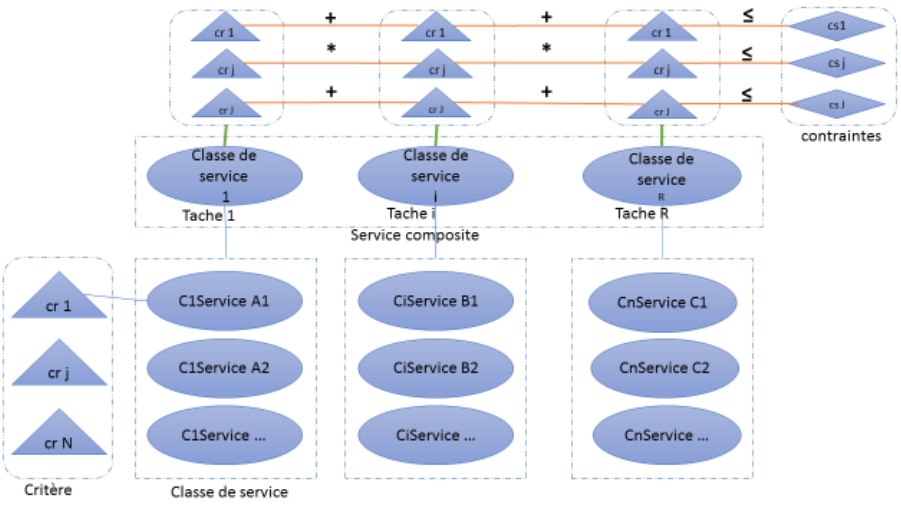

FIG. 3.2 : Approche de sélection de services web composites basée sur le problème MMKP.

L'approche la plus étudiée dans le domaine de l'optimisation de la sélection de services web composites décrit le problème comme un problème de sac à dos multidimensionnel à multiple choix (Multi-dimensional Multichoice Knapsack Problem (MMKP)) [64]. Ce problème recherche la composition qui a le plus grand profit et en même temps satisfait les contraintes QdS [97]. Dans la description du problème MMKP :

- désigne l'ensemble des services web.
- L'ensemble services web est divisé en classes (tâches (voir figure 3.2)).
- Chaque classe (tel que) contient services web candidats.
- Chaque service web est associé à critères.
- le client fixe contraintes (tel que).
- un service unique doit être sélectionné pour chaque classe .
- l'objectif est de maximiser le bénéfice total du service composite qui satisfait les besoins et contraintes du client[120].

MMKP est connu pour être NP-difficile, c'est pourquoi de nombreuses méthodes ont été proposées, Strunk [103] propose une classification des approches en 5 groupes : fonction objectif linéaire ou non linéaire, optimisation locale ou globale, supporte ou ne supporte pas les contraintes QdS, optimisation mono ou multi-objectif, solutions optimales ou sous-optimales (algorithme non heuristiques ou (méta) heuristiques respectivement dans [37]). Sheng et al. [96] ont ajouté un autre groupe, à savoir composition statique ou dynamique.

- **Fonction objectif linéaire ou non linéaire.** Une programmation linéaire tente de maximiser ou minimiser une fonction objectif linéaire en respectant les contraintes du client. Une programmation non linéaire tente de maximiser ou minimiser une fonction objectif non linéaire en respectant les contraintes du client[103].
- **Optimisation locale ou globale**. L'optimisation locale a pour objectif de choisir le meilleur service web pour chaque tâche individuelle en considérant des contraintes de QdS relatives à chaque tâche. L'optimisation globale a pour objectif de choisir le meilleur processus métier pour l'ensemble des tâches en considérant les contraintes QdS globales exprimées par le client[31].
- **Supporte ou ne supporte pas les contraintes QdS**. Une sélection qui ne supporte pas les contraintes QdS sélectionne le meilleur service composite parmi tous les services candidats. Une sélection qui ne supporte pas les contraintes QdS sélectionne le meilleur service composite en prenant en compte les contraintes fixées par le client.
- **Optimisation mono ou multi-objectif**. Une optimisation mono-objectif agrège les valeurs des critères QdS avec une seule fonction d'utilité puis compare les résultats pour sélectionner le meilleur service composite. Une optimisation multi-objectif associe à chaque critère QdS une fonction d'utilité puis compare les services web composites pour chaque critère QdS pour sélectionner un (ou plusieurs) bon(s) service(s) web composite(s)[39].
- **Solutions optimales ou sous-optimales**. Une sélection optimale permet de sélectionner le meilleur service composite possible sans contraintes de temps et d'espace. Une sélection sous-optimale permet de rechercher et sélectionner un service composite dans une partie réduite de l'espace de recherche des services candidats, en adoptant des algorithmes (méta)heuristiques. Une méta-heuristique est un algorithme d'optimisation générique permettant de donner des résultats quasi-optimaux, tout en ayant un temps d'exécution abordable[10].

- **Composition statique ou dynamique**. Dans un environnement dynamique, les fournisseurs de services peuvent apparaître et disparaitre à tout moment. Les fournisseurs de services peuvent aussi améliorer la qualité de leurs services à tout moment pour rester compétitif [51] ce qui permet d'améliorer la qualité et les performances des services composites d'une manière temporelle. Dans un environnement statique, les informations fournies par les fournisseurs de services sur leurs services web sont utilisées à un moment donné pour créer des services composites[96].

6 Indexation de services web pour la découverte de services

La fonctionnalité d'un service web est représentée par des entrées et des sorties telles qu'une fonction algorithmique qui nécessite des paramètres en entrées pour réaliser un algorithme qui ensuite retourne des résultats en sorties. Par exemple, la fonctionnalité d'un service web de réservation d'hôtel nécessite la date, le lieu et l'hôtel en entrées pour effectuer une recherche et retourner si disponible le prix du séjour et la date d'annulation de la réservation en sorties.

6.1 Index

Un index est une structure, entretenue automatiquement, qui permet de localiser facilement des services web dans un répertoire. L'utilisation des index est basée sur l'observation suivante : pour trouver un service dans un répertoire, au lieu d'examiner un à un chaque service, il est plus rapide de consulter le catalogue où ils sont classés par fonctionnalité(ou autre). Chaque entrée d'un index comporte une valeur extraite des données et un pointeur sur son emplacement d'origine. Un service peut être ainsi facilement retrouvé en recherchant sa localisation dans l'index.

6.1.1 ▶ Index inversé

L'index inversé dans son cas général consiste à représenter une structure complexe à l'aide de son contenu qui est généralement la cible de la recherche [35]. Dans le cas de la découverte de service web, la structure est le service web et le contenu recherché est la fonctionnalité de ce dernier. Donc cela consiste à représenter les services web par leurs fonctionnalités. L'index inversé est simple à mettre en œuvre et sa consommation ressources mémoires est raisonnable. Cependant, la complexité devient de plus en plus importante à mesure que le nombre de services web augmente. Ce qui augmente aussi la redondance. La méthode de l'index inversé a été utilisée dans [47] pour indexer les services web.

6.1.2 ▶ MLIM

MultiLevel Index Model (MLIM) est une méthode d'indexation proposée dans[119]. Cette méthode se présente comme un index à quatre niveaux. Dans le niveau 4, tous les services web ayant les mêmes entrées et sorties sont regroupés dans une même classe. Dans le niveau 3, toutes les classes du niveau 4 ayant les mêmes entrées sont

regroupées dans une même classe. Dans le niveau 2, Les classes du niveau 3 sont regroupées dans d'autres classes, en choisissant un de leurs paramètres d'entrées comme représentant. La stratégie du choix des représentants des classes de niveau 3 est laissée aux clients et aux spécialistes. Dans le niveau 1, les classes du niveau 2 sont indexées à l'aide d'une méthode d'indexation classique tels que la table de hachage, le B-Arbre,.... L'index MLIM est très flexible selon le contenu des services web. Il permet de jouer sur le ratio ressource mémoire/performance grâce à la suppression d'un niveau si le besoin se fait ressentir. Il supprime la redondance des paramètres d'entrées. Cependant, un mauvais choix de représentant peut se révéler très couteux en termes de performances de recherches. MLIM a été comparée théoriquement et pratiquement avec la méthode d'index inversé et la méthode itérative pour des répertoires de services web dans [119].

7 Conclusion

Nous avons présenté dans ce chapitre, la problématique de sélection de services web simples et composites à base de QdS. Nous avons aussi mis en évidence, un état de l'art illustrant les différentes étapes pour une sélection optimale. Plus précisément, nous avons étudié les étapes suivantes : la réduction de l'espace de recherche, la normalisation des préférences client sur les critères QdS, la normalisation des valeurs des services web, le calcul de la fonction d'agrégation des valeurs des services web, la sélection de services web composites. Dans le chapitre suivant, nous présentons notre approche optimale de sélection de services web simples.

4

Sélection de services web basée sur des méthodes AMCD

Lors de la sélection de services web, les approches basées sur des méthodes AMCD (Aide MultiCritères à la Décision) suivent généralement les étapes suivantes :

- Trouver les services web candidats qui répondent aux exigences fonctionnelles exprimées dans la requête client en utilisant des ontologies ou d'autres approches classiques (découverte de service).
- Traiter les exigences non fonctionnelles (paramètres Qualité de Service (QdS)) exprimées dans la requête client. Pour cela, les critères QdS doivent être normalisés. Une telle tâche peut être réalisée en utilisant une approche de normalisation telle que la normalisation de la somme des valeurs QdS, par exemple, ou une approche plus sophistiquée lorsque les paramètres QdS sont à la fois qualitatifs et quantitatifs avec des contraintes complexes.
- Normaliser les valeurs des services web. Une telle tâche peut être réalisée en utilisant une approche de normalisation telle que la normalisation basée sur la valeur maximale.
- Traiter le classement des services web candidats. Une fonction d'utilité multicritères est utilisée pour classer et trier les services web selon le schéma de pondération normalisé des poids des critères et les valeurs des services web.

Certaines méthodes AMCD comme AHP (Analytical Hierarchy Process) et ANP (Analytical Network Process) ont été principalement utilisées dans le processus de normalisation des poids des critères QdS (Qualité de service) [44, 66, 21, 95, 48, 104, 41, 26]. Néanmoins, elles peuvent également être utilisées pour classer les services web[104, 26]. D'autres approches telles que TOPSIS (The Technique for Order of Preference by Similarity to Ideal Solution) et VIKOR (VIse Kriterijumska Optimizacija kompromisno Resenje) sont utilisées pour le classement des services web [45, 43, 126, 66, 52, 125, 95, 48, 54, 25]. Ces dernières nécessitent que la normalisation du poids des critères QdS soit effectuée au préalable par une autre méthode AMCD. La présence de critères divergents rend le problème de sélection très complexe. La comparaison de ces approches montre que certaines d'entre elles prennent beaucoup de temps de traitement, comme par exemple PROMETHEE (The Preference Ranking Organization METHod for Enrichment of Evaluations), MAUT (Multi Attribute Utility Theory) et ELECTRE (ELimination et Choix Traduisant la REalité) pour la sélection du meilleur service [94, 72]. Ajoutons à cela que certaines techniques semblent retourner des résultats plus cohérents lorsqu'elles sont appliquées au stade de la normalisation des poids alors que d'autres donnent des résultats plus satisfaisants au stade du classement. La dernière problématique de la sélection de services web est que les classements obtenus en utilisant différentes méthodes AMCD peuvent être divergents même opposés. Cela pose la question de savoir quelle approche choisir ? et comment évaluer la pertinence et la précision de cette approche ? Récemment, de nouveaux travaux ont proposé de composer plusieurs techniques AMCD pour la sélection de services web. Cela a permis de fournir des résultats plus pertinents et satisfaisants au client tout en considérant des schémas de requêtes complexes [52] [125] [41] [38] [21] [72] [95] [48]. Cependant, l'évaluation et l'amélioration de la complexité de la sélection de services ainsi que la pertinence du classement obtenu sont toujours un problème d'actualité. Dans ce chapitre, nous proposons la combinaison de différentes méthodes AMCD à différends niveaux pour sélectionner des services web. Dans notre solution, différentes méthodes AMCD sont testées au niveau du classement et les temps de reponses sont mesurés afin d'évaluer la complexité de chaque approche. Pour évaluer la précision des différentes méthodes considérées, nous utilisons une méthode de vote appelée Borda [82] pour calculer une solution de compromis des différents classements des services web candidats obtenus. Enfin, les différents classements sont comparés avec la solution de compromis à l'aide de ratios.

1 Description de notre approche de sélection

Notre approche [94] consiste à combiner différentes méthodes d'aide multicritères à la décision (AMCD) que sont :

- Skyline [1] : L'approche Skyline est utilisée en premier lieu pour réduire le nombre de services web à prendre en compte lors de la sélection et cela en supprimant les services web à faible performance.

- BWM *(Best Worst Method)* [80] : Cette méthode est utilisée par le client pour exprimer ses préférences sur les critères QdS afin de calculer et de normaliser les poids des critères QdS [91].

- COPRAS (COmplex PRoportional ASsessment)[76], SAW (Simple Additive Weight), TOPSIS et VIKOR : Ces quatre méthodes AMCD sont appliquées séparément pour classer les services web dominants selon les poids fournis par l'approche BWM.
- L'approche Borda est utilisée pour proposer une solution de compromis à partir des classements obtenus par les quatre approches de classement. Comme chaque méthode AMCD propose un classement différent des services web dominants, la méthode Borda peut être considérée comme la solution pour proposer un classement final à partir des autres classements.
- Ratios de similarité : Deux ratios sont utilisés pour évaluer la précision de chaque méthode de classement en calculant le degré de similarité entre le classement final Borda et le classement de chacune des quatre méthodes AMCD.

La motivation de notre proposition est de proposer une solution de compromis avec un faible coût dû au fait que la sélection des services web doit généralement être effectuée en temps réel et que les résultats doivent être restitués avec un temps de réponse acceptable. La complexité de l'algorithme de sélection est principalement due à la complexité des méthodes utilisées lors de la phase de classement. Notre approche de sélection passe par cinq étapes. Chaque méthode AMCD est appropriée pour être utilisée à une étape spécifique des trois premières étapes. Pour les deux premières étapes, le choix de la méthode à utiliser est limité comme montré dans le chapitre 3, contrairement à la phase de classement où de nombreuses approches AMCD peuvent être envisagées. La figure 4.1 décrit les différentes étapes de notre approche qui sont :

1. **L'etape de filtrage** permet de réduire l'espace de recherche de l'ensemble des services web ayant les mêmes fonctionnalités sans avoir à prendre en compte le profil de QdS demandé par le client. Il a été prouvé dans [1] que la solution optimale est nécessairement dans l'espace Skyline réduit.
2. **L'étape de normalisation** permet de calculer et de normaliser les poids des critères QdS considérés dans la requête du client. Nous aurions pu choisir AHP plutôt que BWM, mais avec un risque plus accru d'incohérences pendant le processus de normalisation. BWM et AHP parviennent à calculer les mêmes poids lorsque le nombre de critères QdS n'est pas important. En plus, BWM est plus facile à mettre en œuvre et moins complexe que AHP au regard du client [80].
3. **L'étape de classement** permet de classer les services web. Étant donné qu'il n'y a pas de lien entre les méthodes AMCD utilisées lors de cette étape, nous choisissons de ne considérer que des méthodes (quatre méthodes : SAW, VIKOR, TOPSIS, COPRAS) ayant une complexité réduite. Dans la liste des méthodes AMCD discutées dans le chapitre 3, nous avons écarté PROMETHEE, ELECTRE et MAUT en raison de leur grande complexité. De plus, SAW peut être vue comme une simplification de la méthode MAUT.
4. **L'étape de compromis des classements** se concentre sur le calcul de la solution de compromis de Borda. L'utilisation de la solution Borda en tant que classement final devrait être préconisée tant que le temps de calcul du classement des services web reste acceptable. Les résultats de la simulation montrent que les méthodes de classement AMCD ont tendance à proposer des résultats différents lorsque le nombre de services web est augmenté.

5. **L'étape d'évaluation** permet d'évaluer la pertinence de chaque méthode en utilisant deux types de ratio de similarité. Un ratio de similarité compare chaque classement d'une méthode AMCD (**l'étape de classement**) avec le classement final Borda (**l'étape de compromis des classements**).

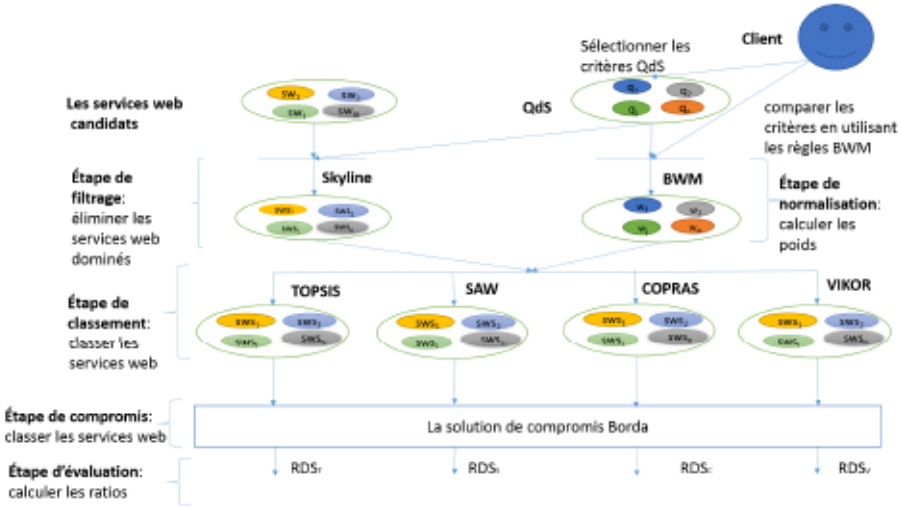

FIG. 4.1 : Description de notre approche de sélection.

1.1 Utilisation de Skyline pour réduire l'espace de recherche

L'opérateur de Skyline s'inspire du front de Pareto [49] qui est basé sur la relation de dominance, définie comme suit : **La relation de dominance** $(\prec)$: Étant donné un ensemble de services web avec des fonctionnalités similaires $SW = \{sw_1, sw_2, ..., sw_i, ..., sw_j, ..., sw_p\}$ et un ensemble de critères QdS $Q = \{q_1, q_2, ..., q_k, ..., q_l, ..., q_n\}$, on dit que sw_i domine sw_j, noté par $sw_j \prec sw_i$, si sw_i est meilleur ou égal à sw_j pour tous les attributs QdS, ou est strictement meilleur pour au moins un critère QdS.

Un critère est dit positif si sa valeur maximale représente sa meilleure valeur (le débit est un critère positif). Un critère est dit négatif si sa valeur minimale représente sa meilleure valeur (le temps de réponse est un critère négatif).

En termes mathématiques, un service web $sw_i \in SW$ domine un autre service web $sw_j \in SW$, si pour tous les attributs q_k dans Q, $k \in \{1, 2, ..., n\}$ critère

$$\begin{cases} q_k(sw_i) \geq q_k(sw_j) \text{ si } q_k \text{ est un critère positif} \\ q_k(sw_i) \leq q_k(sw_j) \text{ si } q_k \text{ est un critère négatif} \end{cases}$$

et pour au moins un attribut q_l of Q, $l \in \{1, 2, ..., n\}$

$$\begin{cases} q_l(sw_i) > q_l(sw_j) \text{ si } q_k \text{ est un critère positif} \\ q_l(sw_i) < q_l(sw_j) \text{ si } q_k \text{ est un critère négatif} \end{cases}$$

L'opérateur Skyline retourne tous les services web sw_i, de sorte que chaque service web sw_i ne soit pas dominé par d'autres services web sw_j.

1.2 Best Worst Method

La méthode BWM a été proposée par Rezaei [80]. BWM effectue des comparaisons de tous les critères QdS avec le meilleur critère et le pire critère prédéfinis par le client. BWM a deux avantages en comparaison avec AHP. Elle réduit le nombre de comparaisons par paires des critères QdS et elle réduit les chances d'avoir des incohérences. Les étapes procédurales de la méthode BWM sont décrites ci-dessous.

- **Étape 1** : Déterminer le pire et le meilleur critère QdS représentés respectivement par et .
- **Étape 2** : Évaluer la préférence du meilleur critère par rapport à tous les autres critères en utilisant une note entre 1 et 9. La table 4.1 donne un modèle de table de comparaison pour comparer les critères Qds avec le meilleur critère.

TAB. 4.1 : Meilleur critère.

Critères	

Où : est le coefficient résultant de la comparaison du meilleur critère avec un critère . est le coefficient comparant le meilleur critère avec le pire critère.

- **Étape 3** : Évaluer la préférence de tous les critères avec le pire critère en utilisant une note entre 1 et 9. La table 4.2 donne un modèle de table de comparaison pour comparer les critères Qds avec le pire critère.

TAB. 4.2 : Pire critère.

Critères	

Où est le coefficient résultant de la comparaison du pire critère avec un critère .

- **Étape 4** : Déterminer les poids optimaux , , .., , associés respectivement aux critères , , .., , .., , en résolvant le problème suivant :

—— *pour tout j*

—— *pour tout j*

pour tout j (4.1)

où est une variable dont la valeur est utilisée pour calculer le ratio de cohérence. est le poids du meilleur critère. est le poids du pire critère.

- **Étape 5** : Calculer le ratio de cohérence, où :

_____

où *InddeCoh* est l'indice de cohérence (max) déterminé selon la table établie par Rezaei[80] (voir la table 4.3) :

TAB. 4.3 : Indice de cohérence.

	1	2	3	4	5	6	7	8	9
InddeCoh	0	0.44	1	1.63	2.3	3	3.73	4.47	5.23

Le ratio de cohérence prend ses valeurs dans l'intervalle [0,1]. Plus le ratio est proche de zéro, plus la solution est cohérente [80].

1.3 Utilisation AMCD pour classer les services web

Quatre différentes méthodes AMCD sont utilisées pour classer les services web dominants résultant de la méthode Skyline en utilisant les poids des critères QdS fournis par l'approche BWM. Nous donnons ici une brève description du fonctionnement des quatre approches :

- **Étape 1** : Normaliser les valeurs des paramètres QdS des services web candidats afin que leurs nouvelles valeurs soit sur une même échèle d'unité entre 0 et 1. Chaque méthode AMCD peut utiliser une ou plusieurs méthodes parmi les méthodes de normalisation décrites dans le chapitre 3.

- **Étape 2** : Calculer le produit des valeurs des services web par les poids des différents critères afin de prendre en compte les préférences du client.

- **Étape 3** : Calculer la distance de chaque service web avec la distance optimale ou la distance idéale. Chaque méthode AMCD peut utiliser une ou plusieurs distances parmi les nombreuses distances proposées par la communauté de chercheurs telles que la distance euclidienne…

- **Étape 4** : Classer les services web dans un ordre croissant ou décroissant suivant la ou les distance(s) utilisée(s).

1.3.1 ▶ VIKOR :

VIKOR considère la matrice de décision suivante de dimension telle que est le nombre de services web non dominés obtenus par Skyline et est le nombre

de critères déjà considérés dans BWM et Skyline.

⋱

Le poids des différents critères obtenus par BWM est noté avec

VIKOR suit les étapes suivantes pour classer les services web [71] :

- **Étape 1** : Déterminer et respectivement la meilleure et la pire valeur pour chaque critère , .

  Si le critère représente un critère positif alors :

  Si le critère représente un critère négatif alors :

  avec et

  Les meilleures valeurs des critères QdS de la matrice seront stockées dans un vecteur de dimension . Les pires valeurs des critères QdS de la matrice seront stockées dans un vecteur de dimension .

- **Étape 2** : Normaliser les attributs des services web dominants avec la formule " Normalisation basée sur la valeur maximale et la valeur minimale " [71] :

  ——— (4.2)

  avec et Les résultats de la normalisation de la matrice seront stockés dans une matrice de dimension .

- **Étape 3** : Calculer le produit des attributs des services web dominants par les poids des attributs respectifs obtenus avec BWM :

  (4.3)

  avec et avec Les résultats du produit seront stockés dans une matrice de dimension .

- **Étape 4** : Calculer la distance Manhattan et la distance Chebyshev notées respectivement et , avec en utilisant les formules suivantes [71] :

  (4.4)

  max (4.5)

où est le poids du critère obtenu par la méthode BWM. Le résultat de la distance Manhattan sera stocké dans un vecteur de dimension . Le résultat de la distance Chebyshev sera stocké dans un vecteur de dimension .

- **Étape 5** : Déterminer les valeurs de , , et définies par : max min max min

- **Étape 6** : Calculer les valeurs , , en utilisant la formule suivante [71] :

(4.6)

où est défini comme le poids de la stratégie d'utilité maximale du groupe, alors que est le poids du regret individuel. La valeur de est considérée comme une valeur de compromis. Les résultats du score VIKOR des services web dominants seront stockés dans un vecteur de dimension .

- **Étape 7** : Classer les services web avec la formule dans un ordre croissant.

- **Étape 8** : Comme solution de compromis, déterminer le service web qui est le mieux classé par la mesure , de sorte que les deux conditions suivantes soient satisfaites [71] :

  – C1 (*Avantage Acceptable*) :

    où respectivement est le service web dans la première respectivement deuxième position dans la liste de classement de .

  – C2 (*Stabilité acceptable dans la prise de décision*) : Il doit également être le mieux classé par ou/et . Cette solution de compromis est stable dans un processus de prise de décision, qui pourrait être : *vote par règle de majorité* (quand est nécessaire), ou *par consensus* ( ), ou *avec veto* ( ). est le poids de la stratégie de prise de décision.

Si l'une des deux conditions précédentes n'est pas satisfaite, un ensemble de solutions de compromis est proposé, qui consiste en :

– Les services web et si seule la condition C2 n'est pas satisfaite, ou

– Les services web si la condition C1 n'est pas satisfaite ; et est déterminé par la relation

pour un maximum de (les scores de ces services web sont proches).

Le service web classé en premier par , est celui avec la valeur minimale de .

1.3.2 ▶ TOPSIS :

On utilise la matrice de décision de dimension comme définie dans VIKOR. Par conséquent, TOPSIS suit les étapes suivantes pour classer les services web [34] :

- **Étape 1**: Normaliser les attributs des services web dominants avec la formule " Normalisation vectorielle ".

(4.7)

avec Les résultats de la normalisation de la matrice seront stockés dans une matrice de dimension .

- **Étape 2** : Calculer le produit des attributs des services web dominants par les poids des attributs respectifs obtenus avec BWM :

(4.8)

avec et avec Les résultats des produits seront stockés dans une matrice de dimension .

- **Étape 3** : Déterminer les solutions positives idéales et les solutions négatives idéales , , définies par : Si la fonction représente un critère positif, alors

max

min

Si la fonction représente un critère négatif, alors

min

max

Les solutions positives idéales seront stockées dans un vecteur de dimension . Les solutions négatives idéales seront stockées dans un vecteur de dimension .

- **Étape 4** : Calculer la distance euclidienne aux valeurs idéales positives et aux valeurs idéales négatives des attributs parmi les services, , avec la relation : Distance euclidienne des solutions idéales positives :

(4.9)

Distance euclidienne des solutions idéales négatives :

(4.10)

Les résultats de la distance euclidienne des solutions idéales positives seront stockés dans un vecteur de dimension . Les résultats de la distance euclidienne des solutions idéales négatives seront stockés dans un vecteur de dimension .

- **Étape 5** : Calculer la proximité à la valeur positive des attributs des services web notée , , avec la relation : .

———— (4.11)

Les résultats du score TOPSIS des services web dominants seront stockés dans un vecteur de dimension .

- **Étape 6** : Classer les services web dans l'ordre décroissant en fonction des valeurs de .

1.3.3 ▶ COPRAS :

On considère la matrice de décision de dimension . COPRAS suit les étapes suivantes pour classer les services web dominants [76] :

- **Étape 1** : Normaliser les attributs des services web dominants avec la formule de " Normalisation basée sur la somme des valeurs ".

———— (4.12)

Les résultats de la normalisation de la matrice seront stockés dans une matrice de dimension .

- **Étape 2** : Calculer le produit des attributs des services web dominants par les poids des attributs respectifs obtenus avec BWM :

avec et

Les résultats du produit seront stockés dans une matrice de dimension

- **Étape 3** : Calculer le score positif (critères positifs) des services individuels et le score négatif (critères négatifs) des services individuels ,
.

(4.13)

(4.14)

Les scores positifs seront stockés dans un vecteur de dimension . Les scores négatifs seront stockés dans un vecteur de dimension .

- **Étape 4** : Calculer l'importance ou les priorités relatives des services web dominants :

  (4.15)

  les résultats de la distance relative seront stockés dans un vecteur de dimension .

- **Étape 5** : Calculer la valeur de l'indice de performance du service :

  (4.16)

  les résultats du score COPRAS des services web dominants seront stockés dans un vecteur de dimension .

- **Étape 6** : Classer les services web dans l'ordre décroissant en fonction des valeurs de .

1.3.4 ▶ SAW :

De même que les méthodes de classement précédemment utilisées, nous considérons la matrice de décision de dimension . SAW suit les étapes suivantes pour classer les services web [123] :

- **Étape 1** : Normaliser les attributs des services web dominants avec la formule " Normalisation basée sur la valeur maximale et la valeur minimale ":

  (4.17)

  Si le critère représente un critère positif.

  (4.18)

  Si le critère représente un critère négatif.

  avec et

  où et représente respectivement la valeur maximale et la valeur minimale de chaque critère ,

  Les résultats de la normalisation de la matrice seront stockés dans une matrice de dimension .

- **Étape 2** : Calculer le produit des attributs de services web dominants par les poids des attributs respectifs obtenus avec BWM :

  avec et Les résultats du produit seront stockés dans une matrice de dimension .

- **Étape 3** : Calculer le score individuel des services :

(4.19)

avec . Les résultats du score SAW des services web dominants seront stockés dans une vecteur de dimension .

- **Étape 4** : Classer les services web dans l'ordre décroissant en suivant les valeurs de .

1.4 Utilisation de Borda pour le classement final

Pour évaluer la pertinence et l'exactitude des quatre approches AMCD utilisées, nous appliquons d'abord la technique pour calculer la solution de compromis.
Calcul de la solution de compromis Borda : Supposons que les quatre classements, , , et ont été obtenus en appliquant respectivement les méthodes de classement VIKOR, COPRAS, TOPSIS, SAW sur la matrice de décision . donne le classement du service web obtenu en utilisant la méthode , avec et . La méthode est une méthode de vote basée sur le classement [82]. Elle suppose qu'il existe électeurs (votants) qui votent pour les services candidats . Comme nous avons quatre méthodes AMCD de classement (VIKOR, SAW, TOPSIS, COPRAS) alors , chaque électeur étant associé à une approche AMCD. Chaque service web reçoit des votes à différents niveaux de classement. Ensuite, le score Borda du service web est obtenu en agrégeant ses différents classements , comme suit :

- **Étape 1** : Calculer le score Borda en ajoutant le classement du service web correspondant à chacune des quatre méthodes AMCD.
  En d'autres termes avec un exemple, supposons qu'un service web noté "1" est classé à la troisième position avec la méthode VIKOR, deuxième position avec la méthode COPRAS, huitième position avec la méthode TOPSIS, première position avec la méthode SAW. Nous avons donc le score Borda
  .

- **Étape 2** : Classer les services dans l'ordre croissant en fonction du score Borda . En cas d'égalité dans le score, nous considérons le nombre de fois où le service web obtient un meilleur classement, et ainsi de suite. Le classement obtenu est une solution de compromis des quatre approches de classement.

1.5 Utilisation de ratios pour l'évaluation de la pertinence de chaque approche AMCD

En analysant le classement obtenu par chaque service dans chacune des quatre méthodes , on peut mesurer la similarité entre ce dernier et le compromis Borda . Dans cette optique, nous introduisons deux fonctions de similarité pour calculer les ratios de similarité entre les deux classements. Le premier ratio est appelé ratio de similarité dure qui est une comparaison stricte. Le second, appelé ratio de similarité doux, limite l'analyse de la similarité aux seuls dix meilleurs services web dans chaque classement.

1.5.1 ▶ a-Calcul du ratio de similarité dur :

- **Étape 1** : Calculer le score de similarité en comptant le nombre de fois où les deux classements se correspondent : où, si , autrement
- **Étape 2** : Calculer le ratio de similarité, noté , obtenu en divisant le coefficient par le nombre de services web candidats :

——

Cette fonction de similarité est stricte et ne permet pas de mesurer objectivement la pertinence de chaque méthode dans certains cas. Par exemple, un service web pour une méthode de classement donnée peut être classé avec un décalage d'une ou deux positions par rapport à sa position dans la solution Borda. Le ratio de similarité dur évaluera les résultats comme étant complètement différents de la solution de compromis alors qu'ils sont très proches.

1.5.2 ▶ b- Calcul du ratio de similarité doux :

Généralement, les méthodes de sélection fournissent à l'utilisateur un sous-ensemble d'éléments optimaux qui répondent aux besoins du client. Le client doit décider lesquels considérer. Pour faciliter le choix, un pourcentage sur le nombre de services web dans l'espace de recherche où un nombre fixe à l'avance peuvent être définis. Afin de fournir un meilleur outil pour évaluer la pertinence de chaque méthode de classement, le ratio doux de similarité , considère l'intersection des dix premiers services web dans chaque classement avec l'ensemble des dix premiers services web du classement Borda.

- **Étape 1** : Calculer le nombre de services web communs résultant de l'intersection : où, est un service web dont le classement avec la méthode est entre la première et dixième position. est une des méthodes de classement parmi les quatre méthodes utilisées (, , et). est un service web dont le classement avec la méthode est entre la première et dixième position.
- **Étape 2** : Calculer le ratio de similarité doux, noté , obtenu en divisant le coefficient par 10 : —— où, 10 représente le nombre des dix premiers services pris en considération pour chaque méthode .

En d'autres termes, le ratio doux de similarité permet de mesurer le nombre de services web qui sont communs à la fois à la solution Borda et à toute autre méthode de classement considérée indépendamment de leur position dans le classement.

2 Exemple d'illustration du modèle proposé

Pour illustrer l'application de notre approche dans le contexte des services web, nous considérons un exemple de services web offrant "les cours de la bourse" pris de la base de données réelle appelée QWS DataSet [59]. Comme le montre la table 4.4,

les services web sont caractérisés par un ensemble de quatre critères QdS, qui sont : **Temps de réponse** : C'est le temps écoulé entre l'envoi de la requête par le client et la réception de la réponse fournie par le service web (ms) ; c'est un critère négatif. **Débit** : Il représente le nombre d'invocations pendant une période donnée. L'unité de mesure est (invocations / seconde) ; c'est un critère positif. **Fiabilité** : Il mesure le ratio du nombre de messages d'erreur sur le nombre total de messages ; c'est un critère positif. **Meilleures pratiques** : Il mesure le nombre de points pratiques pour chaque service web suivant le profil WS-I [59] ; c'est un critère positif.

TAB. 4.4 : Un échantillon de services web de la base de données QWS.

Nom	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
XigniteStockQuotes	225.36	10.9	62.6	77
HistoricalStockQuotes	119.33	2.9	76.5	80
DelayedStockQuotes	123.5	8.1	78.6	80
QuoteofTheDay	631	10	77.1	84
XigniteQuotes	999.71	0.8	38.5	74
StockQuotes	409	8.2	67.6	84
StockQuotes	544	4.7	70.5	80
RealTimeQuotes	677.69	0.8	22.5	74
StockQuotes	281.07	2	8.7	82

2.1 Réduire l'espace de recherche en utilisant Skyline

En appliquant Skyline nous obtenons une liste limitée de services web non dominés (voir la table 4.5, où la marque Oui dénote un service web dominant et la marque Non dénote un service web dominé). Plus concrètement, selon l'algorithme Skyline, nous avons : Le service est dominé par les services , , , et . Le service est dominé par le service . Le service est dominé par les services , , , et . Ensuite, seulement les services web dominants (Skyline), à savoir , , , et seront considérés dans l'étape de classement.

TAB. 4.5 : Les services web dominants.

	Nom	Temps de réponse	Débit	Fiabilité	Meilleures pratiques	Est Skyline
	XigniteStockQuotes	225.36	10.9	62.6	77	OUI
	HistoricalStockQuotes	119.33	2.9	76.5	80	OUI
	DelayedStockQuotes	123.5	8.1	78.6	80	OUI
	QuoteofTheDay	631	10	77.1	84	OUI
	XigniteQuotes	999.71	0.8	38.5	74	NON
	StockQuotes	409	8.2	67.6	84	OUI
	StockQuotes	544	4.7	70.5	80	NON
	RealTimeQuotes	677.69	0.8	22.5	74	NON
	StockQuotes	281.07	2	8.7	82	OUI

2.2 Détermination et normalisation des poids des critères QdS

Nous appliquons la méthode BWM afin de déterminer et normaliser les poids des quatre paramètres QdS. - Nous considérons le paramètre *Temps de réponse* comme étant le meilleur critère. La table 4.6 illustre la comparaison du meilleur critère avec les autres paramètres QdS.

TAB. 4.6 : Comparaison du meilleur critère avec les autres critères en utilisant BWM.

	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
Temps de réponse	1	8	2	4

- Nous considérons le paramètre *Débit* comme étant le pire critère. table 4.7 illustre la comparaison du pire critère avec les autres paramètres QdS.

TAB. 4.7 : Comparaison du pire critère avec les autres critères en utilisant BWM.

	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
Débit	8	1	4	2

- Nous définissons , , et comme étant les poids des critères *Temps de réponse*, *Débit*, *Fiabilité* et *Meilleures pratiques* respectivement. En résolvant le problème suivant : — — — — — Nous obtenons : , , , et . Nous avons = . D'où, nous déterminons le ratio de cohérence(consistance). Selon la table 4.3, l'index(indice) de cohérence(consistance) dans notre cas est égal à , et donc le ratio de cohérence(consistance) est égal à . Ce qui dénote une très bonne cohérence comme la valeur est proche de zéro.

2.3 Classement des services Skyline

Grâce à la normalisation des poids des critères QdS obtenus avec la méthode BWM (voir la table 4.9), nous pouvons classer les services web Skyline (voir la table 4.8) obtenus dans la première étape en utilisant une des quatre méthodes AMCD. Dans cette section, nous allons utiliser les approches SAW, VIKOR, TOPSIS et COPRAS pour classer les services web.

TAB. 4.8 : Les services web dominants (matrice).

Nom	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
XigniteStockQuotes	225.36	10.9	62.6	77
HistoricalStockQuotes	119.33	2.9	76.5	80
DelayedStockQuotes	123.5	8.1	78.6	80
QuoteofTheDay	631	10	77.1	84
StockQuotes	409	8.2	67.6	84
StockQuotes	281.07	2	8.7	82

TAB. 4.9 : Poids des critères QdS.

Temps de réponse	Débit	Fiabilité	Meilleures pratiques
0.533	0.067	0.267	0.133

2.3.1 ▶ Utiliser SAW pour le classement

- **Normalisation basée sur la valeur maximale et la valeur minimale des valeurs des services web**: La normalisation du critère meilleures pratiques du est calculée comme suit :

  En appliquant le même raisonnent sur toute la matrice D, nous obtenons la matrice F(voir la *4.10*)

TAB. 4.10 : Valeurs des services web normalisées (matrice F).

Nom	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
XigniteStockQuotes	0.7928	1	0.7711	0
HistoricalStockQuotes	1	0.1011	0.97	0.4286
DelayedStockQuotes	0.9919	0.6854	1	0.4286
QuoteofTheDay	0	0.8989	0.9785	1
StockQuotes	0.4339	0.6966	0.8426	1
StockQuotes	0.6839	0	0	0.7143

- **Calcul du produit des valeurs des services web normalisées par les poids des critères respectifs obtenus avec BWM.** : Le produit du critère temps de réponse du par le poids du critère meilleures pratiques est calculé comme suit :

  En appliquant le même raisonnement sur toute la matrice F, nous obtenons la matrice P (matrice contenant les résultats du produit des attributs des services web dominants par les poids de leurs attributs respectifs (voir la *table 4.11*)

TAB. 4.11 : Valeurs du produit des services web normalisées par les poids des critères (matrice P).

Nom	Temps de réponse	Débit	Fiabilité	Meilleures pratiques
XigniteStockQuotes	0.4226	0.067	0.2059	0
HistoricalStockQuotes	0.533	0.0068	0.259	0.057
DelayedStockQuotes	0.5287	0.0459	0.267	0.057
QuoteofTheDay	0	0.0602	0.2613	0.133
StockQuotes	0.2313	0.0467	0.225	0.133
StockQuotes	0.3645	0	0	0.095

- **Calcul du score SAW des services web candidats noté** .: Le score SAW du est calculé comme suit :

  En appliquant le même raisonnement sur la matrice P, nous obtenons le vecteur S(voir la *table 4.12*)

TAB. 4.12 : Valeurs du score SAW (vecteur S).

Nom	
XigniteStockQuotes	0.6955
HistoricalStockQuotes	0.8558
DelayedStockQuotes	0.8986
QuoteofTheDay	0.4545
StockQuotes	0.636
StockQuotes	0.4595

- **Classement des services web suivant les valeurs du score SAW dans un ordre décroissant (voir la table 4.13).**:

TAB. 4.13 : Classement des services web suivant SAW.

Nom	Classement
XigniteStockQuotes	3
HistoricalStockQuotes	2
DelayedStockQuotes	1
QuoteofTheDay	6
StockQuotes	4
StockQuotes	5

2.3.2 ▶ Classement final obtenu avec COPRAS, VIKOR, SAW et TOPSIS.

La *table 4.14* fournit les classements obtenus pour les quatre méthodes testées. Comme nous pouvons voir, les quatre méthodes AMCD de classement donnent le même classement pour les trois premières positions. Cependant, des différences sont observées pour les trois dernières positions.

TAB. 4.14 : Le classement des services web avec methode AMCD utilisée (la matrice).

Nom	COPRAS	SAW	VIKOR	TOPSIS
XigniteStockQuotes	3	3	3	3
HistoricalStockQuotes	2	2	2	2
DelayedStockQuotes	1	1	1	1
QuoteofTheDay	5	6	6	6
StockQuotes	4	4	4	5
StockQuotes	6	5	5	4

2.4 Calculer la solution de compromis de Borda

- **Calcul du Score Borda noté des services web candidats.** Le score Borda du service web est calculé comme suit :

  En appliquant le même raisonnement sur la matrice RK, nous obtenons le vecteur B(voir la *table 4.15*)

Tab. 4.15 : Valeurs du score Borda (vecteur).

	Nom	Score Borda
	XigniteStockQuotes	12
	HistoricalStockQuotes	8
	DelayedStockQuotes	4
	QuoteofTheDay	23
	StockQuotes	17
	StockQuotes	20

- **Classement des services web suivant les valeurs du score Borda dans un ordre croissant (voir la table 4.16):**

Tab. 4.16 : Le classement final Borda .

	Nom	Classement Borda
	XigniteStockQuotes	3
	HistoricalStockQuotes	2
	DelayedStockQuotes	1
	QuoteofTheDay	6
	StockQuotes	4
	StockQuotes	5

2.5 Calculer le ratio de similarité.

La table 4.17 décrit le ratio de similitude de chaque approche de classement. Comme nous pouvons le constater, VIKOR et SAW réalisent les mêmes classements que la solution de compromis(Borda) ; d'où . D'autre part, TOPSIS et COPRAS échouent à correspondre deux éléments sur six ; d'où .

Tab. 4.17 : Ratio de similarité de chaque approche de classement.

ratio	COPRAS	SAW	VIKOR	TOPSIS
	4	6	6	4
	4/6=0.67	1	1	0.67

3 Résultats numériques

3.1 Modèle de la simulation

Afin d'étudier les avantages concrets qu'apporte la solution de compromis des classements des services web, nous avons réalisé une série de simulations sur un ordinateur portable avec un processeur CPU i7 1,90 GHz et 4 Go de mémoire à l'aide de notre programme implémenté en C ++ sur la plateforme Qt. Les résultats obtenus sont présentés dans cette section. Ils sont issus de tests simulant la sélection de services web sur deux ensembles de données et . reprend la description de l'ensemble de données decrites dans [121] avec des données artificielles. considére la base de données réelles contenant 2507 services web [61]. Pour mesurer l'efficacité de notre approche, nous avons effectué plusieurs simulations avec notre programme. La table 4.18 indique les valeurs (ou les fourchettes de valeurs) des critères QdS considérés dans pour générer des données artificielles.

TAB. 4.18 : Intervalle des valeurs .

QdS	Intervalle des valeurs
Temps d'exécution	[0, 300]
Réputation	[0, 5]
Coût	[0, 30]
Fiabilité	[0.5, 1]
Disponibilité	[0.7, 1]

3.2 Évaluation du temps de réponse des méthodes de classement AMCD

Pour la première simulation, des données artificielles de l'ensemble sont générées tout en faisant varier le nombre de services web candidats afin d'évaluer le temps d'exécution des quatre méthodes AMCD de classement utilisées dans notre approche sans exécuter Skyline en amont. Nous avons également évalué le temps d'exécution de la méthode PROMETHEE pour montrer sa complexité. La figure 4.2 présente les courbes de chaque méthode de classement représentant la moyenne de plusieurs simulations effectuées avec différents services web candidats. Les résultats montrent que la complexité des quatre méthodes AMCD considérées dans notre approche est assez similaire et le temps d'exécution est assez faible ; il suit une progression linéaire lorsque le nombre de services web est augmenté. PROMETHEE (qui a été abandonnée dans notre approche) suit une progression exponentielle (l'exécution de PROMETHEE atteint 1647ms quand le nombre de services web est égal à 750). En ce qui concerne notre approche, SAW parvient à avoir légèrement de meilleurs résultats en comparaison avec les trois autres méthodes ; Suivent VIKOR et COPRAS. TOPSIS semble être la méthode la moins rapide lorsque le nombre de services est augmenté. L'addition des quatre temps semble être acceptable pour fournir le classement Borda comme résultat final à la demande du client.

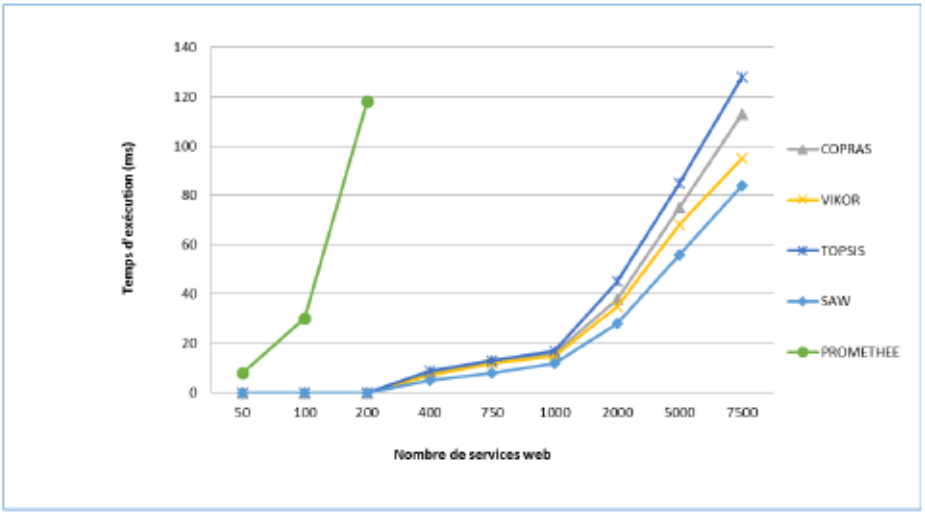

FIG. 4.2 : Comparaison des temps des réponses en utilisant $DS1$.

3.3 Évaluation des ratios de similarité de deux échantillons de données

Dans la deuxième série de simulations, les deux ensembles de données sont considérés pour classer les services web en appliquant notre approche. Pour ce faire, une liste de 100 services web est générée à partir de $DS1$, alors que la liste des 2507 services web est pris en compte à partir de $DS2$. Nous avons exécuté d'abord Skyline sur les deux ensembles de données pour réduire l'espace de recherche. Comme résultat, nous avons obtenu 10 services web dominants avec $DS1$ et 226 services web dominants avec $DS2$. Ensuite, nous avons appliqué BWM pour calculer les poids des critères QdS et enfin VIKOR, TOPSIS, SAW et COPRAS pour classer les services Skyline. Les résultats obtenus avec $DS1$ sont présentés dans la table 4.19, alors que ceux obtenus avec $DS2$ sont présentés dans la table 4.20. Dans la table 4.20, nous donnons le classement des services web classés aux 10 premières places pour chaque méthode de classement ainsi que le ratio de similarité dur RS_r pour chaque méthode AMCD. Dans la table 4.21, nous donnons seulement le nom original de chaque service web de la table 4.20.

TAB. 4.19 : Les services Skyline considérés en utilisant le avec les résultats du classement.

	Temps d'exécution	Réputation	Prix	Fiabilité	Disponibilité	COPRAS	SAW	TOPSIS	VIKOR	Borda
temps										
	13.3	2.6	2.1	0.8	0.9	1	1	1	1	1
	15.2	4.3	11.6	0.6	0.9	3	2	2	6	2
	24.4	2.8	1.6	0.7	0.9	4	3	3	3	3
	7.6	3	18.2	0.8	0.8	6	4	6	2	5
	11.7	0.3	17.2	0.6	0.9	10	9	10	9	10
	9.9	0.9	3.3	0.7	0.8	2	6	4	5	4
	5.8	1.3	15.2	0.8	0.8	9	5	8	4	7
	31.5	2.9	1.6	0.9	0.7	5	7	5	7	6
	20.8	2	7.6	0.9	0.7	7	8	7	8	8
	22.9	4.4	27.5	0.8	0.7	8	10	9	10	9
						0.2	0.4	0.6	0.3	

TAB. 4.20 : Les résultats du classement final en utilisant le .

Classement	VIKOR	SAW	TOPSIS	COPRAS	Borda
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
	0	0.17	0.035	0.022	
	0.6	0.7	0.6	0.2	

TAB. 4.21 : Les noms originaux des services web de la table 4.20.

Nom de code	Nom original
	getJoke
	Service1
	GuidGenerator
	ProTagService
	SERVICE1
	XEMBLService
	CogoService
	TemperatureService
	FaxService
	RegisterResearchServices
	TEMPERATURESERVICE
	ZipCodeLookup
	net.xmethods.services.stockquote.StockQuoteService
	promotionService
	InvitationService
	eBayWatcherService
	DataEnhancement
	GetOwamp
	Distance
	GuidGenerator
	WitwService
	WQS
	DOTSEmailValidate

Comme nous pouvons le constater, il existe des différences dans les résultats. TOPSIS parvient à avoir le meilleur ratio de similarité en comparaison avec les trois autres approches en considérant les deux ensembles de données. Cependant, tous les classements obtenus sont globalement proches sur la valeur . Le ratio doux de similarité fournit une meilleure indication alors que le ratio dur de similarité est très variable. Par exemple, dans , bien que VIKOR ne parvienne pas à une correspondance avec l'une des 226 positions du classement Borda, elle parvient à promouvoir 6 services web sur les dix sélectionnés dans la solution Borda, contrairement à COPRAS qui permet de sélectionner seulement deux éléments sur les dix meilleurs services. Le classement fourni par COPRAS tend à être un peu divergent comparativement aux trois autres classements fournis. Plus particulièrement, le de COPRAS dans est faible comparé aux trois autres. Cependant, ce n'est pas toujours le cas lors de la génération de données artificielles avec .

3.4 Évaluation du ratio de similarité dur des services web

Dans la troisième série de simulations, la solution de compromis Borda et le ratio de similarité dur RS_r avec chaque méthode sont calculés afin d'évaluer le classement de chaque méthode AMCD de classement par rapport aux autres. Les courbes illustrées par les figures 4.3 pour l'ensemble de données $DS1$ et 4.4 pour l'ensemble de données $DS2$, représentent l'évolution du ratio de similarité de chaque méthode de classement en faisant varier le nombre de services web candidats. Nous avons ajouté 10 nouveaux services web dans le processus de classement à chaque nouvelle simulation pour calculer les différentes valeurs du RS_r. Dans les deux ensembles de données,

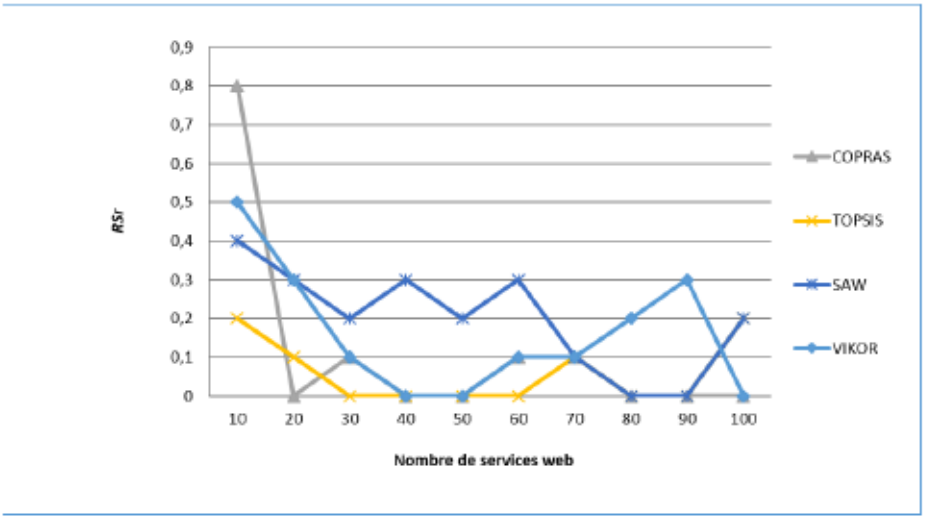

FIG. 4.3 : Évaluation du ratio dur de similarité avec $DS1$.

les résultats montrent que RS_r fluctue et que la tendance est à la baisse en augmentant le nombre de services web. A certains points, deux méthodes peuvent se correspondre dans leurs classements et à un autre moment, elles peuvent diverger.

De plus, il apparaît que plus le nombre de services web est élevé plus la probabilité d'avoir des classements divergents est grande. Par conséquent, Borda peut fournir une solution plus précise, comme un compromis des quatre classements, lorsque ceux-ci sont plus susceptibles d'être différents.

3.5 Évaluation du ratio de similarité doux des services web

Dans la quatrième série de simulations, nous avons procédé de la même manière que sur la figure4.3 pour calculer les ratios doux de similarité RDS_r afin d'avoir une meilleure évaluation du classement des méthodes de classement AMCD. Les courbes tracées sur les figures 4.5 et 4.6 montrent l'évolution de RDS_r en augmentant le nombre de services candidats. Ces nouvelles simulations confirment la première affirmation, les quatre méthodes sont plus susceptibles de diverger lorsque le nombre de services est augmenté. Cela signifie que la liste des dix meilleurs services web sélectionnés par une méthode de classement donnée est plus susceptible de différer des trois autres classements lors de l'augmentation du nombre de services web. Cette découverte soutient que le calcul du classement Borda fournit des résultats plus perti-

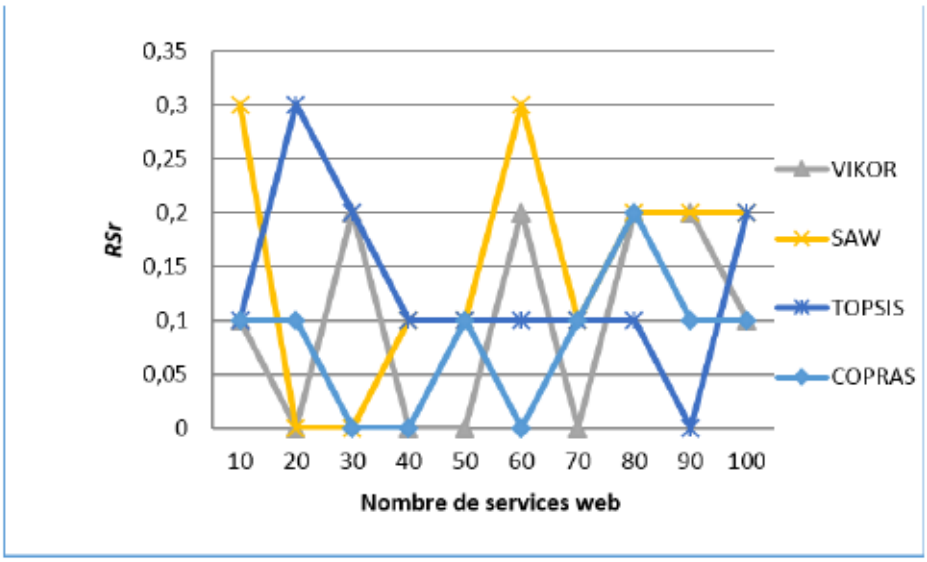

FIG. 4.4 : Évaluation du ratio dur de similarité avec $DS2$.

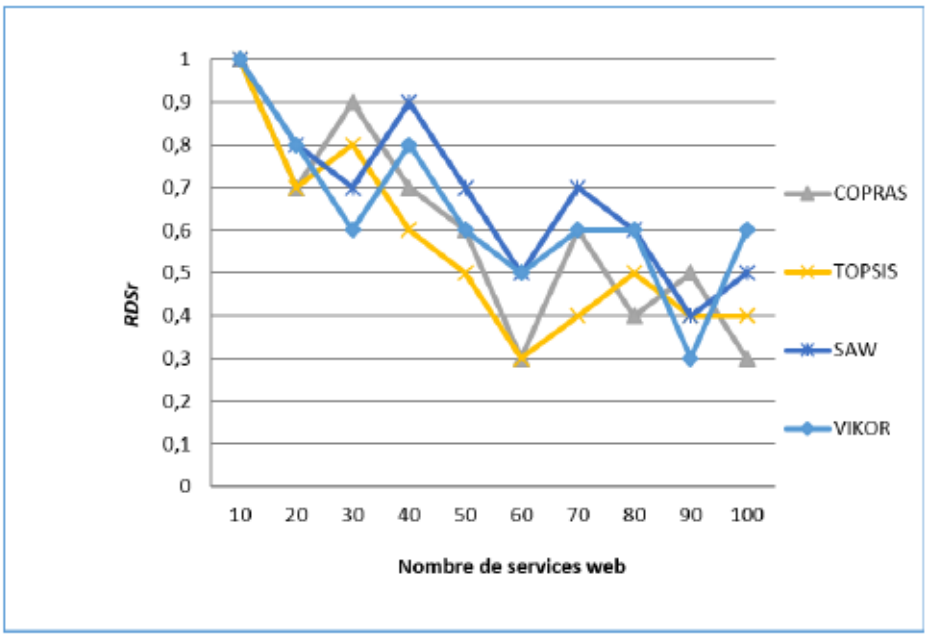

FIG. 4.5 : Évaluation du ratio doux de similarité avec $DS1$.

nents dans la sélection de services web basés sur n'importe quel choix de méthode AMCD de classement.

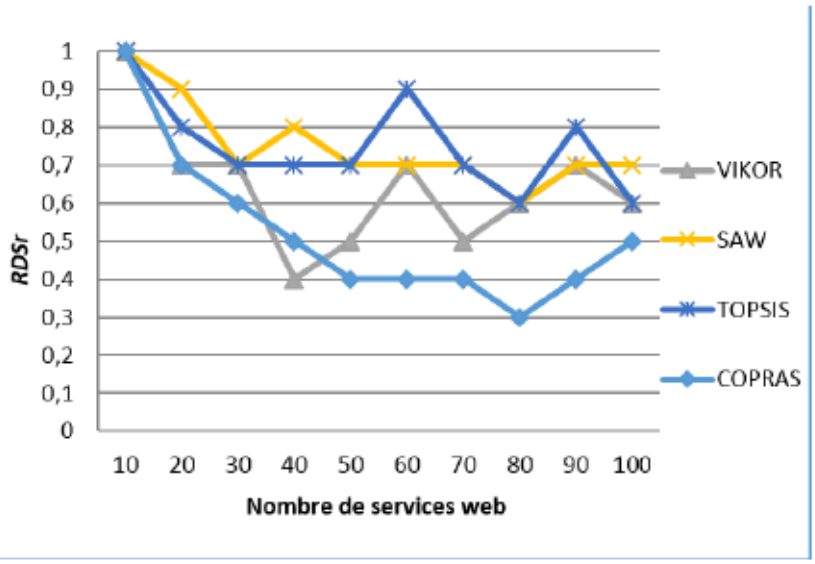

FIG. 4.6 : Évaluation du ratio doux de similarité avec $DS2$.

4 Conclusion

Dans ce chapitre, nous avons proposé de combiner différentes méthodes AMCD à différents niveaux afin d'optimiser le processus de sélection de services web dans un premier temps. Nous avons utilisé la méthode Skyline pour réduire l'espace de recherche. Ensuite, nous avons appliqué la méthode BWM pour calculer et normaliser les poids des critères QdS. Puis, nous avons proposé l'utilisation de quatre méthodes qui ont une faible complexité de calcul, à savoir : SAW, VIKOR, COPRAS et TOPSIS pour classer les services web. Dans un deuxième temps, la solution de compromis Borda est calculée à partir des quatre classements obtenus. Puis, les deux ratios de similarité sont calculés afin d'évaluer la précision de chaque méthode AMCD de classement par rapport à la solution de Borda. Nous avons fourni une étude de cas pour illustrer le fonctionnement de l'approche globale. Nous avons évalué les performances de la solution en réalisant différentes séries de simulations avec des bases de données artificielles et réelles. Les résultats montrent que les méthodes de classement AMCD sont plus susceptibles de diverger lorsque le nombre de services web est augmenté et ainsi valide la solution de compromis Borda pour proposer un compromis des quatre classements au client. Le Chapitre suivant traite un autre problème de sélection qu'est la sélection de services web dans le cas où aucun service web ne répond aux contraintes client.

5

Sélection de services web basée sur les contraintes de QdS

1 Introduction

Lors de la sélection de services web à base de contraintes client, les approches existantes suivent généralement les étapes suivantes :

1. Éliminer les services qui ne répondent pas aux contraintes client.
2. Traiter les exigences non fonctionnelles (Critères Qualité de Service (QdS)) exprimées dans la requête du client. Pour cela, les préférences du client sur les critères QdS doivent être normalisées.
3. Classer les services web candidats en utilisant des méthodes AMCD (Aide MultiCritères à la Décision) de classement traditionnel comme celles décrites dans le chapitre 4.

Lors du traitement de requêtes réelles, les clients souhaitent souvent limiter la plage des valeurs QdS dans leur environnement. Pour faire face à ce besoin spécifique, la majorité des approches s'efforce à supprimer en amont tous les services qui ne répondent pas aux contraintes QdS afin de limiter ainsi l'espace de recherche uniquement sur ceux qui répondent à toutes les exigences. Cependant, l'espace résultant

peut être vide ou trop petit pour être de qualité. En outre, parmi les services web rejetés, on pourrait trouver des candidats pouvant être promus et proposés au client comme solution intéressante en termes de performances. De nombreuses méthodes de normalisation ont été proposées dans la littérature, comme par exemple, la normalisation basée sur la valeur maximale, la normalisation basée sur la somme des valeurs, la normalisation basée sur la valeur maximale et minimale, la normalisation vectorielle, etc. Trouver une nouvelle technique de normalisation des valeurs QdS afin de prendre en compte les contraintes client pour la sélection de services web semble être une solution adéquate pour prendre en compte les services rejetés. Récemment dans ce contexte, une nouvelle méthode appelée RIM (Reference Ideal Method) [13] a été proposée dans la littérature pour traiter le problème des contraintes. Pour cela RIM propose d'utiliser une nouvelle technique de normalisation pour prendre en compte ces contraintes. Ensuite, elle adapte la méthode TOPSIS (The Technique for Order of Preference by Similarity to Ideal Solution) pour proposer un classement. Dans ce chapitre, nous proposons l'utilisation de la méthode RIM pour sélectionner des services web à base de contraintes client. Ensuite, nous proposons notre propre technique de normalisation appelée OMRI (Optimized Method of Reference Ideal). Différentes méthodes AMCD modifiées utilisant notre normalisation seront testées au niveau du classement et les temps de reponses seront mesurés afin d'évaluer la complexité de chaque approche. Pour évaluer la précision des différentes méthodes considérées, nous utilisons la méthode Borda [82] pour calculer la solution de compromis. Dans la suite de ce chapitre, nous présenterons nos deux propositions pour résoudre cette nouvelle problématique.

2 Description de notre approche de sélection

Notre approche [93] consiste à combiner différentes méthodes :

- Nous utilisons une version optimisée de AHP pour calculer et normaliser les poids des critères QdS en tenant compte des recommandations de vafaei. *et al* décrite dans [106]. Cette extension de AHP (Analytical Hierarchy Process) offre une meilleure cohérence des choix du client.
- Nous utilisons OMRI qui est une optimisation de la normalisation RIM. Elle permet d'éliminer les incohérences observées dans RIM.
- Nous adaptons quatre méthodes de classement AMCD, à savoir SAW, WPM (Weighted product model)[105], TOPSIS et VIKOR, pour gérer les contraintes client. Nous remplaçons les techniques de normalisation originales utilisées dans chaque méthode AMCD par OMRI. Nous notons respectivement les méthodes résultantes : SAW *, WPM *, TOPSIS * et VIKOR *.

La principale motivation de la définition de la méthode OMRI est d'offrir une sélection efficace et efficiente des meilleurs services web parmi ceux qui ne répondent pas aux contraintes client. La figure 5.1 décrit les différentes étapes de notre approche que sont :

1. **L'étape de normalisation des poids QdS** permet de calculer et de normaliser les poids des paramètres QdS considérés dans la requête du client. Nous avons

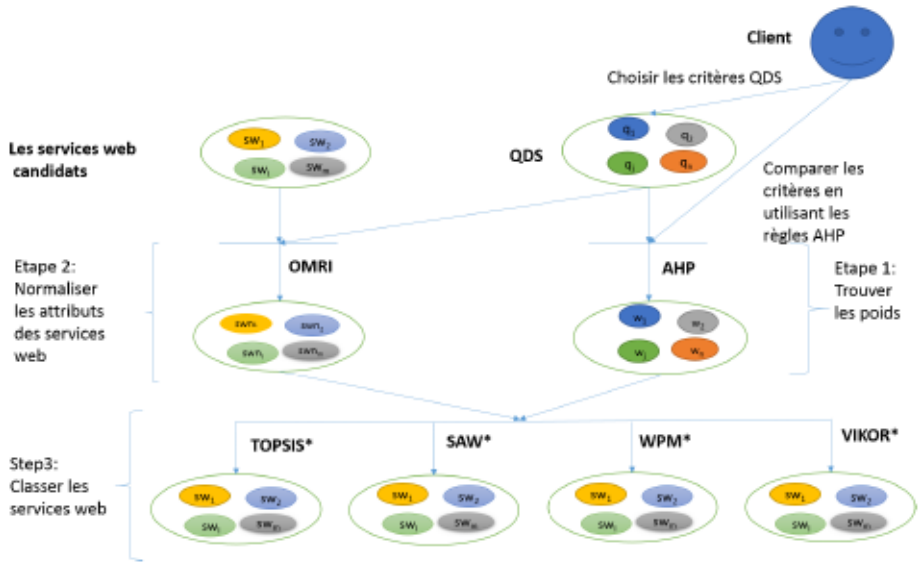

FIG. 5.1 : Description de notre approche de sélection de services web.

utilisé la méthode AHP modifiée. L'utilisation de AHP permet de faciliter le calcul et la normalisation des poids des critères QdS en prenant en compte les préférences du client et en réduisant le degré d'incohérence.

2. **L'étape de normalisation des valeurs QdS** permet de calculer et de normaliser les valeurs des paramètres de QdS des services web en prenant en compte les contraintes client exigées dans la requête du client. Nous avons utilisé la méthode OMRI et la méthode RIM [92].

3. **L'étape de classement** permet de classer les services web. Nous choisissons de considérer quatre méthodes(SAW *, WPM *, TOPSIS * et VIKOR *) ayant une complexité réduite. Il convient de noter que les méthodes SAW *, WPM *, TOPSIS * et VIKOR * ne peuvent être utilisées que pour classer les services web qui ne respectent pas les contraintes client. Tous les services qui satisferont ces derniers seront classés en haut, ils obtiendront le même indice de performance maximale, à savoir 1. Par conséquent, pour sélectionner le meilleur service parmi l'ensemble des services web répondant aux contraintes de valeur, nous devons utiliser l'une des méthodes de classement traditionnelles SAW, COPRAS, TOPSIS ou VIKOR, etc.(voir Chapitre 4).

2.1 Calculer et normaliser les poids des critères QdS avec AHP

AHP *(Analytical Hierarchy Process)* est méthode AMCD a été introduite par *Saaty* dans [86]. Les étapes de AHP sont décrites ci-dessous. Soit q_1,q_2.. q_n l'ensemble des critères QdS.

- **Étape 1** : Déterminer la structure du modèle. À ce stade, le client construit une hiérarchie des différents critères. La figure 5.2 illustre un exemple de hiérarchie à deux niveaux de critères.

FIG. 5.2 : Exemple de hiérarchie AHP.

- **Étape 2** : Comparer les éléments de chaque niveau hiérarchique un à un et construire la matrice de comparaison (C).

TAB. 5.1 : Matrice de comparaison des critères QdS.

	q_1	q_2	⋯	q_j	⋯	q_n
q_1	c_{11}	c_{12}	⋯	c_{1j}	⋯	c_{1n}
q_2	c_{21}	c_{22}	⋯	c_{2j}	⋯	c_{2n}
⋮	⋮	⋮	⋱	⋮	⋱	⋮
q_j	c_{j1}	c_{j2}	⋯	c_{jj}	⋯	c_{jn}
⋮	⋮	⋮	⋱	⋮	⋱	⋮
q_n	c_{n1}	c_{n2}	⋯	c_{nj}	⋯	c_{nn}

Les valeurs de la matrice de comparaison (voir la *table 5.1*) sont obtenues par la transformation de la subjectivité des choix du décideur en valeurs numériques selon les neuf Règles *Saatys*(voir *table 5.2*)

TAB. 5.2 : Les neuf règles d'AHP.

Définitions	Degré d'importance
importance égale	1
Modérément plus important	3
Fortement plus important	5
Très fortement plus important	7
Extrêmement plus important	9
Valeurs intermédiaires	2,4,6,8

- **Étape 2** : Calculer les valeurs normalisées pour chaque critère en tenant compte

des recommandations de [106]. Commencer avec la normalisation maximale :

—— (5.1)

ou, Finaliser avec la normalisation de la somme :

——— (5.2)

- **Étape 3** : Déterminer les poids , ,..., ,..., , associés respectivement aux critères , ,..., ,..., , :

– (5.3)

- **Étape 4** : Calculer le ratio de cohérence et vérifier si les poids sont cohérents, en utilisant les équations suivantes : , ——— —, where, est l'*indice de cohérence aléatoire* [84] et est obtenu comme à partir de la *table 5.3*.

TAB. 5.3 : Les valeurs de .

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.52	0.89	1.11	1.25	1.35	1.40	1.45	1.29

est l'*indice de cohérence*. est la plus grande valeur propre de la matrice et représente le nombre de critères. devrait avoir une valeur plus petite que 0.10 pour valider la conformité des résultats d'AHP.

2.2 Utiliser RIM pour la normalisation des données

Cette normalisation a été proposée par Cables *et al* dans [13]. C'est la première approche de normalisation définie pour gérer les contraintes client. Cette normalisation se fait en divisant la distance entre les valeurs de performance par la distance entre la valeur de performance maximale (ou minimale) et la valeur de performance idéale de référence pour ce critère. L'idéal de référence peut être un point ou un intervalle. L'idéal de référence est généralement utilisé comme un intervalle ; il représente les contraintes de valeur fixées par le client.

si *[A,B]*

——— *si* *[,A]*

——— *si* *[B,]*

L'idéal de référence peut varier entre les deux valeurs extrêmes de performance pour le critère . Cette normalisation a été combinée avec TOPSIS pour définir la méthode RIM.

2.3 Pourquoi OMRI ?

Avant de présenter notre normalisation, appelée OMRI, nous soulignons d'abord, à travers des exemples, les inconvénients de l'utilisation de la normalisation RIM. Dans la normalisation RIM , si alors les services qui sont dans l'intervalle ont plus de chances d'être sélectionnés que les services qui prennent leurs valeurs dans l'intervalle et vice versa. Pour souligner ce fait, considérons l'exemple suivant.

Exemple 1 : Considérons un ensemble de services web caractérisé par le critère "*prix*" qui prend ses valeurs dans l'intervalle . Supposons que le client définit des contraintes telles que le critère *prix* doit être à l'intérieur de l'intervalle . Considérons deux services web et tel que et . Par conséquent, comme et ne satisfont pas les contraintes sur le critère "*prix*", le but est de déterminer quel service web est le plus proche répondant à ces contraintes. Lorsque nous utilisons l'approche RIM, la normalisation des valeurs des services web se déroule sur deux intervalles différents indiquant les plages de valeurs situées en dehors de l'intervalle idéal des valeurs requises, à savoir : . Par conséquent, dans notre exemple, les deux intervalles sont : et . Nous avons et .
La normalisation RIM donne les résultats suivants : ——
qui est supérieur à —— . Cependant, logiquement, ces deux services auraient dû bénéficier du même score, car ils sont à une distance égale des contraintes, .

Exemple 2 : Considérons le même exemple précédent avec des contraintes différentes, en supposant que celles-ci soient dans l'intervalle . Supposons les deux services web et , tels que et . Nous avons et . Après application de la normalisation RIM, nous obtenons : —— < —— .

Nous arrivons à la même conclusion que l'exemple précédent. Logiquement, et devraient obtenir le même score car les deux services sont équidistants aux contraintes client : . Les exemples précédents soulignent le fait que la normalisation RIM ne garantit pas un classement correct lorsqu'il s'agit de contraintes. Par conséquent, nous introduisons ci-après la normalisation OMRI pour éliminer l'incohérence observée dans RIM.

2.3.1 ▶ Présentation de OMRI

Comme pour RIM, l'idéal de référence d'un critère dans OMRI peut être un intervalle compris entre les valeurs minimales et maximales, que nous notons .
La normalisation des données dans OMRI se déroule comme suit :

$$\begin{array}{l} \qquad\qquad\qquad\qquad\quad si \quad [A,B] \\ \_\_\_\_\_ \quad si \quad [\quad ,A] \\ \_\_\_\_\_ \quad si \quad [B,\quad] \end{array} \tag{5.4}$$

Comme nous pouvons le constater, dans OMRI, plutôt que d'utiliser deux intervalles pour normaliser les données, nous considérons la taille maximale des deux intervalles. . Cela garantit un classement équitable pour tous les services web qui ne répondent pas aux contraintes client. Par exemple, revenons à l'exemple 1. En appliquant OMRI, on obtient (voir la figure 5.3) :

$f(s_1, prix) = 1 - \frac{4-2}{MAX(9-7,4-1)} = 0.33$, qui est égal à $f(s_2, prix) = 1 - \frac{9-7}{MAX(9-7,4-1)} = 0.33$.

De même, en appliquant OMRI dans l'exemple 2, nous obtenons : $f(s_1, prix) = f(s_3, prix) = 0.8$

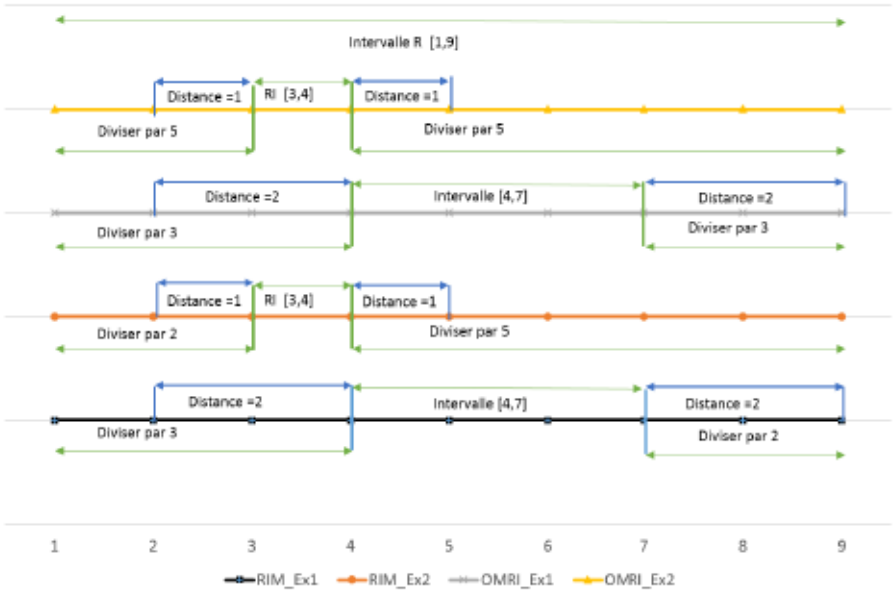

FIG. 5.3 : La différence entre la normalisation RIM et OMRI.

2.4 Extension des méthodes de classement AMCD pour gérer les contraintes de valeur.

L'approche prend en compte quatre méthodes AMCD différentes : SAW, TOPSIS, WPM et VIKOR. Ces dernières utilisent désormais la normalisation OMRI pour gérer les contraintes. Les méthodes étendues notées respectivement SAW *, TOPSIS *, WPM * et VIKOR * procèdent au classement des services web en fonction des poids des critères fournis par AHP et des valeurs normalisées des services web candidats fournies par OMRI. Soit F la matrice de décision normalisée OMRI de dimension (m, n) (Voir la *table 5.4*). Soit $w_1, .., w_n$ le poids des critères fourni par AHP. n indique le nombre de critères de qualité de service pris en compte dans la demande du client et m le nombre de services web candidats.

TAB. 5.4 : La matrice .

		...		...	
		...		...	
		...		...	
⋮	⋮	⋱	⋮	⋱	⋮
		...		...	
⋮	⋮	⋮	⋱	⋮	⋮
		...		...	

Nous donnons ci-après une brève description du fonctionnement des quatre méthodes :

2.4.1 ▶ VIKOR*

Les étapes procédurales de l'algorithme VIKOR * sont décrites ci-dessous.

- **Étape 1** : Calculer les valeurs et , avec en utilisant les relations :

(5.5)

max (5.6)

Le résultat de la distance Manhattan[46] sera stocké dans un vecteur de dimension . Le résultat de la distance Chebyshev[15] sera stocké dans un vecteur de dimension .

- **Étape 2** : Déterminer les valeurs de , , et défini comme suit : max min max min

- **Étape 3** : Calculer les valeurs , , par la relation :

——————— ——————— (5.7)

est défini comme le poids de la stratégie d'utilité maximale du groupe, alors que est le poids du regret individuel. La valeur est considérée comme une valeur de compromis. Les résultats du score VIKOR des service web dominants seront stockés dans un vecteur de dimension .

- **Étape 4** : Classer les services web en fonction de ensuite et finalement dans un ordre décroissant.

2.4.2 ▶ TOPSIS*

Les étapes procédurales impliquées dans TOPSIS * sont décrites ci-dessous.

- **Étape 1** : Calculer le produit des valeurs normalisées de la matrice par les poids des paramètres de QdS respectifs obtenu avec AHP :

(5.8)

avec et avec Les résultats du produits seront stockés dans une matrice de dimension .

- **Étape 2** : Calculer la distance euclidienne entre l'idéal positif et l'idéal négatif des attributs parmi les services, : **Distance euclidienne des solutions idéales positives**: Sachant que le vecteur qui représente la solution idéale positive (idéal de référence) est alors :

(5.9)

Distance euclidienne des solutions idéales négatives: Sachant que le vecteur qui représente la solution idéale négative est alors :

(5.10)

Les résultats de la distance euclidienne des solutions idéales positives seront stockés dans un vecteur de dimension . Les résultats de la distance euclidienne des solutions idéales négatives seront stockés dans un vecteur de dimension .

- **Étape 3** : Calculer la proximité à la valeur positive des attributs parmi les services , , avec la relation :

(5.11)

Les résultats du score TOPSIS* des services web seront stockés dans un vecteur de dimension .

- **Étape 4** : Classer les services web dans un ordre décroissant suivant les valeurs de .

2.4.3 ▶ WPM*

WPM *(The Weighted Product Model)* a été défini dans [105]. Les étapes procédurales impliquées dans l'algorithme WPM sont décrites ci-après.

- **Étape 1** : Calculer le score des services individuels :

(5.12)

avec Les résultats du score WPM* des services web seront stockés dans une vecteur de dimension .

- **Étape 2** : Classer les services Web dans un ordre décroissant selon .

2.4.4 ▶ SAW*

Les étapes procédurales de l'algorithme SAW * sont décrites ci-après.

- **Étape 1** : Calculer le score des services individuels :

(5.13)

  Les résultats du score SAW* des services web seront stockés dans une vecteur de dimension .

- **Étape 2** : Classer les services web dans un ordre décroissant suivant les valeurs de .

2.5 Utiliser Borda pour comparer les différents classements

Pour évaluer la pertinence et la précision des quatre approches de classement AMCD testées, on considère la même solution définie dans [94] qui utilise la technique [83] pour calculer la solution de compromis. Soit , , et les classements des services web obtenus en appliquant respectivement VIKOR*, WPM*, TOPSIS*, et SAW*. donne le classement du service web obtenu en utilisant la méthode . La méthode est une méthode basée sur le vote. Elle suppose qu'il y a méthodes (votants) qui votent sur les services . Chaque service web reçoit des votes à différentes positions dans le classement. Ensuite, le score Borda du service web est obtenu en agrégeant les différents classements , comme suit : [94] :

- **Étape 1** : Calculer le score de Borda en ajoutant le classement de service correspondant à chaque méthode.
- **Étape 2** : Classer les services dans un ordre croissant en fonction du score . En cas d'égalité dans le score, nous considérons le nombre de fois où les services web obtiennent une meilleure note. Le classement obtenu est une solution de compromis entre les 4 méthodes de classement.

2.6 Calcul de la pertinence de chaque méthode de classement

Déterminer la pertinence des résultats obtenus par chaque classement . On peut mesurer la similarité entre ce dernier et le compromis de Borda en classant . Pour cet effet, nous considérons les mêmes ratios de similarité introduits dans [94], appelés ratio de *similarité dur* et ratio de *similarité doux*.

2.6.1 ▶ a-Calculer du ratio de similarité dur :

- **Étape 1** : Calculer le score de similarité en comptant le nombre de fois où deux classements se correspondent : ou, si , sinon
- **Étape 2** : Calculer le ratio de similarité, noté , obtenu en divisant le score par le nombre de services web candidats : ——

2.6.2 ▶ b-Calculer le ratio de similarité doux :

Afin de fournir un meilleur outil pour évaluer la pertinence de chaque méthode de classement, le ratio de similarité doux, noté , considère les premiers services web dans chaque classement. Les points d'intersection avec l'ensemble des premiers services de la solution de Borda sont d'abord déterminés. Le nombre services présents dans l'intersection sur services détermine le ratio de similarité doux. En d'autres termes, permet d'évaluer le nombre de services communs à la solution contenus dans les premières communes à la solution Borda et à toute méthode de classement quel que soit leur classement exact au premières positions. Pour calculer nous procédons comme suit :

Soit l'ensemble de éléments supérieurs de la méthode .
Soit l'ensemble de éléments clés de la solution Borda.

_____,

3 Une étude de cas

Pour illustrer l'application de notre approche dans le contexte des services web, nous considérons une étude de cas utilisant le "QWS DataSet" [60]. Comme le montre la table 5.5 , les services web sont caractérisés par trois critères de qualité de service, à savoir : - *Temps de réponse* (ms) : Il représente le temps écoulé entre l'envoi d'une demande par le client et la réception d'une réponse fournie par le service web ; c'est un critère négatif. - *Débit* (hits/sec) : Il représente le nombre total d'invocations pour une période donnée. L'unité de mesure est le nombre d'appels par seconde pour un service donné ; c'est un critère positif. - *Documentation* (%) : Il désigne le rapport de documentation (c'est-à-dire les balises de description) présent dans WSDL ; c'est un critère positif.

Tab. 5.5 : Notre étude de cas.

	Nom	Temps de réponse	Débit	Documentation
	DictionaryService	45	27.2	58
	MyService	71.75	14.6	86
	aba	117	23.4	59
	AlexaWebSearch	70	5.4	91
	ErrorMailer	105.2	18.2	91
	getJoke	224	24.6	88

3.1 Détermination et normalisation des poids des critères QdS

Au cours de cette étape, nous appliquons la méthode AHP pour déterminer et normaliser les poids des trois paramètres de qualité de service. Nous fournissons des notes (entre 1 et 9). Lorsque nous comparons les trois critères deux à deux (voir la table 5.6).

TAB. 5.6 : Choix de préférence du client.

	Temps de réponse	Débit	Documentation
Temps de réponse	1	3	6
Débit	1/3	1	2
Documentation	1/6	1/2	1

- Nous calculons les poids normalisés pour tous les critères. En utilisant d'abord la normalisation basée sur la valeur maximale (voir la table 5.7)

TAB. 5.7 : Normalisation basée sur la valeur maximale.

	Temps de réponse	Débit	Documentation
Temps de réponse	————	1	1
Débit	1/3	1/3	1/3
Documentation	1/6	1/6	1/6

- Ensuite, Nous appliquons la normalisation basée sur la somme des valeurs (voir la table 5.8).

TAB. 5.8 : Normalisation basée sur la somme des valeurs.

	Temps de réponse	Débit	Documentation
Temps de réponse	————	0.67	0.67
Débit	0.22	0.22	0.22
Documentation	0.11	0.11	0.11

Soit , et les poids normalisés de respectivement *Temps de réponse* , *Débit* et *Documentation*. Nous obtenons donc : — — — ———— , — — — ———— , — — — ———— . - L'indice de cohérence est calculé comme suit : Comme , Nous avons , alors nous déterminons

———— = ——— L'indice de cohérence est égal à qui est inférieur à . Cela signifie que le schéma de pondération associé aux critères de QoS est cohérent. Par conséquent, les poids QdS ainsi calculés peuvent être utilisés dans la prochaine étape pour classer les services Web.

3.2 Normalisation des données avec OMRI en fonction des contraintes client

La table 5.9 donne pour chaque critère : *(i)* la plage des valeurs lorsqu'on considère tous les services web ; *(ii)* l'idéal de référence dénote les valeurs des contraintes expri-

mées par le client ; *(iii)* et enfin la distance de normalisation OMRI calculée. Nous remarquons qu'aucun des 6 services web ne répond à toutes les valeurs des contraintes. En effet,

- le service web 1 ne répond à aucun des 3 critères.
- le service web 2 répond uniquement au critère temps de réponse.
- le service web 3 ne répond à aucun des 3 critères.
- le service web 4 répond uniquement au critère temps de réponse.
- le service web 5 ne répond à aucun des 3 critères.
- le service web 6 ne répond à aucun des 3 critères.

Par conséquent, l'objectif est de sélectionner le service qui répond le mieux à toutes les contraintes.

TAB. 5.9 : Distance de normalisation OMRI.

	Plage des valeurs	l'idéal de référence	la distance de normalisation OMRI
Temps de réponse	[45,224]	[70,90]	max(70-45, 224-90)=134
Débit	[5.4,27.2]	[12,14]	max(12-5.4, 27.2-14)=13.2
Documentation	[58,91]	[70,75]	max(70-58, 91-75)=16

En appliquant la formule de normalisation OMRI pour le et le critère débit nous avons :

En appliquant le même calcul sur toute la matrice D, nous obtenons la matrice F (matrice contenant les valeurs de la normalisation (voir la *table 5.10*)

TAB. 5.10 : Normalisation des données de services web à l'aide d'OMRI (matrice F).

	Temps de réponse	Débit	Documentation
DictionaryService	0.81	0	0.25
MyService	1	0.95	0.31
aba	0.8	0.29	0.31
AlexaWebSearch	1	0.5	0
ErrorMailer	0.87	0.68	0
getJoke	0	0.2	0.19

3.3 Classement des services web

Grâce à la normalisation des poids des critères QdS et des valeurs des services web en utilisant respectivement AHP et OMRI, nous sommes en mesure de classer les services web en appliquant l'une des quatre méthodes de classement sélectionnées.

3.3.1 ▶ WPM*

- **Étape 1** : Calcul du score des services individuels . En appliquant la formule pour le nous avons :

  En appliquant le même calcul sur la matrice F, nous obtenons la matrice S (voir la *table 5.11*).

TAB. 5.11 : Valeurs du score WPM (vecteur S).

	Nom	
	DictionaryService	0
	MyService	0.87
	aba	0.58
	AlexaWebSearch	0
	ErrorMailer	0
	getJoke	0

- **Étape 2** : Classer les services web suivant les valeurs de dans un ordre décroissant (voir la *table 5.12*).

TAB. 5.12 : Classement des services web.

	Nom	Classement
	DictionaryService	3
	MyService	1
	aba	2
	AlexaWebSearch	3
	ErrorMailer	3
	getJoke	3

3.3.2 ▶ Comparaison des différents classements avec VIKOR *, SAW *, WPM * et TOPSIS *.

La table 5.13 présente les différents classements obtenus avec WPM*, VIKOR*, SAW* et TOPSIS* , ainsi que le classement obtenu avec Borda. Comme on peut le constater, les quatre méthodes sélectionnent le même meilleur service. La solution Borda est similaire à presque tous les classements en ce qui concerne le ratio de similitude dur, à l'exception de WPM *.

Tab. 5.13 : Comparaison des différents classement.

Nom	WPM*	VIKOR*	SAW*	TOPSIS*	Borda
DictionaryService	3	5	5	5	5
MyService	1	1	1	1	1
aba	2	4	4	4	4
AlexaWebSearch	3	2	2	2	2
ErrorMailer	3	3	3	3	3
getJoke	3	6	6	6	6
Ratio de similarité dur	2/6	6/6	6/6	6/6	
Ratio de similarité doux	1	1	1	1	

4 Résultats numériques

4.1 Modèle de la simulation

Afin d'étudier les avantages concrets qu'apporte la solution de compromis des classements des services web, nous avons réalisé une série de simulations sur un ordinateur portable avec un processeur CPU i5 1,90 GHz et d'une mémoire de 4 Go à l'aide de notre programme implémenté en C ++ sur la plateforme Qt. Les résultats sont issus de tests simulant la sélection de services web sur deux ensembles de données et . reprend la description de l'ensemble de données decrites dans [121] avec des données artificielles. considére la base de données réelles contenant 2507 services web [61]. Pour mesurer l'efficacité de notre approche, nous avons effectué plusieurs simulations avec notre programme. La table 5.14 indique les valeurs (ou les fourchettes de valeurs) des critères QdS ainsi que la plages des valeurs des contraintes d'un client considérés dans pour générer des données artificielles. La table 5.15 indique les valeurs (ou les fourchettes de valeurs) des critères QdS tel que décrite dans ainsi que la plage des valeurs des contraintes d'un client.

Tab. 5.14 : Intervalle des valeurs des données artificielles .

QdS	Intervalle des valeurs	Plage de valeurs de la contrainte
Temps d'exécution	[0, 300]	[50, 60]
Réputation	[0, 5]	[3,5]
Coût	[0, 30]	[15, 20]
Fiabilité	[0.5, 1]	[0.7, 9]
Disponibilité	[0.7, 1]	[0.9, 1]

TAB. 5.15 : Intervalle des valeurs des données réelles EDR.

QdS	Plage de valeurs du critère	Plage de valeur de la contrainte
Temps de réponse	[37, 4989.67]	[50, 100]
Disponibilité	[7, 100]	[92,100]
Débit	[0.1, 43.1]	[39, 42.5]
succès	[8, 100]	[86, 94]
Fiabilité	[33, 89]	[72, 82]
Conformité	[33, 100]	[75, 100]
meilleures pratiques	[50, 95]	[62,84]
Latence	[0.25, 4140.14]	[99, 100]
Documentation	[1, 97]	[82, 93]

4.2 Évaluation du temps d'exécution

Pour la première simulation, des données artificielles de l'ensemble EDA sont générées tout en faisant varier le nombre de services web candidats afin d'évaluer le temps d'exécution des quatre méthodes AMCD de classement utilisées dans notre approche. La figure 5.4 présente les courbes de chaque méthode de classement représentant la moyenne de plusieurs simulations effectuées avec différents services web candidats. A noter que nous avons ajouté 2500 nouveaux services dans le processus pour chaque nouvelle simulation. Les résultats montrent que la complexité des quatre méthodes AMCD considérées dans notre approche est assez similaire et le temps d'exécution est assez faible ; il suit une progression linéaire lorsque le nombre de services web est augmenté. SAW* parvient à avoir légèrement de meilleurs résultats en comparaison avec les trois autres méthodes ; Suivent WPM* et VIKOR*. TOPSIS* semble être la méthode la moins rapide lorsque le nombre de services est augmenté.

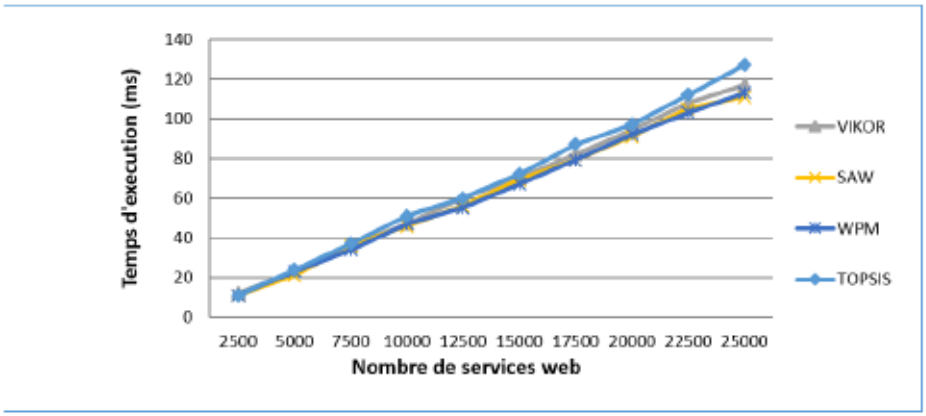

FIG. 5.4 : Camparaison des temps de réponse de EDA.

4.3 Évaluation des ratios de similarité sur la base de données réelles

Dans la deuxième série de simulations, l'ensemble de données réelles est considéré pour classer les services web qui se rapprochent des contraintes client en appliquant notre approche. Nous avons appliquer d'abord AHP pour calculer le poids des critères QdS et OMRI pour la normalisation des valeurs des services web en fonction des contraintes de valeurs fixées par le client. Notez qu'à ce stade, aucun service web ne répond à toutes les contraintes du client décrites dans la table 5.15. Par conséquent, nous exécutons VIKOR *, TOPSIS *, SAW * et WPM * pour classer les services web afin de déterminer les services qui répondent le mieux aux besoins du client. Les résultats obtenus avec sont présentés dans la table 5.16. Dans la table 5.16, nous donnons la liste des dix meilleurs services spécifique à chaque méthode de classement. Dans la table 5.18, nous donnons le nom original uniquement pour les services avec un long nom de la table 5.16 et la table 5.17.

TAB. 5.16 : Résultat du classement final avec .

	VIKOR*	SAW*	WPM*	TOPSIS*
Temps d'exécution	20 ms	18 ms	17 ms	18 ms
1	Service1(xprogramming)			
2	GuidGenerator			
3	Sydication			
4	hunting and consevation			CogoService
5	ImgCutout			hunting and consevation
6	SdssFields(dhu)		NewsAndUpdates	magic
7	NewsAndUpdates		SdssFields(dhu)	getJoke
8	ImageService		CogoService	Testtool
9	SdssFields(dr2fields)		ImageService	ImgCutout
10	SendEmailService		SdssFields(dr2fields)	interop2C

Pour comparer les différents classements obtenus avec chaque méthode, nous avons calculé la solution Borda et donc les ratios et . Les résultats sont présentés dans la table 5.17.

TAB. 5.17 : Comparaison des différents classements.

	Borda*	VIKOR*	SAW*	WPM*	TOPSIS*
1	Service1	1	1	1	1
2	GuidGenerator	2	2	2	2
3	Sydication	3	3	3	3
4	hunting and consevation	4	4	4	5
5	ImgCutout	5	5	5	9
6	NewsAndUpdates	7	7	6	11
7	SdssFields(dhu)	6	6	7	14
8	CogoService	12	12	8	4
9	ImageService	8	8	9	12
10	SdssFields(dr2fields)	9	9	10	17
		0.5	0.5	1	0.4
		0.9	0.9	1	0.6

Comme nous pouvons le constater, toutes les méthodes permettent de promouvoir les trois mêmes meilleurs services. Cependant, des différences minimes sont observées entre TOPSIS * et les trois autres méthodes sur l'ensemble des données réelles . WPM* parvient à avoir les meilleurs ratios de similarités en comparaison avec les trois autres approches. Cependant, tous les classements obtenus sont globalement proches sur la valeur . Le classement fourni par TOPSIS* tend à être un peu divergent comparativement aux trois autres classements fournis. Plus particulièrement, le de TOPSIS* est faible comparé aux trois autres. Il est à noter que VIKOR* et SAW* fournissent le même classement.

TAB. 5.18 : Les noms originaux des services web avec des noms longs de la table 5.16 et la table 5.17.

Nom de code	Nom original
Sydication	
hunting and consevation	
NewsAndUpdates	

4.4 Évaluation du ratio de similarité dur des services web

Dans la troisième série de simulations, la solution de compromis Borda et le ratio de similarité dur avec chaque méthode sont calculés afin d'évaluer le classement de chaque méthode AMCD de classement par rapport aux autres. Les courbes illustrées à la figure 5.5 représentent l'évolution du ratio de similarité de chaque méthode de classement en faisant varier le nombre de services web candidats dans l'ensemble de données . Nous avons ajouté 250 nouveaux services web dans le processus de classement à chaque nouvelle simulation pour calculer les différentes valeurs du . Les résultats montrent que fluctue quand le nombre de services web aug-

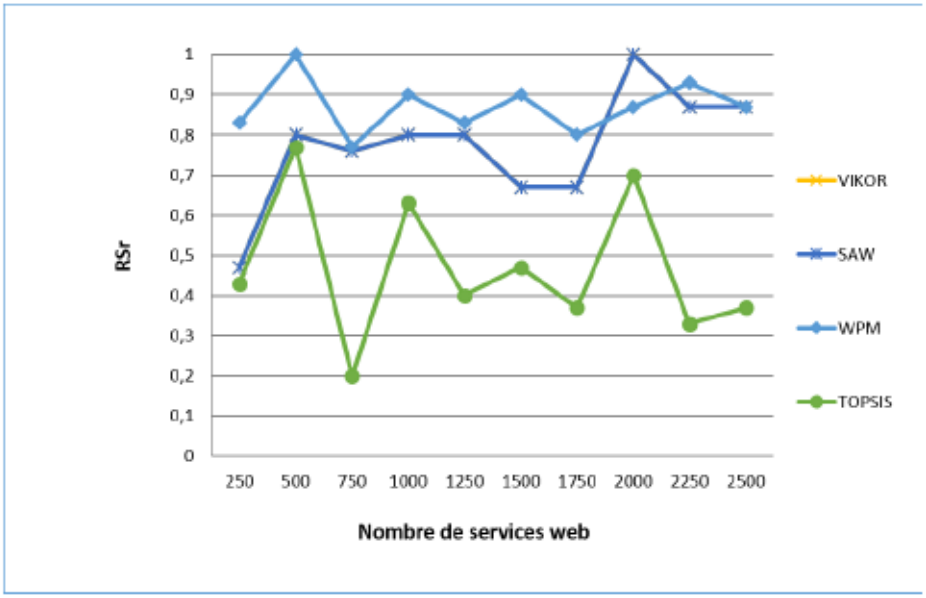

FIG. 5.5 : Évaluation du ratio de similarité dur.

mente. SAW * WPM * et VIKOR * semblent avoir le même classement, contrairement à TOPSIS *, qui a tendance à diverger un peu. Aussi, nous avons effectué de nombreuses autres simulations sur différents ensembles de données réelles pour confirmer ou infirmer ce résultat. Le point positif est qu'en considérant OMRI dans les méthodes testées, les quatre classements obtenus (calculés dans ce contexte spécifique des contraintes client) parviennent à maintenir leur convergence lorsque le nombre de services est augmenté. Malheureusement, cela ne ressemble pas au cas général. En effet, nous avons montré dans le chapitre 4 que les méthodes de classement AMCD utilisées conjointement avec leurs techniques de normalisation standards originales ont tendance à diverger lorsque le nombre de services augmente. En outre, il convient de tenir compte du fait qu'évaluer la pertinence et l'exactitude de chaque classement obtenu est très subjectif, car il repose principalement sur une évaluation de la Qualité d'Expérience (QdE) qui n'est pas facile à mettre en œuvre.

4.5 Évaluation du ratio de similarité doux des services web

Dans la quatrième série de simulations, nous avons procédé de la même manière que sur la figure5.5 pour calculer les ratios doux de similarité RDS_r afin d'avoir une meilleure évaluation du classement des méthodes de classement AMCD. Les courbes tracées sur la figure 5.6 montrent l'évolution de RDS_r en augmentant le nombre de services candidats. A noter que chaque point est la moyenne de plusieurs simulations avec des données différentes. Comme on peut le constater, ces nouvelles simulations confirment la première affirmation. WPM *, VIKOR * et SAW * permettent de calculer à peu près le même classement. TOPSIS * a tendance à diverger légèrement par rapport aux autres. Cependant, toutes les méthodes réussissent globalement à sélectionner les mêmes meilleurs services web, RDS^r restée au-dessus de 0.8, quelle que soit la méthode choisie et le nombre de services considérés. Cela signifie qu'au moins 8 des 10 services considérés dans la solution de compromis Borda sont également promus dans

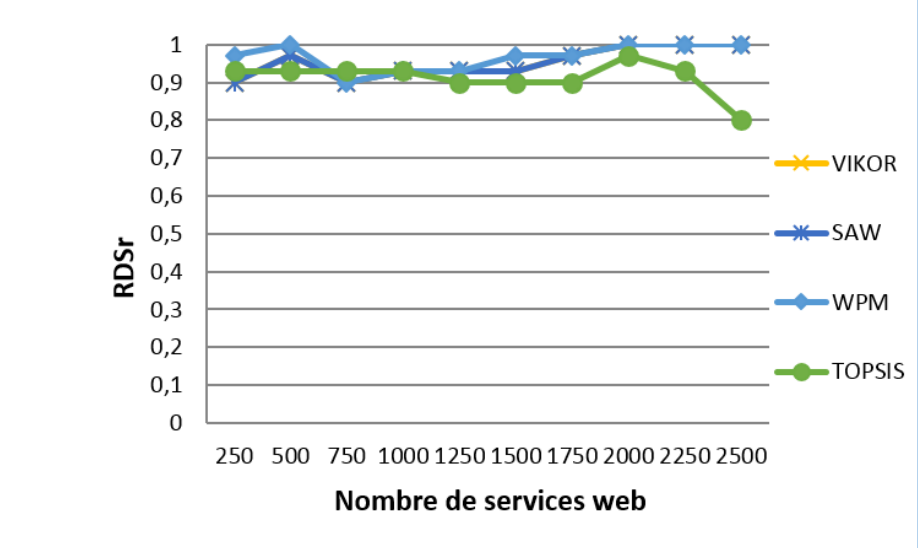

FIG. 5.6 : Évaluation des ratios de similarité doux.

n'importe quelle méthode de classement. De plus, toutes les simulations effectuées confirment que SAW * et VIKOR * affichent le même classement final.

5 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle technique normalisation, appelée OMRI, permettant de normaliser les valeurs des services web en fonction de contraintes client. Ensuite, nous avons étendu différentes méthodes de classement AMCD avec OMRI afin de promouvoir des services web et ceci même lorsque les contraintes ne sont pas satisfaites. Dans notre approche, nous avons utilisé une version optimisée d' AHP pour calculer et normaliser les poids associés aux critères de qualité de service. Nous avons considéré une extension des méthodes SAW, TOPSIS, VIKOR et WPM pour classer les éléments qui ne respectent pas les contraintes client. Pour évaluer la cohérence du processus de classement, nous avons calculé la solution de compromis Borda ainsi que deux ratios de similarité. Pour valider notre approche, nous avons effectué plusieurs simulations sur un ensemble de données réelles et artificielles. Le chapitre suivant traite une problématique de sélection de services web composites. Ce problème est la sélection de services web composites qui répondent à des contraintes spécifiques à chaque profil client.

6

Sélection de services web composites pour un groupe de clients

1 Introduction

Une coopérative est la combinaison d'un regroupement de personnes et d'une entreprise fondée sur la participation économique des membres, en capital et en opérations. Son organisation et son fonctionnement sont caractérisés par des principes et des valeurs, par exemple la démocratie à travers l'égalité de voix entre les membres. Ce groupement de personnes cherche à créer pour chacun de ses membres un service web composite suivant ses contraintes (moyens de chacun) comme par exemple "un site de réservation de voyages" d'une manière équitable c'est à dire que chaque membre ne sera pas lésé pour la simple raison que son budget est faible ou favorisé parce que son budget est trop élevé. Le but est donc d'optimiser la satisfaction du groupe. Le problème consiste à repartir d'une manière efficace un

ensemble de services web simples dans plusieurs services composites de contraintes différentes. Dans ce contexte, nous avons proposé deux approches de sélection ; la première est une approche optimale basée sur des méthodes AMCD et un score pour maximiser la satisfaction globale du groupe ; la deuxième est une méthode approchée basée sur les communautés et la théorie des jeux pour maximiser la satisfaction globale du groupe dans un environnement dynamique tout en proposant un temps de réponse rapide en comparaison avec la première approche.

2 Description de notre problématique

Dans cette section, Nous présentons le problème de la sélection de services web composites distribués. Pour le définir mathématiquement, nous avons besoin des hypothèses suivantes concernant le problème.

- Un groupe de clients qui recherchent services web composites avec des fonctionnalités similaires.
- Chaque service web composite est composé de services web issus de classes de services web.
- Chaque client fixe contraintes pour un service web composite.
- Chaque classe de services web propose une fonctionnalité requise par le service web composite.
- Chaque classe de services web est composée de services web.
- Un service web appartient au maximum à un et un seul service composite.

En notation mathématique, le problème peut être décrit comme suit.

- maximiser , où , , représente la valeur du service web appartenant à la classe.
- , où , représente la valeur du premier critère du service web appartenant à la classe assigné au client et représente la valeur de la première contrainte fixée par le client.
- ...
- , où
- ...
-
- =1, où , prend la valeur 1 si le service web appartenant à la classe est pris par un service composite. 0 sinon.

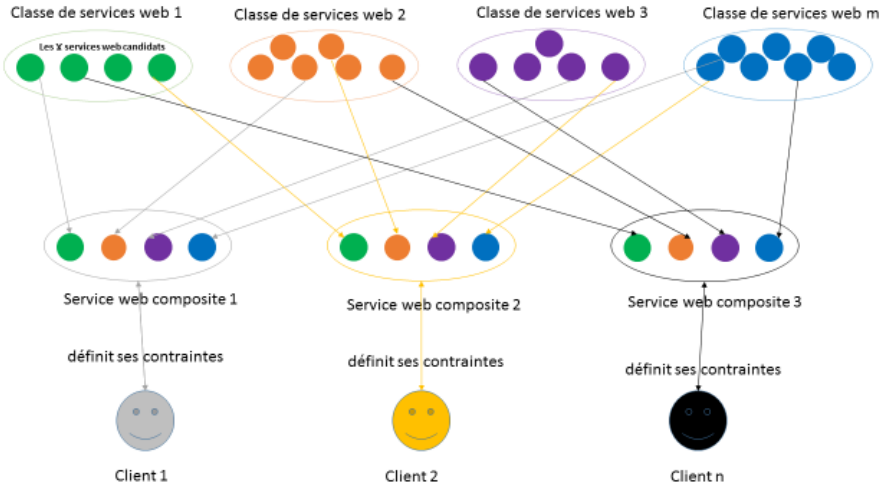

FIG. 6.1 : Problématique de sélection de services web composites distribués.

En résumé, le but est de maximiser la distribution de services web sur les services web composites souhaités de tel sorte que chaque membre du groupe de clients soit satisfait du service composite retourné. La figure 6.1 décrit la problématique de la sélection de services web composites distribués.

La motivation de notre contribution est de proposer deux solutions pour optimiser la satisfaction globale d'un groupe de clients suivant deux critères : le temps de réponse et l'optimalité de la réponse. Nous proposons dans la première solution un algorithme de sélection de services web composites distribués avec une satisfaction du groupe optimale mais avec un temps de réponse exponentiel suivant le nombre de services composites et le nombre de classes de services. Inversement dans la deuxième solution, nous proposons une heuristique de sélection de services web composites distribués avec un temps de réponse très rapide mais avec une satisfaction du groupe sous-optimale.

La figure 6.2 décrit les différentes étapes d'un processus de sélection de services web composites distribués. Nous citons ci-dessous les principales que sont :

- **L'étape de recherche** permet de rechercher toutes les configurations de services web composites possibles. En supposant qu'il existe services web candidats dans chaque classe, classes de services web et clients alors il existe configurations possibles. Nous appelons une configuration, un groupe de services composites distincts en termes de services web (voir figure6.1).

- **L'étape des éliminations** permet de réduire le nombre de configurations en éliminant celles qui ne respectent pas les contraintes clients. Si un service web composite dans une configuration ne respecte pas une ou plusieurs contraintes d'un ou plusieurs clients alors elle est retirée du groupe des configurations candidates.

- **L'étape de classement** permet de classer les configurations candidates en fonction d'un nouveau score appelé score de satisfaction. Il permet de sélectionner

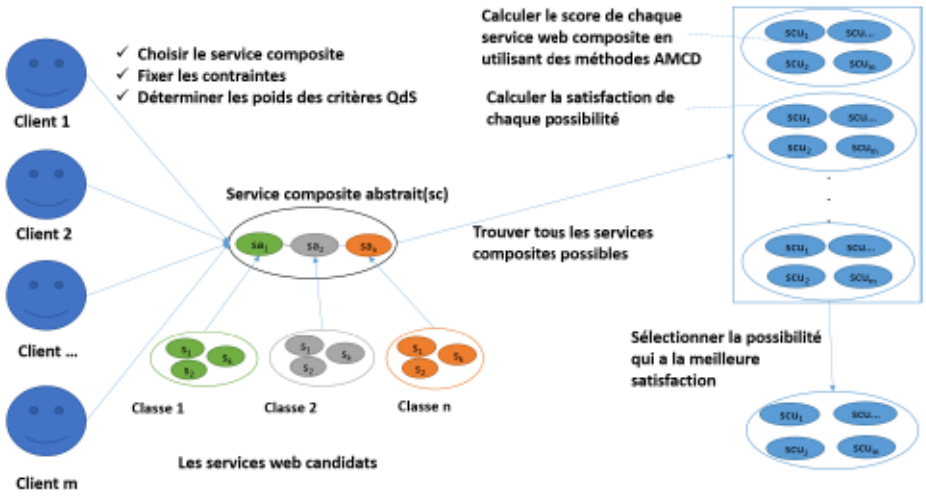

FIG. 6.2 : Processus de sélection de services web composites distribués.

la possibilité qui optimise au maximum la satisfaction du groupe formé par les N clients.

3 Approche optimale d'un groupe de clients pour la sélection de services composites

3.1 Étapes procédurales de notre approche

Les étapes procédurales de notre algorithme de la première solution décrivent d'une manière détaillée le processus de la figure 6.2.

- **Étape 1** : M clients choisissent des services composites ayant les mêmes paramètres fonctionnels avec leurs J contraintes. Chaque client i ($i \in [1, M]$) fixe la valeur de chacune de ces contraintes cs_j ($j \in [1, J]$).

- **Étape 2** : Trouver toutes les configurations possibles, sachant qu'un service appartient au maximum à un et un seul web composite ; ensuite éliminer les configurations qui ne respectent pas les contraintes.

- **Étape 3** : Trouver la solution idéale et la solution non idéale. La solution idéale représente les meilleures valeurs des critères QdS dans chaque classe de services web. La solution non idéale représente les pires valeurs des critères QdS dans chaque classe de services web. Techniquement, il s'agit de trouver les deux matrices respectivement des valeurs maximales v^{max} et des valeurs minimales v^{min} à partir des services web candidats pour chaque classe de services r.

  Si $j^{ème}$ critère représente un critère négatif comme le prix, le temps de réponses,... alors

Si critère représente un critère positif comme la réputation, la disponibilité, ... alors

avec et .

- **Étape 4 :** Calculer les valeurs des nouveaux services web composites (également les deux services composites et , respectivement, pour les deux matrices de valeurs maximales et de valeurs minimales) en utilisant des formules comme celles utilisées dans [122] pour les critères temps d'exécution, disponibilité, fiabilité et coût (voir la table 6.1). Un service web composite peut contenir quatre types de relation entre ses fonctionnalités (classes). Une relation peut être séquentielle, parallèle, conditionnelle ou en boucle [122]. Canfora et al. [14] donnent plus de détails sur les différentes formules utilisées pour calculer les valeurs d'un service web composite pour chaque type de relation. Pour plus de simplicité, nous supposons dans notre approche que le service web composite utilise exclusivement des relations séquentielles.

TAB. 6.1 : Calcul des valeurs d'un service web composite (séquentielles).[122].

Critère	Formules
Temps d'exécution	
Disponibilité	
fiabilité	
Cout	

- **Étape 5** : Calculer les poids des critères pour chaque client. Des techniques comme AHP, BWM et autres [94] sont recommandées pour faciliter et optimiser le processus de pondération des différents critères. Comme vu dans le chapitre précédent, une méthode comme AHP propose une simple comparaison par paire en donnant une note entre 1 et 9 pour comparer deux critères.

- **Étape 6** : Calculer le score de chaque service web composite pour chaque configuration en utilisant des techniques AMCD telles que SAW, VIKOR, TOPSIS, COPRAS, WPM et autres [94]. Pour chaque configuration qui satisfait les clients, nous avons calculé le score de chaque service web composite. Nous donnons ci dessous les étapes de la méthode SAW. Les étapes de la méthode SAW prennent en compte la matrice de décision de dimension (,), de

sorte que représente le nombre de services web composites distincts qui répondent aux besoins des clients. N est le nombre de critères pris en compte dans l'étape 5. est une configuration telle que .

- **Étape A** : Normaliser les valeurs des services web composites avec la formule suivante :

  Si critère représente un critère négatif, alors

  ———————— (6.1)

  Si critère représente un critère positif, alors

  ———————— (6.2)

  avec et

- **Étape B** : Calculer le score des services web composites :

  (6.3)

  avec , et est le poids du critère pour le client obtenu dans l'étape 5.

- **Étape 7** : Calculer le score de satisfaction de chaque configuration formée par services web composites distincts en utilisant les formules suivantes :

  (6.4)

  — (6.5)

  avec ,

- **Étape 8** : Sélectionner la possibilité avec la valeur maximale de .

4 Approche d'un groupe de clients pour la sélection de services web composites à base de communautés

4.1 Architecture de notre approche

Notre architecture de communautés de services web est représentée dans la figure 6.3. Les composants de cette architecture sont le fournisseur de services web,

le registre UDDI des services web simples, la configuration des communautés, le registre UDDI des services composites et le client de services web. Les communautés sont crées et détruites selon des contraintes spécifiques détaillées dans la section 4.3. Le fournisseur de services publie son service dans le registre UDDI ; UDDI permet de conserver les services web dans un annuaire. Ces informations sont transparentes pour les clients et peuvent être utilisées pour créer des services web composites (communautés). Deux configurations de communautés sont montrées dans la figure 6.3. Il peut s'agir, par exemple, d'un service de réservations de voyages ou d'un service de ventes en ligne . Les informations sur les configurations et les communautés sont stockées et publiées dans un nouveau registre UDDI dédié à la composition ; le client de services exprime son processus métier et ses exigences pour trouver son service composite dans le nouveau registre UDDI. La manière de publier et d'appeler des services web composites est similaire à l'architecture standard orientée service bien que les services web soient maintenant associés en communautés pour offrir un bon service composite dans des délais acceptables. Le nouveau registre UDDI garde l'historique des demandes clients afin de mieux cibler les exigences d'un profil client.

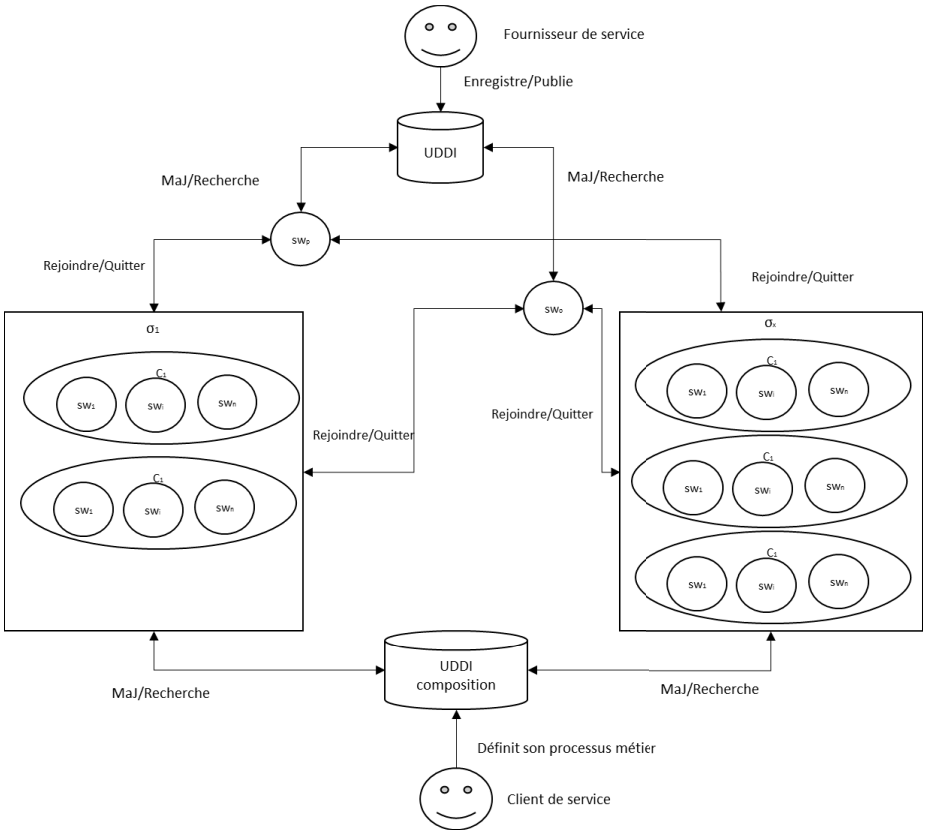

FIG. 6.3 : Architecture de notre approche de communautés.

4.2 Modèle formel de communautés

Certaines tâches complexes nécessitent l'exécution de plusieurs tâches plus simples, chacune pouvant être déléguée à un *service* spécialisé. Ce service peut être fourni par un ou plusieurs fournisseurs. Tous ces services web offrant les mêmes fonctionnalités peuvent être rassemblés dans une classe de services notée *sc* : tout service de cette classe peut être remplacé par un autre service de cette même classe pour exécuter cette tâche. Les services peuvent toutefois varier en qualité de service. Pour des raisons de simplicité, nous ne considérons que deux critères de qualité : le coût, noté *sco*, et la réputation, notée *srep*. Un client d'un service préfère un coût associé inférieur, mais une réputation plus élevée à ses exigences.

Définition 1 (Service). Un *service* est un quadruplet *sc srep sco* ou est son identifiant unique, *sc* est sa classe de services, *srep* est sa réputation, et *sco* est son cout.

L'ensemble de tous les services est noté . Bien que les services offrent une grande flexibilité, certaines classes de services sont souvent combinées pour proposer des services composites. Nous considérons que la composition de services nécessite exactement des classes de services dans l'ensemble . Nous notons l'ensemble des services web présents dans une classe de services : *sc* . Le but d'une *communauté* est de prendre un (et un seul) service web par classe afin de fournir une composition dynamique limitée par un budget donné :

Définition 2 (Communauté). Une *communauté* notée *cons* est définie par un identifiant unique et une contrainte de coût *cons* .

On note l'ensemble de toutes les communautés. Nous considérons qu'il existe un ensemble constant de communautés,*i.e.* .

Une communauté ne contient aucun service, mais des services peuvent être attribués à des communautés. Comme ces affectations varient dans le temps, nous les appellerons configurations :

Définition 3 (Configuration). Une *configuration de communautés* est une cartographie telle que pour *cons* *cons* et *sc* , si *sc* et *sc* , alors : *sc* *sc* *sc* et .

Dans une configuration, pour chaque communauté, au plus un service est attribué de chaque classe (ou si aucun service n'est attribué). De plus, chaque service ne peut être attribué qu'à une seule communauté à la fois. Lorsqu'un service est affecté à une communauté dans une configuration , on le note , omettant parfois l'identifiant si cela ressort clairement du contexte. dénote le fait que pour une classe de services *sc*, *sc* , c'est-à-dire que la communauté n'a pas de service pour une classe, donc on ne peut pas effectuer la composition entière. Nous étendons les projections *sco* et *srep* sur en mettant *sco* et *srep* . Nous notons l'ensemble des services inutilisés dans une configuration : . Nous notons comme l'ensemble de toutes les configurations.

Le coût d'une communauté dans une configuration est la somme des coûts de ses services : *cco* *sco* . Sa réputation est la moyenne de la réputation de ses services : *crep* $\frac{srep}{}$. À condition de ne pas dépasser son budget et de pouvoir fournir le service composite, une communauté reçoit une récompense en fonction de son coût et de sa réputation :

Définition 4 (Récompense). La *Récompense* d'une communauté dans une configuration , est définie par

if or *cco* *cons*
$\frac{crep}{cco}$ sinon

La récompense globale d'une configuration est la moyenne des récompenses des communautés :

L'objectif, lors de l'attribution d'un service à une communauté, est de maximiser cette récompense globale. Cependant, chaque communauté souhaite d'abord maximiser sa propre récompense. Nous supposons par conséquent que les communautés sont des agents rationnels en ce sens qu'elles évitent, si possible, des configurations défavorables. Ceci est modélisé en donnant aux communautés la possibilité d'échanger un de leur service par un autre service inutilisé :

Définition 5 (Échange). Dans une configuration , un *echange* de communauté pour un service est la configuration *swap* définie comme

pour *sc* *sc* si et *sc* *sc*
sc sinon

Par souci d'exhaustivité, si , nous définissons *swap* .

À partir d'une configuration de départ, une communauté qui souhaite modifier certains de ses services ne peut pas voler ceux déjà attribués à d'autres communautés. Par conséquent, il ne peut échanger ces services qu'avec des services inutilisés. A noter qu'il peut être fait au maximum échanges : un pour chaque classe de services. Par conséquent, nous définissons le macro-échange comme *swap*
tel que

swap *swap* *swap* *swap*

Nous notons *Swap* comme l'ensemble des configurations obtenues dans par macro-échange de communauté :

Swap *swap*

Une configuration est alors stable si aucun échange de ce type n'est avantageux pour aucune communauté.

Définition 6 (configuration avec équilibre de Nash). Une configuration a un *équilibre de Nash* si pour chaque , pour chaque *Swap* ,
.

Cette définition est basée sur celle de l'équilibre de Nash pour un jeu dans lequel toutes les communautés doivent sélectionner leurs services et recevoir chacune une récompense si le résultat est une configuration valide ou sinon tout le monde obtient si plusieurs communautés ont choisi le même service.

Remarque :

- **Nombre de services** : En réalité, il y a beaucoup plus de services disponibles pour chaque classe de services qu'il n'y a de communautés. Par conséquent, les configurations dans lesquelles une communauté ne dispose pas d'un service pour chaque classe générant une récompense nulle pour cette communauté pourraient être remplacées par des configurations dans lesquelles un service de la bonne classe est attribué à la communauté. Même dans le cas où cela dépasserait la contrainte budgétaire de la communauté, sa récompense ne pourrait pas baisser. Par conséquent, il existe toujours une configuration Nash stable où toutes les communautés ont toutes les classes de services. C'est pourquoi les configurations, où les communautés auxquelles ils manquent des services, peuvent être ignorées.

- **Combinaison des configurations** : En supposant qu'il existe communautés, classes de service, services disponibles dans chaque classe de services et en supposant que alors il existe configurations possibles où toutes les communautés ont un service par classe.

4.3 Communautés dynamiques

Trouver la configuration de communauté optimale pour un ensemble de services s'apparente à une variante du problème de sac à dos, connu pour être NP-difficile [97]. Bien que certaines heuristiques existent pour résoudre ce problème, ces solutions sont approximatives et un calcul exact, bien que optimal dans la récompense obtenue, peut prendre beaucoup de temps. De plus, les services web sont en pratique instables (ils peuvent être créés et arrêtés par les fournisseurs). Ainsi, les méthodes dynamiques qui permettent plus de flexibilité et de fiabilité que les méthodes statiques [62] sont devenues un domaine de recherche actif ces dernières années. Les opérations d'ajouts (A) ou de suppressions (D) peuvent à leur tour affecter les communautés existantes. Une possibilité est de recommencer le calcul de la configuration de communauté optimale en fonction de ce nouvel ensemble de services. Cela est toutefois peu pratique en raison du coût élevé en temps de calcul de la configuration optimale en ce qui concerne le rythme d'ajout ou de suppression de services. En conséquence, la solution que nous proposons est de changer dynamiquement les configurations à la volée, guidé par la récompense globale.

En cas de suppression d'un service appartenant à une communauté, cette récompense de communauté passe à 0 à moins qu'elle puisse choisir un autre service de la même classe de services dans la limite du budget alloué. Dans le cas d'un ajout, une communauté (ou plusieurs) peut vouloir échanger l'un de ses services contre le nouveau. Cela pourrait conduire à d'autres échanges, car le service qui a été remplacé est maintenant disponible pour d'autres communautés.

Afin de limiter le nombre de ces échanges et de maximiser la récompense globale, nous appliquons les contraintes suivantes aux échanges :

Nash-stabilité : Pour une communauté donnée, un échange n'est autorisé que s'il augmente sa récompense.

Optimalité globale : Si plusieurs communautés souhaitent échanger avec le même service, celui-ci est attribué afin de maximiser la récompense globale.

La condition de stabilité de Nash garantit qu'il peut y avoir au plus échanges après un ajout. Et comme le nombre de communautés pouvant vouloir un service donné

est également limité par m, seul un nombre linéaire de configurations potentielles est testé avant chaque échange. Cette approche est illustrée dans la figure 6.4 : le service avec une croix est ajouté et peut potentiellement être revendiqué par les quatre communautés. Cependant, la récompense globale augmente davantage si C_3 le reçoit, libérant ainsi un service pouvant être revendiqué par toutes les autres communautés.

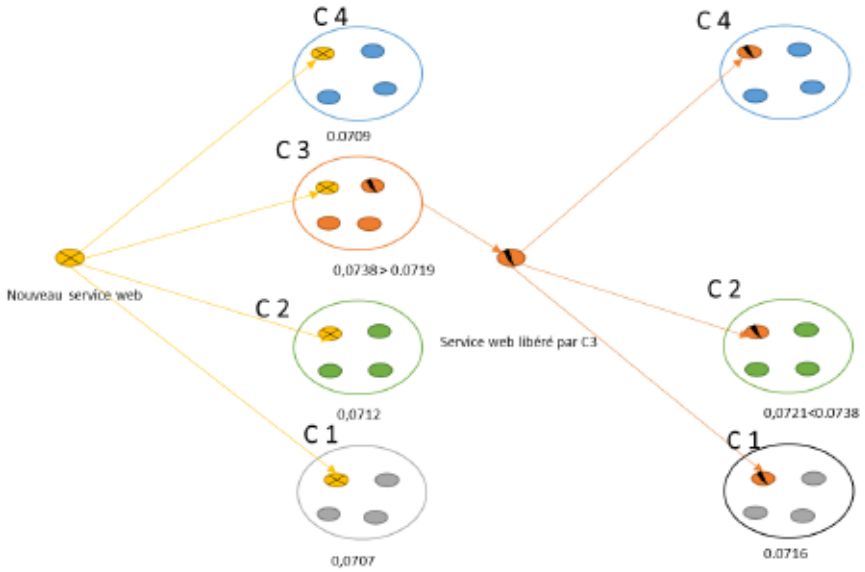

FIG. 6.4 : Ajout d'un nouveau service web.

En conséquence, les configurations disponibles après une suppression ou un ajout sont restreintes et deviennent traitables, en fonction de la rapidité des ajouts et des suppressions de services. Néanmoins, cette approche locale peut ne pas générer la valeur optimale de la récompense globale, seuls les changements locaux étant autorisés.

4.3.1 ▸ Création de communautés

Le choix de l'ordre de priorité pour la création de communautés est une tâche difficile. En fait, nous avons beaucoup de possibilités. Nous citons les deux plus importantes d'entre elles :

1. La communauté avec la plus grande contrainte en premier : cette stratégie est la plus appliquée dans le monde réel. Cependant, le point faible de cette méthode est le temps nécessaire à la recherche de la meilleure composition.

2. La communauté avec la contrainte la plus faible en premier : Cette stratégie est intéressante car elle augmente les chances de satisfaire toutes les communautés en termes de contraintes et consomme moins de temps que la stratégie citée précédemment car le nombre de services composites respectant la contrainte est moins élevé.

   Après la création des communautés, leurs récompenses sont déséquilibrées, notre système dynamique tend à équilibrer les bénéfices des communautés (jeu coopératif).

4.3.2 ▶ Addition

Le problème de l'addition de communautés est considéré du point de vue de la théorie des jeux en définissant le jeu suivant :

Joueurs : Considérez chaque communauté comme un joueur. le nombre de communautés détermine le nombre de joueurs impliqués dans la partie. Soit le nombre de communautés. Un joueur est noté , .

Stratégies : La stratégie proposée à chaque joueur consiste à ajouter un nouveau service web.

Gains : Le gain considéré pour chaque joueur sera le score de la communauté.

Dans ce jeu, chaque joueur (communauté) cherche à maximiser son gain en choisissant le service web avec le raisonnement "le plus à gagner en l'incluant ou le plus à perdre en ne l'ayant pas en tant que membre".

L'équilibre de Nash de ce jeu peut être une situation dans laquelle aucun joueur (aucune communauté) ne peut améliorer ses gains par déviation unilatérale (en modifiant ses services web uniquement par lui-même).

La définition "configuration Nash-stable " donne le principe de base pour la sélection d'un nouveau service web. Sur la base de ce principe, une méthode de la théorie des jeux est proposée pour ajouter un nouveau service web sur le marché des communautés. Lorsqu'un nouveau service est ajouté dans les communautés :

1. Calculer les récompenses de chaque communauté.
2. Calculer les récompenses globales.
3. Choisir la meilleure récompense globale.

4.3.3 ▶ Suppression

Le problème de suppression de communautés peut être considéré comme un problème de substitution susceptible de remplacer le service supprimé par un service appartenant aux mêmes classes de services.

Un algorithme de base est donné pour ajouter un nouveau service dans la communauté. Lorsqu'un service est supprimé de la communauté :

1. Éliminer les services web de la classe contenant le service supprimé qui ne respectent pas la contrainte de la communauté.
2. Calculer la récompense de la communauté en testant chaque service de la classe contenant le service supprimé.
3. Choisir le meilleur service avec la meilleure récompense communauté.
4. Calculer la nouvelle récompense de la nouvelle configuration.

4.4 Une étude de cas

Dans cette section, nous donnons un exemple simple pour expliquer le fonctionnement de nos algorithmes. Dans la table 6.2, nous avons quatre classes de services avec des fonctionnalités différentes appelées Vérifier la météo, Réservation d'une salle, Réservation d'une voiture et Paiement. Chaque classe de service contient quatre services différents ws (**a**, **b**) où **a** représente le coût et **b** la réputation du service web.

Tab. 6.2 : Notre étude de cas.

Nom de la classe	Vérifier la météo	Réserver une chambre	Réserver une voiture	Paiement
	sw(12,2)	sw(7,3)	sw(13,4)	sw(17,5)
	sw(15,4)	sw(4,1)	sw(1,0)	sw(14,4)
	sw(2,1)	sw(19,5)	sw(9,2)	sw(5,1)
	sw(8,3)	sw(9,3)	sw(11,2)	sw(10,3)

Dans la table 6.3, nous proposons une configuration composée de quatre communautés. chaque communauté est composée de quatre services avec des fonctionnalités complémentaires afin de proposer une nouvelle fonctionnalité appelée réservation de voyage.

Tab. 6.3 : Contrainte de chaque communauté.

Nom de la communauté	contrainte cout
	80
	60
	40
	20

- **Création de communautés** : Nous avons utilisé dans l'étape de création de communautés, le principe de Shortest job next (SJN) pour réduire le nombre d'opérations visant à former une communauté ainsi que pour réduire le temps nécessaire à la création de communautés. En fait, les communautés avec la contrainte de coût la plus basse ont le plus petit nombre possible de services composites qui respectent les contraintes. Suivant ce raisonnement, l'ordre de création de la communauté est le suivant :

  1. La création de la communauté 4.
  2. La création de la communauté 3.
  3. La création de la communauté 2.
  4. La création de la communauté 1.

  La table 6.4 donne toutes les compositions possibles qui respectent la contrainte de coût de la communauté 4. Nous faisons la même opération pour la création des autres communautés .

TAB. 6.4 : Création de la communauté 4.

Vérifier la météo(*sco*,*srep*)	Réserver une chambre	Réserver une voiture	Paiement	,	
(12,2)	(4,1)	(1,0)	(5,1)	(20,1)	0.05
(2,1)	(7,3)	(1,0)	(5,1)	(15,1.25)	0.0833
(2,1)	(7,3)	(1,0)	(10,3)	(20,1.75)	0.0875
(2,1)	(4,1)	(1,0)	(5,1)	(12,0.75)	0.0625
(2,1)	(4,1)	(1,0)	(10,3)	(17,1.25)	0.0735
(2,1)	(4,1)	(9,2)	(5,1)	(20,1.5)	0.075
(2,1)	(9,3)	(1,0)	(5,1)	(17,1.25)	0.0735
(8,3)	(4,1)	(1,0)	(5,1)	(18,1.25)	0.0694

La table 6.5 donne la configuration finale de la réservation de voyage. la récompense de la configuration est :

TAB. 6.5 : Configuration résultante de la création des communautés.

Nom de la classe	Contraintes	Vérifier la météo	Réserver une chambre	Réserver une voiture	Paiement	,	
	80	(12,2)	(4,1)	(11,2)	(14,4)	(41,2.25)	0.0549
	60	(15,4)	(19,5)	(9,2)	(17,5)	(60,4)	0.0667
	40	(8,3)	(9,3)	(13,4)	(11,2)	(35,2.75)	0.0786
	20	(2,1)	(7,3)	(1,0)	(10,3)	(20,1.75)	0.0875

- **Ajout d'un service web** :

  Supposons que le service (8,2) soit ajouté à la classe de service nommée "Paiement".

  Chaque communauté essaie d'ajouter le nouveau service pour promouvoir la configuration. Dans la table 6.6, nous avons un exemple d'ajout d'un service web dans la communauté . Nous faisons la même opération pour le reste des communautés.

TAB. 6.6 : Configuration résultante de l'ajout du nouveau service dans la communauté 1.

Nom de la classe	Contraintes	Vérifier la météo	Réserver une chambre	Réserver une voiture	Paiement	,	
	80	(12,2)	(4,1)	(11,2)	(8,2)	(35,1.75)	0.05
	60	(15,4)	(19,5)	(9,2)	(17,5)	(60,4)	0.0667
	40	(8,3)	(9,3)	(13,4)	(11,2)	(35,2.75)	0.0786
	20	(2,1)	(7,3)	(1,0)	(10,3)	(20,1.75)	0.0875

La table 6.7 donne le score de satisfaction globale de chaque configuration suite à l'ajout d'un nouveau service.

TAB. 6.7 : Satisfaction globale de chacune des quatre configurations.

Nom de la classe	Contraintes	Vérifier la météo	Réserver une chambre	Réserver une voiture	Paiement	,		
	80	(12,2)	(4,1)	(11,2)	(8,2)	(35,1.75)	0.05	0.0707
	60	(15,4)	(19,5)	(9,2)	(8,2)	(51,3.25)	0.0637	0.0712
	40	(8,3)	(9,3)	(13,4)	(8,2)	(32,2.75)	0.0859	0.0738
	20	(2,1)	(7,3)	(1,0)	(8,2)	(18,1.5)	0.0833	0.0709

La table 6.8 donne le score de satisfaction globale de chaque configuration suite à l'ajout du service libéré par la communauté 3.

TAB. 6.8 : Satisfaction globale de chacune des deux configurations.

Nom de la classe	Contraintes	Vérifier la météo	Réserver une chambre	Réserver une voiture	Paiement	,	
	80	(12,2)	(4,1)	(11,2)	(11,2)	(38,1.75)	0.0461	0.0716
	60	(15,4)	(19,5)	(9,2)	(11,2)	(54,3.25)	0.0602	0.0721

Le service libéré par la communauté 3 est ajouté à la communauté 2 car 0.0716 > 0.0721.

5 Résultats numériques

5.1 Modèle de la simulation

Afin d'étudier les avantages concrets qu'apporte la solution optimale et la solution heuristique pour la séléction de services web composites pour un groupe de clients, nous avons calculé la complexité de la solution optimale puis nous avons réalisé une série de simulations sur un ordinateur portable avec un processeur CPU i5 1,90 GHz et d'une mémoire de 4 Go à l'aide de notre programme implémenté en C ++ sur la plateforme Qt. Les résultats sont issus de tests simulant la sélection de services web composites sur l'ensemble de données . considére la base de données réelles contenant 2507 services web [61]. Pour mesurer l'efficacité de notre approche heuristique, nous avons effectué plusieurs simulations avec notre programme. Nous avons considéré les trois cas suivants pour toutes nos simulations :

- **cas 1** : deux communautés et deux classes de services web.
- **cas 2** : trois communautés et deux classes de services web.
- **cas 3** : deux communautés et trois classes de services web.

5.2 Évaluation de la complexité du pire cas des trois cas étudiés

En supposant que le nombre de services web par classe de services soit égale à k alors le nombre de configurations notée :

- nc(cas1) = = pour
- nc(cas2) = = pour
- nc(cas3) = = pour

Comme le nombre de services par configuration est égale à 2 pour la cas 1 et le cas 3 quand il est égale à 3 pour la cas 2 alors la complexité notée :

- comp(cas1) = =
- comp(cas2) = =
- comp(cas3) = =
- où t représente le temps de calcul du score d'un service web composite.

Dans le pire cas, c'est à dire que toute les configurations répondent aux contraintes des clients. La complexité du cas 2 est plus grande que la complexité du cas 3 qui est plus grande que la complexité du cas 1.

La table 6.9 indique le nombre de configurations ainsi que le nombre de web services composites par cas. Comme on peut le constater, la complexité du cas 2 devient supérieure au cas 3 à partir de k=9.

TAB. 6.9 : Le nombre de configurations et le nombre services web composites pour chaque cas.

Nombre de services par classe	(cas 1)	(cas 2)	(cas 3)	(cas 1)	(cas 2)	(cas 3)
3	36	36	216	72	108	432
4	144	576	1728	288	1728	3456
5	400	3600	8000	800	10800	16000
6	900	14400	27000	1800	43200	54000
7	1764	44100	74088	3528	132300	148176
8	3136	112896	175616	6272	338688	351232
9	5184	254016	373248	10368	762048	746496
10	8100	518400	729000	16200	1555200	1458000
100						

5.3 Évaluation du temps d'exécution et du ratio de satisfaction

La table 6.10 indique les valeurs (ou les fourchettes de valeurs) des critères QdS ainsi que la contrainte (temps de réponse) des clients considérés dans chaque cas.

TAB. 6.10 : Les données en entrée.

Données	Valeurs
Temps de réponse	[37, 4989.67]
Disponibilité	[7, 100]
Cas1(C1,C2)	(300,400)
Cas2(C1,C2,C3)	(200,300,400)
Cas3(C1,C2)	(400,600)

5.3.1 ▶ Ajout

Pour la première série de simulations, le temps d'exécution de la solution heuristique et la solution optimale est calculé suivant le nombre de communautés et le nombre de classes de services web pour l'ajout de nouveaux de services web.

Les courbes illustrées dans la figure 6.5 représentent l'évolution du temps de réponse de chaque méthode de classement en faisant varier le nombre de services web ajoutés dans chaque cas. Nous avons ajouté des nouveaux services web dans le processus à chaque nouvelle simulation pour calculer les différentes valeurs du temps d'exécution. Les résultats montrent que la méthode heuristique a un temps d'exécution presque nul quand la méthode optimale augmente d'une manière exponentielle lorsque le nombre de services web ajoutés est augmenté. Comme constaté dans la complexité des trois cas, le nombre de communautés influe plus que le nombre de classes de services sur le temps d'exécution.

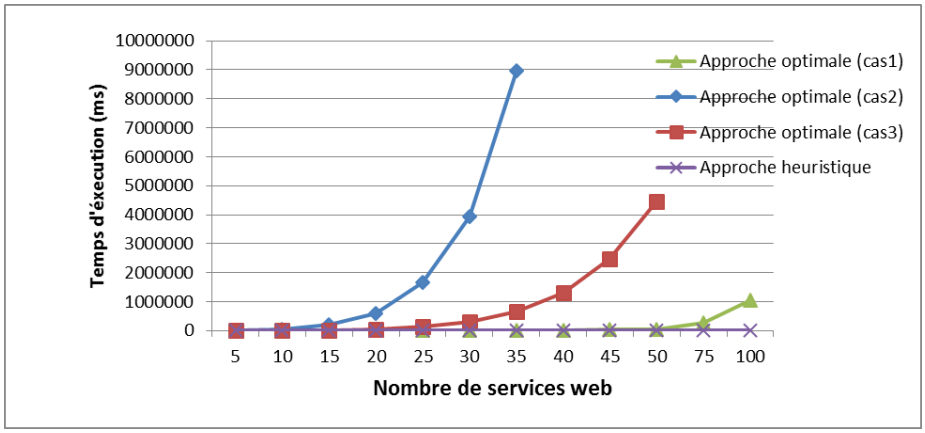

FIG. 6.5 : Comparaison des temps de réponse de l'ajout de services web.

5.3.2 ▶ Ratio de satisfaction des deux approches

Dans la deuxième série de simulations, le ratio de satisfaction des deux approches est calculé afin d'évaluer la satisfaction des clients par rapport aux deux approches.

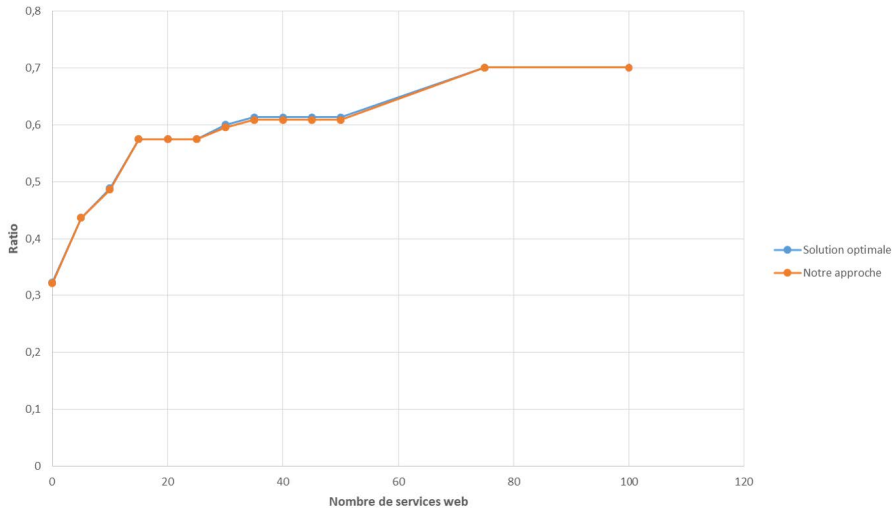

FIG. 6.6 : Comparaison des ratios de satisfaction des deux approches dans le cas de deux communautés et deux classes de services web.

Les courbes illustrées dans les figures 6.6, 6.7 et 6.8 représentent l'évolution du temps de réponse de chaque méthode de classement en faisant varier le nombre de services web supprimés dans chaque cas. Nous avons ajouté des services web dans le processus à chaque nouvelle simulation pour calculer les différentes valeurs du temps d'exécution. Les résultats montrent que la méthode heuristique a un ratio de satisfaction très proche du ratio de satisfaction de la solution optimale ; plus le nombre de service web augmente plus les deux ratios se rapprochent jusqu'à devenir égaux à certains points. Plus le nombre de communautés et le nombre de classes augmentent

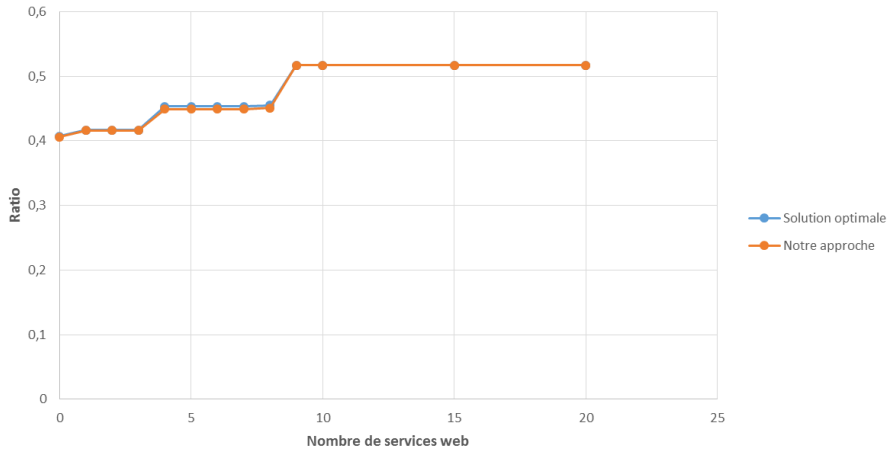

FIG. 6.7 : Comparaison des ratios de satisfaction des deux approches dans le cas de trois communautés et deux classes de services web.

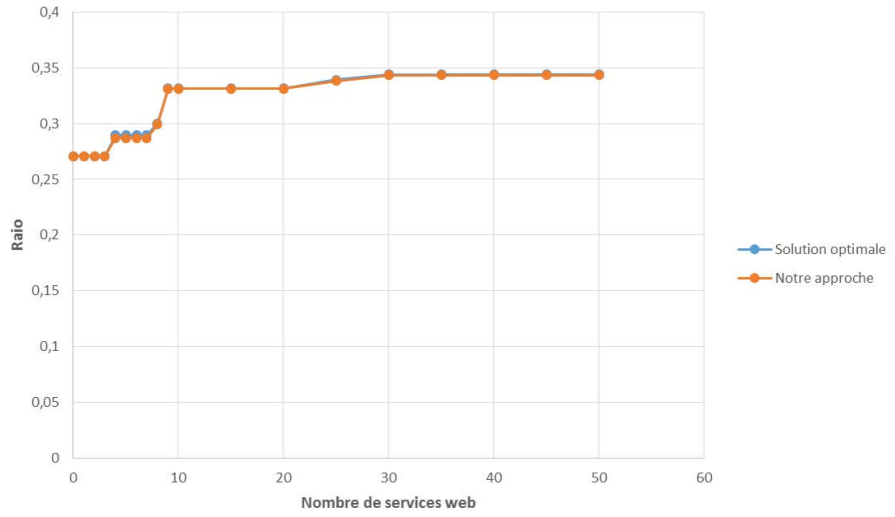

FIG. 6.8 : Comparaison des ratios de satisfaction des deux approches dans le cas de deux communautés et trois classes de services web.

plus le ratio de satisfaction augmente. Le nombre de classes influe plus que le nombre de communautés sur le ratio de satisfaction.

6 Conclusion

Dans ce chapitre, nous avons décrit une problématique liée à la sélection de services composites distribués sur un groupe de clients. Nous avons décrit puis propo-

sé deux approches pour résoudre ce problème. Ces approches permettent de donner aux clients une satisfaction optimale ou sous-optimale. Dans notre deuxième solution, nous avons proposé une approche basée sur un système dynamique, elle peut contenir une séquence d'opération d'addition et de suppression. L'addition est basée sur la théorie des jeux pour ajouter un nouveau service à une communauté. La suppression est basée sur la sélection de service web simple optimal (voir chapitre 4) pour remplacer un service web supprimé. Des algorithmes de suppression et d'addition sont proposés pour gérer et maintenir les communautés en fonction de notre modèle proposé. Une étude de cas a été présentée pour expliquer le fonctionnement de la méthode heuristique. L'analyse théorique et expérimentale des résultats confirment que le modèle heuristique proposé est plus performant en termes de temps de réponse avec des résultats satisfaisant pour la sélection de services composites distribués. En particulier, les avantages du modèle proposé deviennent plus clairs et significatifs lorsque le nombre de services augmente. Le Chapitre suivant propose des méthodes d'indexation pour optimiser le temps de découverte des services web simples ou composites.

7

Découverte de services web basée sur l'indexation

1 Introduction

Nous décrivons dans ce chapitre nos approches d'indexation de découverte de services qui reposent sur la construction d'index à plusieurs niveaux que nous avons appelées ITACOM (Indexation de Tous les Arbres des services web COMposites) pour l'indexation des services web composites stockés dans le registre UDDI de composition(voir l'architecture chapitre 6) et INWEB (Indexation des arbres d'eNtrées des services WEB), INOWEB (Indexation des arbres d'eNtrées et de sOrties des services WEB) pour l'indexation de services web simples stockés dans le registre UDDI.

2 Description de nos approches d'indexation de services web simples

Nos deux approches d'indexation de services web simples se concentrent sur les paramètres des services web et se présentent sous forme d'un index à plusieurs niveaux. Nos index représentent chaque service par chacun de ses paramètres et pour optimiser la recherche, nous avons décidé de trier les paramètres de chaque service puis de construire un arbre avec les dits paramètres où chaque nœud de l'arbre est représenté par un paramètre alors que chaque feuille est un ensemble de services ayant comme paramètres les clés des nœuds nous ayant conduit à cette feuille. Le fait de trier les paramètres nous permet de choisir lequel représenter en premier dans l'arbre mais aussi, à l'arrivée d'une requête, de savoir lequel rechercher. Nous rajoutons deux niveaux supérieurs afin de faciliter la recherche de la bonne feuille. Le premier consiste au regroupement des services ayant le même nombre de paramètres dans un même arbre (un sous arbre est construit pour chaque nombre de paramètres existants dans le répertoire). Le second facilite la détection du bon arbre grâce à la mise en place d'une structure de recherche liant chaque arbre au nombre de paramètres que contient les services qu'il regroupe.

Dans ce chapitre, nous distinguons le niveau et la profondeur de l'index :

- Le niveau d'index indique la décomposition de la hiérarchie d'index en termes logiques. Un niveau peut donc consister en une arborescence impliquant la même structure logique qui se répète.
- La profondeur de l'index indique la taille globale de la hiérarchie physique de l'index en termes d'étapes physiques. Un même niveau logique peut donc contenir plusieurs étapes et avoir une profondeur supérieure à un.

2.1 Notation et définitions

Nous avons utilisé la notation et les définitions suivantes pour définir notre approche :

Notation 1. *Soit S l'ensemble contenant tous les services web de notre annuaire.*

Definition 1. *Un service est représenté par = (entrées, sorties) où entrées correspond à l'ensemble des paramètres d'entrées et sorties à l'ensemble des paramètres de sorties. Il peut également être représenté par entrées sorties*

Definition 2. *La découverte de services est représentée par (, ,) = entrées sorties , le symbole est utilisé sur le contexte de l'ontologie. Par exemple est vrai si et seulement si est une sous-classe de dans l'ontologie. Nous n'irons pas plus loin avec les ontologies car cela ne concerne pas notre travail.*

Definition 3. *L'ajout de service est défini par Ajouter (,) où l'objectif est simplement d'ajouter le service web " " dans l'ensemble " " tout en respectant les règles imposées par la structure (la méthode) utilisée.*

Definition 4. *$l = |S|$ est le nombre de services contenus dans un annuaire. $S = \{s_1, s_2, s_3, ..., s_l\}$*

Definition 5. *n est le nombre maximal d'entrées qu'un service peut avoir, noté $n = \max(|s_i.inputs|)$ avec $i \in [1, l]$*

Definition 6. *n' est le nombre maximal de sorties qu'un service peut avoir, noté $n' = \max(|s_i.outputs|)$ avec $i \in [1, l]$*

Definition 7. *m est le nombre total d'entrées uniques dans le référentiel (chaque entrée est comptée une fois), noté $m = |Single(\{s_i.inputs\})|$*

Definition 8. *m' est le nombre total de sorties uniques dans le référentiel (chaque sortie est comptée une fois), noté $m' = |Single(\{s_i.outputs\})|$*

En utilisant l'exemple suivant, nous définissons :
$S = \{s_1, s_2, s_3, s_4\}$, $s_1 = \{\{b, a\}, \{c, d\}\}$ $s_2 = \{\{d, b, c\}, \{a\}\}$ $s_3 = \{\{d, c\}, \{a, c\}\}$ $s_4 = \{\{d\}, \{d\}\}$ nous avons de la définition précédente :

1. $l = |\{s_1, s_2, s_3, s_4\}| = 4$,
2. $n = |\{d, b, c\}| = 3$,
3. $n' = |\{c, d\}| = |\{a, c\}| = 2$,
4. $m = |\{a, b, c, d\}| = 4$,
5. $m' = |\{a, c, d\}| = 3$.

2.2 Indexation des paramètres d'entrées des services web

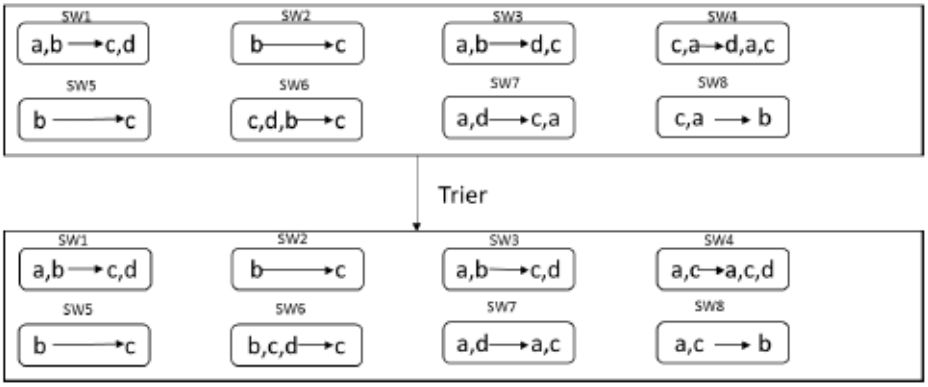

FIG. 7.1 : Exemple d'un échantillon de services web.

INWEB se concentre sur les entrées des services web. La construction est faite en prenant en compte les entrées seulement pour représenter les feuilles de l'arbre par un ensemble de classes où chacune regroupe les services offrant les mêmes sorties. Pour simplifier les choses, lorsqu'une demande arrive, nous calculons d'abord le nombre d'entrées recherchées, puis nous les trions (voir la figure 7.1). Nous parcourons les nœuds de l'arbre à la recherche de la feuille appropriée. Après l'avoir trouvée, il suffit de comparer les sorties recherchées avec les sorties de chaque classe regroupant la

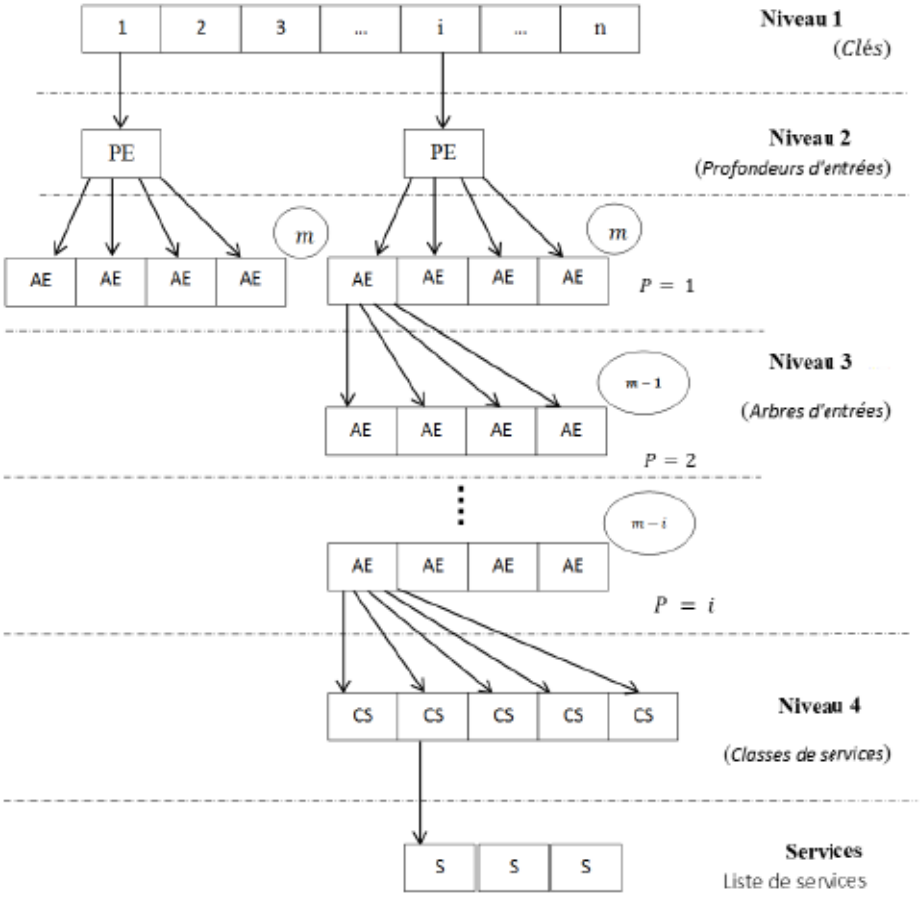

FIG. 7.2 : Architecture de l'index INWEB.

feuille et si la concordance est faite alors chaque service contenu dans cette classe correspond aux fonctionnalités de la recherche.

Comme on peut le constater à la figure 7.2, INWEB comporte quatre niveaux. Le niveau le plus bas est l'ensemble des classes regroupant les services en fonction de leurs entrées et de leurs sorties. Le niveau supérieur est un ensemble d'arbres destiné à faciliter la découverte des classes du dernier niveau. Le deuxième niveau est un ensemble de classes regroupant les arbres du troisième niveau par leurs profondeurs et enfin le premier niveau est une structure de recherche qui facilite la découverte des classes du deuxième niveau. Le nombre d'entrées associées à une *classe de services* n détermine la profondeur de l'entrée d'index pour cette classe. Ainsi, la classe $n = 1$ a une profondeur d'indice égale à un, etc. La largeur des arbres du niveau 3 diminue à

mesure que chaque arbre est parcouru de haut en bas (m, à $m - i$), où i est le nombre d'entrées dans la *classe de services*. Le niveau 2 détermine l'entrée dans les différents arbres.

Nous donnons à chaque niveau décrit(4 niveaux) un exemple sur le processus d'indexation suivant l'échantillon de services composites de la figure7.1 composé des services s_1, ..., s_8 caractérisés par leurs paramètres d'entrée et de sortie. Ces paramètres sont triés pour chaque service par ordre alphabétique. Le niveau 1 de l'index représente les entrées triées par ordre croissant du nombre de paramètres d'entrée par service. L'entrée 1 regroupe ainsi tous les services web ayant un seul paramètre d'entrée, etc. Le niveau 2 regroupe les entrées dans les arborescences accédant aux différentes listes de services web ayant les mêmes paramètres d'entrée. Le niveau 3 contient des arbres de paramètres d'entrée triés de gauche à droite. Un chemin existe dans cette arborescence si et seulement s'il existe au moins un service web dans la liste identifiée par l'ensemble des entrées constituant le chemin. Le niveau 4 contient les entrées pointant vers les listes de services web ayant les mêmes entrées et sorties. La figure 7.3 montre le processus d'indexation INWEB de l'échantillon de la figure 7.1.

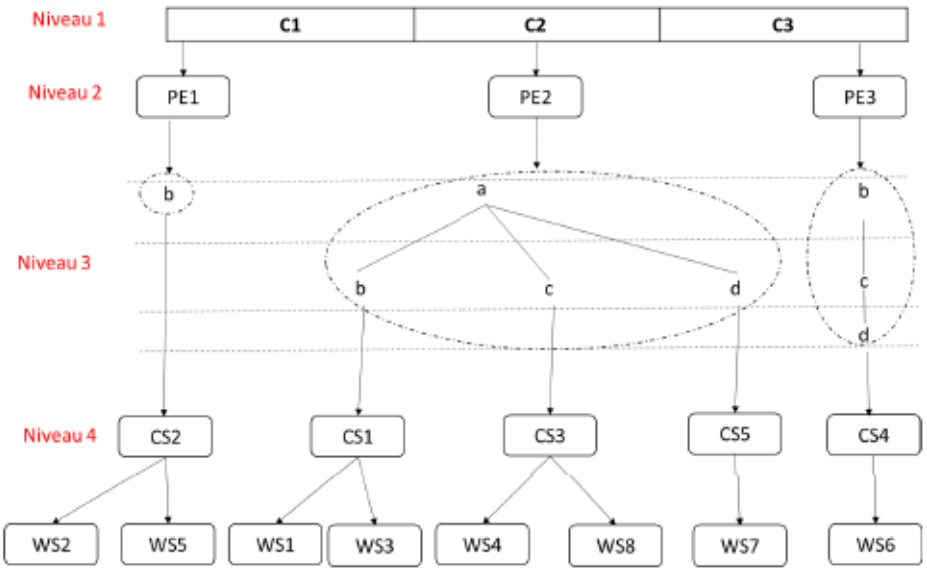

FIG. 7.3 : Exemple d'indexation INWEB de l'échantillon de services web.

2.2.1 ▸ Niveau 4 (*Classes de services*) :

Ce niveau est identique au niveau 3 de la méthode MLIM [119]. Par conséquent, nous construisons N_4 à partir de S de la même manière. Les classes obtenues dans cet ensemble s'appellent *classes de services*. Pour rappel, il s'agit de regrouper des services ayant les mêmes entrées dans la même classe.

2.2.2 ▸ Niveau 3 (*Arbres d'entrées*) :

Ce niveau consiste en la construction d'un arbre qui représente les entrées triées de chaque *classe de services*(*CS*) de l'ensemble N_4. Les différents nœuds de l'arbre sont appelés "*Arbre d'entrées*" ou "*AE*".*Nous représentons AE par : AE= {k,*

p, entrées, ASE, CSS}, entrées est un ensemble de paramètres, ASE est un ensemble de AE, et enfin CSS est l'ensemble de Classes de services.

Definition 10. *est un ensemble d'AE défini par les règles suivantes :*

(a) *: Toute classe de services de est représentée au moins une fois dans , et si une classe de services est représentée dans alors elle doit forcément appartenir à .*

(b) *: La profondeur, la clé et les inputs rendent un AE unique dans l'ensemble .*

La stratégie du choix de clé représentant les *AE* :

Le choix de la clé de chaque *AE* est crucial dans cette méthode car il dirige la construction de l'index et par conséquent la recherche. Comme expliqué précédemment, nous avons choisi d'adopter l'ordre alphabétique dans notre cas, ce qui veut dire que lorsque nous voulons choisir une clé nous prenons la clé la mieux classée alphabétiquement si l'index se construit de haut en bas pour représenter un *AE*, et l'inverse dans le cas où l'index se construit de bas en haut (cela importe peu tant que nous respectons l'ordre choisi dans la recherche).

2.2.3 ▶ Niveau 2 (*Profondeurs d'entrées*) :

A ce niveau, nous regroupons les arbres d'entrées (*AE*) ayant la même profondeur dans la même classe nommée " *Profondeur d'entrées*" ou "*PE*".

Definition 11. *une Profondeur d'entrées est définie par PE = (p, classes), où p est la profondeur et classes est un ensemble de AE tel que . elle peut également être représentée par PE: .*

Definition 12. *est un ensemble PE défini par les règles suivantes :*

(a) *PE : Tout AE de est représenté au moins une fois par une PE dans , et si une PE représente un AE alors cette AE appartient à l'ensemble .*

(b) *: La profondeur d'une PE la rend unique dans l'ensemble .*

2.2.4 ▶ Niveau 1(*Clés*) :

Une structure qui facilite la recherche est mise en place (table de hachage, B-Arbre). Elle relie chaque profondeur à sa *PE*. La profondeur ici dénote le nombre d'*entrées* caractérisant les SW contenus dans la *PE* correspondante.

Definition 13. *une clé d'une profondeur est définie par C = (c,p), où c est une clé et p est une profondeur de l'ensemble P qui est défini par : : Toute PE de est représentée par sa profondeur dans P, et toute profondeur de P représente une seule PE de .*

2.3 Indexation des paramètres d'entrées et de sorties des services web

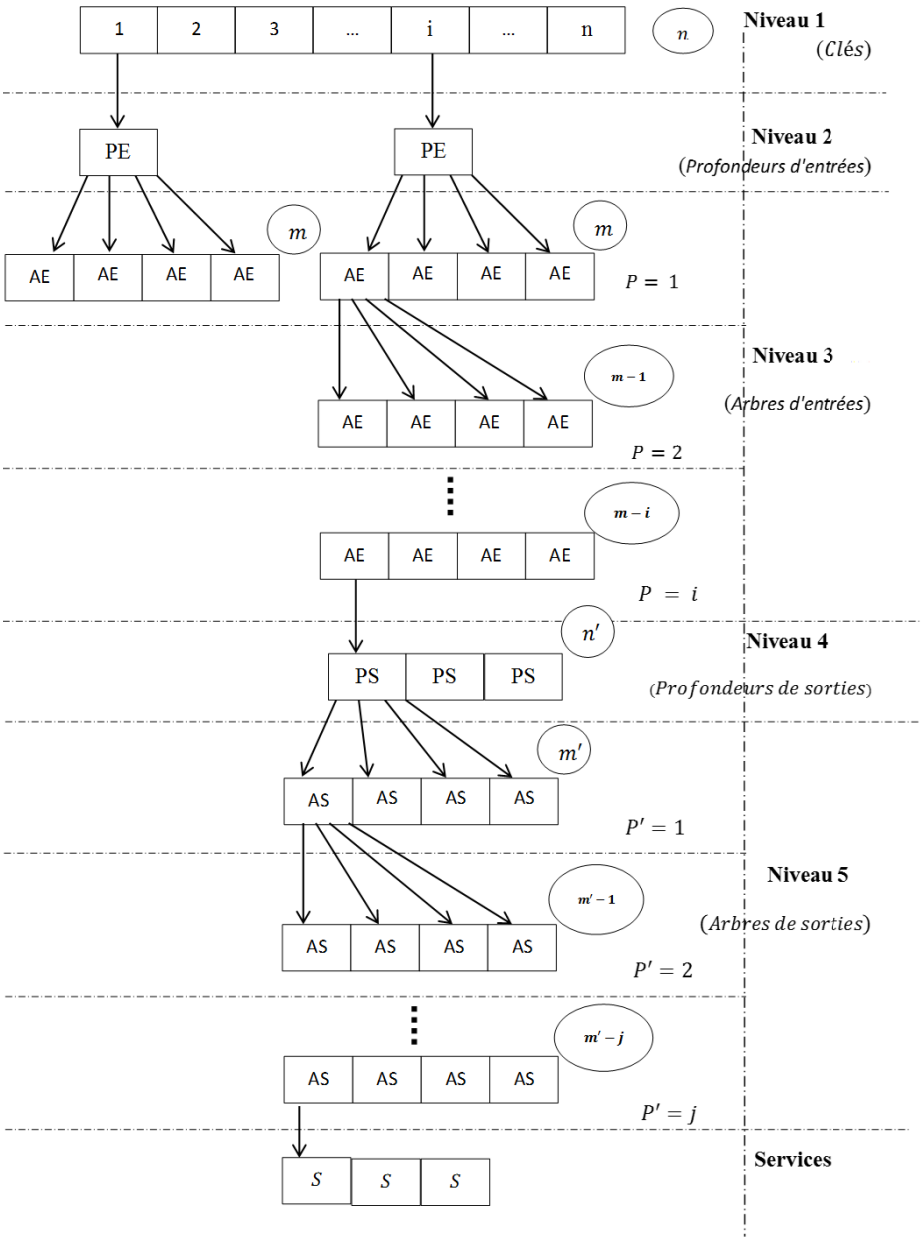

FIG. 7.4 : Architecture de INOWEB.

INOWEB se différencie de la première dans le dernier niveau. Au lieu de construire des *classes de services* nous construisons des *Arbres de sorties* basés sur les *sorties* en reprenant les mêmes principes que pour la construction des *Arbres d'entrées* (voir la

figure 7.4). Cela peut être intéressant s'il y a une grande redondance dans la fonctionnalité globale des SW dans le répertoire. Cela conduira aussi à une utilisation assez grande en ressource mémoire. Dans la première variante, l'accès aux *sorties* des services web se faisait séquentiellement lorsque nous atteignons la feuille de l'arborescence *AE*, à la recherche des différentes configurations des paramètres de *sorties* caractérisant les services de la *classe de services* recherchée. Cela peut prendre énormément de temps si le nombre de *services web* par feuille *classe de services* est très grand. L'ajout des *profondeurs de sorties* pour dénoter le nombre de paramètres de *sorties* et des *Arbres de sorties* pour construire l'arborescence des chemins, permet d'accélérer l'accès aux listes de services ayant les mêmes *entrées* et *sorties*. Les niveaux 4 et 5 sont introduits pour affiner l'index. Le niveau 4 contient les *profondeurs de sorties*. Le niveau 5 les *Arbres de sorties* menant vers les *classes de services*.

Contrairement à INWEB où l'accès aux *services* est effectué en parcourant de manière séquentielle la table du dernier niveau (*Classe de services*) à la recherche de la clé du *services web* correspondant aux paramètres de sortie, dans INOWEB, l'accès est indexé par le chemin de l' *Arbre de sorties* qui conduit aux *classe de services* correspondant aux paramètres de sortie inclus dans ce même chemin.

La figure 7.5 montre le processus d'indexation INOWEB de l'échantillon de la figure 7.1.

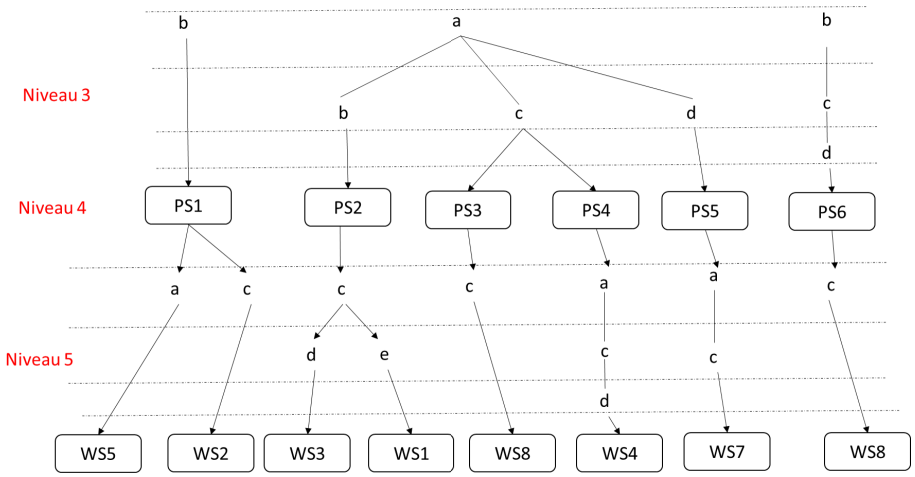

FIG. 7.5 : Exemple d'indexation INOWEB de l'échantillon de services web.

2.3.1 ▶ Opérations sur les index proposés

Nous présentons ci-dessous les différentes étapes des algorithmes pour l'ajout, la recherche, la suppression et la mise à jour de services dans notre solution pour l'algorithme INWEB. Il est à noter que les algorithmes peuvent être facilement étendus à l'algorithme INOWEB.

1. Addition

   Pour ajouter un nouveau service web à la structure, il faut :

   - Trier les *entrées* du nouveau service.

- Vérifier si une *PE* existe pour le nombre d'*entrées* correspondant. Sinon, il faudra la créer et l'ajouter à la structure de recherche.
- Pour la *PE* sélectionnée, vérifier s'il existe un *AE* ayant comme clé la première entrée . Sinon, il faudra le créer et l'ajouter à la *PE*, puis supprimer le premier élément de l'ensemble des *entrées* du nouveau service.
- Pour l'*AE* sélectionné, vérifier s'il existe un *AE* ayant comme clé le premier élément des entrées. Sinon, le créer et l'ajouter à la structure de recherche contenue dans l' *AE* sélectionné, puis supprimer le premier élément de l'ensemble des *entrées* du nouveau service.
- Répéter l'opération précédente tant que l'ensemble des entrées du nouveau service n'est pas vide.
- Pour l'*AE* sélectionné, parcourir toutes les classes de services qu'il contient d'une manière séquentielle et vérifier pour chacune si l'ensemble des sorties correspond à l'ensemble des sorties du nouveau service. Si aucune classe ne remplit cette condition, nous devons créer une nouvelle classe de services et l'ajouter à l'ensemble des classes de services de l'*AE* sélectionné.
- Pour la *classe de services* sélectionnée, ajouter le nouveau service.

2. Recherche

   Pour rechercher un service web existant dans la structure, il faut :

   - Trier les *entrées* du service recherché.
   - Vérifier s'il existe une *PE* qui correspond au nombre d'*entrées* recherchées. Si tel est le cas, la sélectionner, sinon retourner NULL.
   - Pour la *PE* sélectionnée, vérifier si elle contient un *AE* qui a comme clé le premier élément de l'ensemble des *entrées* du service recherché. Si c'est le cas, le sélectionner et supprimer le premier élément des entrées, sinon, retourner NULL.
   - Pour l'*AE* sélectionné, vérifier s'il contient un *AE* qui a pour clé le premier élément des *entrées*. Si c'est le cas, le sélectionner et supprimer le premier élément des entrées, sinon, retourner NULL.
   - Répéter l'opération précédente tant que l'ensemble des *entrées* du service recherché n'est pas vide.
   - Pour l'*AE* sélectionné, parcourir les services web de toutes les *classes de services* qu'il contient et vérifier pour chacune si ses sorties correspondent à l'ensemble des sorties du service recherché. Dans le cas positif, retourner le *service* sinon retourner NULL.

3. Suppression

   - Appliquer l'algorithme de recherche.
   - Supprimer le service de la *classe de services* correspondante.
   - S'il n'y a pas d'autres services dans la *classe de services*, supprimer la *classe de services*.
   - S'il n'y a pas d'autres *classes de services* dans l'*AE*, supprimer l'*AE*.

- Si le père de l' *AE* supprimé ne contient pas d'autres *AE*, le supprimer de la structure de recherche.
- Répéter l'opération précédente tant que l'ensemble de l'*AE* du service supprimé n'est pas vide.
- S'il n'y a pas d'autres *AE* dans la *PE*, alors supprimer *PE*.

4. Mise à jour

   Pour mettre à jour les entrées d'un service web existant, il faut :

   - Appliquer l'algorithme de suppression.
   - Appliquer l'algorithme d'addition.

- **Complexité des algorithmes pour la recherche** :

  Nous considérons comme étant le nombre moyen de *services web* dans une *classe de services* .Nous considérons comme le nombre d'entrées et comme le nombre de paramètres de sorties d'un service web recherché (est le nombre total de paramètres).
  La complexité de l'algorithme de recherche de INWEB est calculée comme suit :
  -Trier les entrées. (=)
  -Interroger une fois le dernier niveau pour savoir s'il existe une qui a comme clé . (= 1)
  -Interroger fois la structure de recherche des pour trouver le bon . (=)
  -Comparer fois deux ensembles de paramètres de tailles . (= * —)

  Le nombre de calculs pour traiter les entrées est le même pour les deux algorithme. La différence est dans le traitement des sorties. Là où INWEB prône une approche itérative pour trouver la bonne *classe de services*, INOWEB suit les mêmes étapes que pour les entrées.

  Pour MLIM, nous considèrerons comme étant le nombre moyen de *services web* dans une *classe de services*, comme étant le nombre moyen de *classes d'entrées* dans une *classe de clés*, comme le nombre d'entrées et comme le nombre de paramètres de sorties d'un service web recherché.
  La complexité de l'algorithme de recherche de MLIM est calculée comme suit :
  - recherche dans la structure. (=)
  - ( ) comparaisons entre les entrées recherchées et les entrées des classes d'entrées. (= * *—)
  - comparaisons entre deux ensembles de paramètre taille . (= * —)

  Pour l'index inversé, nous considérons comme étant le nombre moyen de *services web* pour chaque paramètre d'entrée et de sortie et comme le nombre d'entrées et comme le nombre de paramètres de sorties d'un service web recherché.

  La complexité de l'algorithme de recherche de l'index inversé est calculée comme suit :

- recherche dans la structure. (=) -Comparer fois deux ensembles de paramètres de tailles . (= *—)

La complexité de INWEB dans le pire cas = + 1 + + *—.
La complexité de INOWEB dans le pire cas = 2 (+ 1 +). La complexité de MLIM dans le pire cas = + * *— + * — La complexité de l'index inversé dans le pire cas= 2(+ *—)

est le nombre de paramètres qui composent les entrées ou sorties d'un service ou d'une requête. Ce nombre ne dépasse généralement pas les 8 paramètres et a une moyenne encore plus faible. le nombre moyen de services dans une classe de paramètres. Ce nombre varie en fonction du nombre total de services dans l'index et de la diversité fonctionnelle que proposent ses services. Le but principal d'un index étant de faciliter la recherche dans des répertoires volumineux, nous pouvons supposer que est largement plus grand que . donc ( ) ce qui implique que — —.

Pour la recherche d'un service web, INOWEB est plus rapide que INWEB, MLIM et l'index inversé. INWEB est plus rapide que MLIM et l'index inversé. MLIM est plus rapide que l'index inversé.

3 Description de notre approche d'indexation de services web composites

La création d'une approche d'indexation pour les services web composites pour notre approche du chapitre 6 (communautés) permet d'accélérer le processus de découverte et de sélection de services web composites. Cependant la création d'une telle approche est relativement complexe. Le problème principal d'indexation de services composites est qu'un service composite est représenté par la combinaison de plusieurs arbres de services (une foret). Cela nous oblige à parcourir tous les arbres à plusieurs reprises (suivant le nombre d'arbres du service composite recherché) ensuite faire l'union des différents résultats ce qui peut être extrêmement coûteux dans le temps. Pour éviter ce problème, nous décidons de proposer un algorithme de transformation de la représentation de services composites sous formes de chemins, ensuite, ces chemins seront regroupés sous formes d'arbres suivant les profondeurs triées par le nombre de services web. Chaque nœud de l'arbre représente un service web. Chaque feuille est un ensemble de services web composites ayant pour paramètres les clés des nœuds ayant en tête cette feuille. Le tri suivant la profondeur nous permet, lorsqu'une requête arrive, de savoir dans quel arbre rechercher le service composite requis.

3.1 Notation et définitions

Nous avons utilisé la notation et les définitions suivantes pour définir notre approche :

Notation 2. *Soit l'ensemble contenant tous les services web composites de notre annuaire.*

Definition 14. *Les relations d'un service web avec d'autres services web dans une composition sont représentées par =( ). Elles peuvent également être représentées sous forme d'un arbre de relations par : . C'est à dire que le service web de pointe vers les services web .*

Definition 15. *Un service composite peut être vu comme une suite d'arbres. Il est représenté par = (, , …) où correspond à l'arbre des relations du service web 1 et correspond à l'arbre des relations du service web m où m correspond au nombre de services web dans . Il peut également être représenté par .*

Definition 16. *La découverte de services composites est représentée par la paire DC(A,SC) = , le symbole est utilisé sur le contexte de l'ontologie. Par exemple est vraie si et seulement si est une sous-classe de dans l'ontologie. Nous n'irons pas plus loin avec les ontologies car cela ne concerne pas notre travail.*

Definition 17. *L'ajout de service web composite est défini par Ajouter (sc, SC) où l'objectif est simplement d'ajouter un service web composite "sc" dans l'ensemble "SC" tout en respectant les règles imposées par la structure (la méthode) utilisée.*

Definition 18. *est le nombre de services web composites contenus dans un annuaire.*

Definition 19. *est le nombre maximum d'arbres de relations qu'un service composite peut avoir, noté* max *avec*

Definition 20. *est le nombre maximal de services web qu'un service composite peut avoir, noté* max *avec*

Definition 21. *est le nombre total d'arbres uniques dans l'annuaire (chaque arbre est compté une fois), noté*

Definition 22. *est le nombre total de services web uniques dans l'annuaire (chaque service web est compté une fois), noté*

En utilisant l'exemple suivant :

,

,

,

.

Nous avons à partir des définitions précédentes :

1. ,
2. ,
3. ,
4. ,
5. .

3.2 Indexation des arbres de services web des services web composites

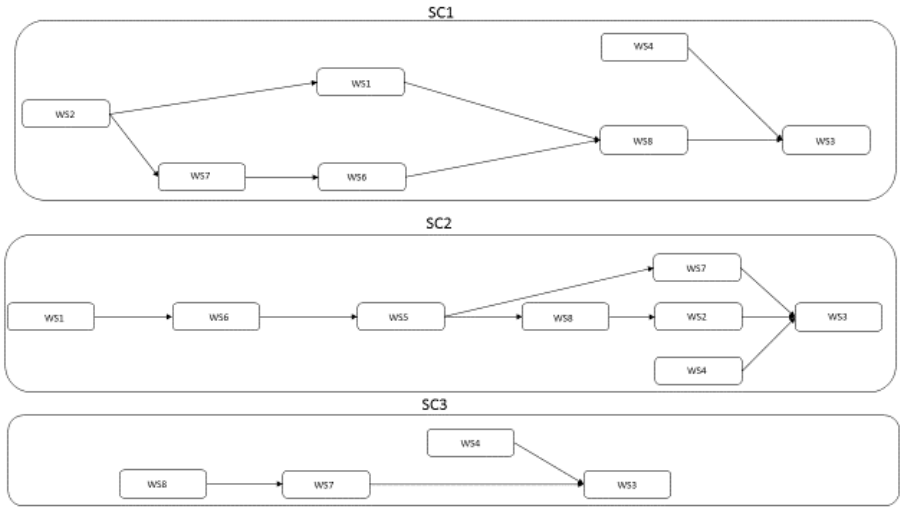

FIG. 7.6 : Exemple d'un échantillon de services web.

ITACOM se concentre sur l'indexation de tous les services web composites sous forme d'un arbre. ITACOM représente chaque service composite par un chemin unique de services web et pour optimiser la recherche, nous avons décidé de trier suivant les noms des services web puis de construire un arbre avec les dits services où chaque nœud de l'arbre est représenté par un service web alors que chaque feuille est un ensemble de services web composites ayant comme paramètres les clés des nœuds nous ayant conduit à cette feuille. Le fait de trier les noms des services web nous permet, à l'arrivée d'une requête, de savoir dans lequel rechercher.

Pour représenter un service web composite sous forme d'un chemin il faut :

- Trouver tous les arbres de relations des services web du service web composite.
- Trier par les noms des services web.
- Construire l'arbre de services web du service web composite.
- Construire le chemin de services web du service web composite (voir figure 7.8) :

  Pour chaque arbre de services web d'un service web composite (voir figure 7.9) :

  - Si la racine d'un arbre n'a pas de fils alors la lier à la racine suivante. (cas 1)
  - Si la racine d'un arbre a un seul fils alors lier le fils à la racine suivante. (cas 2)
  - Si la racine d'un arbre a plus qu'un fils alors garder le lien avec le premier fils et enlever les autres liens. Tant que ce n'est pas le dernier fils alors lier chaque fils à son frère suivant. Si dernier fils alors le lier à la racine suivante. (cas 3)

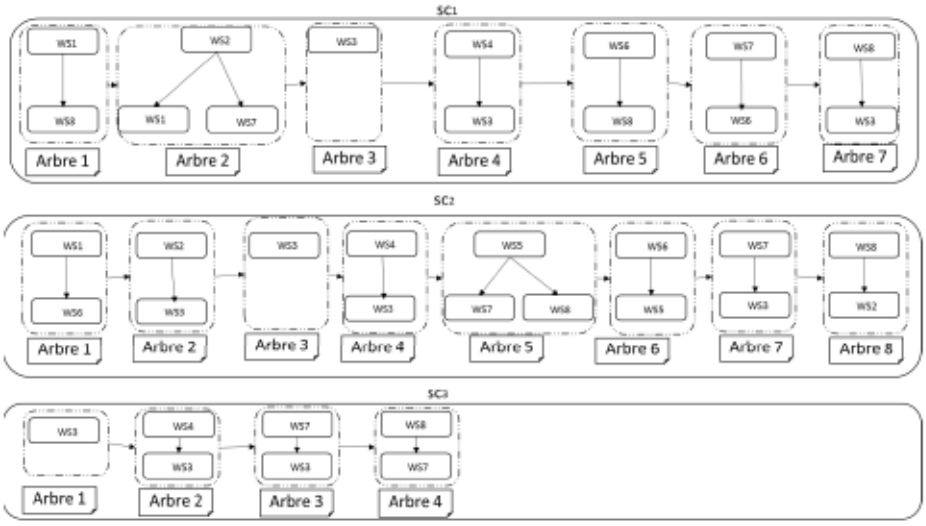

FIG. 7.7 : Exemple d'un arbre de services web triés d'un service web composite.

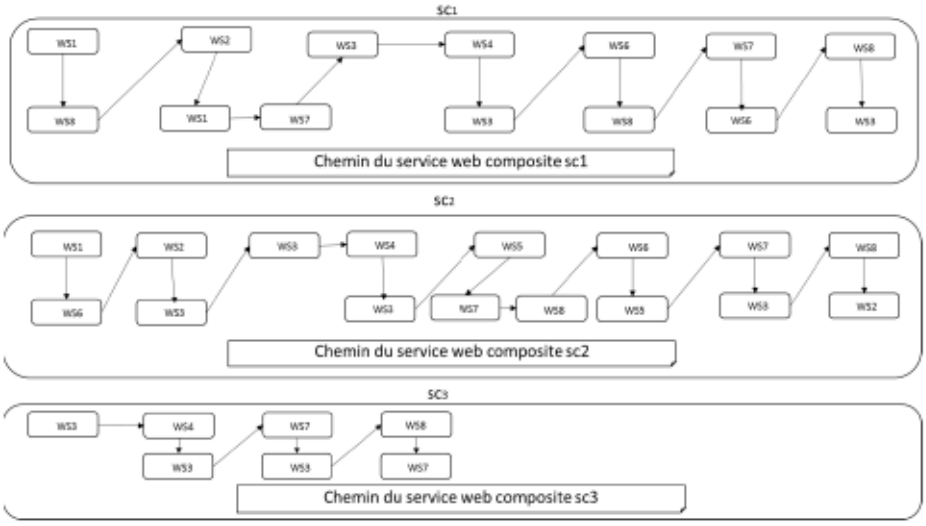

FIG. 7.8 : Exemple d'un chemin de services web du service web composite.

ITACOM comporte quatre niveaux. Le niveau le plus bas est l'ensemble des classes regroupant les services composites en fonction de leurs services. Le niveau supérieur est un ensemble d'arbres destiné à faciliter la découverte des classes du dernier niveau. Le deuxième niveau est un ensemble de classes regroupant les arbres du troisième niveau par leurs profondeurs et enfin le premier niveau est une structure de recherche qui facilite la découverte des classes du deuxième niveau.

Le nombre de services associés à une *classe de services composites* n détermine la profondeur de l'arbre d'index pour cette classe. Ainsi, la classe $n = 1$ a une profondeur d'indice égale à un, etc. La largeur des arbres du niveau 3 diminue à mesure que chaque arbre est parcouru de haut en bas (m, à $m - i$), où i est le nombre services dans la

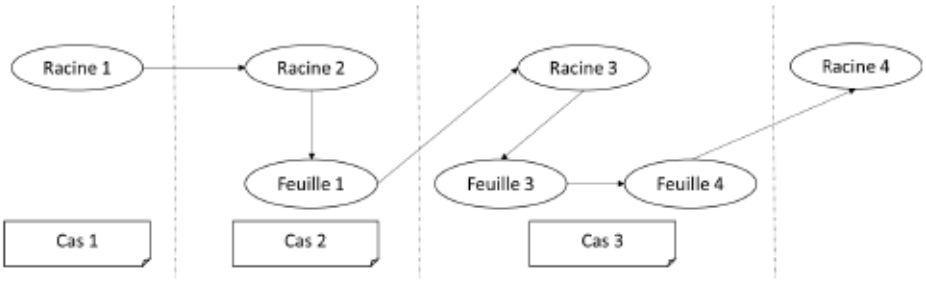

FIG. 7.9 : un chemin de services web du service web composite.

classe de services. Le niveau 2 détermine le point d'entrée dans les différents arbres.

Nous donnons un exemple sur le processus d'indexation suivant l'échantillon de services composites de la figure7.6 composé des services composites sc_1, sc_2 et sc_3 caractérisés par leurs services web. Le niveau 1 de l'index représente les clés permettant d'accéder aux différentes profondeurs de services. Le niveau 2 regroupe les services dans les arborescences accédant aux différentes listes de services web composites ayant les mêmes profondeurs de services. Ainsi dans notre exemple, PS_1 regroupe tous les services web composites ayant une profondeur de chemin égal à sept, la profondeur PS_2 regroupe tous les services composites ayant une profondeur de chemin égal à quatorze et la profondeur PS_3 regroupe tous les services composites ayant une profondeur de chemin égal à seize. Le niveau 3 contient des arbres de services web triés de gauche à droite. Un chemin existe dans cette arborescence si et seulement s'il existe au moins un service web composite dans la liste identifiée par l'ensemble des services constituant le chemin. Le niveau 4 contient les services pointant vers les listes de services web ayant les mêmes chemins de services web. (voir figure 7.13)

3.2.1 ▸ Niveau 4 (*Classes de services*) :

Ce niveau regroupe des services web composites ayant les mêmes services web dans la même classe.

3.2.2 ▸ Niveau 3 (*Arbres de services*) :

Ce niveau consiste en la construction d'un arbre qui représente les services web triés de chaque *classe de services composites*(*CC*) de l'ensemble N_4. Les différents nœuds de l'arbre sont appelés "*Arbre de services*" ou "*AS*".

Definition 23. *Nous représentons AS par : AS= {k, p, services, ASS, CC}, servicesest un ensemble de services web, ASS est un ensemble de ASS, et enfin CCS est l'ensemble de Classes de services composites.*

Definition 24. *N_3 est un ensemble d'AS défini par les règles suivantes :*

1. $(\forall CC \in N_4) \Leftrightarrow ((\exists A \in N_3) \wedge (CC \in A.CCS))$*: Toute classe de services de N_4 est représentée au moins une fois dans N_3, et si une classe de services est représentée dans N_3 alors elle doit forcément appartenir à N_4.*

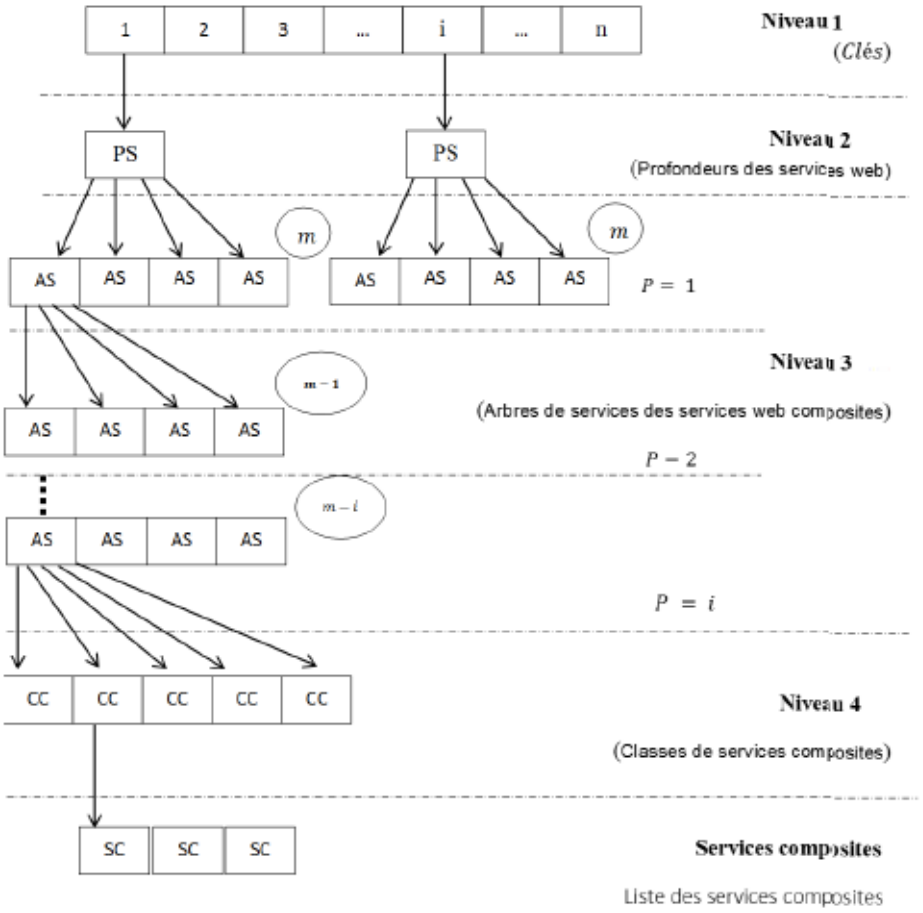

FIG. 7.10 : Architecture de l'index ITACOM.

2. $((\forall A1, A2 \in N_3) \wedge (A1.p = A2.p) \wedge (A1.k = A2.k) \wedge (A1.sevices = A2.services)) \Leftrightarrow (A1 = A2)$: *La profondeur, la clé et les services web rendent un AS unique dans l'ensemble* N_3.

La stratégie du choix de clé représentant les *AS* :

Le choix de la clé de chaque *AS* est crucial dans cette méthode car il dirige la construction de l'index et par conséquent la recherche. Comme expliqué précédemment, nous avons choisi d'adopter l'ordre alphabétique dans notre cas, ce qui veut dire que lorsque nous voulons choisir une clé, nous prenons la clé la mieux classée alphabétiquement si l'index se construit de haut en bas pour représenter un *AS*, et

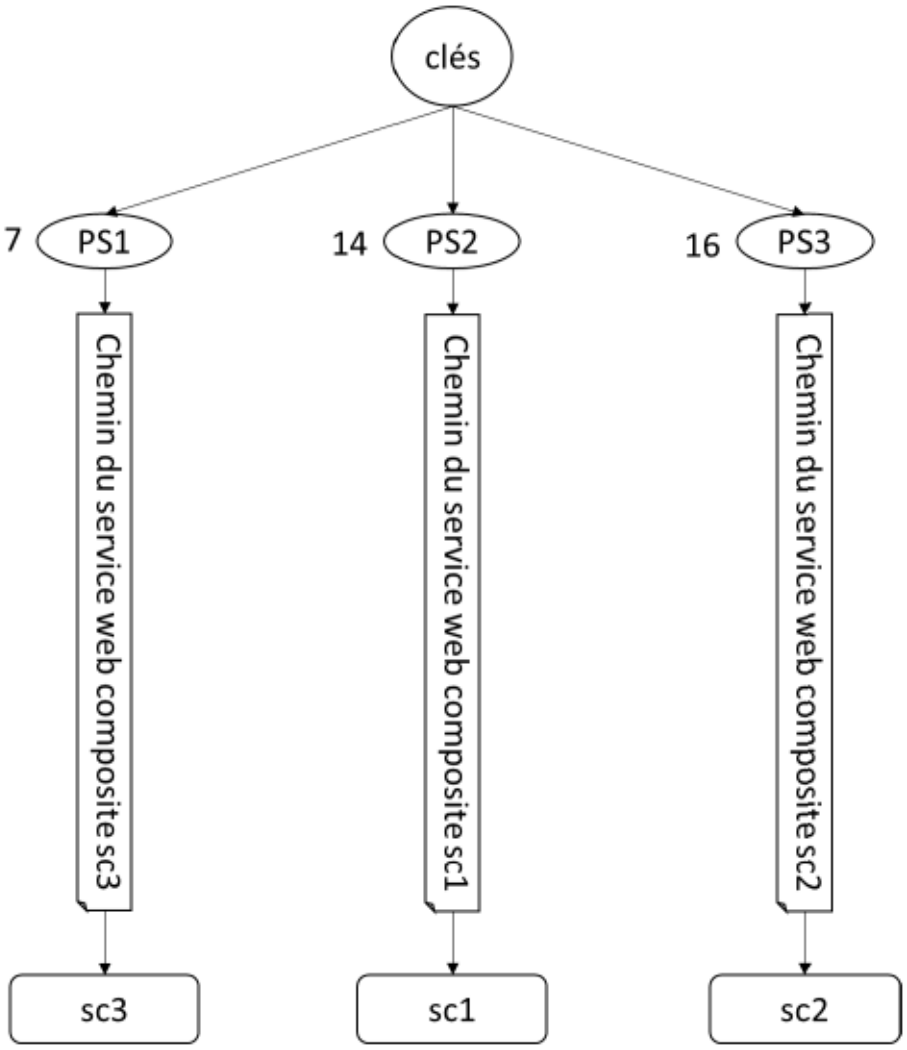

FIG. 7.11 : Exemple d'indexation ITACOM.

l'inverse dans le cas où l'index se construit de bas en haut (cela importe peu tant que nous respectons l'ordre choisi dans la recherche).

3.2.3 ▸ Niveau 2 (*Profondeurs de services*) :

A ce niveau, nous regroupons les arbres de services (*AS*) ayant la même profondeur dans la même classe nommée " *Profondeur de services*" ou "*PS*".

Definition 25. *une Profondeur de services est définie par PS = (p, classes), où p est la profondeur et classes est un ensemble de AS tel que* $\forall A \in classes \rightarrow (A.p = p)$. *Elle peut également être représenté par PS:* $p \rightarrow classes$

Definition 26. N_2 *est un ensemble PS défini par les règles suivantes :*

1. $(\forall A \in N_3) \Leftrightarrow (\exists P \in N_2) \wedge (A \in PS.classes)$): *Tout AS de* N_3 *est représentée*

au moins une fois par une PS dans , et si une PS représente un AS alors cette AS appartient à l'ensemble .

2. *: La profondeur d'une PS la rend unique dans l'ensemble .*

3.2.4 ▶ Niveau 1(*Clés*) :

Une structure qui facilite la recherche est mise en place (table de hachage, B-Arbre). Elle relie chaque profondeur à sa *PS*. La profondeur ici dénote le nombre de *services web* caractérisant les SW contenus dans la *PS* correspondante.

Definition 27. *une clé d'une profondeur est définie par C = (,), où est une clé et est une profondeur de l'ensemble qui est défini par :*
: Toute PS de est représentée par sa profondeur dans P, et toute profondeur de P représente une seule PS de .

3.2.5 ▶ Opérations sur les index proposés

Nous présentons ci-dessous les différentes étapes des algorithmes pour l'ajout, la recherche, la suppression et la mise à jour de services dans notre solution pour l'algorithme ITACOM.

1. Addition

   Pour ajouter un nouveau service web composite à la structure, il faut :

   - Construire le chemin du service composite.
   - Vérifier si une *PS* existe pour le nombre de *services* correspondant. Sinon, il faudra la créer et l'ajouter à la structure de recherche.
   - Pour la *PS* sélectionnée, vérifier s'il existe un *AS* ayant comme clé le premier service. Sinon, il faudra le créer et l'ajouter à la *PS*, puis supprimer le premier élément de l'ensemble des *services* du nouveau service.
   - Pour l'*AS* sélectionné, vérifier s'il existe un *AS* ayant comme clé le premier élément des services. Sinon, le créer et l'ajouter à la structure de recherche contenue dans l' *AS* sélectionné, puis supprimer le premier élément de l'ensemble des *services* du nouveau service.
   - Répéter l'opération précédente tant que l'ensemble des services du nouveau service n'est pas vide.
   - ajouter une nouvelle classe de services web composites si le nouveau service n'est pas dans la liste des classes de services composites.
   - Pour la *classe de services composites* sélectionnée, ajouter le nouveau service composite.

2. Recherche

   Pour rechercher un service web composite existant dans la structure, il faut :

   - Trier les *services* du service composite recherché.
   - Vérifier s'il existe une *PS* qui correspond au nombre de *services* recherchés. Si tel est le cas, la sélectionner, sinon retourner NULL.

- Pour la *PS* sélectionnée, vérifier si elle contient un *AS* qui a comme clé le premier élément de l'ensemble des *services* du service composite recherché. Si c'est le cas, le sélectionner et supprimer le premier élément des services, sinon, retourner NULL.
- Pour le *AS* sélectionné, vérifier s'il contient un *AS* qui a pour clé le premier élément des *services*. Si c'est le cas, le sélectionner et supprimer le premier élément des *services*, sinon, retourner NULL.
- Répéter l'opération précédente tant que l'ensemble des *services* du service composite recherché n'est pas vide.
- Retourner la *classe de services composites* de l' *AS* sélectionné sinon retourner NULL.

3. Suppression

   Pour supprimer un service web composite existant dans la structure, il faut :

   - Appliquer l'algorithme de recherche.
   - Supprimer le service composite de la *classe de services composites* correspondante.
   - S'il n'y a pas d'autres services composites dans la *classe de services*, supprimer la *classe de services composites*.
   - S'il n'y a pas d'autres *classes de services composites* dans l'*AS*, supprimer l'*AS*.
   - Si le père de l' *AS* supprimé ne contient pas d'autres *AS*, le supprimer de la structure de recherche.
   - Répéter l'opération précédente tant que l'ensemble de l'*AS* du service supprimé n'est pas vide.
   - S'il n'y a pas d'autres *AS* dans la *PS*, alors supprimer *PS*.

4. Mise à jour

   Pour mettre à jour les entrées d'un service existant, il faut :

   - Appliquer l'algorithme de suppression.
   - Appliquer l'algorithme d'addition.

- **Complexité de l'algorithme ITACOM pour la recherche** :

  Nous considérons comme étant le nombre moyen de relations d'un service web avec d'autres services web du service composite recherché.
  Nous considérons comme le nombre de services web d'un service composite à rechercher.
  La complexité de l'algorithme de recherche pour ITACOM est calculée comme suit :
  -Trier les services web. (=)
  -Trier les relations des services web. (=) -Interroger une fois le dernier niveau pour savoir s'il existe une qui a comme clé . (= 1)
  -Interroger fois la structure de recherche des pour trouver le bon .

(=)

Pour l'index inversé, nous considérons comme étant le nombre moyen de *services web composites* pour chaque *service web*, comme étant le nombre de services web d'un service composite recherché, comme étant le nombre moyen de relations d'un service web avec d'autres services web du service composite recherché et comme étant le nombre de services web composites différents ayant les mêmes services web.

La complexité de l'algorithme de recherche pour l'index inversé est calculé comme suit :
- recherche dans la structure. (=) -Comparer fois deux ensembles de paramètres de tailles . (= *—)

-Comparer fois deux ensembles de services web de tailles . (= *———)

La complexité de ITACOM dans le pire cas = + + 1 + .

La complexité de l'index inversé dans le pire cas = + *— + *———.

Pour la recherche d'un service web composite, ITACOM est plus rapide que l'index inversé.

4 Résultats numériques

4.1 Modèle de la simulation

Afin d'étudier les avantages concrets qu'apportent nos solutions d'indexation pour la découverte de services web, nous avons réalisé une série de simulations sur un ordinateur avec un processeur CPU i5 1,7 GHZ et d'une mémoire de 4 Go à l'aide de notre programme implémenté en C ++ sur la plateforme Qt. Les résultats obtenus sont présentés dans cette section. Ils sont issus de tests simulant la découverte de services web sur les ensembles de données , et . considére la base de données réelles contenant 23409 services web, utilisés dans [7]. La table 7.1 fournit les informations sur les référentiels WSDL fournies par les organisateurs du WS-Challenge [7]. considère la description de l'ensemble de données décrite dans la table 7.2 avec des données générées artificiellement pour des services web simples. considère la description de l'ensemble de données décrite dans la table 7.3 avec des données générées artificiellement pour des services web composites.

TAB. 7.1 : Les paramètres de

Paramètres	Intervalles
Nombre d'entrées par service	[4,35]
Nombre de sorties par service	[4,35]
Nombre de services	23409
Nombre de requêtes	50000

TAB. 7.2 : Les paramètres de

Paramètres	Intervalles
Nombre d'entrées par service	[1,5]
Nombre de sorties par service	[1,3]
Nombre de requêtes	10000

TAB. 7.3 : Les paramètres de

Paramètres	Intervalles
Nombre de services web	1000
Nombre de services par service web composite	[3,15]
Nombre de relations par service	[1,3]

4.2 Évaluation du temps d'exécution pour l'ensemble réel

Pour la première série de simulations, nous avons évalué les différents temps d'exécution des quatre approches d'indexation pour les 23409 services de pour les cas d'ajout et de recherche de services web.

4.2.1 ▶ Ajout

La figure 7.12 présente les différents temps d'exécution des quatre approches d'indexation pour l'ajout de services web. L'index inversé est le plus rapide pour ajouter de nouveaux services, suivent les approches INWEB, MLIM et INOWEB. L'index inversé est le plus rapide du au fait qu'il utilise un seul niveau d'index par rapport aux autres approches. INOWEB est le moins rapide en raison du nombre élevé de paramètres de sorties par service.

4.2.2 ▶ Recherche

La figure 7.13 présente les différents temps d'exécution des quatre approches d'indexation pour la recherche de services web avec 50000 requêtes. Nos deux approches ont les meilleures performances en comparaison avec l'index inversé et MLIM. Comme nous l'avons dit dans la figure 7.12, INOWEB a le temps d'exécution le moins rapide pour ajouter des nouveaux services avec un nombre élevé de sorties, Cependant, INOWEB dispose du plus rapide temps d'exécution pour rechercher des services web.

4.3 Évaluation du temps d'exécution pour l'ensemble artificiel

Pour la deuxième série de simulations, nous avons évalué les différents temps d'exécution des quatre approches d'indexation pour des services web générés aléatoirement tels que décrit dans le tableau table 7.2 pour l'ensemble pour les cas d'ajout et de recherche de services web et le nombre de structures utilisées.

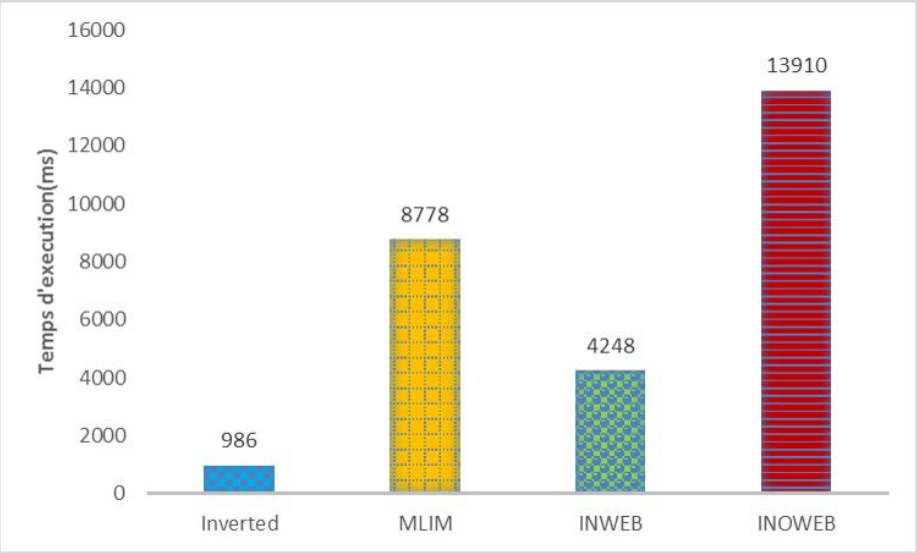

FIG. 7.12 : Addition.

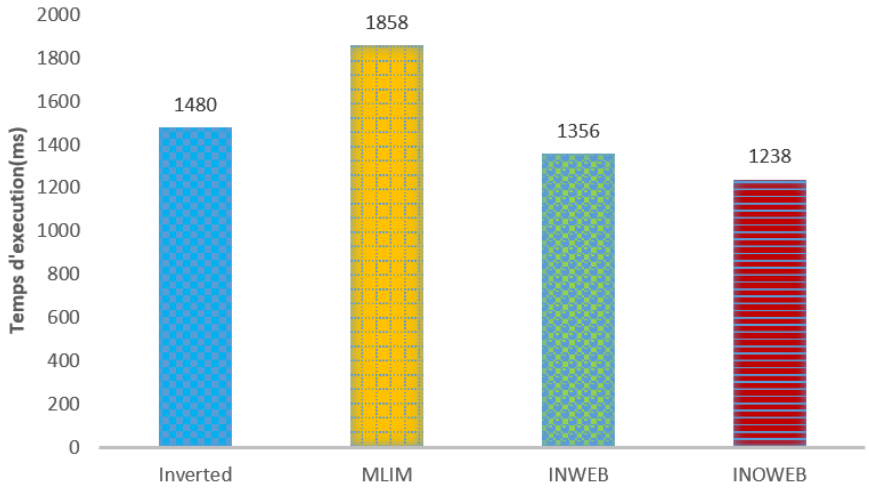

FIG. 7.13 : Recherche.

4.3.1 ▶ Ajout

Dans la figure 7.14, le temps d'exécution des approches d'index inversé, INWEB et INOWEB augmente d'une manière linéaire , tandis que pour l'approche MLIM, son temps d'exécution augmente d'une manière exponentielle. Comme on pouvait s'y attendre, la méthode de l'index inversé est la plus rapide, suivie par INWEB, INOWEB et loin derrière, la méthode MLIM.

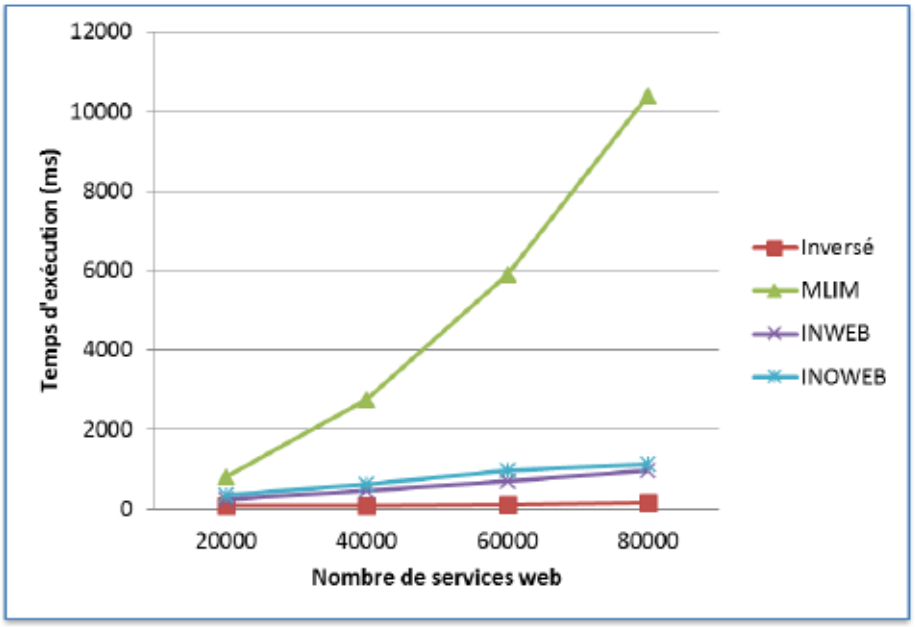

FIG. 7.14 : Addition avec variation du nombre de services web.

4.3.2 ▸ Recherche

Dans la figure 7.15, le temps d'exécution des approches d'index inversé, INWEB et MLIM augmente d'une manière linéaire, tandis que pour l'approche INOWEB, son temps d'exécution est presque constant. La méthode INOWEB est la plus rapide, suivie de près par INWEB et enfin les deux autres méthodes.

4.3.3 ▸ Nombre de structures

Dans la figure 7.16, les ressources de mémoire consommées augmentent d'une manière linéaire avec l'augmentation du nombre de services web dans le répertoire. La méthode de l'index inversé est une exception car le nombre de structures utilisées par cette dernière dépend uniquement du nombre d'entrées répertoriées. Comme prévu, INOWEB est celui qui utilise le plus, puis vient la méthode INWEB suivie de MLIM. INWEB utilise 23 % plus de ressources de mémoire que MLIM, mais le gain en performances est de 2330 % (23 fois plus).

4.4 Évaluation du temps d'exécution pour l'ensemble artificiel $EDAC$

Pour la troisième série de simulations, nous avons évalué les différents temps d'exécution des deux approches d'indexation (index inversé et ITACOM) pour des services web composites générés aléatoirement tels que décrit dans le tableau table 7.3 pour l'ensemble $EDAC$ pour les cas d'ajout et de recherche de services web composites.

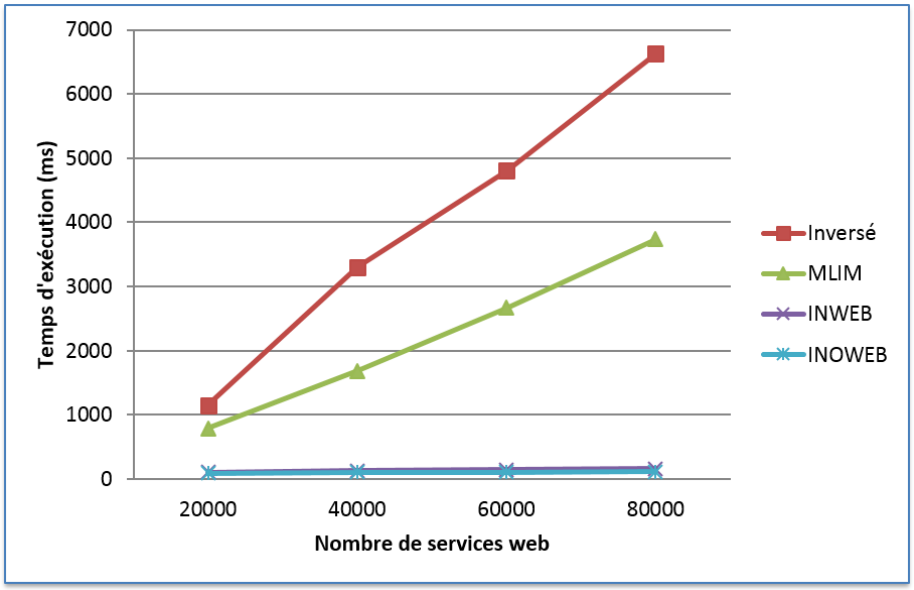

FIG. 7.15 : Recherche avec variation du nombre de services web.

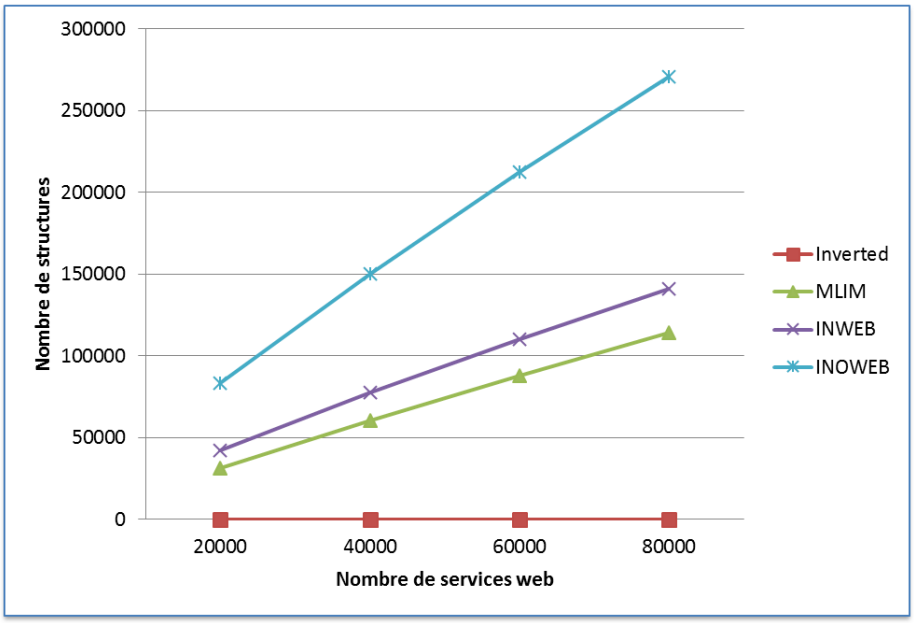

FIG. 7.16 : Nombre de structures utilisées avec variation du nombre de services web.

4.4.1 ▸ Ajout

Dans la figure 7.17, le temps d'exécution des approches d'index inversé et ITACOM augmente d'une manière linéaire. Comme on pouvait s'y attendre, la méthode de l'index inversé est la plus rapide que la méthode ITACOM .

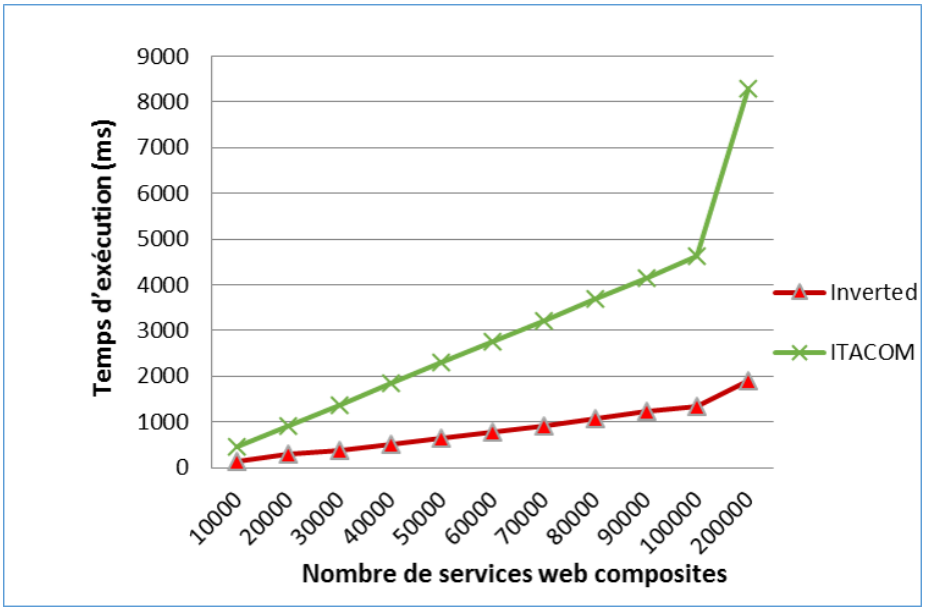

FIG. 7.17 : Addition avec variation du nombre de services web composites.

4.4.2 ▶ Recherche

Dans la figure 7.18, le temps d'exécution des approches d'index inversé et ITACOM augmente d'une manière linéaire. La méthode ITACOM est nettement plus rapide que l'index inversé.

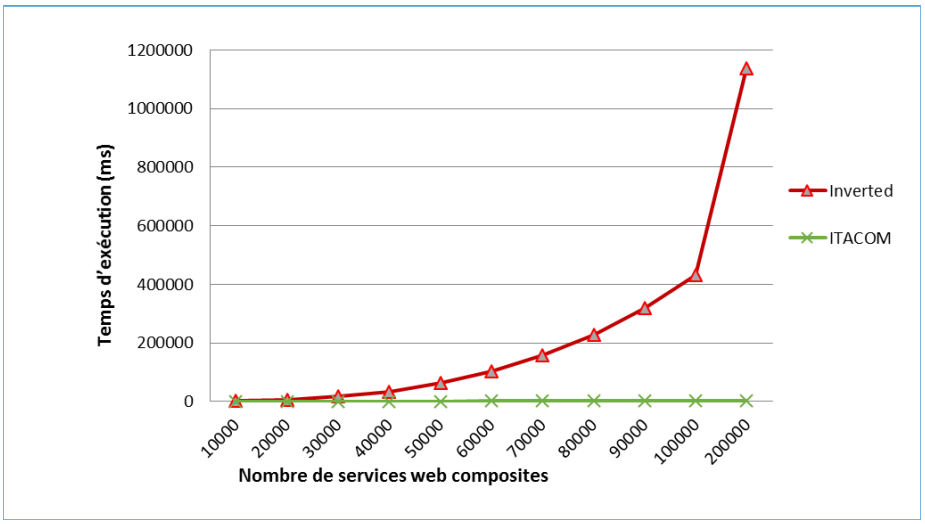

FIG. 7.18 : Recherche avec variation du nombre de services web composites.

5 Conclusion

Dans ce chapitre, nous avons proposé plusieurs solutions à base d'index pour l'optimisation du temps d'accès et la pertinence des résultats dans un répertoire de services web. L' index ITACOM est proposé pour la gestion des services web composites. Les index INWEB et INOWEB sont proposés pour la gestion des services web simples. Pour évaluer les performances de nos approches, nous avons mené différentes expériences en les comparant avec les approches existantes dans la littérature (MLIM et index inversé). Les résultats ont validé les performances des index ITACOM, INWEB et INOWEB.

8

Conclusion et Persepectives

Au cours de ces dernières années, plusieurs types de paradigmes de développement d'applications distribuées ont été proposés. La technologie des services web représente une des meilleures solutions pour le développement d'applications distribuées sur le web. Avec le nombre gigantesque de services web disponible sur le réseau internet, les clients auront toujours besoin de techniques efficaces de découverte, de sélection et de composition pour trouver des services web qui correspondent à leurs exigences et à leurs besoins. Les travaux présentés dans cette thèse s'inscrivent dans ce cadre-là.

1 Synthèse des contributions

Dans cette section, les contributions et les réalisations sont discutées. Les objectifs, décrits dans le chapitre 1, sont examinés individuellement. De plus, cette section explique comment cette contribution atteint les objectifs.

- Objectif 1 : Sélection efficace des services web simples à base de QdS.
  Pour atteindre cet objectif, nous avons tout d'abord effectué une recherche sur les approches AMCD existantes dans la littérature de la sélection de services web. Ensuite, ces approches sont examinées et comparées. Les comparaisons sont effectuées sur deux niveaux. Dans le premier niveau, les approche AMCD sont classées dans trois catégories : Sélection de services, pondération des poids des critères QdS et reduction de l'espace de recherche des services web. Au

deuxième niveau, la complexité des approches est examiné pour déterminer les méthodes rapides et efficaces. Le principal problème après évaluation des méthodes de sélection de services est le classement différent fourni par chaque méthode pour un même échantillon de service web. Par conséquent, pour combler ce problème en matière de sélection, une approche est proposée dans laquelle le classement de chaque méthode AMCD rapide est pris en compte pour élire un classement des meilleures services web à l'aide d'une méthode de vote. Aussi, ce classement est utilisé pour évaluer la précision de classement de chaque méthode à l'aide de ratios.

- Objectif 2 : Sélection efficace des services web simples à base de QdS et à base de contraintes.
  Pour atteindre cet objectif, nous avons tout d'abord effectué une recherche sur les approches existantes dans la littérature de la sélection de services web et la normalisation des valeurs des services web. Ensuite, ces normalisations sont examinées et comparées. Le principal problème après évaluation des méthodes de sélection de services est le cas où aucun service web ne réponds aux contraintes du client. Par conséquent, pour combler ce problème en matière de sélection, une approche de normalisation prenant en compte les contraintes du client est proposée. Cette normalisation est ensuite combinée avec des méthodes AMCD adaptées pour proposer un classement des services web qui s'approchent des contraintes du client. Aussi, les différents classements fournis par les différentes méthodes AMCD adaptées sont évalués à l'aide de ratios.

- Objectif 3 : Sélection efficace de services web composites pour un groupe de clients dans un environnement dynamique.
  Pour atteindre cet objectif, nous avons tout d'abord effectué une recherche sur les approches existantes dans la littérature de la sélection de services composites. Le principal problème après évaluation des approches de sélection de services web est qu'aucune approche n'a été proposée pour ce type de problème. Par conséquent, pour combler ce problème en matière de sélection, une description détaillée du problème ainsi qu'une solution optimale est proposée. Aussi, nous avons proposé une heuristique à base de théorie des jeux et de communautés pour améliorer le temps de réponse de la sélection. Enfin, les deux approches sont évaluées et comparées à l'aide de simulations.

- Objectif 4 : Découverte efficace de services web simples ou composites à l'aide de méthodes d'indexation.
  Pour atteindre cet objectif, nous avons tout d'abord effectué une recherche sur les approches existantes dans la littérature de la découverte de services web. Le principal problème après évaluation des approches de découverte de services web est le temps de réponse pour trouver un service web en terme d'entrées / sorties. Par conséquent, pour combler ce problème en matière de découverte, plusieurs approches d'indexation à plusieurs niveaux sont proposées. Ces approches sont ensuite évaluées et comparées avec les approches existantes à l'aide de simulations.

2 Perspectives pour les travaux futurs

Nous avons déjà mené une étude poussée sur les processus d'optimisation de la sélection des services web simples et composites, mais il reste néanmoins beaucoup de pistes de réflexion à explorer. Une des pistes à explorer est la mise en place d'un système de qualité d'expérience. En effet, le succès d'un service web dépend grandement de sa capacité à offrir une performance applicative exceptionnelle aux clients. Mesurer l'expérience client est un défi majeur dans la littérature des services web souvent résumée avec le critère QdS "réputation" en raison d'une constante évolution du paysage informatique et des clients. En outre, la performance des services web et l'expérience client doivent être gérées de manière à ce que les clients puissent bénéficier pleinement des retours d'expériences. Comment s'assurer qu'un service web soit facilement disponible et réponde parfaitement aux exigences client (performances erronées d'un service web, pannes fréquentes d'un service, efficacité d'un service...etc). Une autre piste à explorer est l'automatisation du système d'attribution des poids des critères QdS. Pour cela, un profil client peut être créé en se recentrant, analysant, étudiant les préférences QdS exprimées par un client pour un service web suivant le domaine, la catégorie, etc. . Des applications de ces méthodes méritent d'être étudiées dans un environnement réel, dynamique et spécifique. En fonction des scénarios déployés, les mécanismes de sélection devront peut-être subir de nouvelles adaptations. Quoi qu'il en soit, les résultats numériques obtenus par simulation profiteraient logiquement de l'usage d'applications plus proches d'une mise en production réelle. Mais nous pourrions faire encore mieux : implémenter de telles approches pour offrir des solutions efficaces aux gestionnaires d'annuaires de services web, tel que AWS, ce qui nous permettrait d'obtenir des valeurs en meilleure adéquation avec une mise en production du système.En fonction des résultats, il serait alors possible de rechercher l'amélioration des approches proposées, ou bien l'utilisation d'architectures plus adaptés pour obtenir des résultats plus fidèles et plus efficaces.Le modèle utilisant les données qualitatifs et quantitatifs constitue une base de travail intéressante que nous aimerions étudier, dans le but d'améliorer le processus de sélection de services, par exemple en parvenant à convertir les données qualitatifs d'une manière efficace sans perdre son échelle de valeur. Toutes ces approches sont autant de perspectives à étudier dans de futurs travaux, afin de concevoir, à partir des solutions apportées au cours de cette thèse, des mécanismes et des modèles toujours plus performants dans la sélection de services web simples ou composites.

Bibliographie

[1] Mohammad Alrifai, Dimitrios Skoutas et Thomas Risse. «Selecting Skyline Services for QoS-based Web Service Composition». In : *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA : ACM, 2010, p. 11–20. ISBN : 978-1-60558-799-8. DOI : 10.1145/1772690.1772693. URL : http://doi.acm.org/10.1145/1772690.1772693 (cf. p. 21, 36, 37).

[2] M. Clement Anand et Janani Bharatraj. «Theory of Triangular Fuzzy Number». In : mar. 2017 (cf. p. 26).

[3] *Applications d'entreprise*. https://aws.amazon.com/fr/business-applications/?nc1=f_dr. Accessed : 2018-11-11 (cf. p. 8).

[4] *AWS*. https://fr.wikipedia.org/wiki/Amazon_Web_Services. Accessed : 2019-02-06 (cf. p. 10).

[5] Saravana Balaji B, Priyadharshini G et Gunasri R. «A Survey on Semantic Web Service Discovery Methods». In : *International Journal of Computer Applications* 82 (nov. 2013). DOI : 10.5120/14158-1759 (cf. p. 17).

[6] Majid Behzadian, S. Khanmohammadi Otaghsara, Morteza Yazdani et Joshua Ignatius. «A state-of the-art survey of {TOPSIS} applications». In : *Expert Systems with Applications* 39.17 (2012), p. 13051–13069 (cf. p. 26).

[7] M. B. Blake, Kwok Ching Tsui et A. Wombacher. «The EEE-05 challenge : a new Web service discovery and composition competition». In : *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*. Mar. 2005, p. 780–783. DOI : 10.1109/EEE.2005.131 (cf. p. 120).

[8] M. A. Bouanaka et N. Zarour. «An approach for an optimized web service selection based on skyline». In : *International Journal of Computer Science Issues* 10.1 (2013), p. 412–418 (cf. p. 21).

[9] Boudjemaa Boudaa. «Vers une Substitution des Services Web sans Inconsistance Sémantique». In : (mai 2013) (cf. p. 8).

[10] Ilhem Boussaid. «Improvement of metaheuristics for continuous optimization». Theses. Université Paris-Est, juin 2013. URL : https://tel.archives-ouvertes.fr/tel-00952774 (cf. p. 31).

[11] J.P. Brans. «L'ingénièrie de la décision ; Elaboration d'instruments d'aide à la décision. La méthode PROMETHEE». In : *L'aide à la décision : Nature, Instruments et Perspectives d'Avenir*. Sous la dir. de R. Nadeau et M. Landry. Québec, Canada : Presses de l'Université Laval, 1982, p. 183–213 (cf. p. 27).

[12] M. R. Brenner et M. R. Unmehopa. «Service-oriented architecture and Web services penetration in next-generation networks». In : *Bell Labs Technical Journal* 12.2 (2007), p. 147–159. ISSN : 1538-7305. DOI : 10.1002/bltj.20243 (cf. p. 10).

[13] Elio Cables, Maria Teresa Lamata et Jose L. Verdegay. «RIM-reference ideal method in multicriteria decision making». In : *Information Sciences* 337 (2016), p. 1–10 (cf. p. 62, 65).

[14] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito et Maria Luisa Villani. «An Approach for QoS-aware Service Composition Based on Genetic Algorithms». In : *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. GECCO '05. Washington DC, USA : ACM, 2005, p. 1069–1075. ISBN : 1-59593-010-8. DOI : 10.1145/1068009.1068189. URL : http://doi.acm.org/10.1145/1068009.1068189 (cf. p. 85).

[15] Sung-Hyuk Cha. «Comprehensive Survey on Distance/Similarity Measures between Probability Density Functionss». In : *INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES* 1.4 (2007), p. 300–307 (cf. p. 68).

[16] Salem Chakhar, Serge Haddad, Lynda Mokdad, Vincent Mousseau et Samir Youcef. «Multicriteria Evaluation-Based Framework for Composite Web Service Selection». In : *Evaluation and Decision Models with Multiple Criteria : Case Studies*. Springer-Verlag Berlin Heidelberg, 2015, p. 167–200 (cf. p. 21, 29).

[17] Subrata Chakraborty et Chung-Hsing Yeh. «A simulation based comparative study of normalization procedures in multiattribute decision making». In : *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*. T. 6. 2007, p. 102–109 (cf. p. 25).

[18] Subrata Chakraborty et Chung-Hsing Yeh. «A simulation comparison of normalization procedures for TOPSIS». In : *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on*. IEEE, 2009, p. 1815–1820 (cf. p. 25).

[19] Liping Chen, Weitao Ha et Guojun Zhang. «Reliable Execution Based on CPN and Skyline Optimization for Web Service Composition». In : *The Scientific World Journal* 2013 (2013) (cf. p. 21).

[20] Chantal Cherifi. «Classification et Composition de Services Web : Une Perspective Réseaux Complexes. Web. Université Pascal Paoli». Thèse de doct. Université Pascal Paoli, 2011 (cf. p. 8).

[21] Cheol-Rim Choi et Hwa-Young Jeong. «A broker-based quality evaluation system for service selection according to the QoS preferences of users». In : *Inf. Sci.* 277 (2014), p. 553–566 (cf. p. 27, 29, 36).

[22] *Clients du cloud de AWS*. https://www.cnbc.com/2017/11/29/amazons-cloud-service-signs-disney-expedia-nfl.html. Accessed : 2019-02-06 (cf. p. 10).

[23] Marco Comuzzi et Barbara Pernici. «An architecture for flexible web service QoS negotiation». In : *EDOC Enterprise Computing Conference, 2005 Ninth IEEE International*. IEEE, 2005, p. 70–79 (cf. p. 30).

[24] MH Dodani. «From objects to services. A journey in search of component reuse nirvana.» In : *Journal of Object Technology* (2004) (cf. p. 8).

[25] H. Feng et Y. d. Cao. «Multiple attribute decision making with intervals for QoS-based web service selection». In : *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*. 2011, p. 1041–1045 (cf. p. 26, 29, 36).

[26] Manish Godse, Rajendra M. Sonar et Shrikant Mulik. «Web Service Selection Based on Analytical Network Process Approach». In : *APSCC*. IEEE Computer Society, 2008, p. 1103–1108 (cf. p. 22, 29, 36).

[27] Kannan Govindan, Joseph Sarkis et Murugesan Palaniappan. «An analytic network process-based multicriteria decision making model for a reverse supply chain». In : *The International Journal of Advanced Manufacturing Technology* 68.1 (2013), p. 863–880 (cf. p. 22).

[28] Muhammet Gul, Erkan Celik, Nezir Aydin, Alev Taskin Gumus et Ali Fuat Guneri. «A state of the art literature review of {VIKOR} and its fuzzy extensions on applications». In : *Applied Soft Computing* 46 (2016), p. 60–89 (cf. p. 27).

[29] X. Han, Y. Liu, B. Xu et G. Zhang. «A Survey on QoS-Aware Dynamic Web Service Selection». In : *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*. Sept. 2011, p. 1–5. doi : 10.1109/wicom.2011.6040618 (cf. p. 17).

[30] Khayyam Hashmi, Amal Alhosban, Zaki Malik, Brahim Medjahed et Salima Benbernou. «Automated negotiation among web services». In : *Web Services Foundations*. Springer, 2014, p. 451–482 (cf. p. 29).

[31] Turki Hazar, Leila Baccouche et Henda Ben Ghezala. «ETUDE DE CAS POUR LA SELECTION DES SERVICES WEB BASEE SUR LES CONTRAINTES TEMPORELLES». In : fév. 2009 (cf. p. 31).

[32] Caroline Herssens, Ivan Jureta et Stéphane Faulkner. «Dealing with Quality Tradeoffs during Service Selection». In : *ICAC*. IEEE Computer Society, 2008, p. 77–86 (cf. p. 27, 29).

[33] Angus F. M. Huang, Chi-Wei Lan et Stephen J. H. Yang. «An optimal QoS-based Web service selection scheme». In : *Inf. Sci.* 179.19 (2009), p. 3309–3322 (cf. p. 29).

[34] Ching-Lai Hwang, Young-Jou Lai et Ting-Yun Liu. «A New Approach for Multiple Objective Decision Making». In : *Comput. Oper. Res.* 20.9 (1993), p. 889–899 (cf. p. 24, 25, 43).

[35] *Inverted index*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec. Accessed : 2018-11-10 (cf. p. 32).

[36] M. C. Jaeger et H. Ladner. «Improving the QoS of WS compositions based on redundant services». In : *International Conference on Next Generation Web Services Practices (NWeSP'05)*. 2005, 6 pp (cf. p. 25, 29).

[37] C. Jatoth, G. R. Gangadharan et R. Buyya. «Computational Intelligence Based QoS-Aware Web Service Composition : A Systematic Literature Review». In : *IEEE Transactions on Services Computing* 10.3 (mai 2017), p. 475–492. issn : 1939-1374. doi : 10.1109/TSC.2015.2473840 (cf. p. 31).

[38] Xing Jian, Qingsheng Zhu et Yunni Xia. «An interval-based fuzzy ranking approach for QoS uncertainty-aware service composition». In : *Optik - International Journal for Light and Electron Optics* 127.4 (2016), p. 2102–2110 (cf. p. 28, 36).

[39] Nicolas Jozefowiez. «Optimisation combinatoire multi-objectif : des méthodes aux problèmes, de la Terre à (presque) la Lune». Habilitation à diriger des recherches. Institut National Polytechnique de Toulouse (INP Toulouse), déc. 2013. URL : https://tel.archives-ouvertes.fr/tel-01104895 (cf. p. 31).

[40] FERKAL Karim et CHAIBI Yassina. «Conception et réalisation d'une application Web Service pour la gestion d'un cabinet médical.» Thèse de doct. Université Abderrahmane Mira de Béjaia, 2017 (cf. p. 9).

[41] Raed Karim, Chen (Cherie) Ding et Chi-Hung Chi. «An Enhanced PROMETHEE Model for QoS-Based Web Service Selection». In : *IEEE SCC*. IEEE Computer Society, 2011, p. 536–543 (cf. p. 28, 29, 36).

[42] R. Keeney et H. Raiffa. «Decision with multiple objectives : preferences and value tradeoffs». In : *Cambridge University Press* (1976) (cf. p. 27).

[43] Mojtaba Khezrian, Ali Jahan, Wan Mohd Nasir Wan Kadir et Suhaimi Ibrahim. «An Approach for Web Service Selection Based on Confidence Level of Decision Maker». In : *PLoS ONE* 9.6 (2014), p. 1–14 (cf. p. 28, 29, 36).

[44] Mojtaba Khezrian, Wan M. N. Wan Kadir, Suhaimi Ibrahim et Alaeddin Kalantari. «A Hybrid Approach for Web Service Selection». In : *International Journal Of Computational Engineering ijcer* 2.1 (2012), p. 190–198 (cf. p. 28, 29, 36).

[45] Mojtaba Khezrian, Wan M. N. Wan Kadir, Suhaimi Ibrahim et Alaeddin Kalantari. «A Hybrid Approach for Web Service Selection». In : *International Journal Of Computational Engineering ijcer* 2.1 (2012), p. 190–198 (cf. p. 29, 36).

[46] Tobias Kretz, Cornelia Bönisch et Peter Vortisch. «Comparison of Various Methods for the Calculation of the Distance Potential Field». In : *Pedestrian and Evacuation Dynamics 2008*. Sous la dir. de Wolfram W. F. Klingsch, Christian Rogsch, Andreas Schadschneider et Michael Schreckenberg. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 335–346. ISBN : 978-3-642-04504-2 (cf. p. 68).

[47] L. Kuang, Y. Li, J. Wu, S. Deng et Z. Wu. «Inverted Indexing for Composition-Oriented Service Discovery». In : *IEEE International Conference on Web Services (ICWS 2007)*. Juil. 2007, p. 257–264. DOI : 10.1109/ICWS.2007.113 (cf. p. 32).

[48] C. Zhang L.C Zhang T. Zhang. «Web Service Composition Algorithm Based on Hybrid-QoS and Pairwise Comparison Matrix». In : *journal of information and comptutional science jocs* 9.1 (2012), p. 135–142 (cf. p. 28, 29, 36).

[49] Matthijs van Leeuwen et Antti Ukkonen. «Discovering Skylines of Subgroup Sets». In : *Machine Learning and Knowledge Discovery in Databases*. Sous la dir. d'Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen et Filip Železný. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, p. 272–287. ISBN : 978-3-642-40994-3 (cf. p. 38).

[50] G. q. Liu, Z. l. Zhu, Y. q. Li, D. c. Li et Y. Liu. «Description and Selection Model Based on Constraint QoS for Web Service». In : *2009 WRI World Congress on Software Engineering*. T. 4. Mai 2009, p. 18–23. DOI : 10.1109/WCSE.2009.180 (cf. p. 20).

[51] Yutu Liu, Anne H. Ngu et Liang Z. Zeng. «QoS Computation and Policing in Dynamic Web Service Selection». In : *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp ; Posters*. WWW Alt. '04. New York, NY, USA : ACM, 2004, p. 66–73. ISBN : 1-58113-912-8. DOI : 10.1145/1013367.1013379. URL : http://doi.acm.org/10.1145/1013367.1013379 (cf. p. 32).

[52] Chi-Chun Lo, Ding-Yuan Cheng, Chen-Fang Tsai et Kuo-Ming Chao. «Service Selection Based on Fuzzy TOPSIS Method». In : *AINA Workshops*. IEEE Computer Society, 2010, p. 367–372 (cf. p. 28, 29, 36).

[53] N. W. Lo et Chia-Hao Wang. «Web services QoS evaluation and service selection framework - a proxy-oriented approach». In : *TENCON 2007 - 2007 IEEE Region 10 Conference*. 2007, p. 1–5 (cf. p. 25).

[54] Fang-chun. YANG Long-chang ZHANG Hua Zou. «Web service composition algorithm based on TOPSIS». In : *the journal of china universities of posts and telecommunications* 18.4 (2011), p. 89–97 (cf. p. 26, 29, 36).

[55] Zakaria Maamar, Quan Sheng, Samir Tata, Djamal Benslimane et Mohamed Sellami. «Towards an approach to sustain web services high-availability using communities of web services». In : *IJWIS* 5 (avr. 2009), p. 32–55. DOI : 10.1108/17440080910947303 (cf. p. 8).

[56] S. Maheswari et G. R. Karpagam. «Enhancing Fuzzy Topsis for web service selection». In : *IJCAT* 51.4 (2015), p. 344–351 (cf. p. 28, 29).

[57] David Martin, Massimo Paolucci, Sheila Mcilraith, Mark Burstein, Drew Mcdermott, Deborah Mcguinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan et Katia P. Sycara. «Bringing Semantics to Web Services : the OWL-S approach». In : t. 3387. Juil. 2004, p. 26–42. DOI : 10.1007/978-3-540-30581-1_4 (cf. p. 11).

[58] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan et Katia Sycara. «Bringing Semantics to Web Services : The OWL-S Approach». In : *Semantic Web Services and Web Process Composition*. Sous la dir. de Jorge Cardoso et Amit Sheth. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 26–42. ISBN : 978-3-540-30581-1 (cf. p. 16).

[59] E. Al-Masri et Q. H. Mahmoud. «QoS-based Discovery and Ranking of Web Services». In : *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*. 2007, p. 529–534 (cf. p. 47, 48).

[60] E. Al-Masri et Q. H. Mahmoud. «QoS-based Discovery and Ranking of Web Services». In : *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*. 2007, p. 529–534 (cf. p. 71).

[61] "Eyhab Al-Masri et Qusay H. Mahmoud. «Investigating web services on the world wide web.» In : *WWW Conference*. 2008, p. 795–804 (cf. p. 54, 75, 96).

[62] Brahim Medjahed. «Semantic Web Enabled Composition of Web Services». Phd. Virginia Polytechnic Institute et State University, avr. 2004 (cf. p. 90).

[63] Michael Papazoglou . *Web Services : Principles and Technology*. Prentice Hall ; 1 edition, 2007. ISBN : 0321155556 (cf. p. 9, 13).

[64] Mahboobeh Moghaddam et Joseph G. Davis. «Service selection in web service composition : A comparative review of existing approaches». In : *Web Services Foundations*. Springer, 2014, p. 321–346 (cf. p. 16–18, 24, 29, 30).

[65] Debajyoti Mukhopadhyay et Archana Chougule. «A Survey on Web Service Discovery Approaches». In : *Advances in Computer Science, Engineering & Applications*. Sous la dir. de David C. Wyld, Jan Zizka et Dhinaharan Nagamalai. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 1001–1012 (cf. p. 8, 17).

[66] Neerja Negi et Satish Chandra. «Web service selection on the basis of QoS parameter». In : *IC3*. IEEE Computer Society, 2014, p. 495–500 (cf. p. 28, 29, 36).

[67] *OASIS Consortium Members Approve UDDI Version 3 as an OASIS Standard*. http://xml.coverpages.org/ni2005-02-02-a.html. Accessed : 2018-11-11 (cf. p. 13).

[68] *OASIS UDDI Specification TC*. https://www.geeksforgeeks.org/inverted-index/. Accessed : 2018-11-11 (cf. p. 13).

[69] *OASIS Web Services Business Process Execution Language (WSBPEL) TC*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. Accessed : 2018-11-11 (cf. p. 17).

[70] *Openclassrooms*. https://openclassrooms.com/fr/courses/219329-les-services-web. Accessed : 2019-01-30 (cf. p. 11).

[71] Serafim Opricovic et Gwo-Hshiung Tzeng. «Extended VIKOR method in comparison with outranking methods». In : *European Journal of Operational Research* 178.2 (2007), p. 514–529 (cf. p. 24, 26, 41, 42).

[72] A. Ouadah, K. Benouaret, A. Hadjali et F. Nader. «Combining skyline and multi-criteria decision methods to enhance Web services selection». In : *Programming and Systems (ISPS), 2015 12th International Symposium on*. IEEE Computer Society, 2015, p. 1–8 (cf. p. 21, 29, 36).

[73] Nomane Ould Ahmed M'Barek et Samir Tata. «Services Web : revue des approches de description sémantique». In : *SIIE 2008 : Système d'Information et Intelligence Economique*. Hammamet, Tunisia, fév. 2008, p. 1–7. URL : https://hal.archives-ouvertes.fr/hal-01378724 (cf. p. 8).

[74] Didier Parigot et Baptiste Boussemart. *Architecture Orienté Service Dynamique : D-SOA*. Research Report. 2008, p. 13. URL : https://hal.inria.fr/inria-00342310 (cf. p. 10).

[75] Vikram Patankar et Rattikorn Hewett. «Automated negotiations in web service procurement». In : *Internet and Web Applications and Services, 2008. ICIW'08. Third International Conference on*. IEEE, 2008, p. 620–625 (cf. p. 30).

[76] V. Podvezko. «The comparative analysis of MCDA methods SAW and COPRAS». In : *Inzinerine Ekonomika Engineering Economics* 22.2 (2011), p. 134–146 (cf. p. 23, 37, 44).

[77] R. J. R. Raj et T. Sasipraba. «Web service selection based on QoS Constraints». In : *Trendz in Information Sciences Computing(TISC2010)*. Déc. 2010, p. 156–162. DOI : 10.1109/TISC.2010.5714629 (cf. p. 20, 25, 29).

[78] Shuping RAN. «A Model for Web Services Discovery with QoS». In : *SIGecom Exch.* 4.1 (mar. 2003), p. 1–10. ISSN : 1551-9031. DOI : 10.1145/844357.844360. URL : http://doi.acm.org/10.1145/844357.844360 (cf. p. 13).

[79] *Reference Model for Service Oriented Architecture 1.0*. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html. Accessed : 2018-11-11 (cf. p. 10).

[80] Jafar REZAEI. «Best-worst multi-criteria decision-making method». In : *Omega* 53 (2015), p. 49–57 (cf. p. 36, 37, 39, 40).

[81] B. ROY. «Classement et choix en présence de points de vue multiples». In : *RAIRO - Operations Research - Recherche Opérationnelle* 2.6 (1968), p. 57–75 (cf. p. 21).

[82] D G. SAARI. *Chaotic elections, a mathematician looks at voting*. American Mathematical Society, 2001 (cf. p. 36, 46, 62).

[83] D G. SAARI. *Chaotic elections, a mathematician looks at voting*. American Mathematical Society, 2001 (cf. p. 70).

[84] Thomas L. SAATY. «How to make a decision : the analytic hierarchy process». In : *European journal of operational research* 48.1 (1990), p. 9–26 (cf. p. 65).

[85] T.L. SAATY. *Decision Making with Dependence and Feedback : The Analytic Network Process*. RWS Publications, 1996 (cf. p. 22).

[86] T.L. SAATY. *Fundamentals of Decision Making and Priority Theory With the Analytic Hierarchy Process*. AHP series. RWS Publications, 2000. ISBN : 9781888603156 (cf. p. 22, 63).

[87] M. SATHYA, P. DHAVACHELVAN et K. VIVEKANANDAN. «Egalitarian based Negotiation model for QoS based Web Service Selection». In : *International Journal of Soft Computing* 8.2 (2013), p. 134–142 (cf. p. 30).

[88] *Semantic Annotations for WSDL and XML Schema*. https://www.w3.org/TR/sawsdl/. Accessed : 2018-11-11 (cf. p. 12).

[89] Young-Jun SEO, Hwa-Young JEONG et Young-Jae SONG. «A Study on Web Services Selection Method Based on the Negotiation Through Quality Broker : A MAUT-based Approach». In : *ICESS 2004, Hangzhou, China, December 9-10*. Springer Berlin Heidelberg, 2004, p. 65–73 (cf. p. 27, 29).

[90] Young-Jun SEO, Hwa-Young JEONG et Young-Jae SONG. «Best Web Service Selection Based on the Decision Making Between QoS Criteria of Service». In : *ICESS*. T. 3820. Lecture Notes in Computer Science. Springer, 2005, p. 408–419 (cf. p. 13, 28).

[91] W. SERRAI, A. ABDELLI, L. MOKDAD et Y. HAMMAL. «An efficient approach for Web service selection». In : *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE Computer Society, 2016, p. 167–172 (cf. p. 36).

[92] W. SERRAI, A. ABDELLI, L. MOKDAD et A. SERRAI. «Dealing with user constraints in MCDM based web service selection». In : *2017 IEEE Symposium on Computers and Communications (ISCC)*. Juil. 2017, p. 158–163. DOI : 10.1109/ISCC.2017.8024522 (cf. p. 63).

[93] W. SERRAI, A. ABDELLI, L. MOKDAD et A. SERRAI. «How to deal with QoS value constraints in MCDM based Web service selection». In : *Concurr. Comput. Pract. Exp.* 31.24 (2019). DOI : 10.1002/cpe.4512. URL : https://doi.org/10.1002/cpe.4512 (cf. p. 62).

[94] Walid Serrai, Abdelkrim Abdelli, Lynda Mokdad et Youcef Hammal. «Towards an efficient and a more accurate web service selection using MCDM methods». In : *Journal of Computational Science* Supplement C (2017), p. 253–267. ISSN : 1877-7503. DOI : https://doi.org/10.1016/j.jocs.2017.05.024. URL : http://www.sciencedirect.com/science/article/pii/S1877750317306154 (cf. p. 36, 70, 85).

[95] N. Y. Setiawan et R. Sarno. «Multi-criteria decision making for selecting semantic web service considering variability and complexity trade-Off». In : *Journal of Theoretical and Applied Information Technology Jatit* 86.2 (2016), p. 316–326 (cf. p. 28, 29, 36).

[96] Quan Z. Sheng, Xiaoqiang Qiao, Athanasios V. Vasilakos, Claudia Szabo, Scott Bourne et Xiaofei Xu. «Web services composition : A decade's overview». In : *Information Sciences* 280 (2014), p. 218–238 (cf. p. 31, 32).

[97] Y. Shi et X. Chen. «A Survey on QoS-aware Web Service Composition». In : *2011 Third International Conference on Multimedia Information Networking and Security*. Nov. 2011, p. 283–287. DOI : 10.1109/MINES.2011.118 (cf. p. 30, 90).

[98] Munindar P. Singh et Michael N. Huhns. *Service-Oriented Computing : Semantics, Processes, Agents*. John Wiley et Sons, 2006 (cf. p. 13).

[99] D. Skoutas, D. Sacharidis, Alkis Simitsis, Verena Kantere et Timos K. Sellis. «Top-k dominant web services under multi-criteria matching». In : *EDBT*. T. 360. ACM International Conference Proceeding Series. ACM, 2009, p. 898–909 (cf. p. 21).

[100] *SOAP*. https://www.tutorialspoint.com/soap/. Accessed : 2019-01-30 (cf. p. 12).

[101] *SOAP Version 1.2 Part 1 : Messaging Framework (Second Edition)*. https://www.w3.org/TR/soap12/. Accessed : 2018-11-11 (cf. p. 12).

[102] Jie Song, Hongying Hou, Tiantian Li, Guoqi Liu et Zhiliang Zhu. «QoS Cube: Management and Navigating Web Services through Multi-dimensional Model». In : *Computational Science and Engineering (CSE), 2011 IEEE 14th International Conference on*. IEEE, 2011, p. 9–15 (cf. p. 20).

[103] Anja Strunk. «[IEEE 2010 IEEE 8th European Conference on Web Services (ECOWS) - Ayia Napa, Cyprus (2010.12.1-2010.12.3)] 2010 Eighth IEEE European Conference on Web Services - QoS-Aware Service Composition : A Survey». In : 2010. ISBN : 978-1-4244-9397-5. DOI : 10.1109/ECOWS.2010.16 (cf. p. 31).

[104] Vuong Xuan Tran, Hidekazu Tsuji et Ryosuke Masuda. «A new QoS ontology and its QoS-based ranking algorithm for Web services». In : *Simulation Modelling Practice and Theory* 17.8 (2009), p. 1378–1398 (cf. p. 22, 29, 36).

[105] Evangelos Triantaphyllou. «Multi-criteria decision making methods». In : *Multi-criteria decision making methods : A comparative study*. Dordrecht, The Netherlands : Kluwer Academic Publishers (now Springer)., 2000, p. 5–21 (cf. p. 62, 69).

[106] Nazanin Vafaei, Rita A. Ribeiro et Luis M. Camarinha-Matos. «Normalization Techniques for Multi-Criteria Decision Making: Analytical Hierarchy Process Case Study». In : *Doctoral Conference on Computing, Electrical and Industrial Systems*. Springer, 2016, p. 261–269 (cf. p. 23, 25, 62, 65).

[107] Velasquez et Hester. «An Analysis of Multi-Criteria Decision Making Methods». In : *IJOR* 10.2 (2013), p. 56–66 (cf. p. 25, 26).

[108] *Web Service d'étiquetage Colissimo*. https://www.colissimo.entreprise.laposte.fr/fr/ws-edito. Accessed : 2019-02-06 (cf. p. 9).

[109] *Web Service Endurancelogistique e-commerce*. https://www.endurancelogistique.fr/webservices-e-commerce.htm. Accessed : 2019-02-06 (cf. p. 9).

[110] *Web Service Modeling Ontology (WSMO)*. https://www.w3.org/Submission/WSMO/. Accessed : 2018-11-11 (cf. p. 11).

[111] *Web Service Planyo reservation en ligne*. https://www.planyo.fr/api.php. Accessed : 2019-02-06 (cf. p. 9).

[112] *Web Service Semantics - WSDL-S*. https://www.w3.org/Submission/WSDL-S/. Accessed : 2018-11-11 (cf. p. 12).

[113] *Web Service SRD e-commerce*. http://www.srd.fr/module-gestion-panier-e-commerce/. Accessed : 2019-02-06 (cf. p. 9).

[114] *Web Services Architecture*. https://www.w3.org/TR/ws-arch/#wsdisc. Accessed : 2018-11-11 (cf. p. 8, 16).

[115] *Web Services Business Process Execution Language Version 2.0*. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. Accessed : 2018-11-11 (cf. p. 8, 17).

[116] *Web Services Description Language (WSDL) 1.1*. https://www.w3.org/TR/2001/NOTE-wsdl-20010315. Accessed : 2018-11-11 (cf. p. 11).

[117] *Web Services Description Language (WSDL) Version 2.0 Part 1 : Core Language*. https://www.w3.org/TR/2003/WD-wsdl20-20031110/. Accessed : 2018-11-11 (cf. p. 11).

[118] *Web Services Policy 1.5 - Framework*. https://www.w3.org/TR/2007/REC-ws-policy-20070904/. Accessed : 2018-11-11 (cf. p. 30).

[119] Y. Wu, C. Yan, Z. Ding, G. Liu, P. Wang, C. Jiang et M. Zhou. «A Multilevel Index Model to Expedite Web Service Discovery and Composition in Large-Scale Service Repositories». In : *IEEE Transactions on Services Computing* 9.3 (mai 2016), p. 330–342. issn : 1939-1374 (cf. p. 32, 33, 105).

[120] Youxin Xia, Chao Gao et JinLong Li. «A Stochastic Local Search Heuristic for the Multidimensional Multiple-choice Knapsack Problem». In : *Bio-Inspired Computing – Theories and Applications*. Sous la dir. de Maoguo Gong, Pan Linqiang, Song Tao, Ke Tang et Xingyi Zhang. Berlin, Heidelberg : Springer Berlin Heidelberg, 2015, p. 513–522. isbn : 978-3-662-49014-3 (cf. p. 31).

[121] Q. Yu et A. Bouguettaya. *Foundations for Efficient Web Service Selection*. Springer us, 2010 (cf. p. 54, 75).

[122] Tao Yu et K. -. Lin. «A broker-based framework for QoS-aware Web service composition». In : *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*. Mar. 2005, p. 22–29. doi : 10.1109/EEE.2005.1 (cf. p. 85).

[123] L. Zadeh. «Optimality and non-scalar-valued performance criteria». In : *IEEE Transactions on Automatic Control* 8.1 (1963), p. 59–60 (cf. p. 25, 45).

[124] Noorul Hassan Zardari, Kamal Ahmed, Sharif Moniruzzaman Shirazi et Zulkifli Bin Yusop. *Weighting Methods and their effects on multi-criteria decision making model outcomes in water ressources management*. SpringerBriefs in Water Science and Technology. Springer, 2015 (cf. p. 22).

[125] Long-chang ZHANG, Chun-jie LI et Zhan-lin YU. «Dynamic Web service selection group decision-making based on heterogeneous QoS models». In : *The Journal of China Universities of Posts and Telecommunications* 19.3 (2012), p. 80–90 (cf. p. 28, 29, 36).

[126] Hua Zou, Longchang Zhang, Fangchun Yang et Yao Zhao. «A Web Service Composition Algorithmic Method Based on TOPSIS Supporting Multiple Decision-Makers». In : *SERVICES*. IEEE Computer Society, 2010, p. 158–159 (cf. p. 26, 29, 36).

Index

G

H

I

L

M

N

O

P

Liste des sigles

AHP *Analytical Hierarchy Process*

AMCD *Aide MultiCritères à la Décision*

ANP *Analytical Network Process*

AOS *Architecture Orientée Service*

BPEL *Business Process Execution Language*

ELECTRE *ELimination et Choix Traduisant la REalité*

HTTP *Hypertext Transfer Protocol*

MAUT *Multi Attribute Utility Theory*

MMKP *Multi-dimensional Multichoice Knapsack Problem*

MOLAP *Multidimensional OnLine Analytical Processing*

OWL-S *Ontology Web Language for Services*

PROMETHEE *The Preference Ranking Organization METHod for Enrichment of Evaluations*

QdS *Qualité de Service*

SAW *Simple Additive Weight*

SAWSDL *Semantic Annotation for Web Service Description Language*

SLA *Service Level Agreements*

SOA *Service Oriented Architecture*

SOAP *Simple Object Access Protocol*

TOPSIS *The Technique for Order of Preference by Similarity to Ideal Solution*

VIKOR *VIse Kriterijumska Optimizacija kompromisno Resenje*

W3C *World Wide Web Consortium*

WSDL *Web Services Description Language*

WSDL-S *Web Service Description Language-Semantic*

WSMO *Web Service Modeling Ontology*

XML *Extensible Markup Language*

Les systèmes logiciels accessibles via le web sont construits en utilisant des services web existants et distribués qui s'interagissent par envoi de messages. Le service web expose ses fonctionnalités à travers une interface décrite dans un format manipulable par ordinateur. Les autres systèmes interagissent, sans intervention humaine, avec le service web selon une procédure prescrite en utilisant les messages d'un protocole. Les services web peuvent être déployés sur des plateformes cloud. Ce type de déploiement occasionne un grand nombre de services à gérer au niveau de mêmes répertoires soulevant différents problèmes : Comment gérer efficacement ces services afin de faciliter leur découverte pour une éventuelle composition. En effet, étant donné un répertoire, comment définir une architecture voire une structure de données permettant d'optimiser la découverte des services, leur composition et leur gestion. La découverte de services consiste à trouver un ou plusieurs services satisfaisant les critères du client. La composition de services consiste quant à elle à trouver un nombre de services pouvant être exécutés selon un schéma et satisfaisant les contraintes du client. Comme le nombre de services augmente sans cesse, la demande pour la conception d'architectures permettant d'offrir non seulement un service de qualité mais aussi un temps de réponse rapide pour la découverte, la sélection et la composition, est de plus en plus intense. Ces architectures doivent de plus être facilement gérables et maintenables dans le temps. L'exploration de communautés et de structures d'index corrélée avec l'utilisation de mesures multi critères pourrait offrir une solution efficace à condition de bien choisir les structures de données, les types de mesures, et les techniques appropriées. Dans cette thèse, des solutions sont proposées pour la découverte, la sélection de services et leur composition de telle façon à optimiser la recherche en termes de temps de réponse et de pertinence des résultats. L'évaluation des performances des solutions proposées est conduite en utilisant des plateformes de simulations.

Software systems accessible via the web are built using existing and distributed web services that interact by sending messages. The web service displays its functionalities through an interface described in a computer-readable format. Other systems interact, without human intervention, with the web service according to a prescribed procedure using the messages of a protocol. Web services can be deployed on cloud platforms. This type of deployment causes a large number of services to be managed at the level of the same directories raising different problems : How to manage these services effectively to facilitate their discovery for a possible composition. Indeed, given a directory, how to define an architecture or even a data structure to optimize the discovery of services, their composition, and their management. Service discovery involves finding one or more services that meet the client's criteria. The service composition consists of finding many services that can be executed according to a scheme and that satisfy the client's constraints. As the number of services is constantly increasing, the demand for the design of architectures to provide not only quality service but also rapid response time for discovery, selection, and composition, is getting more intense. These architectures must also be easily manageable and maintainable over time. The exploration of communities and index structures correlated with the use of multi-criteria measures could offer an effective solution provided that the data structures, the types of measures, are chosen correctly, and the appropriate techniques. In this thesis, solutions are proposed for the discovery, the selection of services and their composition in such a way as to optimize the search in terms of response time and the relevance of the results. The performance evaluation of the proposed solutions is carried out using simulation platforms.