PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF ABOU BAKR BELKAÏD – TLEMCEN –

LMD Thesis

Presented to:

FACULTY OF SCIENCES – COMPUTER SCIENCE DEPARTMENT

Thesis submitted in partial fulfilment of the requirements for the degree of:

DOCTORATE

Specialty: Networks and Distributed Systems

By:

Zeyneb Yasmina REMACI

Theme

Web service composition under uncertain QoS

Thesis publicly defended on April 11th, 2022 at Tlemcen university.

Jury members

Mr. Badr Benmammar	Professor	Univ. Tlemcen	President
Mme. Yassamine Seladji	Associate Professor	Univ. Tlemcen	Examiner
Mr. Nabil Keskes	Professor	ESI Sidi Bel-Abbes	Examiner
Mr. Lyazid Toumi	Associate Professor	Univ. Setif 1	Examiner
Mr. Fethallah Hadjilla	Associate Professor	Univ. Tlemcen	Supervisor
Mme. Fedoua Lahfa	Professor	Univ. Tlemcen	Invited

2021 - 2022

Dedication

Above all, I thank Allah the only one lord, the most merciful, and the most gracious. All praises to Allah and His blessing for the completion of this thesis.

This thesis is dedicated to:

My mother & father,

My dear sisters,

My friends.

Acknowledgment

There are no proper words to convey my deep appreciation and respect for my thesis and research supervisor, ***Dr. Fethallah Hadjila****. I want to thank him for his professional and academic assistance that made this research project much more efficient and rewarding. I am very grateful for his support, and for his willingness to spend his valuable time guiding me.*

I would like to express my sincere thanks to ***Prof. Fedoua Lahfda*** *for her guidance, continuous encouragement, and the warm spirit to finish this thesis.*

I would like to express my profound gratitude to ***Prof. Badr Benmammar****, who has honoured me by accepting to preside the jury of this thesis.*

My sincere thanks must also go to the examination committee: ***Dr. Yassamine Seladji*** *,* ***Dr. Nabil Keskes****, and* ***Dr. Lyazid Toumi****. They generously gave their time to offer me valuable comments toward improving my work.*

Last but above all, I shall be thankful to my family who have always been there for me. Their support was a major contribution to be able to pursue my PhD degree.

Loving thanks to my friends for their motivation, prayers, and unwavering belief that I can achieve so much.

Abstract

Over the last decade, Web services represent the de facto implementation of the Service-Oriented Architecture (SOA). Thereby, this technology increasingly emerges in different areas, which generate an important number of similar Web services offering equivalent functionalities. Indeed, these functionalities are, in many cases, limited regarding the user's needs.

Accordingly, selecting appropriate services according to their quality of service (QoS), and combining them in order to offer a value-added composite service is the most important goal of SOA. It is worth noting that the composition process is much more challenging because of the fluctuating environment which usually lead to QoS uncertainty.

Within this context, we propose three main contributions, each one leverages a local search for selecting the most relevant services and a global search for finding the optimal composition while considering the user's requirements and the uncertain QoS. The first approach adopts the probabilistic dominance heuristic and the backtracking search. The second approach leverages both the Majority Judgment heuristic and the Constraint Programming search. Finally, the third approach consists of Entropy and Cross-Entropy of hesitant fuzzy set and the meta-heuristic called Grey Wolf-Based Composition (GWC). The validation of the proposed approaches has shown very encouraging results.

KEY WORDS : SOA, Web services, Uncertain Quality of Service, Constraint Programming, Majority Judgment, Entropy, Cross-Entropy, Hesitant fuzzy set, GWC.

Résumé

Au cours de la dernière décennie, les services Web ont représenté la réalisation la plus efficace de l'architecture orientée services (SOA). De ce fait, cette technologie émerge de plus en plus dans des domaines différents, ce qui engendre un nombre croissant des services Web similaires en fournissant des fonctionnalités équivalentes. En effet, ces fonctionnalités sont limitées par rapport aux besoins des utilisateurs. Par conséquent, la sélection des services Web appropriés en fonction de leur qualité de service (QoS) et la composition de ces services pour fournir une application ayant une valeur ajoutée est l'objectif principal de la SOA.

Il convient de noter, que le processus de composition est beaucoup plus difficile en raison de la fluctuation permanente de l'environnement; ce qui entraîne souvent à l'incertitude de la QoS.

Pour adresser cette situation, nous proposons trois contributions principales. Chaque contribution s'appuie à la fois sur une recherche locale pour sélectionner les services pertinents et une recherche globale pour trouver la composition optimale en tenant compte des besoins de l'utilisateur et la QoS incertaine. La première approche, adopte l'heuristique de dominance probabiliste et la recherche du retour en arrière. La deuxième approche exploite à la fois l'heuristique du jugement majoritaire et la programmation par contraintes. Enfin, la troisième approche est basée sur l'entropie et l'entropie croisée de l'ensemble flou hésitant et l'approche appelée composition des services Web basée sur le loup gris (Grey Wolf-Based Composition (GWC)). La validation des approches proposées a montré des résultats très encourageants.

Mots clés: SOA, Services Web, Qualité de service incertaine, Programmation par Contraintes, Jugement Majoritaire, Entropie, Entropie Croisée, Ensemble Flou hésitant, GWC.

ملخص

على مدى العقد الماضي، إعتُمِدت خدمات الويب كتطبيق فعلي للبنية الخدمية (نمُودج SOA). هذا ما يجعل هذه التكنولوجيا بارزة بشكل متزايد في مجالات مختلفة . كما انه يؤدي الى وجود عدد مهم من خدمات الويب المماثلة التي تقدم وظائف متكافئة. في كثير من الحالات تكون هذه الوظائف محدودة فيما يتعلق باحتياجات المستخدم. وعليه فإن اختيار الخدمات المناسبة وفقًا لجودة خدمة (QoS) ، و دمج هذه الخدمات لتقديم خدمة مركبة ذات قيمة مضافة هو الهدف الأكثر أهمية بالنسبة للبنية الخدمية (SOA).

إن عملية التركيب تعد أكثر صعوبة بسبب البيئة المتقلبة التي تؤدي عادة إلى عدم اليقين في جودة الخدمة. في هذا السياق، نقترح ثلاث مساهمات رئيسية ، كل واحدة تعتمد على البحث المحلي لاختيار الخدمات الأكثر صلة. اضافة الى البحث الشامل للعثور على التركيب الأمثل مع الأخذ في الاعتبار متطلبات المستخدم و جودة الخدمة غير المؤكدة.

النهج الأول ، يتبنى الهيمنة الاحتمالية (Probabilistic dominance)اضافة الى البحث بالتراجع (Backtracking search). النهج الثاني يعزز كلاً من توجيه أحكام الأغلبية (Majority Judgment) والبحث في البرمجة المقيدة (Constraint Programming search). أخيرًا ، يتكون النهج الثالث من الانتروبيا والانتروبيا المتقاطعة للمجموعة الالضبابية مترددة (Entropy and Cross-Entropy of hesitant fuzzy set) و كذا التركيب المعتمد على الأدلة العليا المسماة تركيب خدمات الويب عن طريق الذئب الرمادي (Grey Wolf-Based Composition (GWC)). التحقق من صحة النهج المقترح أظهر نتائج مشجعة للغاية.

الكلمات الرئيسية : نمُودج SOA، خدمات الواب، جودة الخدمة غير المؤكدة، الهيمنة الاحتمالية، البحث بالتراجع، أحكام الأغلبية، البرمجة المقيدة، الالانتروبيا، انتروبيا المتقاطعة، المجموعة الالضبابية مترددة، الأدلة العليا (GWC) .

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ABC Artificial Bee Colony

EC2 Amazon Elastic Compute Cloud

AWS Amazon Web Services

ANP Analytical Network Process

AS Average Skyline

ASR Average Service Ranking

AWS Amazon Web Services

B2B Business-to-Business

BPEL Business Process Execution Language

BPML Business Process Modeling Language

BWM Best Worst Method

CORBA Common Object Request Broker Architecture

CP Constraint Programming

DCOM Distributed Component Object Model

DP Dynamic Programming

DSR Dominance Service Ranking

EAI Enterprise Application Integration

ESR Entropy Service Ranking

GA Genetic Algorithm

GCE Google Compute Engine

GQC Global QoS Conformance

GWC Grey Wolf-Based Composition

GWO Grey Wolf Optimizer

HFE Hesitant Fuzzy Element

HFS Hesitant Fuzzy Set

HTTP Hypertext Transfer Protocol

HTTPR HyperText Transfert Protocole Reliable

IaaS Infrastructure as a Service

IoT Internet of Think

IP Integer Programming

IWD Intelligent Water Drops

JMS Java Message Service

LP Linear Programming

MCDM Multi-Criteria Decision-Making

MJ Majority Judgment

MJ-CP Majority judgment and Constraints Programming approach

MOO Multi-Objective Optimization

MSR-CP Majority Service Ranking and constraint programming algorithm

NIST National Institute of Standards and Technology

PaaS Platform as a Service

PROMETHEE Preference Ranking Organization Method for Enrichment Evaluation

PS Probabilistic Skyline

p.s.g.c percentage of satisfied global constraints

PSO Particle Swarm Optimization

QoS Quality of Service

RESTful Representational State Transfer

RMI Remote method invocation

SaaS Software as a Service

SAW Simple Additive Weighting

SOA Service-oriented architecture

SOAP Simple Object Access Protocol

SOC Service-Oriented Computing

SOO Single-Objective Optimization

SMTP Simple Mail Transfer Protocol

TGA Triangular Fuzzy Genetic

TOPSIS Technique for Order of Preference by Similarity to Ideal Solution

UDDI Universal Description, Discovery and Integration

WPM weighted Product Method

WSDL Web Service Description Language

WSEL Web Services Endpoint Language

W3C World Wide Web Consortium

XML eXtensible Markup Language

XSD XML Schema Definition

Chapter 1

General Introduction

1.1 Context and motivation

"*Web services are a new breed of Web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web*".

This is how IBM defined Web services and outlined some features behind the proliferation of this technology. Actually, the Web services' goal consists of reusing the existing services while offering a lightweight integration inside the enterprise (Enterprise Application Integration (EAI)) and maintaining a data exchange between the enterprises via the exposed services.

The exchange between these enterprises is known as Business-to-Business (B2B) communication, while the communication between the companies and the final customers is commonly referred to as business-to-consumer (B2C).

Additionally, Web services improve the efficiency and decrease the cost and the time for an agile application realization. The enhanced performance of Web services is due to the fact that the interactions between the modular services are performed without previous knowledge of the internal implementation details.

Over the last years, Web services have emerged as an appealing technology for providing several solutions in different vital areas, such as Electronic Health (e.g., INBIOMED [Lopez-Alonso et al., 2004], [del Rey et al., 2005]), e-government, scientific workflows, and Internet of Things environments (e.g., smart cities). We also indicate the contribution of Web services in e-science, such as the platform of Kyoto Encyclopedia of Genes and Genomes (KEGG)[1], or the platform of the National Center for Biotechnology Information (NCBI)[2] used for analyzing the

[1]KEGG Web services: http://www.genome.jp/kegg/soap/

[2]NCBI Web services: http://www.ncbi.nlm.nih.gov/entrez/query/static/esoap_help.html

biological data.

For instance, the e-government applications (e.g., eMayor [Electronic, 2004] system) allows for the citizens to remotely request their personal and familial certificates and documents, such as birth or marriage papers. By doing so, the people will avoid the traditional cumbersome appointments and the crowding issues in public administrations. Additionally, we notice that the wide-spread of Web services in Internet of Think (IoT) systems, have brought new benefits [Kyusakov, 2014].

For instance, the embedded applications deployed in smart cities can cooperate to measure the congestion rates as well as the pollution degrees; these applications can readjust the duration of red lights to ensure more fluid circulation and less carbon emissions.

It is worth noting that such innovative applications are built using a composition of several existing IoT services [Benomar et al., 2020], and therefore, it will be crucial to design efficient models and algorithms to deal with service compositions.

We argue that the efficient realization of such workflows requires the adoption of sophisticated frameworks and protocols in order to face the challenges of the environment (including the non reliability of the devices, the intermittent connection, the variability, and the fluctuation of QoS).

1.2 Problem statement

Service-Oriented Computing (SOC) can be seen as the core model that defines the guidelines to build complex and distributed applications; it supports the development of rapid, low-cost, flexible, interoperable, and evolvable applications and systems [Papazoglou et al., 2007]. It presents a set of concepts, principles, and methods for ensuring interaction between existing (possibly heterogeneous) software components.

The combination of these principles with additional organizational aspects (such as service life cycle) will form the paradigm of Service-oriented architecture (SOA). SOA allows for building loosely coupled applications with prominent added-value. These objectives came as a remedy for the issues encountered in tightly coupled systems and even monolithic applications.

As a concrete instance of SOA, Web services implement these general principles with a set of standards and protocols [Papazoglou, 2008]. The standards and specifications describing Web services are promoted by the World Wide Web Consortium (W3C). Among the most popular frameworks for developing Web services, we can mention ASP.NET [Liberty and Hurwitz, 2005] and Axis [Tong, 2008].

Nowadays, the majority of public/private companies encapsulate their offered functionalities in terms of web services. Therefore, it is likely to encounter multiple services that provide similar functionalities while the other aspects (nonfunctional properties) may differ.
Moreover, the customers usually request for (complex) applications which are rarely fulfilled by an elementary service (since the services are functionally limited). To address the mentioned limitation, a service composition procedure of atomic services is required. By composing the necessary atomic services, a value-added application is proposed for the users, this latter application may provide emergent properties.
Figure 1.1 shows a user's request for buying a new product. Firstly, the client wants to find a product with specific requirements, then he needs to check the currency rate and exchange if needed, and finally, he desires to make a payment. It is evident, that this request can't be achieved with one service, nevertheless, it is realizable with four cooperating services.
In general, the increasing number of the available Web services on the Web has several positive and negative impacts on the composition process.
As regards, the positive impacts, we can notice that this proliferation will ensure more alternatives for a given functionality and therefore the designers will be more comfortable in choosing or replacing less reliable services. Furthermore, since the number of equivalent services is increasing exponentially, the temporal cost of building and selecting such compositions will be largely intolerable.

According to the literature [Alrifai et al., 2012], [Benouaret et al., 2012], [Dahan et al., 2019], [Remaci et al., 2020], the nonfunctional attributes also known as QoS (e.g., cost, throughput, reliability, and response time) represent the key ingredient for handling the service selection issue. Unfortunately, the QoS is largely influenced by environment conditions. We argue that the QoS is unstable because of different factors, such as the change of the network connectivity, the presence of cyberattacks, the heterogeneous client-side environments, and the service hosting architecture.
For example, due to the network load, the access to an e-commerce service will take an additional time compared to normal situations. Hence, the QoS of the service is fluctuating, nondeterministic, and dynamic; consequently, the QoS uncertainty will change the way of measuring the pertinence of a service composition as well as the selection algorithms.

We notice that the majority of classical composition approaches assume that the QoS is deterministic, ignoring the real-world situation. In this thesis, we assume that the QoS variables

are uncertain and follow an unknown probability distribution. To handle this issue, we suppose that the user's request is modeled as an abstract workflow of services that need to instantiated with concrete services.

The combination of these concrete services must meet a set of QoS requirements that express the upper/lower bounds of the QoS criteria (e.g., the total cost of the concrete services is lesser than the user's budget). In summary, our task consists of searching the Top-k service compositions that best meet the QoS requirements of the user; the QoS requirement satisfaction (also expressed as the Global QoS conformance in Chapter 4) aims to maximize the probability of preserving the global constraints.

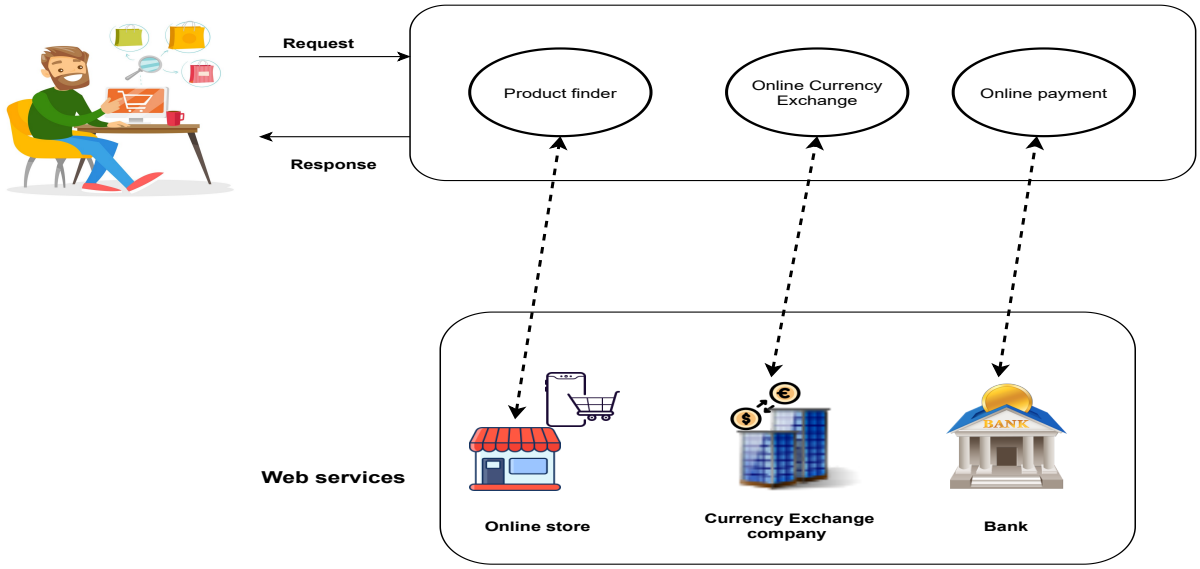

Figure 1.1: An example of Web service composition.

1.3 Contributions

It is worth noting that the processing of the user's request is not a self-evident task, since the search space is increasingly large. Therefore, our proposed composition procedure involves two major steps: a local search (for individual services) and a global search (for service compositions). The first step allows for choosing the appropriate services for each abstract class of the workflow using a given heuristic, the main intuition behind this is the reduction of the search space by filtering out the less pertinent services (See Figure 1.2). Likewise, the second step allows for retrieving the best compositions that optimize the QoS requirements (or global QoS conformance); to do so, we perform a heuristic search that ensures a trade-off between the quality of the retained solutions and the computational time.

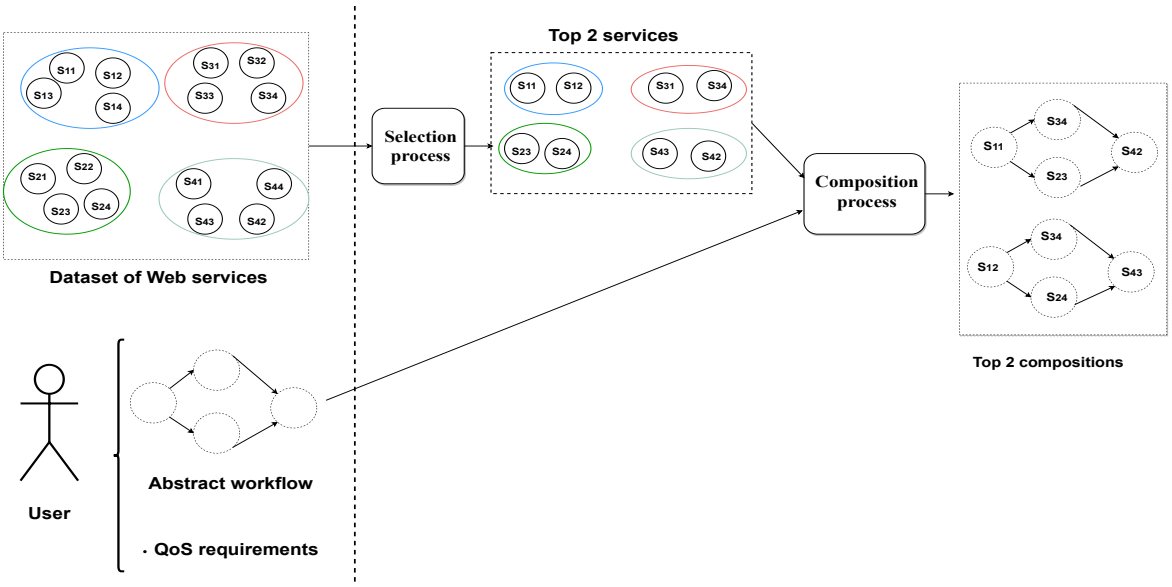

Figure 1.2: General composition procedure.

In addition, to the consideration of the functional and nonfunctional requirements of the user, our framework must handle the fluctuations of the QoS criteria during the two steps. To best refine the aforementioned framework, a literature review is performed to criticize and summarize the existing approaches.

The existing works that tackle the Web services selection and/or composition include Mathematical programming techniques (such as linear programming [Cardellini et al., 2007], mixed programming techniques [Ardagna and Pernici, 2007], Markov decision programming), heuristic and meta-heuristic techniques such as the genetic algorithm [Liu et al., 2010], PSO [Huang et al., 2011]. It is interesting to note that these methods do not consider both the user's requirements and the uncertainty of QoS. Consequently, they cannot provide reliable Web services composition in dynamic and uncertain environments. To face this uncertainty, we design a set of heuristics to handle and compare statistical samples of QoS data.

In this thesis, three major contributions are presented:

1. The first contribution [Remaci et al., 2018], is a two-stage framework that deals with the QoS fluctuations and the large space of possible compositions. The first step leverages the probabilistic dominance relationship as a heuristic to sort the services in order of merit. Based on the ranked services, we use the Backtracking algorithm to handle the service compositions while keeping only the relevant services (Top K services) provided by the first step.

2. The second contribution [Zeyneb Yasmina et al., 2022] is also structured in two major phases; the first one consists of selecting the Top K Web services of each abstract class of the user's workflow according to the Majority Judgment (MJ) heuristic. The second phase leverages the Constraint Programming (CP) approach to search the compositions and eliminate those that do not fulfill the global constraints. The retained Top K compositions must optimize the Global QoS Conformance objective function [Hwang et al., 2015]. We adopt this metric to take into account both QoS fluctuations and user's requirements.

3. The third contribution is based on a recent fuzzy Set theory (called Hesitant Fuzzy Set [Torra and Narukawa, 2009]) that deals with the QoS uncertainty. This contribution is constituted of two stages; the first one allows for ranking the concurrent services using the Cross-Entropy based hesitant fuzzy set. The comparison also leverages the concept of Entropy to compute the weight of each QoS attribute. The second stage performs a global search based on the grey wolf meta-heuristic. The proposed meta-heuristic is an improved version of the grey wolf optimization that is well suited to discrete problems, such as service compositions.

1.4 Organization of the manuscript

The rest of the dissertation is organized as follows:

- **Chapter 2: SOA and Web Service technology .**

  This chapter presents the necessary background and principles on the Web service technology, such as the Service-Oriented architecture, the fundamental standards, and protocols. In addition, the cloud computing technology, and the cloud services are briefly presented.

- **Chapter 3: State of the art.**

  The second chapter briefly introduces the related works and gives an overview of the existing methods and approaches used to solve the Web services selection and/or composition.

- **Chapter 4: Uncertain QoS-based Web service composition approach.**

  This chapter outlines the proposed approaches for solving the service composition problem. It presents the architecture of the proposed framework and details the suggested algorithms.

- **Chapter 5: Implementation and experimental results.**

  This chapter covers the evaluation of the proposed approaches while describing the exper-

imental datasets and the adopted configuration. Furthermore, it presents the effectiveness and the performance rates of all proposed algorithms.

- **Chapter 6: General conclusion and perspectives.**
  The final chapter summarizes the main contributions of the thesis and the achieved objectives, it also discusses the possible directions for future researches.

Part I

Background on Web services

Chapter 2

SOA and Web Service technology

2.1 Introduction

The increasing adoption of Internet-based technology spawned the development of large-scale distributed applications. These applications are constituted of multiple components that interact and communicate with each other to create an elaborated system. By nature, they are heterogeneous, complex, and sensitive to changes. Consequently, the major concern of the distributed computing is to find a software development architecture that handles these limitations.

Actually, SOA is the most desired architecture to enable flexible, interoperable, and loosely coupled solutions. A real realization of SOA was successfully performed through Web services in 2000 by the W3C [1]. Certainly, Web services are not the only way for implementing the SOA. Meanwhile, they achieved the goal of SOA in different domains.

Indeed, Web services are an emerging technology having minimal dependencies between components, since they use low-level standards. One of the main standards is the eXtensible Markup Language (XML) as well as the core protocol of the Internet called Hypertext Transfer Protocol (HTTP). Furthermore, there are other fundamental standards specific to Web Services such as Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI).

The goal of this chapter is to introduce a review of the SOA and present the main elements of Web services. First of all, we present a definition of Web services. Then, we explain the main participants of Web services and the operations. We will cover the two types of Web service architectures (SOAP and REST services) and we explain the three main standards for Web services (SOAP, WSDL, UDDI). Finally, We introduce cloud computing technology, and

[1] https://www.w3.org/TR/ws-arch/

the main relationship between Web services, SOA, and cloud computing.

2.2 Service Oriented Architecture

Nowadays, business organizations need for an efficient, extensible, and distributed systems supporting the high rate of changing requirements. Recently, Service-Oriented Computing (SOC) is the most desired paradigm to ensure these qualities. It provides a distributed computing infrastructure that relies on services, with smooth and consistent communication between them. In fact, this paradigm involves the Service Oriented Architecture (SOA) to offer an architectural model and expose the functionalities of applications as services. SOA is a key paradigm for modernizing the legacy software systems such as Common Object Request Broker Architecture (CORBA) or Distributed Component Object Model (DCOM), Remote method invocation (RMI). These systems are unable to deal with IT/SI challenges, like the growing number of existing complex systems, the competitive systems, and the heterogeneous implementation. Therefore, SOA is leveraged to improve interoperability, loose coupling, flexibility, agility, adaptability, integration, and reusability.

The SOA definition may slightly differ according to the expected goal (see Table 2.1).

Table 2.1: SOA definitions

Authors	**SOA definitions**
[Blanco et al., 2007]	SOA can be defined as an architectural style promoting the concept of business-aligned enterprise service as the fundamental unit of designing, building, and composing enterprise business solutions.
[Dehne and DiMare, 2007]	Service-Oriented Architecture (SOA) is a style of developing and integrating software. It involves breaking an application down into common, repeatable services that can be used by other applications, both internal and external, in an organisation—independent of the applications and computing platforms on which the business and its partners rely.
[Marks and Bell, 2008]	SOA is a conceptual business architecture where business functionality, or application logic, is made available to SOA users, or consumers, as shared, reusable services on an IT network. Services in an SOA are modules of business or application functionality with exposed interfaces, and are invoked by messages
[MacKenzie et al., 2006]	Service-Oriented Architecture (SOA) is a paradigm for organising and utilising distributed capabilities that may be under the control of different ownership domains

In fact, SOA can be implemented using different technologies. Although the most SOA-based solutions in business sectors and IT systems (such as banking, stock trading, automotive systems, and healthcare) are implemented based on Web Services. Web services are based on a set of protocols and languages that facilitate the implementation. Besides of the mentioned advantages, it is important to note that the development costs for the services and the maintenance costs are lower. In addition, the development time is faster.

2.3 Web services

There are many definitions of Web services. In this thesis, we present the W3C and [Dustdar and Papazoglou, 2008], besides IBM definitions as reference. The W3C's Web Services Architecture Working Group has given the following definition of Web service:

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

According to [Dustdar and Papazoglou, 2008]:

"*Services are self-contained processes deployed over standard middleware platforms, e.g., J2EE, or .NET that can be described, published, located (discovered), and invoked over a network... Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled.*"

2.4 Why are Web Services Attractive?

The efficiency of the Web services technology can be summarized in the following points:

- **Modularity and granularity**: In SOA, each service represents a bloc for building the business process. According to [Papazoglou, 2008], the service granularity is the unit of modularity of a system. Modularity means the amount of service functionalities.
- **Reuse**: Usually the services have specific functionality; they are enabled to be used in different and independent business processes for multiple purposes. The reusability has a high dependence on the composition and the interoperability.
- **Composable composition**: The granular services are combined to offer a new powerful service or value-added application.

- **Interoperability**: The composed services work in a heterogeneous environment, including different programmatic languages and platforms which make communication harder between them. However, in SOA there is a high-level of interoperability ensured using some standards and protocols.

- **Loose coupling**: Coupling is exchanging the information between services. In fact, the services are black boxes and commutation between them is done using standards, such as XML, which guarantees a given level of compatibility.

- **Service discovery**: This means that the available service is identifiable and funded based on the information in the service registry.

2.5 Web service Architecture

2.5.1 General Architecture

The basic Web service architecture models the interactions between three roles: the service provider, the service registry (broker), and the service requestor (Service customer).
Consequently, the main operations are: publish, find, and bind (see Figure 2.1).

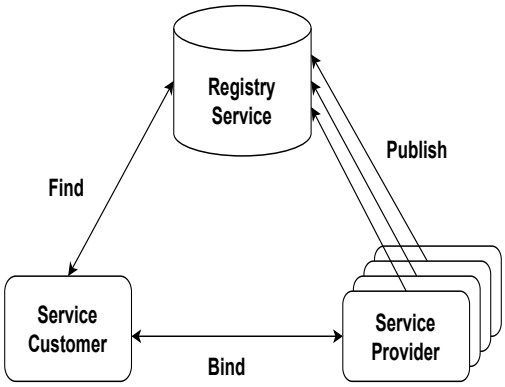

Figure 2.1: General Architecture.

2.5.1.1 Participants of Web services

Web services are based on three participants, each one has to accomplish a set of operations.

- **Service provider:** It is the owner of Web services. The service provider ensures the implementation of the service and makes an advertisement by publishing its description on the service registry.

- **Service customer:** It requests for a specific service provider that can offer the desired service. The description can be found using some service discovery mechanisms for binding to the service.
- **Service registry:** It represents a set of repositories containing the services information. It is accessible for any service customer.

2.5.1.2 Operation of Web services

Based on the description of the participants, we can notice three main operations: publishing the service descriptions, finding the service descriptions, and binding and/or invoking the services based on their service description.

- **Publish operation:** The publishing operation consists of two main operations: the description of the Web service operation (e.g., WSDL); followed by the registration of the Web service.

  The categories of the described and registered information are:

  - **Business information:** It is the information about the owner of the Web service.
  - **Service information:** It specify the categories that describe the Web services.
  - **Technical information:** It describes the realization and the invocation of the Web service.

- **Find operation:** Finding Web services starts with discovering all different services in the registry then selecting the most pertinent Web service(s). The selection can be a manual or an automatic selection. In the Manual Web services selection, the customer selects the Web service from the set obtained from the discovering operation. However, the automatic selection of the Web services is based on a full automatic process; without entailing the customer.

- **Bind operation:** The invocation and localization of Web services by the customer are achieved through the bounded details in the service description from the service registry. The architecture for Web services is founded on principles and standards for connection, communication, description, and discovery. The interoperability between these operations is based on four open technologies including Service Description Language (WSDL), Extensible Markup Language (XML), XML Schemas Definition Language (XSD), and Simple Object Access Protocol (SOAP).

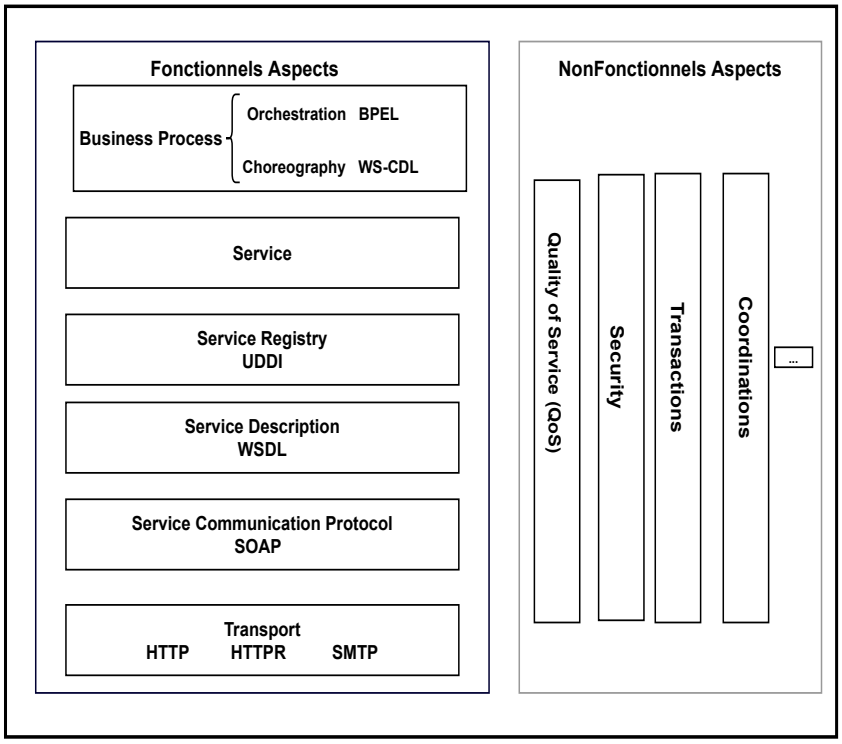

Figure 2.2: Web service stack.

2.5.2 Extended architecture

The extended Web services architecture, also called stack architecture, (see Figure 2.2) is divided into two sets. The first fold concerns the functional aspects of the architecture, and the second fold addresses the quality of service aspects (nonfunctional aspects). The detail of each layer of the stack is explained as follows:

2.5.2.1 Functional aspects

1. **Transport:** The transport layer represents the mechanism leveraged for transporting the service request and response between the service provider and the customer. There are several standards such as Java Message Service (JMS), Simple Mail Transfer Protocol (SMTP), HyperText Transfert Protocole Reliable (HTTPR); however, the most used is HTTP.

2. **Service communication protocol:** This layer is the mechanism used for communication between the service customer and the service provider. Actually, there are two popular methods of Web services implementation, called the Simple Object Access Protocol (SOAP) and the Representational State Transfer (RESTful).

- **Simple Object Access Protocol:** SOAP is W3C recommended as a communication protocol. SOAP 1.1 was originally submitted to the W3C in May 2000. It is an XML-based communication protocol using high level protocols, such as HTTP, FTP, and SMTP. It allows the service customer and the service provider communication in the heterogeneous environment with respect to the interoperability. SOAP defines the organization of the XML message as an envelope form containing two elements:

  (a) **SOAP header:** It is an optional element of the envelope, but if it exists, it must be the first child element. The header block possibly includes the origin and the destination of the document. Furthermore, it can be leveraged for digital signatures, authorizations and authentication, transaction contexts, payments...

  (b) **SOAP body:** It is a mandatory element for all SOAP messages. This element incorporates one or multiple child elements, including the mandatory message (payload) to be delivered and processed. Specifically, the body element consists of application specific data for describing the exchanged information or a fault message in the case of error occurrence.

Figures 2.3 and 2.4 show a simple SOAP request and response in an HTTP POST.

```
POST / temp HTTP / 1 . 1
Host: www.currency_convertor.com
Content−Type: text/xml ; charset ="utf-8"
Content−Length: xxx
SOAPAction: "http://www.currency_convertor.com/currency"

<?xml version ="1.0"?>
      <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
            env: encoding Style=  "http://schemas.xmlsoap.org/soap/encoding/"/>
      <env:Header/>
       <env:Body>
            <m:GetCurrency  xmlns:m="http://www.currency_convertor.com/currency.xsd">
                  <m:From>Euro</m:From>
                  <m:To>Dollar</m:To>
                   <m:Amount>100</m:Amount>
            </m:GetCurrency>
       </env:Body>
     </ env:Envelope>
```

Figure 2.3: A SOAP request.

```
HTTP / 1 . 1 200 OK
Content−Type: t e x t / xml ; charset ="utf-8"
Content−Length: xxx
SOAPAction: "http://www.currency_convertor.com/currency"
<?xml version ="1.0"?>
      <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
            env: encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"/>
        <env:Header/>
        <env:Body>
            <m:GetCurrencyResponse   xmlns:m="http://www.currency_convertor.com/currency.xsd">
                <Value>117,20</ Value>
            </m:GetCurrencyResponse>
        </env:Body>
      </ env:Envelope>
```

Figure 2.4: A SOAP response.

- **Representational State Transfer:** REST [Fielding, 2000] is a recent method for building Web services. However, it is different from SOAP, since REST is an architectural style, not a protocol. REST was firstly founded by [Fielding and Taylor, 2002], and defined as:

  "a coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations".

  This architecture, uses a stateless communication protocol, such as HTTP to exchange resources through a standardized interface. RESTful Web services depend on the nonexhaustive set of the principle discussed in follows:

  - **Resources:** It is considered as the most important concept of the REST architecture. It represents an abstraction of information having at least one representation and unique identification. This identification is performed through a URI (Uniform Resource Locator).
  - **Addressability:** In the REST style, it represents all the exposed resources; i.e., a set of URIs.

- **Stateless Interactions:** This principle uses HTTP to exchange requests between resources while ensuring isolation between them. These requests are independent from each other. Furthermore, the state is not shared between the clients and the servers. However, the context (state) connecting the server and the client is supported by the client. Furthermore, the client application can be used to provide all the information for the server to accomplish the request that occurs at any phase of interactions.
- **Uniform Interface:** Considering that the REST is based on HTTP, the interaction between resources is managed through a uniform interface. This interface considers the following HTTP operations:
  - **PUT:** Creates a new resource (new URI) or updates an available resource (existing URI).
  - **GET:** Returns the representation and the current state of a resource.
  - **POST:** It is a read-write operation leveraged to update the state of the resource.
  - **DELETE:** remove the resource with an invalid URI.
- **Self-Describing Messages:** This means that the messages between services contain both data and metadata describing the method to process this message. Furthermore, it includes the control data to describe the purpose of this message.

- **RESTful WS vs SOAP WS:**

  Actually, we can compare the implementation of Web service using RESTful over the implementation using SOAP. Technically, RESTful WS is a less complex, lighter, and more suitable solution compared to SOAP WS. Specifically, the integration in RESTful WS is lightweight and flexible. Therefore, it is generally adopted for controlling physical things through the Web [Jabbar et al., 2018].

  Table 2.2, presents the main differences between the two implementations.

Table 2.2: SOAP vs REST [Wagh and Thool, 2012].

SOAP	**REST**
Changing services in SOAP Web provisioning often means a complicated code change on the client side.	Changing services in REST Web provisioning not requires any change in client side code.
SOAP has heavy payload as compared to REST.	REST is definitely lightweight as it is meant for lightweight data transfer over a most commonly known interface, - the URI
It requires binary attachment parsing.	supports all data types directly.
SOAP Web services always return XML data.	While REST Web services provide flexibility in regards to the type of data returned.
It consumes more bandwidth because a SOAP response could require more than 10 times as many bytes as compared to REST.	It consumes less bandwidth because it's response is lightweight.
SOAP request uses POST and require a complex XML request to be created which makes response-caching difficult.	Restful APIs can be consumed using simple GET requests, intermediate proxy servers / reverse-proxies can cache their response very easily.
SOAP uses HTTP based APIs refer to APIs that are exposed as one or more HTTP URIs and typical responses are in XML / JSON. Response schemas are custom per object	REST on the other hand adds an element of using standardized URIs, and also giving importance to the HTTP verb used (i.e. GET / POST / PUT etc
Language, platform, and transport agnostic.	Language and platform agnostic.
Designed to handle distributed computing environments.	Assumes a point-to-point communication model - not for distributed computing environment where message may go through or more intermediaries.
Is the prevailing standard for Web services, and hence has better support from other standards (WSDL, WS) and tooling from vendors.	Lack of standards support for security, policy, reliable messaging, etc., so services that have more sophisticated requirements are harder to develop.

3. **Service description:** Web services architecture ensures the loose coupling through the service description that allows the independence between the service provider and the service consumer. Usually, the description includes information about the structures and the functional characteristics of the Web service.

   - **Syntactic description based on WSDL:**

     The Web Services Description Language (WSDL) is an XML language for describing a programmatic interface to a Web service [Christensen et al., 2001]. The information defined by WSDL is what Web Service can do, where it can be found, and how it can be invoked. WSDL is based on two parts of descriptions: the description of the operations and messages (structure), called the service interface definition. Additionally, the message format and the network protocol description, called the service implementation definition.

     WSDL document consists of two parts (see Figure 2.5) [Papazoglou, 2008]:

     – **The abstract interface definition:** It is the part of a Web service description that is independent from any particular protocol and message encoding used for interacting with the service. This part is also independent from the location, in terms of network address, where the service is available. This makes the abstract part reusable for different protocols, message encoding, and locations.

     – **The concrete interface definition:** Describes a specific protocol binding, message encoding, and location binding of a Web service.

     WSDL structure is constituted on the following elements:

     (a) **Definitions:** Associates Web Service with its namespaces.

     (b) **Types:** A container for data type definitions, typically using an XML Schema Definition (XSD) or possibly some other type system.

     (c) **Message:** An abstract, typed definition of the data contained in the message.

     (d) **Operation:** An abstract description of an action that the Web Service supports.

     (e) **PortType:** The set of operations supported by one or more endpoints.

     (f) **Binding:** A specification of the protocol and data format for a particular portType.

     (g) **Port:** An endpoint, defined in terms of a binding and its network address (typically a URL). This is not a TCP/IP port, which is represented by a number.

(h) **Service:** A collection of related endpoints.

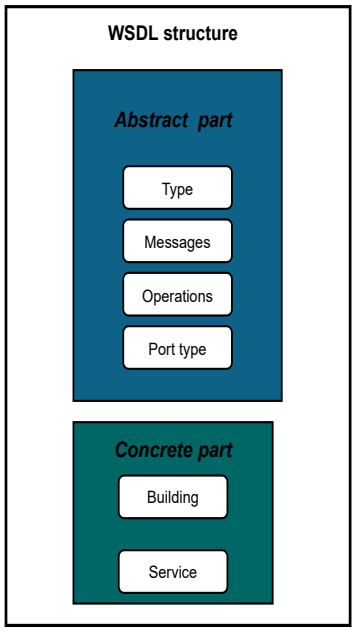

Figure 2.5: WSDL structure.

4. **Service registry:** The service registry is used for registering the service description of the available service, and then locating this description if needed. In this layer, multiple standards are emerging such as UDDI, ebXML, Web Services Inspection Language (WS-Inspection). However, UDDI is most used as a standard for registering and discovering Web services according to their description.

   - **Universal Description, Discovery, and Integration:**
     The UDDI registry allows the service provider to publish a description about the company and its services. On the other hand, it offers a searching mechanism for service customers.

     UDDI encodes three types of information about Web services (see Figure 2.6):

     (a) **White pages:** Includes the list of organizations (names) and contact details such as phone or email; also the details of service.

     (b) **yellow pages:** Includes the information about business categories, characteristics, and capabilities of companies and Web services. It is possible to search for services based on the category they belong to according to a given classification scheme.

(c) **Green pages:** Includes technical data such as the URI of the technical documents including WSDL.

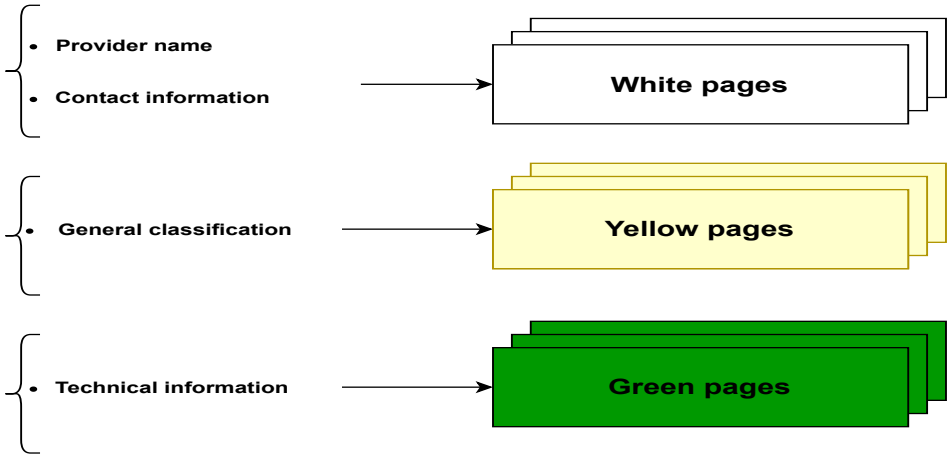

Figure 2.6: UDDI structure.

5. **Service:** It is the principal unit of SOA, which represents a business function that can be used in multiple applications (we refer to reusability). The service can be classified according to its granularity:

   (a) **Atomic service:** It is a computational entity that can offer granular functionality. This atomic service can be part of one or multiple composite services.

   (b) **A composite service:** It is a collection of atomic services in collaboration to achieve complex tasks.

6. **Business process (Composition/Processes):** A business process is a set of activities, functional roles, and relationships that describe a function of the business that accomplishes a business goal [Hollingsworth and Hampshire, 1995]. [Papazoglou, 2008] defines the business process as a set of logically related tasks performed to achieve a well-defined business outcome. For Web service composition, a business process is a collection of service tasks with defined control-flow and data-flow dependencies between them. Service composition can be achieved using one of two paradigms: the service orchestration and the service choreography (see Figure 2.7).

   (a) **Orchestration:** In [Peltz, 2003], orchestration is defined as an executable business process that can interact with both internal and external Web services.

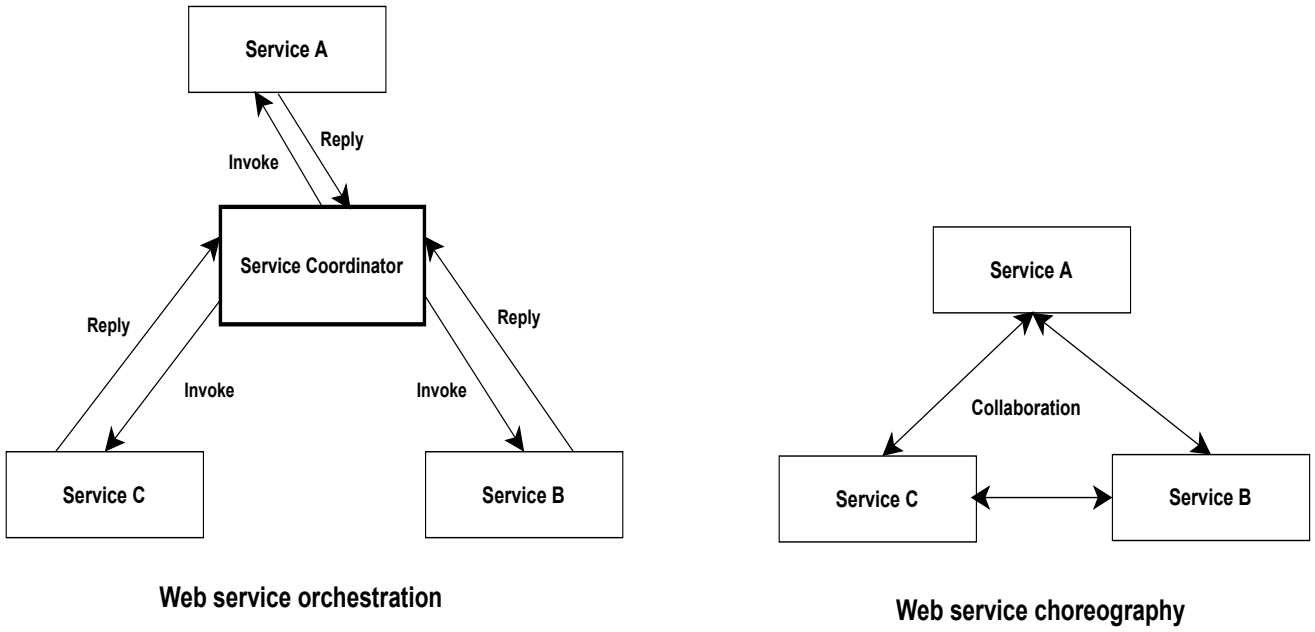

Figure 2.7: Web Services Orchestration and Choreography.

It represents the communication actions and the internal actions. The interactions are controlled by one central workflow coordinator called the orchestrator; this coordinator is responsible for the invocation and composition of Web services. Several languages are used for Web services orchestration including WSFL [Leymann, 2001], Business Process Modeling Language (BPML) [Arkin, 2002], and Business Process Execution Language (BPEL). The most used as orchestration language is BPEL.

BPEL: The Business Process Execution Language for Web Services (BPEL4WS, also known as BPEL or WS-BPEL) proposed by IBM, Microsoft, and BEA. It is an XML-based language that supports both, composition and coordination protocols. BPEL is leveraged to define the behavior of business processes as executable or abstract processes.

i. **Executable processes:** It describes the mean of execution of the process through BPE by defining the external messages exchanged with other services and the complete internal details of the business process.

ii. **Abstract processes:** It models the external message exchange between Web services based on their interfaces without including the specific details of the business process.

(b) **Choreography:** It is a decentralized cooperation where the interaction between Web services is divided across each service through send and received messages without depending on a coordinator service. The W3C Web Services Architecture [Booth, 2003], describes choreography as the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state. Many languages have been proposed for choreography, however, WS-CDL [Kavantzas et al., 2003] is the approved standard by W3C for describing the choreography.

Web Services Choreography Description Language WS-CDL: It is an XML-based language that allows specifying a global viewpoint of the interactions between services. Furthermore, it describes the data and the control dependencies.

2.5.2.2 Nonfunctional aspects (Quality of Service)

The term "Quality of Service" has been used to express nonfunctional attributes. This layer focuses on these attributes rather than the functional properties. The Web service standards, such as WSDL, only support the descriptions of the functional properties. On the other hand, the Web Service Description language (called Web Services Endpoint Language (WSEL)), which is an XML format, is used for describing the quality-of-service attributes. The QoS offered by a Web service is becoming the highest priority for service providers and their customers.
The QoS of Web services can be summarized as follows:

- **Availability:** It is the probability that a service is available and can be used when requested [Al-Shargabi et al., 2010]. It can be computed as: Number of successful invocations / total invocations [Al-Masri and Mahmoud, 2008].

- **Accessibility:** It defines the probability that a service can satisfy a request [Al-Shargabi et al., 2010]. The accessibility is different from the availability because a service might be available but inaccessible.

- **Accuracy:** It is the measure of the service error rate according to the number of errors generated by a Web service, the number of fatal errors, and the frequency of that situation.

- **Integrity:** It assures the correctness and the authorization of the transactional properties in the case of modifications [Kalepu et al., 2003].

- **Reliability:** It represents the ability to respond correctly and consistently to a service request, and it is usually expressed in terms of the number of failures per month or year. It can be measured by: Mean time between failure (MTBF), Mean Time to Failure (MTF), and Mean Time To Transition (MTTT) [Ran, 2003].
- **Security:** It specifies the mechanism proposed by the service provider and leveraged for confidentiality, non-repudiation, encrypts messages, and access control [Kalepu et al., 2003].
- **Cost (Price):** It is the total cost of the requested service.
- **Response Time:** It refers to the amount of time between sending the request and receiving a response [Bochmann et al., 2001].
- **Scalability:** It is the ability to serve the requests despite the variations in the volume of the requests [Al-Shargabi et al., 2010].
- **Performance:** It represents a measure of the speed in completing a service request. It is measured in terms of three factors: throughput, latency, and response time [Ran, 2003].
- **Throughput:** It represents the average rate of successful service requests in a specific period of time [Ran, 2003].
- **Latency:** It represents the time taken between the request sending and handling the response [Ran, 2003]. The best performance of Web service is defined by a higher throughput and a lower latency.

2.6 Cloud computing

The frequent evolution of the hardware and software platforms presents a serious preoccupation of the IT industry. The issues related to the development, deployment, maintenance, and versioning; besides the rising time for consumers as well as for software developers.
Consequently, cloud computing represents a new computing paradigm that hides the complexity of IT infrastructure and offers the opportunity for cost-effectiveness and time savings.

2.6.1 Cloud computing definition

According to the National Institute of Standards and Technology (NIST), Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool

of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [Mell et al., 2011].

2.6.2 Categories of Cloud Computing

Cloud computing is classified into three categories, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). More specifically they represent a stack of three layers (see Figure 2.8).

- **Software as a Service (SaaS):** This service (such as Customer Relationship Management (CRM), Email, Enterprise Resource Planning (ERP), Games) is provided for different categories of users using multiple devices. Furthermore, it is running on a cloud infrastructure and is controlled by service providers.
- **Platform as a Service (PaaS):** It is intended for developers, it provides all required computational resources to build their applications; having full control on the installation and configuration, and access to the environment tools. The PaaS is achieved by several platforms such as Google App Engine, Microsoft Azure Cloud Services.
- **Infrastructure as a Service (IaaS):** It offers the fundamental physical and virtual resources for hosting and developing applications, such as servers, connections, and software. In this layer, the user can control the deployed OS, applications, and data.
  The most popular IaaS applications are Amazon Elastic Compute Cloud (EC2), Microsoft Azure, Google Compute Engine (GCE).

2.6.3 Cloud Deployment Models

- **Public Cloud:** The cloud infrastructure is proposed for the general public.
- **Private Cloud:** The cloud infrastructure is exposed to a single organization that can include multiple users.
- **Hybrid Cloud:** This is a cloud infrastructure that includes both private and public clouds.

2.6.4 Most Popular Cloud Computing Platforms

There are many cloud computing platforms with plenty of advantages. This study considers four popularly adopted platforms namely Amazon Web services, Microsoft windows azure, Google app engine, and IBM cloud.

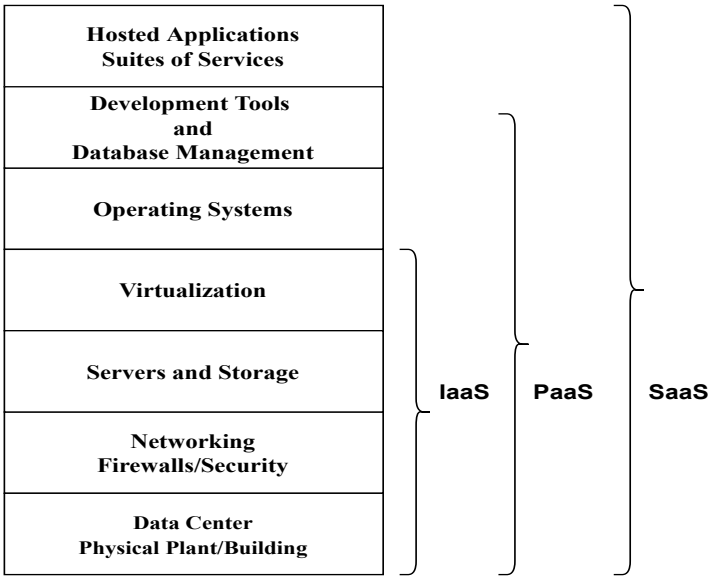

Figure 2.8: Cloud computing stack: IaaS, PaaS, and SaaS [Barry and Dick, 2013].

- **Amazon Web Services (AWS):** It represents the pioneer in cloud computing. It was founded in 2006, and it offers several cloud services to organizations, business enterprises, and normal Customers. The architecture of Amazon Web services includes four parts:
  - **Elastic Compute Cloud Amazon (EC2):** It is an IaaS cloud provider that offers to the user virtual servers (called instances). The running virtual machine on the user side represents an Amazon Machine Image (AMI).
  - **Simple Storage Service Amazon (S3):** It provides data storage as a low-cost. It is a highly available service that provides a flexible data storage space for archiving and recuperating data (Buck up). The data are available and accessible for the user at a lower cost.
  - **Amazon SimpleDB:** This is a form of non-relational database that enables a user to store data. It uses a simple read/write command from the Application program interface [Ampaporn and Gertphol, 2015].
  - **Simple Queue Service Amazon (SQS):** It Provides a hosted queue service that stores the message among the cloud components.
- **Microsoft Windows Azure:** It represents a form of Platform as a Service maintained in 2010 by Microsoft. It offers flexible services and exposes several tools and platforms for

building, executing, and managing applications.

- **Google App Engine:** It is proposed by Google in 2011. This cloud platform is classified as a platform as a service that offers a Google framework for users to develop and execute their applications avoiding the acquisition and maintaining of databases.
- **IBM Clouds:** The company information technology company IBM proposed a cloud computing platform in 2011. It offers the access of considerable computing power.

2.6.5 Web service and Cloud computing

Venn diagram (see Figure 2.9) outlines the relationship between Web services, SOA, and cloud computing. Web service encapsulates Cloud computing since cloud computing uses Web services for connection. However, Web services can be used without cloud computing.
In addition, SOA can be implemented with or without Web services. Similarly, cloud computing can be applied without having the SOA [Barry and Dick, 2013].

However, many types of research indicate that, in cloud computing, Web services are not

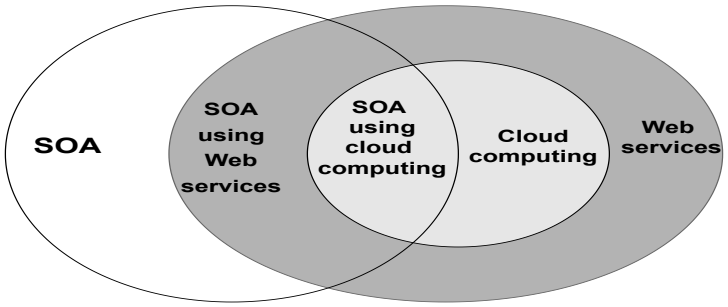

Figure 2.9: Relationship of Web services, SOA, and cloud computing. [Barry and Dick, 2013]

used only for connection. Nevertheless, they present cloud computing as a new Internet-based computing paradigm, whereby a pool of computational resources are deployed as Web services [Ai et al., 2011], [Nadanam and Rajmohan, 2012]. Specifically, the SaaS focuses on exposing software functions as services (i.e. WS) [Wang and Deters, 2009], [Jamal and Deters, 2011]. Moreover, the cloud computing providers expose an IaaS which is considered as Web service such as Elastic Compute Cloud (Amazon EC2). The latter is defined by AWS as a Web service that provides secure, resizable compute capacity in the cloud[2].

We can project the definition of IBM on cloud services, since they are modular, composable, and can be invoked across the Web. In addition, the cloud service composition is some-

[2]Amazon elastic compute cloud (amazon ec2), May 2011. http://aws.amazon.com/ec2/.

how similar to the Web service composition. Therefore, in this thesis, we address the service composition / selection. The service concept includes both cloud services (Specially IaaS, and SaaS) and Web services.

2.7 Conclusion

So far, we have begun by introducing the concept of Service-Oriented Architecture (SOA) and Web services. We have presented the most important standards and protocols that represent the foundation for Web service implementations, such as SOAP, REST, WSDL, and UDDI. Furthermore, we have highlighted the details of the general and the extended architecture.
Finally, we have outlined the cloud computing technology and its relationship with SOA and Web services.

Chapter 3

State of the art

3.1 Introduction

In most cases, the service provider offers an interesting Web service ensuring one specific functionality. Meanwhile, users usually demand complex applications that cannot be accomplished with an elementary Web service. For example, if a user wants to purchase a product, it is not sufficient to search for it, but it is necessary to take care of the payment, delivery, and so on. Invoking multiple services cooperating together involves the service composition process. We notice that the fact of choosing appropriate service components is complicated for the user, because of the proliferation of similar services that propose identical functionalities.
Despite the functional similarity, each service is characterized with the nonfunctional properties called the Quality of Service (QoS). Essentially, QoS represents the key aspect to address the service composition problem. In fact, it consists of choosing the most appropriate services and/or compositions having the best QoS, and respecting the user's requirements.

Nevertheless, the fluctuation of the SOA environment in general, and Web services specifically entails QoS instability over time. Consequently, it disturbs the composition process, since dealing with QoS as a static value is inappropriate.

In this chapter, we outline the basic concepts of Web service composition. We start by introducing some background knowledge. We then explain the role of QoS in the composition process. After that, we present the main classification of the service composition approaches and related work of each category. Finally, we address the researches based on uncertain QoS.

3.2 Service composition

Many definitions are introduced for Web service composition. In [Georgakopoulos and Papazoglou, 2009], the service composition is defined as a task of developing complex services by composing other elementary or composite services. An elementary service provides an access point to Internet-based applications that do not rely on other Web Services to fulfil external requests. A composite service aggregates other elementary services and composite services that collaborate to implement a set of operations.

Another definition by [Paik et al., 2017], mentions that the activity of aggregating Web services to build a new service is known as Web Service Composition. Meanwhile, the most simple definition is that the service composition refers to the process of combining several services to provide a value-added service [Luo et al., 2010].

3.2.1 Service composition constituent

1. **Activities:** An activity indicates the work that has to be done. Activities can be atomic, in which case they are called tasks, or non-atomic, in which case they are named subprocesses [Paik et al., 2017].

2. **Data-flow:** It is the message flowing between the activities, for the composite Web service. It represents the message exchanges (called the message flows) that join the services.

3. **The control flow:** It represents the execution order of atomic Web services in the composition.

4. **Workflows:** A workflow usually contains a set of activities and the sequence among them [Tan and Zhou, 2013]. An activity can be either a manual activity that needs human intervention or an automated workflow activity that will be executed by a software application. As regards Web services, the workflow for Web service composition is usually composed of tasks and transitions which denote the dependencies between tasks and are associated with an enabling probability [Cardoso et al., 2004]. According to recent studies, the elemental structures for the workflow model include sequence, parallel, choice, and loop.

5. **Global constraints:** It is one of the most important concepts for filtering alternatives. It represents the requirements over some sequence of variables.

6. **Customer:** Organization or person that receives a product or service [1].

7. **End-user:** Person or persons who will ultimately be using the system for its intended purpose [2].

8. **Objective Function:** The general objective function intended by the research community for service selection is to maximize the service requester's satisfaction of the composite service execution [Moghaddam and Davis, 2014].

3.2.2 QoS roles in Web service selection

Quality of service is widely addressed in the past few years; starting from ensuring the best quality for telecommunication and Computer networking, up to extending this concept to cover several domains such as Web services, cloud services, and IoT systems. The international quality standard ISO 8402 describes quality as the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.

Actually, the QoS is a collection of characteristics (i.e,. criterion) describing the nonfunctional attributes. Each one is represented as quantitative values and limited by range and boundaries. In the Web services' context, the values of these QoS attributes can be either directly collected from service providers (e.g., price), recorded from previous executions (e.g., response time), or collected from user's feedback (e.g., reputation) [Liu et al., 2004].

The set of QoS attributes can be divided into positive and negative QoS attributes.
The values of positive attributes need to be maximized (e.g., throughput, and availability), whereas the values of negative attributes need to be minimized (e.g., price, and response time) [Alrifai et al., 2010], [Alrifai and Risse, 2009]. According to [Zheng et al., 2010], there are only two categories that incorporate the user-dependent and user-independent. In advance, the user dependent categories consist of all QoS attributes that are different for each user, such as throughput and response time. The user independent category includes the attributes that are similar for all users. QoS properties in this category include popularity, price, etc. According to [Wang et al., 2012], the QoS attributes can be classified as measurable or immeasurable QoS. Measurable QoS attributes are quantifiable using a QoS metric, such as average execution time, while immeasurable QoS attributes are naturally qualitative (e.g. flexibility, reputation, etc.).

In [KangChan et al., 2003], the classification is different, where the QoS attributes can be viewed as an application and network attributes. The network categories include the QoS

[1] SOURCE: ISO/IEC 19770-5, 3.10
[2] SOURCE: ISO/IEC 19770-5, 3.13

attributes such as the Network delay, delay variation, packet loss. The latter attributes are those that indicate the communication between a Web service and the outside world.

Nevertheless, the application QoS attributes are descriptive of the Web services application, such as availability, reliability, cost, etc. On the other hand, the QoS can be seen as deterministic or nondeterministic [Liu et al., 2004]. An attribute is considered deterministic if its value is recognized before the service invocations (e.g., price, security). However, if the attribute is unknown at service invocation time (e.g., response time or availability), then this attribute is considered Nondeterministic.

Table 3.1 summarizes the different classifications of QoS properties.

Frequently, each user request includes two specifications, the functional description and QoS requirements that represent the QoS constraints over the requested services.

On the other hand, many service providers can functionally respond to the user request, meanwhile, each one rushes to offer the best QoS or at least reach the QoS target specified by them. Managing the provider's concurrence during the composition process can be achieved by differentiating between multiple candidate services based on the offered QoS, this step represents the local selection. Likewise, to control the number of possible compositions, the similarity between the QoS of the composition and QoS requirements requested by the user is considered, which represents the global selection.

Actually, the composition process may include the local or the global selection; however, if it includes both, it is considered as hybrid selection (see section 3.4 for more details). Leveraging the QoS in the composition process is known as QoS-aware service composition.

3.2.2.1 QoS aggregation functions

Practically, moving from the QoS of the atomic services to the QoS of the composite services is handled according to the workflow model and a QoS aggregation function. The latter is a mechanism used to compute the overall QoS of the composition based on a set of functions. It depends on the workflow structures. Broadly speaking, aggregating the QoS in sequential workflows is different from parallel or loop workflows.

In this thesis, we leverage the aggregation function proposed in [Hwang et al., 2015] (see Table 3.2).

3.2.2.2 Weights of QoS Attributes

Usually, the user has some preferences or importance concerning each QoS attribute during the composition process. Nevertheless, many researchers claim that the user has an incomplete

Table 3.1: QoS classifications

QoS	Positive	Negative	Mesurable	Immeasurable	Application attributes	Network attributes	Deterministic	Nondeterministic
Response time		✓	✓		✓			✓
Cost		✓	✓		✓		✓	
Network delay		✓	✓			✓		✓
Throuthput	✓		✓	✓				✓
Reliability	✓					✓		✓
Reputation	✓		✓			✓	✓	
Avalability	✓		✓			✓		✓

Table 3.2: QoS aggregation functions [Hwang et al., 2015]

Function	Attribute		
	Response time	Reliability	Fidelity
Sequential	$\sum_{j=1}^{n} q(s_j)$	$\prod_{j=1}^{n} q(s_j)$	$1/n * \sum_{j=1}^{n} q(s_j)$
Loop	$\sum_{i=1}^{k} q(s)$	$\prod_{j=1}^{n} q(s_j)$	$q(s_j)$
Parallel	$\max_{j=1}^{n} q(s_j)$	$\prod_{j=1}^{n} q(s_j)$	$1/n * \sum_{j=1}^{n} q(s_j)$
Conditional	$\max_{j=1}^{n} q(s_j)$	$\min_{j=1}^{n} q(s_j)$	$\min_{j=1}^{n} q(s_j)$

idea about the QoS preferences. Therefore, many works proposed solutions for computing QoS weights. Accordingly, we can distinct four main approaches depending on the leveraged QoS weights:

- **Subjective weights:** These approaches use the weights enforced by the users. These weights are extracted from their feedback data [Karim et al., 2011].
- **Objective weights:** In these approaches, some systematic methods are adopted to compute the QoS weights for adjusting the user preferences, such as Entropy proposed in [Sun et al., 2016].
- **Combined subjective and objective weights:** These approaches integrate both subjective and objective weights, namely the work proposed in [Ouadah et al., 2018].
- **Equitable QoS weights:** In these methods (such as [Remaci et al., 2018]), equitable weights are assigned for each QoS.

3.2.3 Composition categories

Web service composition can be categorized according to the involvement of the user in the composition process into three categories: manual, automatic, and semi-automatic compositions. It can be also categorized according to the flexibility and dynamicity of the composition process, where we notice the static and dynamic composition.

- **Static composition:** A static composition means that the composition process is statically formed, deployed, and fixed by building the process model including the atomic services and the dependencies between them. In this case, there is no possibility of modification after execution, even if the service functionalities or the composition requirements change.
- **Dynamic Composition:** Since the environment is dynamic during the runtime and the composition process should adapt to this flexibility, the dynamic composition is per-

formed. According to [Rao and Su, 2004], the dynamic composition is achieved by creating the abstract model of tasks and selecting the atomic Web services automatically without the interference of the service requestor in the composition process.

- **Manual composition:** The Web service composition called manual is the basic method for creating a composition where the user is involved to select and build the interconnection (workflow) between Web services. In addition, the execution of the composition is made by executing the services one by one. Consequently, the modification or extension of the composition is complicated. Furthermore, this type of composition is usually handled by the programmer through the use of business process languages, such as Web Ontology Language for Web Services (OWL-S), IBM's Web Services Flow Language (WSFL), and Business Process Execution Language for Web Services (BPEL4WS), to specify the composition schema. The presence of many tools (such as ZenFlow, Sedna, and Tavern), and techniques does not prevent that creating a manual Web service composition is a hard task, error-prone, and time-consuming.

- **Semi-automatic composition:** The semi-automatic composition is an enhanced and faster method for composing service, compared to manual composition where both system and users are involved in the process of finding, selecting, and composing automatically the services. In this approach, the workflow is designed before starting the composition process.

- **Automatic composition:** Automatic service composition is a growing area leveraged to overcome the drawback of manual composition. The automatic Web service composition aims to eliminate or reduce the user's implication in the composition process that accelerates the composition process. Automatic composition is the process where the workflow is automatically implemented through algorithms based on a set of constraints and preferences. In fact, the process model will be automatically generated and the composition will be dynamic and would change during the run-time.

The automatic Web service composition is characterized by four groups of approaches:

- Workflow-based approaches.
- Model-based approaches.
- Mathematics-based approaches.

- Artificial Intelligence (AI) planning approaches.

In this section, we will focus on the Workflow-based approaches and AI planning approaches.

- **Workflow-based approaches:** One of the main approaches of automatic Web service composition is workflow-based planning. In fact, the service components are considered as activities and the composition is considered as a workflow including the data and the control flow. The workflows generation can be either static or dynamic. In the static workflow-based approaches, the abstract process model is specified by the user in advance. Meanwhile, the process model is handled automatically in dynamic workflow-based approaches. The user is enquired to define the constraints and the preferences.

- **AI planning approaches:** These approaches are leveraged for fully automated Web service composition. The service composition problem is considered as a planning problem. Whence, the aim of the planner agent is generating a planning process (composition) based on a set of possible actions to move from the starting state to the final state.

  In the literature, we can find many AI planning approaches to solve service composition problems, which are devised into three main categories of planning techniques: Classical planning also called state space-based planning, Neoclassical planning, and Control Strategies.

3.3 Service composition life cycle

In this section, we present the Web service composition process. Generally, the service composition process proceeds in four subsequent phases (see Figure 3.1).

1. **Service request and planning:** In this phase, the end-user describes the composite service requested by designating the preferences and the constraints. Afterward, the request will be decomposed into abstract sets of tasks using specific methods. Besides the decomposition, the control flow is identified to determine the invoking order of the component services.

2. **Service Discovery:** It is the process of finding Web services from registries according to the user's requests. The research of services is performed according to the available description. Eventually, a set of services is discovered.

3. **Service composition:** This step is composed of two sub-phases. In fact, the previous phase returns multiple similar services offering similar functionalities. Therefore, the first

sub-phase called the service selection is adopted to perform the local selection of the optimal services according to the nonfunctional properties (QoS). The second sub-phase is the generation of the possible composition according to the specific workflow, then the selection of the optimal composition.

4. **Execution:** The last phase of the life cycle is the execution of the composition. The service components are invoked under a defined order and executed to exchange the messages. This composition is controlled and monitored in the case of failure.

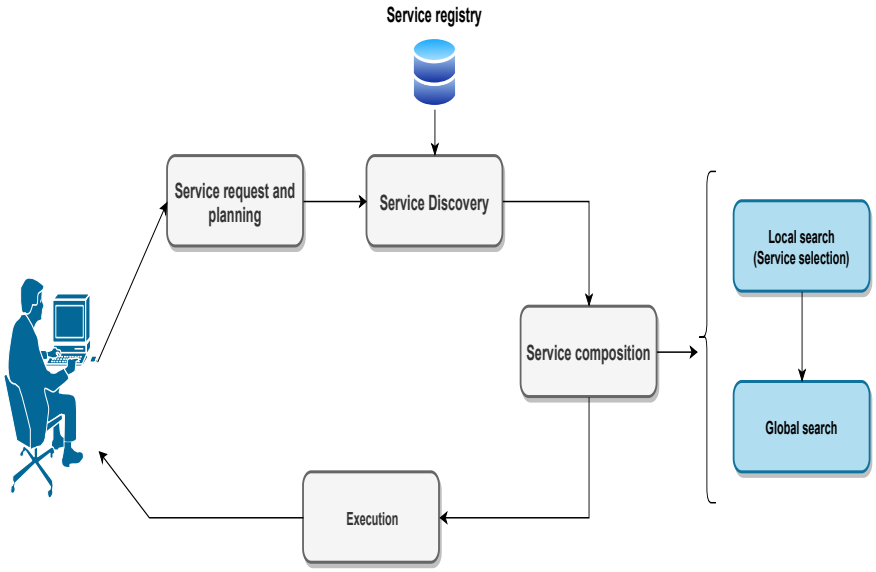

Figure 3.1: Web services composition life cycle.

3.4 Web service composition solutions

The augmentation of the functionally similar Web services represents a major challenge for Web service composition. Broadly speaking, to offer a value-added application, we need to generate m^n combinations (n is the number of tasks and m is the number of services per task). Then we select the optimal solutions (Top K compositions). In this case, local Web service selection is suitable to reduce the number of services related to each task. The main key of characterization between services is Quality of Service (e.g., the response time, availability, and reputation).

Up to now, QoS-aware Web service composition is widely handled by several researchers, in which it is considered as a Single-Objective or a Multi-Objective Optimization [Hadjila et al., 2020].

- **Single-Objective Optimization (SOO):** SOO can be described as an optimization technique based on a single objective function where the latter is maximized or minimized. The objective function includes a combination of variables with or without the constraints.

- **Multi-Objective Optimization (MOO) :** (also called multi-criteria optimization, multi-performance, or vector optimization problem) is an optimization technique that involves a set of constraints to be satisfied and multiple objectives to be either maximized or minimized. Solving MOO problems is solving multiple sub-problems. The provided solutions are known as Pareto-optimal solutions that provide a good trade-off between the objectives.

  In general, there are four fundamental notions in MOO: Pareto dominance, Pareto front, Pareto optimum, skyline.

  - **Pareto dominance:** Given the choice between two objects, with one object being better concerning at least one attribute but at least equal with respect to all other attributes, users will always prefer the first object over the second one (the first object dominate the second one). In general, we can say that solution i dominates another solution j; if the solution i is better than or equal to solution j in all objectives and is strictly better than solution j in at least one objective. The definition of the following concepts is based on Pareto dominance.
  - **Pareto optimum solution (nondominated set):** It is the best solution having at least one best objective with preserving the goodness of the other objectives. Broadly speaking, this solution is not dominated by any other solution. A Pareto optimal solution denotes that there exists no other solution that can increase the quality of a given objective without decreasing the quality of at least one other objective [Talbi, 2009].
  - **Pareto front:** It represents the set of all equivalent Pareto-optimal solutions where another better solution does not exist.
  - **Skyline**: The skyline includes all objects that are not dominated by any other object. Web service with multiple QoS attributes is similar to multidimensional object. Hence, we can formulate the skyline set denoted as WS_sky of a set of Web services S as follows:

$$WS\_sky = \{s_i \in S | \nexists s_j \in S : s_j \prec s_i\}. \tag{3.1}$$

The Pareto dominance concept can be used to implement an intuitive retrieving since the dominated objects can be retrieved from the data collection or the final data in the Skyline set. For example, there is a user's request to find products that are both trendy and cheap. Since a trendy product is usually more expensive than another, it is hard to define an "optimal" product satisfying both conditions. However, the expensive products and not trendy will be eliminated from the desired list of products. The rest of them represents the skyline set.

In fact, Pareto dominance is adopted to establish preferences among a given set of solutions. Therefore, it is leveraged to select the most suitable Web service according to the user's request. Figure 3.2 illustrates the dominated set, nondominated set, and Pareto front in the case of a request for Web services with high accuracy and availability (maximization).

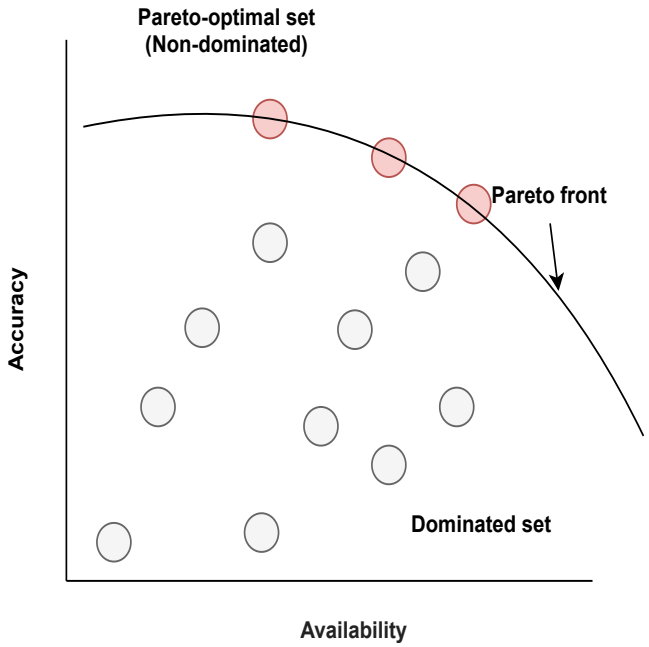

Figure 3.2: Example of Pareto dominance.

Different methods are proposed for solving the MOO problem; these methods are divided into three categories:

1. **Classical methods:** Also called Aggregative Methods, it consists on transforming MOO to SOO through aggregating the objectives into a scalar function.

Many techniques are proposed, such as :

- **Weighted Aggregation:** This technique is performed by leveraging a weight for each objective, then transforming all objectives into one objective function to be solved by a single-objective optimization method.
- ϵ**-Constraint:** The basic idea of this technique is finding Pareto optimal solutions through the consideration of one objective and handling the other objectives as constraints bounded by vector ϵ.
- **The goal-programming method:** It is an optimization program. It can be thought of as an extension or generalization of linear programming to handle multiple, normally conflicting objective measures. Each of these measures is given a goal or target value to be achieved [Bhattacharya and Chatterjee, 2014].

2. **Non-Pareto-Based Approaches:** These methods address multiple objectives at the same time [Prakash and Kumar, 2021], without leveraging the transformation into SOO or integrating directly the concept of Pareto optimality. In general, these methods are not capable of reaching certain fragments of the Pareto Front. Various resolution methods are proposed such as Vector Evaluated Genetic Algorithm (VEGA) [Schaffer, 1985].
3. **Pareto-Based Approaches:** These methods are classified as a posteriori methods [Branke et al., 2008]. In this category, all objectives are considered when handling MOO problems. However, there are no preferences on any objective; this means that finding a trade-off between objectives is primordial. This trade-off is handled through the concept of dominance. The resolution techniques aim to find all possible nondominated solutions (such as Nondominated sorting genetic algorithm-II (NSGA-II) [Deb et al., 2002]).

3.4.1 Composition strategies

We can mention three main composition strategies to preform the Web service composition process:

- **Global selection:** The global selection aims to select near-optimal compositions that ensure meeting the user's requirement, called global QoS constraints. Nevertheless, it has an exponential complexity and the computational cost is NP-hard (i.e., Nondeterministic Polynomial-time Hard).

- **Local selection:** A local selection strategy focuses on selecting the most relevant service for each activity in the abstract process, that will be combined to form a composition.
  The selected services have the best trade-off between the required QoS attributes. The concerned approaches have linear complexity and low computational cost. However, there is no guarantee that the global constraint is satisfied since it handles only the local constraints.

- **Hybrid selection:** It is a compromise of the two approaches, it has a reduced complexity in comparison with the global approach, and it can also handle the global constraints.

3.4.2 Classification of Web service composition methods

According to [Hadjila, 2014], the QoS-aware Web service composition can be branched into four main classes according to the leveraged approach.

1. **Exact methods**

   The main characteristic of these methods is the optimality of the final solution.
   Generally, the exact methods are leveraged for reasonable size problem instances. In the case of large-scale problem instances, the computation time with regard to these methods is increasingly high.
   Many exact methods have been proposed and have demonstrably improved the ability to obtain optimal solutions such as Integer Programming (IP), Dynamic Programming (DP), Linear Programming (LP), or graph algorithm. As regards Exact methods, [Gabrel et al., 2014] leverage a new ILP model for transactional QoS-aware Web service composition. The service repository is modeled by a Service Dependency Graph (SDG). From this graph, the 0-1 linear programming model is leveraged for finding the optimal Web service composition.

   In the same context, the work [Liu et al., 2012] presents a branch and bound-based approach called (BB4EPS). This approach is used to solve the QoS-aware service composition problem by considering user preferences and constraints. In literature [Alrifai et al., 2012], the researchers introduce a centralized QoS-aware service composition approach with considering a complex composition model (sequential, iteration, parallel, and conditional). The proposed approach starts by leveraging MIP-solving techniques to find the optimal decomposition of global QoS constraints into local constraints. Then, the distributed local selection aims to find the best Web services that satisfy these local constraints. The

proposed MIP model requires a few QoS data of available services to improve the overhead of the composition process. In [Wang et al., 2017], the authors propose a new approach called SCORE-QoS based on Correlated QoS Requirements. In this approach, the Service Selection problem is modeled as a Constraint Optimization Problem (COP) with integrating two types of QoS requirements (with or without users' optimization goals). To find the optimal solution for COP the Integer Programming (IP) is employed.

In [Fan et al., 2018], the authors leverage an efficient mechanism to address the Web service composition problem. The service dependency graph is generated to describe services and the relations among them. In addition, the composition problem represents the searching for a reachable path in the service dependency graph, such as the knapsack-variant algorithm. Furthermore, an optimization strategy is proposed to reduce the complexity of the knapsack-variant algorithm.

The authors in [Ghobaei-Arani and Souri, 2019], propose the linear programming-based approach, called "LP-WSC", to handle the QoS-aware Web service composition in a geographically distributed cloud environment. Firstly, the Z-score normalization method is applied to normalize the QoS-data since Z-score is adapted to handle the outlier data. Afterward, the nearest center is selected according to the user's geographical distance. Finally, the linear programming technique is used to select the most appropriate composition that meets the user requirements delivered in the form of a Service Level Agreements (SLAs). However, the increasing number of decision variables leads to exponentially increasing complexity and exacerbated cost.

2. **Heuristic methods** The approaches that belong to this category are advised for a specific optimization problem, but propose a near-optimal solution in lower computation time.

   In [Baranwal and Vidyarthi, 2016], the authors consider multiple QoS criteria and introduce QoS metrics to measure the cloud services. Furthermore, an improved ranked voting method (IRVM) is leveraged with considering the importance of QoS metrics for a voting framework to select the best provider with the highest score.

   In [Wang et al., 2017], the authors propose a new approach called SCORE-QoS based on Correlated QoS Requirements. In this approach, the Service Selection problem is modeled as a Constraint Optimization Problem (COP) with integrating two types of QoS requirements (with or without users' optimization goals). To find the optimal solution for

COP the Integer Programming (IP) is employed.

The QoS-based Web service selection can be reported to the Multi-Criteria Decision Making (MCDM) methods to find the most pertinent Web service from multiple and similar Web services such that the QoS is optimized and users' QoS requirements are satisfied.

In [Purohit and Kumar, 2018], an improved Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) [Mareschal et al., 1984] is presented to address the problem of Web Services Selection (WSS). Based on the classification techniques, the number of equivalent services is reduced; additionally, the hybrid QoS weight evaluation scheme depending on a Maximizing Deviation Method (MDM) and PROMETHEE-PLUS are leveraged to select the Top K Web services with respecting end-user QoS requirements.

The work in [Ouadah et al., 2018], introduces the Skyline-Entropy-Fuzzy-Ahp-Promethee (SEFAP) approach for skyline services selection. This approach is based on the Block Nested Loops (BNL) algorithm to generate the skyline Web service. Furthermore, the Entropy and Fuzzy AHP (Analytic Hierarchy Process) are used to extract the objective and subjective weights. Finally, the PROMETHEE method is proposed to rank the skyline Web services.

The work presented in [Al-Faifi et al., 2019] employs a hybrid MCDM method to select the best cloud service provider (CSP) from smart data. The hybrid method integrates the clustering by adopting the k-means algorithm that regroups services providers into k clusters. Then, DEMATEL is adopted to obtain one proxy from each cluster and to determine interdependency and relations between criteria. Finally, the Analytical Network Process (ANP) is used to rank the clusters and provide the pertinent CSP. The hybrid multi-criteria decision-making model is proposed in [Jatoth et al., 2019] for the QoS based cloud service selection. The hybrid model called (EGTOPSIS) consists of both AHP and extended versions of Grey TOPSIS to compute the weights of criteria; the latter is then used for ranking the services. The research in [Serrai et al., 2018], exploits the AHP method to generate normalized weights of the QoS criteria derived from the user's request. Then, they introduce a novel approach called OMRI (Optimized method of Reference idea). OMRI is an improvement of the data normalization technique called RIM (Reference Ideal Method), since RIM does not guarantee a good ranking due to the data

normalization technique used in the process. Additionally, the OMRI method is combined with different appropriate MCDM ranking methods (weighted Product Method (WPM), Simple Additive Weighting (SAW), VIKOR, and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)) for ranking Web services.

In [Youssef, 2020], the Best Worst Method (BWM) is leveraged to acquire the weights of criteria and relative scores for alternatives. These weights and scores are used in the TOPSIS method to rank Cloud services.

In a more recent work [Tiwari and Kumar, 2021], the authors adopt the Gaussian TOPSIS (G-TOPSIS) for a robust reversal ranking. This method is leveraged to address the MCDM-based cloud service selection based on QoS. To evaluate the proposed approach, the authors use the CloudHarmony dataset as a benchmark service provider.

3. **Meta-heuristic methods:** The term of **"meta-heuristic"** was firstly proposed in [Glover, 1986]. Where, **meta** means "beyond" or "higher level" and **heuristic** is the Greek word that means "heuriskein" or "to search". Generally, these approaches perform better than simple heuristics. They are leveraged to solve a large part of different NP-hard problems since the latter require a high computation time. In fact, most of meta-heuristic approaches represent an iterative and random process. Moreover, many of them are inspired from the natural systems. The main characteristic of the meta-heuristics is that they are not proposed for a particular problem, but they are approximative and search for a good enough solution.

   Meta-heuristic algorithms mimic the exploitation and exploration behaviors (also called intensification and diversification). The exploration means exploring the search space by generating diverse solutions, while exploitation means exploiting the local area and searching for good solutions in this region.

   The meta-heuristics can be classified into two classes (see Figure 3.3); population-based methods and point-to-point methods. The population-based methods (such as genetic algorithms and particle swarm optimization) are characterized by invoking multiple particles. However, point-to-point methods use a single agent or solution and improve this solution using local search. For instance, the simulated annealing and Tabu search are point-to-point methods.

   The work by [Zhao et al., 2012], proposes an improved ant colony optimization algorithm

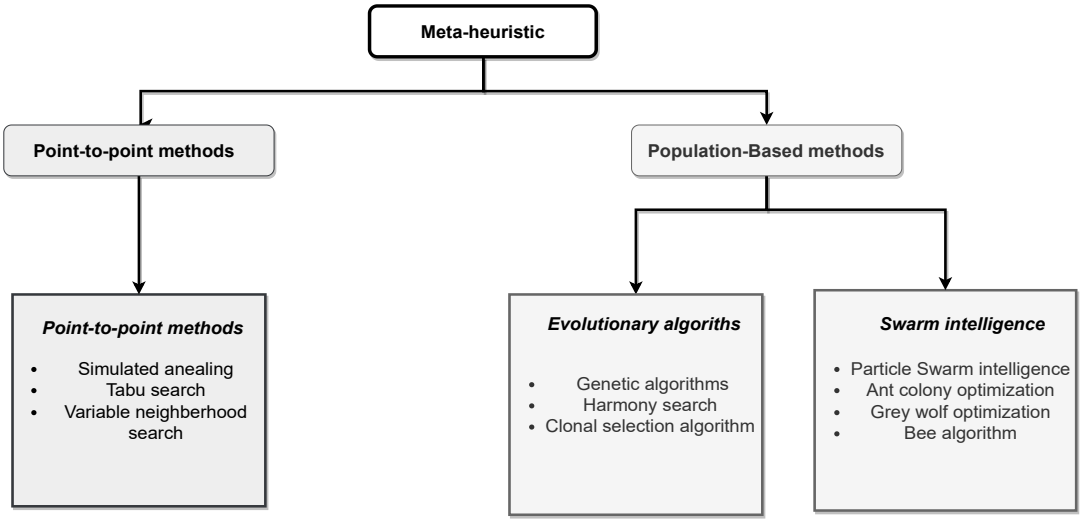

Figure 3.3: Classification of meta-heuristic methods.

to solve the multi-dimension multi-objective composition optimization. The implemented ACO algorithm introduces certain changes in the pheromone and transition probability.

The authors in [Merzoug et al., 2014] suggest an approach based on the harmony search. This approach addresses the mono-objective service selection to retrieve the near-optimal compositions while considering the user's constraints. Similarly, in [Bekkouche et al., 2017], the authors propose a new framework for QoS-aware automated Web service composition. They employ a planning graph that contains all possible compositions that satisfy the QoS constraints. Then, the compositions are ranked using the harmony search meta-heuristic.

Genetic algorithms and their extensions are widely used to tackle the QoS-aware Web service selection problem. To overcome the limitation of the basic genetic algorithm, the work in [Li et al., 2018] presents a cooperative evolutionary genetic algorithm (CGA) based on a real coding method. Furthermore, the entropy method is used to determine the weights of each QoS attribute. [Ranjan and Sahoo, 2020] is another work that introduces an approach based on the genetic algorithm for selecting the optimum composition in a fog environment. This approach considers both the user's preference and QoS attributes mentioned in the service level agreement (SLA). The work in [Thangaraj and Balasubra-

manie, 2021], introduces a hybrid meta-heuristic approach for the QoS-aware selection of services. This approach is based on both genetic algorithm and Tabu search for reducing the space search while respecting the QoS constraints.

In [Li et al., 2020], the authors propose a new approach to address the QoS-aware Web service composition problem. First, a novel method called Fuzzy Optimal Continuity Construction (FOCC) is proposed to represent fuzzy continuous neighbourhood relations between the services, which represents a pre-processing, to make the meta-heuristic algorithms effective in a discrete space. Then, the Chaos Harris Hawk Optimization (CHHO) algorithm is proposed to retrieve the best composition using Logistic Chaotic Single-Dimensional Perturbation (LCSDP) strategy and a meta-heuristic called Harris Hawk Optimization (HHO). The authors in [Dahan et al., 2019] improve the performance of the Artificial Bee Colony (ABC) algorithm by adopting the neighbouring node selection and swapping strategies.
In order to take into account interval-based QoS, the original crowding distance of NSGA-II is improved. In [Dahan, 2021], an agent-based ant colony optimization (MAACS) method is leveraged. They propose searching decomposition and real-time population re-disruption techniques to reduce the complexity of the cloud service composition.

The work by [Hosseinzadeh et al., 2020], handles a service composition model in cloud-edge computing through a hybrid method that combines the Artificial Neural Network (ANN) and the meta-heuristic Particle Swarm Optimization (Particle Swarm Optimization (PSO)) algorithm. Furthermore, to support the correctness and to improve the proposed ANN-PSO, a formal verification method is used. This latter one is based on a labelled transition system to check some critical Linear Temporal Logics (LTL) formulas.

For the cloud service composition, the authors in [Naseri and Navimipour, 2019] present a hybrid approach based on an agent-based method and PSO. The agent-based method is used to compose the cloud services by identifying the QoS parameters, and the PSO algorithm is employed to select the best-combined cloud services based on a fitness function.

In [Fekih et al., 2016], the authors use the skyline method to reduce the number of services candidates. Then, they propose a Valued Constraint Satisfaction Optimization approach (VCSOP) based on the PSO meta-heuristic, to model the Web service composition problem while considering the user's needs and context changes.

In [Dahan et al., 2021], the authors introduce an improved meta-heuristic, called Enhanced Flying Ant Colony Optimization (EFACO). It is proposed to restrict the flying process and a novel neighbouring selection process to overcome the execution time problem, meanwhile it reduces the quality of selection. Therefore, they leveraged a multi-pheromone method to increase the exploration by attributing a pheromone value for each QoS attribute. The work in [Barkat et al., 2021] tackles the Web service composition in the multi-cloud environment. Therefore, the authors leveraged an Intelligent Water Drops (IWD) algorithm based on the quality of service (QoS).

4. **Pareto dominance-based methods** They are leveraged to handle the increasing number of services over the Web. Moreover, they deal with the involvement of the user to introduce his preferences. Specifically, These approaches are proposed to solve multi-objective problems by seeking a set of optimal solutions.

   In [Alrifai et al., 2010], the authors propose an approach for the service selection and composition based on the dominance relationship. First, the search space is reduced by eliminating the nonskyline services of each abstract class. Second, the K-means algorithm is leveraged for hierarchical clustering to downsize the number of skylines. Finally, the composition space is explored using the clusters heads combination.
   The work by [Halfaoui et al., 2015], proposes a heuristic based on a new concept for local selection, called the Averaged-Fuzzy-Dominated-score (denoted AFDetS()). This approach is adopted to rank services based on QoS. In [Yu and Bouguettaya, 2013], the authors propose three algorithms for searching skyline compositions (denoted C-SKY): One Pass Algorithm (OPA), Dual Progressive algorithm (DPN), and Bottom-Up Algorithm (BUA). Based on the OPA algorithm, all possible compositions are generated by combining the skyline services (C-Skies). DPN algorithm sorts the skyline composite services via a fitness function where the QoS criteria are aggregated according to the sum. However, this approach assumes that the QoS of different services are independent of each other. In [Benouaret et al., 2011], the authors propose an approach based on the fuzzy dominance relationship through a new concept, called α-dominance. This concept is used for pruning the α-Pareto-dominated services and selecting the α-Pareto-dominant services. besides allowing users to control the size of the skylines through the modification of α degree, the selected Web services have a good compromise between QoS attributes. However, this approach does not consider the service composition.

In [Wang et al., 2016], the KD-tree (K-Dimensional tree) is adopted to find skyline Web services from clusters. In fact, each cluster regroups the services having the same functionalities. Since, the number of services in the clusters is large, finding the skyline services is handled by employing the KDS algorithm. Furthermore, lottery scheduling is leveraged to schedule and select services. The key idea is to distribute lottery tickets to services; services with higher quality will get more tickets and have more chances to be executed.

In [Serrai et al., 2016], the authors propose a hybrid Web service selection approach. First, the skyline method presented in [Alrifai et al., 2010] is employed to eliminate the dominated services and helps to reduce the search space. Secondly, Best-Worst Method (BWM) is adopted to assign weight to QoS criteria, since BWM is less complex, more consistent, and reliable than AHP. Finally, the VIKOR method (Multi-criteria Optimization and Compromise Solution) is leveraged to rank and sort the skyline services based on their pertinence relatively to the QoS weights.

In [Permadi and Santoso, 2018], skyline techniques are employed to address the service composition. This method consists of computing dynamic skylines using Sort and Limit Skyline algorithm (SALSA) to find the optimal solution that meets the users' preferences with respect to the budget constraints. This approach is proposed to overcome large search space problems. Nevertheless, the computational time still increases when the number of compositions is large.

In [Liang et al., 2019], the authors propose an improved skyline algorithm using the Bitmap method to handle the QoS-based Web service selection. This algorithm can reduce the dominance checks in regions and the candidate service set besides saving memory space.

3.4.3 Dynamic QoS-aware service composition

Web service composition is a challenging task. The concurrence between the service providers is the major obstacle. Furthermore, the time available for the composition process is limited, and the complexity of the composition problem is considered as NP-hard. This means that complexity and timeliness are also challenging. Therefore, the service selection and composition approach must be efficient. On the other hand, finding the service composition that respects all users' QoS requirements is a difficult task. Moreover, the dynamic nature of the Web service environment has an impact on the QoS of the ser-

vice and consequently on the QoS of the composition. Broadly speaking, the QoS vary over time and this represents another challenge to deal with in the composition process.

In the following, we focus on service selection and/or composition works that handle the uncertain QoS.

The work proposed in [Pei et al., 2007], extends the traditional notion of skylines [Börzsönyi et al., 2001] to probabilistic skylines to retrieve the most dominant uncertain object services having a degree of at least p. To compute the p-skyline, two algorithms are proposed based on three techniques: bounding, pruning, and refining. However, the noisy services have a high impact on the P-skyline [Pei et al., 2007]. Therefore, the p-dominant skyline concept is introduced in [Yu and Bouguettaya, 2010] to overcome the mentioned limitation. To compute the p-dominant skyline, the authors propose a dual pruning process through two stages algorithm that uses an extension of the R-tree [Guttman, 1984] called p-R-tree. This approach does not handle the composition process.

[Benouaret et al., 2012] is another research that introduces an improved skyline-based approach. It addresses the uncertain QoS through a probability distribution representation to select the most relevant candidate services. In this approach, the extensions pos-dominates and the nec-dominates skyline was presented to replace the traditional P-skyline. Furthermore, Optimizing the dominance computation is leveraged to overcome the high computational cost. The works presented in [Wen et al., 2014] also address the QoS uncertainty for the service selection using probabilistic dominance. In this context, the concept of dominance ability is introduced to compute the Top K dominant compositions. The proposed approach leverage two algorithms called Bounding Batch Computing (BBC) and Top-k Services Retrieving (TSR) to select top k dominating services. Furthermore, half independent selection of component services approach (HIS) is proposed to improve the accuracy of obtained top-k composite services. It is important to note that this work does not consider global constraints.

The work in [Seghir et al., 2019] proposes an Interval-based Multi-Objective Artificial Bee Colony (IM_ABC) approach to handle the QoS uncertainty in the service composition process. In fact, the composition problem is considered as an interval constrained multi-objective optimization, where a new uncertain constrained dominance relation is introduced to deal with the interval-valued objective functions and rank the candidate ser-

vices. Furthermore, controlling the diversity of the nondominated solutions is tackled by improving the original crowding distance of NSGA-II to a new interval distance definition.

The service composition in a multi-Cloud environment is addressed in [Haytamy and Omara, 2020] by a modified PSO with considering the uncertainty of QoS attributes. The work advanced in [Xu et al., 2018], integrated both fuzzy set theory (FST) and Genetic Algorithm (GA) to solve the uncertain QoS service composition. In fact, the proposed approach is called a Triangular Fuzzy Genetic (TGA) algorithm where it is used in the specification specification of the uncertain information of QoS properties with triangular fuzzy numbers. Additionally, it presents a feasible method for QoS normalization and propose a practicable method of defuzzification. The fitness function based on the Pareto dominance is employed in TGA for evaluating the criterion and selecting the solution that best meets the user's preferences. In [Mostafa and Zhang, 2015], modelling multi objective service compositions in a dynamic environment is handled with the Partially Observable Markov Decision Process (POMDP). To solve this issue, the Multi-Objective Reinforcement Learning (MORL) methods are employed through the algorithm called Baseline Multi-Objective Reinforcement Learning for Service Composition. In addition, two approaches were proposed to address the multiple policies multi-objective service composition. The first approach is based on user preferences, and the second approach is based on a convex-hull operator to find the Pareto optimal solutions. The experimental results have shown good effectiveness for solving the service composition in a dynamic environment. However, it suffers from high cost and complexity.

For solving QoS-aware service composition, the work in [Elsayed et al., 2017] proposes a hybrid approach that combines the GA and the Q-learning. Q-Learning algorithm is used to generate the initial population instead of the random generation. Furthermore, the genetic operators are leveraged for the population to find the optimal composition in terms of QoS.

The authors in [Tripathy et al., 2015], propose a representation of Web-Service Based Systems (SBS) through the service cluster graph. Additionally, a dynamic time-aware service selection is efficiently performed by employing Bellman ford's single-source shortest path graph algorithm. Indeed, dealing with dynamic QoS requirements is handled by employing a run-time adaptation scheme based on a service reselection approach.

[Hadjila et al., 2020] is a recent work that handles the QoS uncertainty in Web service

composition. The proposed approach adopts multiple fuzzy heuristics and global optimization. The optimization leverages a set of rules to prune the search space such as fuzzy dominance-based ranking, probabilistic skyline-based ranking, P-dominant skyline-based ranking, and average skyline-based ranking. In addition, for providing the Top K compositions, the authors leverage the constraint programming-based search.

A pioneer work is proposed in [Hwang et al., 2015], to handle the service selection with uncertain QoS. This work addresses the modelling of fluctuating QoS for both atomic and composite services. To identify the best services, the approach starts with global constraints decomposition into local constraints where the computed local thresholds are the fraction of the global constraint with respect to the average of QoS attributes of each class. Afterward, the composition is performed and evaluated according to a local objective function. The resulted solution is improved based on the simulated annealing method by replacing some selected services with others whenever the global constraints are violated.

In [Kim et al., 2016], the authors address an important problem in service selection where they claim that the outliers affect the service selection quality. In this context, the proposed framework deals with outliers and proposes global constraints' decomposition into local constraints. The local constraints are further used to eliminate the candidate services that violate these constraints. Furthermore, service selection is performed through a ranking score that maximizes the probability of satisfying a global QoS constraint of a composite service.

Table 3.3 summarizes the main approaches for service selection and/ or composition (i.e., the service composition may include the service selection). We specify if the researchers emphasize the static QoS or uncertain QoS. Furthermore, we indicate if the researchers concern Web services or cloud services.

Table 3.3: State of the art classification.

Approaches	**SS**	**CS**	**QoS**	**Static QoS**	**Uncertain QoS**	**Web service**	**Cloud service**
[Hwang et al., 2015]		✓	Response Time, Reliability, Fidelity		✓	✓	
[Wen et al., 2014]		✓		✓	✓		

[Hadjila et al., 2020]	✓			✓	✓
[Gabrel et al., 2014]	✓	Response Time, Throughput	✓		✓
[Liu et al., 2012]	✓	Execution duration, Reputation, Reliability, Cost,Availability	✓		✓
[Alrifai et al., 2012]	✓	Response Time, Throughput, Availability, Cost	✓		✓
[Fan et al., 2018]	✓	Cost	✓		✓
[Ghobaei-Arani and Souri, 2019]	✓	Response Time, Cost, Availability, Reliability	✓		✓
[Halfaoui et al., 2015]	✓		✓		✓
[Baranwal and Vidyarthi, 2016]	✓	Cloud QoS	✓		✓
[Wang et al., 2017]	✓	Cost, Throughput	✓		✓
[Purohit and Kumar, 2018]	✓	Response Time, Availability,Cost, Throughput, Reliability, Successability, Compliance, Best practice, Latency	✓	✓	
[Ouadah et al., 2018]	✓	Cost, Availability, Reliability	✓	✓	

[Serrai et al., 2018]	✓		Response Time, Reputation, Cost, Availability, Reliability	✓	✓	
[Youssef, 2020]	✓		Scalability, Cost, Sustainability, Usability, Interoperability, Maintainability, Response Time, Reliability	✓		✓
[Tiwari and Kumar, 2021]	✓		CPU performance, Disk performance, Disk I/O consistency, Memory performance Cost	✓		✓
[Zhao et al., 2012]		✓	Response Time, Cost, Reliability	✓	✓	
[Merzoug et al., 2014]		✓	Cost, Response Time, Fiability, Availability, Reputation	✓	✓	
[Bekkouche et al., 2017]		✓	Response Time, Cost, Availability, Fiability, Reputation	✓	✓	
[Li et al., 2018]		✓	Response Time, Cost, Availability, Reputation	✓	✓	

[Thangaraj and Balasubramanie, 2021]	✓	Availability, Response Time, Throughput, Interoperability	✓	✓	
[Li et al., 2020]	✓	Response Time, Availability, Throughput, Latency	✓	✓	
[Ranjan and Sahoo, 2020]	✓	Availability, Reliability, Cost, Delay	✓	✓	
[Dahan et al., 2019]	✓	Cost, Response Time, Throughput, Reliability	✓	✓	
[Dahan, 2021]	✓		✓		✓
[Hosseinzadeh et al., 2020]	✓	Availability, Response Time, Cost		✓	✓
[Naseri and Navimipour, 2019]	✓	Response Time, Cost, Reliability, Makespan, Resource count	✓		✓
[Dahan et al., 2021]	✓	Cost, Response Time, Throughput, Reliability	✓	✓	
[Yu and Bouguettaya, 2013]	✓	Response Time, Cost, Availability, Reliability	✓	✓	
[Benouaret et al., 2011]	✓		✓	✓	

[Wang et al., 2016]	✓		Reliability, Response Time, Cost	✓		✓
[Serrai et al., 2016]	✓		Response Time, Throughput, Reliability, Best Practice	✓		✓
[Permadi and Santoso, 2018]		✓	Delay, Bandwidth capacity		✓		✓
[Al-Faifi et al., 2019]		✓	Response Time, Cost, Reliability		✓	✓	
[Liang et al., 2019]	✓		Response time, Availability, Throughput, Success rate, Reliability, Compliance, Best practices, Latency, Documentation	✓		✓	
[Jatoth et al., 2019]	✓		Cost, Processing performance, Operational consistency, Disc storage performance, I/O Memory performance	✓			✓
[Benouaret et al., 2012]	✓				✓	✓	
[Seghir et al., 2019]		✓	Response Time, Cost, Throughput, Reliability		✓		✓
[Haytamy and Omara, 2020]		✓	Response Time, Cost, Throughput		✓	✓	

[Xu et al., 2018]	✓	Response Time, Reputation, Throughput, Latency, Reliability, Success rate	✓	✓
[Mostafa and Zhang, 2015]	✓	Response Time, Cost, Availability	✓	✓
[Tripathy et al., 2015]	✓	Response Time, Cost, Safety, Reliability	✓	✓
[Kim et al., 2016]	✓	Response Time, Availability, Reliability	✓	✓

3.5 Conclusion

In this chapter, we have introduced some basic concepts of web service composition and related works in this field according to the four categories. Subsequently, we have outlined the uncertain QoS-aware service composition, and we have reviewed some related works regarded to this context. In the next chapter, we will present our proposed approaches to deal with the service composition problem.

Part II

Contributions

Chapter 4

Uncertain QoS-based Web service composition approaches

4.1 Introduction

The service composition problem is known as a NP-hard problem [Moghaddam and Davis, 2014]. Therefore, it attracted an increasing interest by many researchers in both academy and industry.

In fact, we notice that a few researchers address all the WSC aspects. More specifically, we mention the consideration of the complex user requirements, the emergence of similar services over the Web, and the presence of uncertain QoS. Actually, the increasing number of concurrent service providers entails an exponential increase of the possible compositions.
Broadly speaking, to answer a user's request we need to examine m^n combinations (n is the number of required tasks, and m is the number of concurrent services per task). Even though the Quality of Service is the main characteristic used to distinguish between the concurrent Web services as well as to design the best composition, it represents another challenge for solving the service composition problem. The challenge lies in the technique that deals with the uncertainty of QoS, especially dealing with the different realizations of each QoS attribute.

In order to handle all mentioned challenges, we propose effective approaches for uncertain QoS-based Web service composition.

In this chapter, we present a problem formulation; then we suggest a scenario that shows the motivation behind the proposed solutions. Afterwards, we propose a global architecture of the Web Service Composition framework. We then present the details of our contributions to

solve WSC problem, while describing the main three algorithms. Finally, we summarize the contributions in the conclusion.

4.2 Problem formulation

To simplify the problem of the service composition with uncertain QoS, we leverage the notation shown in Table 4.1. We consider that the user requests some specific functionalities (e.g., finding a product or making a payment); each functionality represents a task. The entire workflow has *n* tasks CL_1, CL_2,...,CL_n, each task is performed by a service s_i. There exist m concurrent services that achieve the same task. All services are characterized by *r* QoS attributes. Each QoS criterion is represented by a sample of *l* realizations.

Table 4.1: Notations [Remaci et al., 2018]

Notations	Meaning
n	The number of abstract classes.
m	The number of services per classes.
r	The number of QoS attributes.
l	The number of service instances (i.e., the number of QoS realizations or the sample size).
$\mathrm{CL_1}, \mathrm{CL_2}, ..., \mathrm{CL_n}$	The set of abstract classes, each class contains atomic Web services with the same functionality and different QoS. The size of each $\mathrm{CL_j}$ is m.
S_i	Stands for the id of the selected service related to $\mathrm{CL_j}$.
$\mathrm{QoS_{piju}}$	The value of the $\mathrm{p^{th}}$ QoS attribute related to the $\mathrm{u^{th}}$ instance of the service $\mathrm{S_i} \in \mathrm{Cl_j}$.
b_1,b_2,...,b_r	The user's global constraints (i.e., the bounds that must be fulfilled by the QoS of the composition).
$\mathrm{w_1}, ..., \mathrm{w_r}$	The weight of the QoS criteria.
K	The size of the returned list (of compositions).

4.3 Motivation scenario

The challenge of providing a more sophisticated application is how to retrieve the most relevant services from a set of ssimilar ones, and based on these services how to select the optimal composition that satisfies both functional and nonfunctional requirements, especially in the uncertain environment. In order to clarify the above-mentioned challenges, we present

the example specified in [Zeyneb Yasmina et al., 2022].

Let us consider an e-health application used by a health centre. This application includes a module for authentication and access control, an annotation of medical images module, a disease diagnosis module, a module for obtaining multimodal medical images, a medical image sharing module, and a medical scheduling module (see Figure 4.1).

Firstly, the doctor can access the cloud storage service using authentication and access control services. Then, using the annotation service, the doctor can localize the region of interest and label the organs through the use of bounding boxes with different colors. If the lesion is clear, the doctor mentions the anomaly or the disease related to the images (e.g., inflammation, fibrillation, etc.). However, if the lesion is ambiguous or noisy, the doctor asks for images with different modalities such as ultrasound images, CT scan images, MRI images.
In parallel, the doctor shares the medical image with the physicians via the medical image sharing service. Finally, an appointment is fixed for the patient through the medical scheduling service [Zeyneb Yasmina et al., 2022].
Time is essential in the e-health application, since it usually is a life or death case. Therefore, this health centre requires that the total response time should not exceed 1.3 ms.

Let us assume that for each service there are two concurrent providers that offer similar functionalities. Additionally, due to the fluctuating environment the QoS of each service change over time. The QoS realizations of each service are summarized Table 4.2.

In this case, we have to handle four aspects:

- We must to represent the user's requirements (such as the global constraints, the number of tasks, the control flow).
- We must to find the best metric for QoS aggregation since the QoS value is nondeterministic.
- We must to reduce the space search for the concurrent service and the possible compositions.
- In order to retrieve the most stable services, and select the best composition, we must to manage the nondeterministic QoS.

The contributions of this thesis are summarized as follows:

1. We select the Top K Web services of each abstract class of the user's workflows by leveraging a given set of heuristics.

2. We use the global search approaches to eliminate the compositions that do not fulfill the global constraints, then we find the Top K compositions.

3. We conduct a series of experiments to evaluate our framework on real and synthetic datasets. Moreover, we consider different workflows.

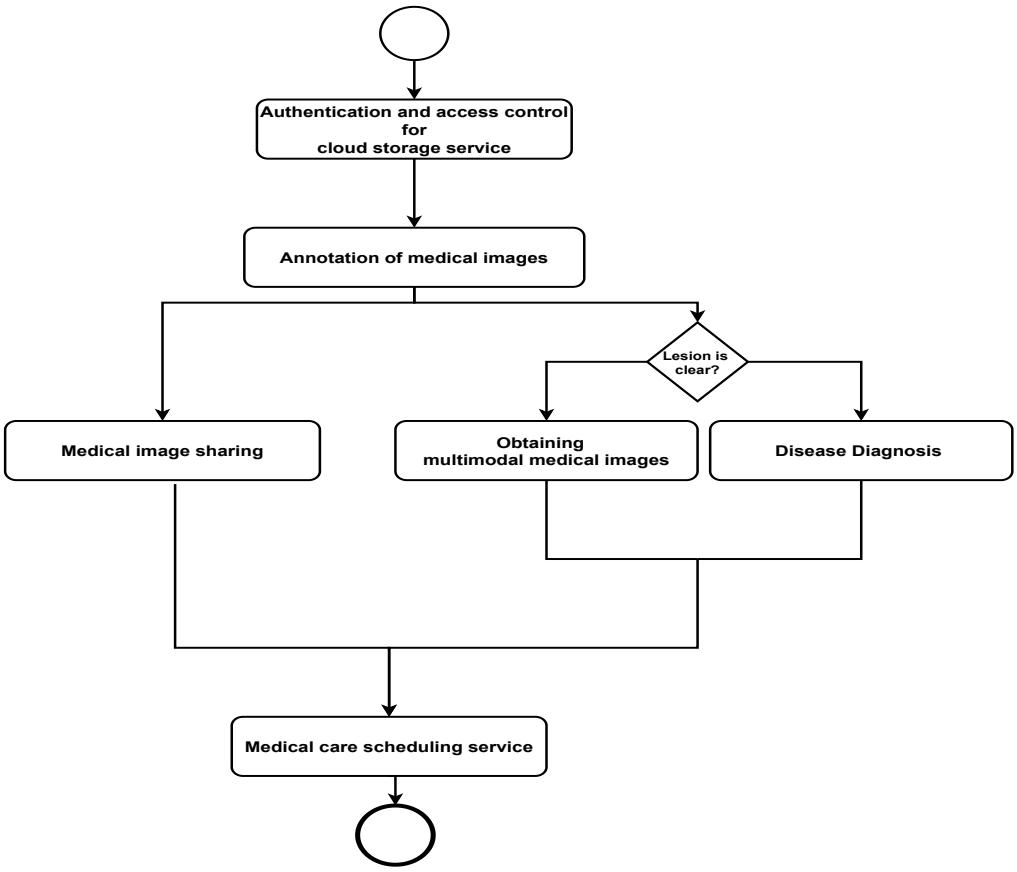

Figure 4.1: e-health workflow.

4.4 Composition framework

Figure 4.2, illustrates the framework adopted for the Web service composition process. This framework assists users to find the best service composition from a set of concurrent services, in order to achieve their requirements. It consists of three main modules:

1. **Class management module:** To deal with the increasing number of services exposed over the Web we organize them into specific groups called abstract classes. Therefore, our framework includes the class management module that ensures the assignment of each service to an abstract class and guarantees recurrent updates to add new services and eliminate the unavailable components.

Table 4.2: Normalized QoS realization (Response time).

	Authentication and access control		Annotation of medical image		Medical diagnosis		Obtaining multimodal medical images		Medical image sharing		Medical scheduling	
	S_{11}	S_{12}	S_{21}	S_{22}	S_{31}	S_{32}	S_{41}	S_{42}	S_{51}	S_{52}	S_{61}	S_{62}
t_1	0.2	0.2	0.1	0.8	0.5	0.2	0.2	0.4	0.3	0.6	0.1	0.3
t_2	0.5	0.8	0.1	0.9	0.6	0.4	0.7	0.9	0.6	0.1	0.4	0.5

2. **QoS management and integration module:** The QoS is usually represented as a set of quantitative values since the QoS is nondeterministic. These values are usually collected from the Web service providers and social networks, and this module is stored in some data warehouse.

3. **Selection module:** In fact, this module is composed of two main sub-modules: atomic service selection and composition selection modules. The first one is leveraged to select the best services from each abstract class based on the uncertain QoS, where the second module allows for building a composition from the returned set of components from the previous sub-module. Afterward, we retrieve the optimal composition that better meets the user's requirements.

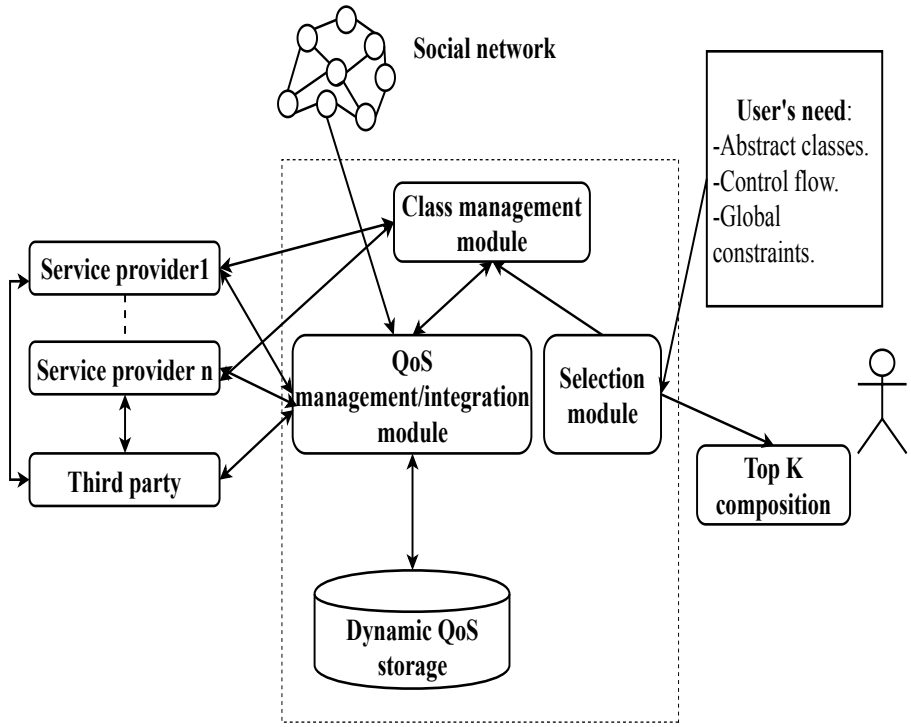

Figure 4.2: Service composition framework.

4.5 Probabilistic dominance approach

In this section, we introduce the first proposed approach included in the selection module. In fact, this module incorporates two algorithms: dominance service ranking algorithm which is based on probabilistic dominance, and the backtracking search algorithms that address the WSC problem.

4.5.1 Probabilistic dominance relationship

This concept is an extension of deterministic dominance. Meanwhile, it is leveraged in the presence of uncertainty. Selecting one or Top-K compositions is a time-consuming process, since it requires exploring and evaluating m^n compositions. Therefore, we proposed to achieve a pre-process to reduce the service search space. Thus, we sort and rank services of each abstract class, by comparing the services with respect to the probabilistic dominance. In fact, this relation measures the average fraction of the instances of S_i that are weakly dominated by an instance of $S_{i'}$. The weak dominance relationship is defined between two QoS realizations vectors.

Let X and Y be two vectors of R^r:

$X >> Y$ iff for each dimension $q \in 1, .., r, X(q) \geq Y(q)$.

$$prob - dom(S_i^{'}, S_i) = \frac{1}{l} * \sum\nolimits_{u'=1}^{l} individual - prob - dom(u^{'}, i^{'}, i) \tag{4.1}$$

Where

$$individual - prob - dom(u^{'}, i^{'}, i) = \\ (|(QoS_{1iju}, .., QoS_{riju})/(QoS_{1i'ju'}, .., QoS_{ri'ju'}) >> (QoS_{1iju}, .., QoS_{riju})|l), u \in 1, .., l. \tag{4.2}$$

Where the $>>$ denotes "better than".

4.5.2 Dominance Service Ranking algorithm

Dominance Service Ranking (DSR) algorithm aims to rank the services of the same abstract class depending on the computed score. This score is incremented during each pairwise comparison through the probabilistic dominance test.

The pseudo-code of Algorithm 1 is given below:

This Algorithm requires only the abstract classes that contain the services as an input.

1. In lines 1-3, we initialize the ranked Class $RankedCl_j$ (with an empty structure).
2. In lines 5-7, we initialize the ranking score of each service of the current class j.
3. In lines 8 up to 11, we perform a pairwise comparison between each service (S_i,$S_{i'}$) of the same class j, through the use of the probabilistic dominance formula 4.1.

Algorithm 1 DominanceServiceRanking

Input: $Cl_1, ..., Cl_n$.
Output: $RankedCl_1, ..., RankedCl_n$
Begin

```
1:  for j=1 to n do
2:      RankedCl_j =<>;
3:  end for
4:  for j=1 to n do begin
5:      for y=1 to m do
6:          score(y) = 0;
7:      end for
8:      for i = 1 to m do begin
9:          for i' = 1 to m do begin
10:             if (i! = i') then begin
11:                 if (prob-dom( S_i, S_i') ≥ prob-dom( S_i',S_i)) then begin
12:                     score(i) = score(i) + 1;
13:         end for
14:     end for
15:     RankedCl_j=decreasing-sort(Cl_j)
16: end for
17: return < RankedCl_1, ..., RankedCl_n >
18: end
```

4. In line 12, we update the score of S_i if probabilistic dominance between the services (S_i,$S_{i'}$) is better than the probabilistic dominance between ($S_{i'}$,S_i).

5. We sort the elements of $RankedCl_j$ according to the scores updated in line 15.

6. We return the ranked classes in line 17.

By employing algorithm 1 for ranking the services of the class **Obtaining multimodal medical images service (see Table 4.2)**, we can notice than s_{41} is better ranked that s_{42} since :
$prob - dom$ (s_{41}, s_{42}) = $\frac{3}{4}$ (since 0.2 >> 0.4, 0.2 >> 0.4 and 0.7 >> 0.9).
$prob - dom$ (s_{42}, s_{41}) = $\frac{1}{4}$ since 0.4 >> 0.7).

4.5.3 Backtracking search algorithm

This algorithm is one of the main algorithms for solving constraint satisfaction problems. This algorithm browses all alternatives in the search space through two recursive phases: moving forward and going back (termed back-track).

In order to handle the service composition problem, we propose this algorithm to perform an exhaustive search (i.e., for generating and browsing the possible compositions). At the end of the process, the optimal composition or Top K compositions are selected.

Choosing a composition from a set of candidate compositions is based on the objective function (utility function), used for comparing the compositions. This function attempts to minimize the negative QoS values and maximize the positive QoS ones. Furthermore, the result is a normalized score in the range [0,1]; the closer the score to 1, the better the composition. This means that the composition's objective value depends on the QoS values.

The traditional objective function (see Equation 4.3), represents a weighted sum of all QoS values is widely applied [Fethallah et al., 2012], [Merzoug et al., 2014]. The latter is appropriate if the QoS is static, however, the QoS is nondeterministic. Therefore, we improved the objective function (see Equation 4.6) to consider all QoS realizations by leveraging the median value as a representative value.

The composition that violates the user's requirement is unfeasible, even if it has a high score. Therefore, we integrate another objective function called percentage of satisfied global constraints (p.s.g.c) in the backtracking search algorithm, according to inequalities 4.12, and 4.13. The p.s.g.c is adopted by [Remaci et al., 2018], [Hadjila et al., 2020], and it is defined as the fraction of global constraints with respect to the median QoS that is preserved by the composition.

$$U(C) = \sum_{p=1}^{r} w_p * \frac{(Q_p^{'}(C) - Qmin^{'}(p))}{(Qmax^{'}(p) - Qmin^{'}(p))} \tag{4.3}$$

$$U_{Positive}(C) = \sum_{p=1}^{r} w_p * \frac{(MedianQ_p^{'}(C) - Qmin^{'}(p))}{(Qmax^{'}(p) - Qmin^{'}(p))} \tag{4.4}$$

$$U_{Negative}(C) = \sum_{p=1}^{r} w_p * \frac{(Qmax^{'}(p) - MedianQ_p^{'}(C))}{(Qmax^{'}(p) - Qmin^{'}(p))} \tag{4.5}$$

$$U(C) = U_{Positive}(C) + U_{negative}(C) \tag{4.6}$$

$$Qmin^{'}(p) = \sum_{j=1}^{n} Qmin(j,p) \tag{4.7}$$

$Qmin^{'}(p)$ is the minimal aggregated QoS of the p^{th} attribute for all possible compositions.

$$Qmax^{'}(p) = \sum_{j=1}^{n} Qmax(j,p) \tag{4.8}$$

$Qmax^{'}(p)$ is the maximal aggregated QoS of the p^{th} attribute for all possible compositions. Equations $Qmin(j,p), Qmax(j,p)$ are defined as follows :

$$Qmin(j,p) = Min_{u\in\{1,...,l\},s_i\in CL_j}(QoS_{piju}) \tag{4.9}$$

$Qmin(j,p)$ is the minimal QoS value of the p^{th} attribute of all services related to the i^{th} task.

$$Qmax(j,p) = Max_{u\in\{1,...,l\},s_i\in CL_j}(QoS_{piju}) \tag{4.10}$$

$Qmax(j,p)$ is the maximal QoS value of the p^{th} attribute of all services related to the i^{th} task.

$$MedianQ_p^{'}(C) = \sum_{j=1}^{n} Median_{u\in\{1,...,l\}} QoS_{ps_jju} \tag{4.11}$$

If the criterion *p* is positive, the global constraint related to p is defined as:

$$\mathrm{MedianQ'_p(C)} \geq \mathrm{b_p}; \forall \mathrm{p} \in \{1, ..., \mathrm{l}\} \tag{4.12}$$

If the criterion *p* is negative, the global constraint related to p is defined as:

$$\mathrm{MedianQ'_p(C)} \leq \mathrm{b_p}; \forall \mathrm{p} \in \{1, ..., \mathrm{l}\} \tag{4.13}$$

Where $\mathrm{MedianQ_p^{'}(C)}$ represents the aggregated p^{th} QoS value of each component in the composition *C* with respect to the median QoS value, the aggregation changes according to the QoS attributes, see Table 3.2.

The main advantage is that the evaluation function considers both the uncertain QoS and the user's requirements, to retrieve the Top-K composition that maximizes, the chance of satisfying the global constraints (see inequalities 4.12, 4.13), and maximize the function $U(.)$ (see Equation 4.6). Broadly speaking, a composition C is ranked above the composition $C^{'}$ if the degree of satisfying the global constraints of C with respect to Equations 4.12, and 4.13 is higher than the degree of $C^{'}$. If the degrees of C and $C^{'}$ tie, then we leverage the Equation 4.6 to rank these compositions.We can notice that both the evaluation functions depend on the median value.
The motivation behind choosing this value is the uncertain QoS that needs to be considered in the composition process.

In general, the distribution probability that generates QoS data is usually unknown, consequently, we can notice the existence of some extreme values called outliers that perturb the selection and ranking of compositions. Therefore, we leverage the median value as a representative value since it is not sensitive to variation or extreme values.

Algorithm 2 BacktrackingSearch

Input: $RankedCl_1, ..., RankedCl_n$,$b_1, b_2, ..b_r$: global constraints, k: size of the results set; t: the minimum

Output: TopKCompositions **Begin**

1: Top-K Compositions=<> // the result is initially empty.
2: **for** i=1 to k^n **do**
3: $c = GetNextComposition(RankedCl_1, \ldots, RankedCl_n)$;
4: $degree = 1/r * \sum_{p=1}^{r} CC(p, c)$
5: **if** $(degree \geq t)$ **then**
6: **if** better (c, Top-K Compositions) **then**
7: Update (c, Top-K Compositions)
8: **end for**
9: **return** (Top-K Compositions)
10: **end**

The explanation of algorithm 2 is given as follows:
The inputs of this algorithm are the ranked classes obtained from algorithm 1. In order to reduce the search space, we only retain Top-k services from each abstract class. Thus, we can reduce the possible composition from m^n to k^n.

1. In line 2, we browse all the possible compositions.
2. In line 3, we get the current composition C.
3. In line 4, we compute the fraction of satisfied global constrained.
4. In line 5, we check that the fraction of the preserved global constraints is above the threshold.
5. In line 6, we compare C with the existing "Top-K Composition" elements through the use of Equations 4.12, 4.13, and 4.6 .
6. In line 7, we update the result Top-K Compositions if C is better than an existing composition.
7. We return the final result in line 9.

4.6 Majority judgment and Constraints Programming approach (MJ-CP)

In this heuristic, the selection module address the Top K service compositions problem based on two main algorithms: Majority Service Ranking and constraint programming algorithm (MSR-CP).

Practically, the dataset is introduced as input, the algorithm 3 is invoked to sort the services of the abstract classes according to a specific score. Following that, we invoke the second algorithm, called Constraint Programming (CP), where we retain only the Top K services having the highest scores from the ranked services. This algorithm allows for the selection of the Top K compositions by considering the user's requirements, and the fact that the QoS is uncertain. A general overview of the proposed selection module is described in Figure 4.3.

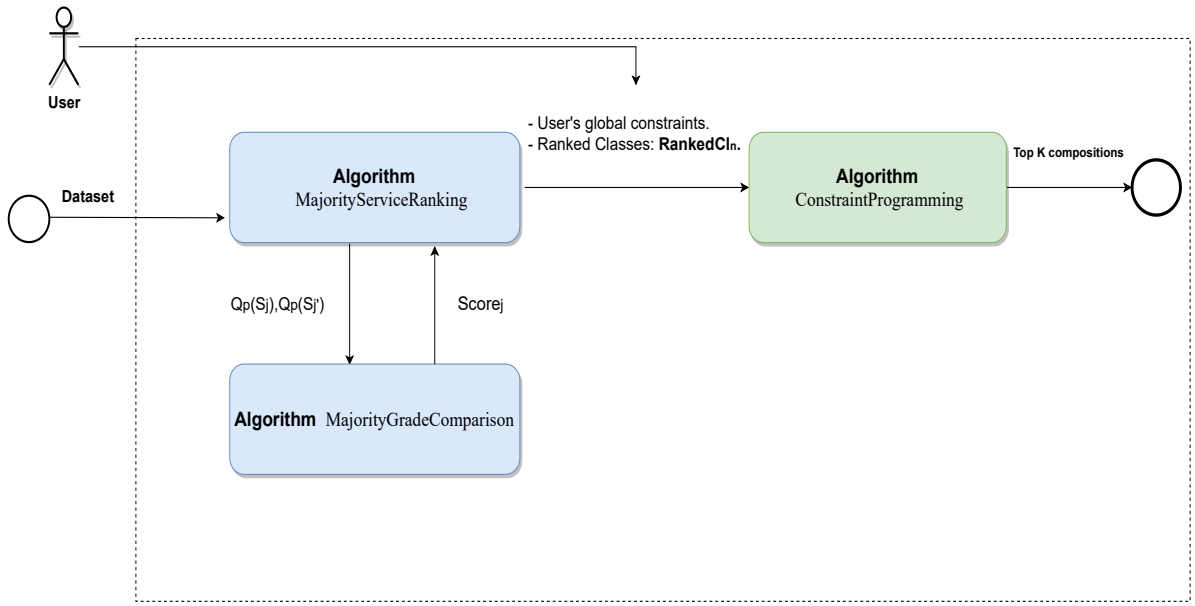

Figure 4.3: Selection module.

4.6.1 Majority judgment heuristic

The majority judgment is a recent voting theory proposed in 2007 by Balinski and Laraki in [Balinski and Laraki, 2007]. This procedure is proposed for election allowing all voters to evaluate each candidate, compared to other voting methods where the voter might only choose one candidate. In this method, the voter provides much more valuable information than in the traditional methods.

In this thesis, we were inspired by this voting theory. We proposed a heuristic for ranking the services with considering the QoS fluctuation by employing the Majority judgment procedure.

Actually, this procedure is based on two main concepts: the majority-grade, the tie-breaking rule.

1. **Majority-grade:** It represents the median value. Nevertheless, it imposes a specific rule: the majority-grade is the middle of the list if the number of instances is odd, and it is equal to the lower of the two middle grades if the number of the list is even then. According to [Balinski and Laraki, 2007], The choice of the smallest grade of the middle-interval when the number of judges is even is the logical consequence of a principle of consensus.

2. **Tie-breaking rule:** It is applied iteratively when two or more service's majority-grades are equal: the grades are dropped from each service; then the new grade is computed by repeating the same process.

The majority judgment heuristic is leveraged for ranking the services and outlined in algorithm 4. As well, algorithm 3 is a part of a program for service ranking. This algorithm takes as inputs the two vectors related to the services to be compared (i.e., services H and G), each vector contains l QoS realizations of the same QoS attribute, where the output can be either 0, 0.5, or 1. This value is used in algorithm 4 to update the score of the winning service.

Algorithm 3 MajorityGradeComparison

Input: $H =< h_1, .., h_l >, G =< g_1, .., g_l >$.
Output: $Score \in \{0, 0.5, 1\}$.
Begin

i=L;
while i >0 **do**
  if (odd(i)) **then**
    $middle = i \div 2 + 1$;
    if ($i == 1$ **and** $h_i == g_i$) **then return** 0.5;
  else if (even (i)) **then**
    $middle = i \div 2$;
  if ($h_{middle} \geq g_{middle}$ **and** $H.type == positive$) **then return** 1;
  else if ($h_{middle} \neq g_{middle}$ **and** $H.type == negative$) **then return** 1;
  else if ($h_{middle} == g_{middle}$) **then return** 0;
  else if ($h_{middle}! = g_{middle}$) **then**
    remove h_{middle} and g_{middle} from both H and G (respectively);
    $i - -$;
end while
end

Algorithm 4 MajorityServiceRanking

Input: $< CL_1, ..., CL_n >$,
Output: $RankedCl_1, ..., RankedCl_n$
Begin

1: **for** j=1 to n **do**
2: $RankedCl_j =<>$;
3: **end for**
4: **for** j=1 to n **do**
5: **for** i=1 to m **do**
6: score(y)=0;
7: **end for**
8: **for** i=1 to m **do**
9: **for** $i^{'} = 1$ to m **do**
10: **if** $i \neq i^{'}$ **then**
11: **for** p=1 to r **do**
12: $Score(i) = score(i) + 1/r * MajorityGradeComparison(Qp(S_i), Qp(S_{i'}))$;
13: **end for**
14: **end for**
15: **end for**
16: RankedCl$_j$=decreasing-sort (Cl$_j$)
17: **end for**
18: **return** $< RankedCl_1, ..., RankedCl_n >$
19: **end**

The internal process of algorithm 4 is presented in follows:

1. In lines 1-3, we assign an empty arrangement to the ranked class $RankedCl_j$.
2. In lines 5-7, we set the score of all services of class j to 0.
3. In lines 8-11, we invoke algorithm 3 to compare each pair of services ($s_i, s_i^{'}$) of the current class j
4. In line 12, we compute the score of s_i.
5. In line 16 we rank the elements of $RankedCl_i$ according to the scores updated in 12.
6. We return the ranked classes in line 18.

Example

$H =< 14, 15, \mathbf{21}, 23, 24 >$ and $G =< 11, 17, \mathbf{19}, 21, 22 >$, since $majorityGrade_H = median(H) = 21$, and $majorityGrade_G = median(G) = 19$, then h is ranked above g (because $21 > 19$); additionally, the algorithm 3 returns 1.

Example

$H =< 17, \mathbf{18}, 21, 21 >$ and $G =< 13, \mathbf{18}, 22, 23 >$. Since the median interval of h is [18,21]

and the median interval of G is [18,22], then the majority grade of both series is 18.
To resolve the issue, we delete the majority grade (i.e., 18) from H and G, and we repeat the same process. Thus, $H =< 16, \mathbf{21}, 21 >$ and $G =< 13, \mathbf{22}, 23 >$, and $majorityGrade_H = median(H) = \mathbf{21}$, and $majorityGrade_G = median(G) = \mathbf{22}$.

Based on this information, G will be ranked above H (because $22 > 21$); in addition, the algorithm 3 returns 0.

4.6.2 Constraints Programming

The real-life problems (such as scheduling, networks, data mining...) are usually considered as constraint satisfaction problems CSP. Considering this, the constraint programming (CP) is used for modelling and solving CSP. Constraint Programming (CP) has been defined as a technology of software used to describe and solve combinatorial problems [Apt, 2003].

CP is based on four concepts: the decision variables; the inputs which are a finite domain of values, a set of constraints; and the objective function. For example, the price of the product is the decision variable, and the constraint might be expressed as searching for the product where the cost is less than 100 dollars.

In CP, we can search for one or several feasible solutions. However, the exhaustive search is time-consuming. Therefore, we can use this feature for retrieving the infeasible solutions.

For solving the service composition problem, we can notice that even when reducing the number of services through the use of ranking algorithms from m service per class to k services, the number of possible compositions is constantly high (with k^n compositions).
Therefore, we propose an effective algorithm based on the CP concept, which allows reducing the search space by eliminating partial compositions that violate the global constraints.

In this approach, we notice that it will be useful to reduce the search space by eliminating partial compositions that violate the user's constraints. This means that during the composition process, we combine the services one by one, we evaluate the feasibility of the partial composition, and we remove this partial composition if it violates the global constraints.
The evaluation is performed through Inequalities 4.12, and 4.12 characterized by one sample of QoS realization designated by the median value. Besides that, we address the QoS fluctuations problem by selecting the Top K compositions by employing the Global Quality Conformance as the objective function. Furthermore, we tackle in this algorithm four composition models: sequential, loop, parallel, and choice using the aggregation functions presented in Table 3.2.

4.6.2.1 Global QoS Conformance (GQC)

The concept of Global QoS Conformance (GQC) is proposed in [Hwang et al., 2015], for computing the probability of respecting the global constraints by the composite service (see Equation 4.14). The reason for using this function is to search a composition C that maximizes GQC, consequently maximizing the ratio of respecting the users' constraints. Indeed, this criterion is widely adopted in the literature [Hwang et al., 2015, Kim et al., 2016, Hadjila et al., 2020].

$$\mathrm{GQC}((\mathrm{S_{ij}},...,\mathrm{S_{in}}),(\mathrm{b_1},...,\mathrm{b_r})) = \prod_{\mathrm{i=1}}^{\mathrm{r}} \mathrm{Pr}((\mathrm{S_{i1}},...,\mathrm{S_{in}}),\mathrm{b_i}) \tag{4.14}$$

$$\begin{aligned}\mathrm{Pr}((\mathrm{S_{i1}},...,\mathrm{S_{in}}),\mathrm{b_p}) =& 1/\mathrm{l^n} * \textstyle\sum_{\mathrm{u_1=1}}^{\mathrm{l}} ... \sum_{\mathrm{u_n=1}}^{\mathrm{l}} \\ & \mathrm{Step}(\mathrm{Aggregate}(\mathrm{QoS_{p\,s_{i1u_1}}},...,\mathrm{QoS_{p\,s_{inu_n}}}),\mathrm{b_p})\end{aligned} \tag{4.15}$$

Step is defined as:

$\mathrm{Step}(\mathrm{Aggregate}(\mathrm{QoS_{p\,s_{1nu_1}}},...,\mathrm{QoS_{p\,s_{inu_n}}}),\mathrm{b_p}) =$

$$\begin{cases} 1 \text{ if } \mathrm{Aggregate}(\mathrm{QoS_{p\,s_{i1u_1}}},...,\mathrm{QoS_{p\,s_{inu_n}}}) \geq \mathrm{b_p} \\ \text{and the criterion } \boldsymbol{p} \text{ is positive} \\ 1 \text{ if } \mathrm{Aggregate}(\mathrm{QoS_{p\,s_{i1u_1}}},...,\mathrm{QoS_{p\,s_{inu_n}}}) \leq \mathrm{b_p} \\ \text{and the criterion } \boldsymbol{p} \text{ is negative} \\ 0\ otherwise. \end{cases} \tag{4.16}$$

We adopt the product of the r probabilities of Equation 4.14, to address the satisfaction of r global constraints as a whole. Each individual probability $Pr((s_{w_1},...,s_{w_n}),b_i)$ measures the chance that the aggregated Quality of Service (QoS_i) of the composition $C = (s_{w_1},...,s_{w_n})$ is greater than or equal to the user's bound b_i (in the case where QoS_i is a positive criterion).

In the context of CP algorithm, if the GQC of C is higher than that of $C^{'}$ with respect to Equation 4.14, then the composition C is better ranked then the composition $C^{'}$. Nevertheless, if the composition C ties with $C^{'}$, we rank them according to the objective function (see Equation 4.6), the higher the score of $U(.)$, the better the rank.

The computation of GQC, and Equation 4.6 is explained through the previously mentioned Example (see Table 4.2, Figure 4.1):

We have the following QoS realizations (i.e., response time):

$s_{11} < 0.2, 0.5 >$, $s_{12} < 0.2, 0.8 >$, $s_{21} < 0.1, 0.1 >$, $s_{31} < 0.5, 0.6 >$, $s_{41} < 0.2, 0.7 >$, $s_{51} < 0.3, 0.6 >$, $s_{61} < 0.1, 0.4 >$.

By evaluating the composition $C1 =< s_{11}, s_{21}, s_{31}, s_{41}, s_{51}, s_{61} >$, we obtain:

1. $MedianQ'_1(C1 =< s_{11}, s_{21}, s_{31}, s_{41}, s_{51}, s_{61} >) = 0.35+0.1+Max(0.55, Max(0.45, 0.45)+ 0.25 = 1.25 \leq 1.3$ (the composition C1 is feasible).

2. $GQC(C1) = \frac{1}{2^6} * 33 = 0.52$ (we have 33 feasible combinations among 64 possible cases, see Table A.1 for more details).
   If $Qmin(j, 1) = 0$ and $Qmax(j, 1) = 1$ whare $j \in \{1, ..., n\}$, then

3. $U(C1) = \frac{1.25-0}{(6-0)} = 0.21$.

   However, if we consider $C2 =< s_{12}, s_{21}, s_{31}, s_{41}, s_{51}, s_{61} >$, then

4. $MedianQ'_1(C2 =< s_{12}, s_{21}, s_{31}, s_{41}, s_{51}, s_{61} >) = 0.50+0.1+Max(0.55, Max(0.45, 0.45)+ 0.25 = 1.4 \geq 1.3$ (The composition $C2$ is not feasible, because it violates the global constraint).

5. $GQC(C2) = 0.23$.

6. $U(C2) = \frac{1.4-0}{(6-0)} = 0.23$.

The overview of CP is summarized in algorithm 5.

Algorithm 5 ConstraintProgramming

Input: $C, TopKC, RankedCl_1, ..., RankedCl_n$
$,b_1, ..., b_r$.
Output: $TopKC$:the liste of the best K compositions in terms of GQC;
Begin

```
1:  if (Size(C) == n) then
2:      Update1 (C, TopKC)
3:  else
4:      j=GetFirstunassignedClass(C)
5:      for i=1 to k do
6:          Update2 (C, RankedCl_j, i)
7:          if (GQC(C, b_1, ..., b_r) == True) then
8:              TopKC= CP(C, TopKC, Cl_1, ..., RankedCl_n, b_1, ..., b_r)
9:      end for
10:     if (j==1) then return TopKC;
11: end
```

This pseudo-code is explained as follows:

1. Lines 1-2 : If C is fully assigned and already feasible, then we update the Top KC and return to the previous *CP* call (in which we handle the next service of the previous class i.e., the class with ID n-1).

2. Lines 4-13 : In this case, the assignment of C is only partial (i.e., C is empty or contains less than or equal to *n-1* services).

3. In line 5, we get the first class of C which is unassigned (its rank is denoted as j). Then, we explore the first k services of $RankedCl_j$.

4. Following this, we assign the j^{th} component of C with the service i (See line 7). If the global constraints of C are preserved, we call *CP* to process the next class.

5. If all services of the first class are processed and consequently all compositions are examined, then we return $TopKC$ (see line 13).

Update1(): This function is called for updating $TopKC$ list by inserting the current composition C if it contains less than K elements, or the GQC of C is better than the existing solutions, then the worst solution is removed.

Update2(): This function is employed to attribute the i^{th} service of $RankedCl_j$ to the j^{th} component of C. To explain the concept of the partial elimination of CP, we continue with the example shown in the motivation scenario (see Figure 4.1, and Table 4.2).

Let us consider the lower value of QoS realization as median value (according to majority judgment concept), thus, the evaluation of the composition $C =< s_{11}, s_{22}, s_{32}, s_{41}, s_{51}, s_{61} >$, is performed as follows:

1. $MedianQ'_{Responsetime}(C =< s_{11} >) = 0.2 \leq 1.3$ (GC1 is still preserved).
   If C1 is updated with s_{22} (i.e, $C =< s_{11}, s_{22} >$), then

2. $MedianQ'_{Responsetime}(C =< s_{11}, s_{22} >) = 0.2 + 0.8 = 1 \leq 1.3$ (GC1 is still preserved).
   If C1 is updated with s_{31} (i.e, $C =< s_{11}, s_{22}, s_{31} >$), then

3. $MedianQ'_{Responsetime}(C =< s_{11}, s_2, s_{31} >) = 0.2 + 0.8 + Max(0.5, 0.2) = 1.5 \geq 1.3$
   (GC1 is violated).

Consequently, it will be useless to process the following classes (i.e., CL_5 and CL_6), since GC1 is violated. Furthermore, if we have another QoS (e.g., reliability) we can skip the composition process.

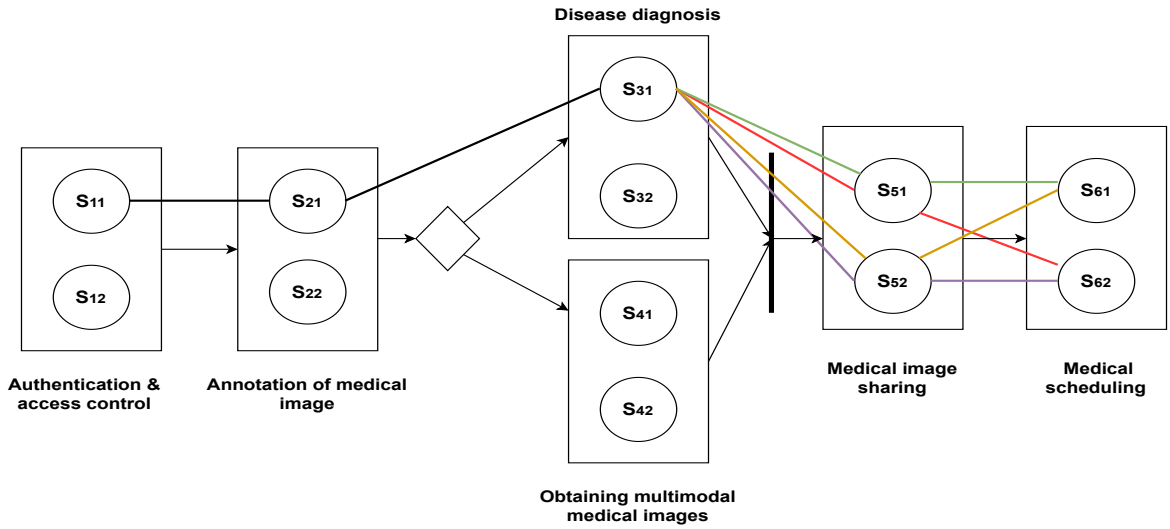

Figure 4.4: Service composition pruning.

By performing this pruning method, we can avoid k^{n-p} possible compositions (p is the number of assigned classes); in our example, we prune four compositions (colored compositions in Figure 4.4):

<$s_{11}, s_{22}, s_{31}, s_{51}, s_{61}$>, <$s_{11}, s_{22}, s_{31}, s_{51}, s_{62}$>, <$s_{11}, s_{22}, s_{31}, s_{52}, s_{61}$>, and <$s_{11}, s_{22}, s_{31}, s_{52}, s_{62}$>.

4.7 Grey Wolf-Based Composition (GWC) approach

Generally, the service composition problem is considered as a Multi-Criteria Decision-Making (MCDM) problem. Since we have to retrieve the optimal services from a set of alternatives, then we have to select the best composition based on multiple criteria (or QoS attributes).

There are several MCDM methods to solve such a problem. Meanwhile, the environment fluctuations represent another real-world problem that involves critical troubles on the services applications exposed over the Web. More specifically, we mention drawbacks caused by the imprecise and vague information of QoS, especially during the service composition process.

Fuzzy set (FS) theory was introduced by [Zadeh, 1996] and used as a key method to solve MCDM problems [Yager, 1977]. Therefore, we propose a new framework based on the Cross-Entropy of the Hesitant Fuzzy Set, which is a recent extension of FS. In addition, we introduce the approach called Grey Wolf-Based Composition (GWC) for selecting the Top K compositions.

4.7.1 Hesitant Fuzzy Set (HFS)

Fuzzy sets theory attracts larger attention by researchers, since it is mainly proposed for handling several real-world problems. Dealing with the uncertainty of the information is always an interesting research field in the Fuzzy sets theory.

[Torra and Narukawa, 2009] proposed an extension of the fuzzy sets, called Hesitant fuzzy set, to deal with the limitation of the habitual Fuzzy sets extensions regarding the uncertain information. In fact, Hesitant Fuzzy Set (HFS) represents a set of possible values in the range [0,1] instead of one membership degree. Accordingly, we can adopt the hesitant fuzzy set to represent the QoS realizations of Web services.

Let X be a fixed set, a HFS on X defined in terms of a function that is when applied to X returns a subset of [0,1].

The HFS is represented mathematically by [Xia and Xu, 2011] as:

$$\mathrm{A} = \{< \mathrm{x}, \mathrm{h_A}(\mathrm{x}) > |\mathrm{x} \in \mathrm{X}\} \tag{4.17}$$

The range of the set $\mathrm{h_A(x)}$ is [0,1], denoting the possible membership degrees of the element $\mathrm{x} \in \mathrm{X}$ with respect to the set A.

According to [Xia and Xu, 2011], $\mathrm{h_A(x)}$ is called a Hesitant Fuzzy Element (HFE) and θ the set of all HFEs.

In following, we model the service s_{11} of the class **Authentication and access control** with HFS:

- An abstract class denoted as $\mathrm{CL_1}$ (**Authentication and access control**).
- One QoS attribute (i.e., response time).
- The QoS realizations of the service s_{11} are given as follows :
  $\mathrm{Q} = \mathrm{q_1(i.e.,\ response\ time)}$,
  $\mathrm{A_{s_{11}}} = \{< \mathrm{q_1}\{0.2, 0.5\}\}$.
  $\mathrm{h_{s_{11}}(q_1)} = \{0.2, 0.5\}$ is the HFE.

In the following, we introduce some well-known hesitant fuzzy sets basics [Torra, 2010]:

1. **Empty set:** $\mathrm{h(x)} = \{0\}$ for all x in X.
2. **Full set:** $\mathrm{h(x)} = \{1\}$ for all x in X.

3. **Complete ignorance** for a $x \in X$ (all is possible): $h(x) = [0, 1]$.

4. **Nonsense set:** $h = \varnothing^*$.

4.7.1.1 QoS Normalization

The service composition process deals with several QoS; each one has different measurement units and magnitudes [Polska et al., 2021]. Therefore, transforming the QoS to an uniform range is primordial.

We leverage the following normalization procedure (see Equation 4.18) which enables us to map the QoS values into one range from 0 to 1.

$$N_{QoS_{ps_{i}ju}} = (QoS_{ps_{i}ju} - Qmin(p))/(Qmax(p) - Qmin(p)) \tag{4.18}$$

$$Qmin(j, p) = MIN_{S_i \in Cl_j}, u \in 1, ..l(QoS_{ps_{i}ju}) \tag{4.19}$$

$$Qmax(j, p) = MAX_{S_i \in Cl_j, u \in 1,..l}(QoS_{ps_{i}ju}) \tag{4.20}$$

4.7.1.2 Entropy and Cross-Entropy for Hesitant fuzzy set

Entropy and Cross-Entropy concepts are widely used for solving the MCDM problems. Indeed, the Entropy is leveraged to compute the weights of criteria. However, Cross-Entropy is applied to compute the divergence between two probability distributions. Hence, we leverage the Cross-Entropy for computing divergence between QoS realizations and best and/or worst solutions.

According to [Xu and Xia, 2012], Cross-Entropy for the hesitant fuzzy set is defined as follow:

Let $h_{s_{j1}}(p)$ and $h_{s_{j2}}(p)$ be two HFEs of the services s_{j1} and s_{j2} respectively, then the Cross-Entropy $C(h_{s_{j1}}(p), h_{s_{j2}}(p))$ is specified as:

$$
\begin{aligned}
\mathrm{C}(\mathrm{h}_{\mathrm{s}_{\mathrm{j}1}}(\mathrm{p}), \mathrm{h}_{\mathrm{s}_{\mathrm{j}2}}(\mathrm{p})) = \; & \tfrac{1}{lT_0} \sum_{u=1}^{l} \\
& \left(\tfrac{(1+tQoS_{ps_1ju}) \ln(1+tQoS_{ps_1ju}) + (1+tQoS_{ps_2ju}) \ln(1+tQoS_{ps_2ju})}{2} - \right. \\
& \tfrac{2+tQoS_{ps_1ju}+tQoS_{ps_2ju}}{2} \ln(\tfrac{2+tQoS_{ps_1ju}+tQoS_{ps_2ju}}{2}) + \\
& \tfrac{(1+t(1-QoS_{ps_1j(l-u+1)})) \ln(1+t(1-QoS_{ps_1j(l-u+1)}))}{2} + \\
& \tfrac{(1+t(1-QoS_{ps_2j(l-u+1)})) \ln(1+t(1-QoS_{ps_2j(l-u+1)}))}{2} - \\
& \left(\tfrac{2+t(1-QoS_{ps_1j(l-u+1)}+1-QoS_{ps_2j(l-u+1)})}{2} \right. \\
& \left. \left. \ln(\tfrac{2+t(1-QoS_{ps_1j(l-u+1)}+1-QoS_{ps_2j(l-u+1)})}{2}) \right) \right)
\end{aligned}
\tag{4.21}
$$

Where

$$\mathrm{T}_0 = (1+\mathrm{t}) \ln(1+\mathrm{t}) - (2+\mathrm{t})(\ln(2+\mathrm{t}) - \ln(2)), \mathrm{t} \geq 0. \tag{4.22}$$

Based on the definition of the Cross-Entropy and the realizations of q_1 (i.e., response time) presented in Example1, the Cross-Entropy between the service s_{11} and s_{12} will be:
$\mathrm{t} = 2, \mathrm{T}_0 = 0.52,$
$\mathrm{C}(\mathrm{h}_{\mathrm{s}_{11}}(1), \mathrm{h}_{\mathrm{s}_{12}}(1)) = 0.719$

According to [Xu and Xia, 2012] the entropy (or disorder quantity) of a fuzzy hesitant set is defined as follows:
Let h be a HFE, then entropy is specified as:

$$\mathrm{E}_{\mathrm{a}}(\mathrm{h}) = 1 - \mathrm{C}_1(\mathrm{h}, \mathrm{h}^{\mathrm{c}}). \tag{4.23}$$

$$E_{s_{ji}q}(h) = 1 - \frac{2}{lT_0}\sum_{u=1}^{l}$$

$$\left(\frac{(1+tQoS_{ps_1ju})ln(1+tQoS_{ps_1ju})+(1+t(1-QoS_{ps_1j(l-u+1)}))ln(1+t(1-QoS_{ps_1j(l-u+1)}))}{2}\right. \tag{4.24}$$

$$\left.-\frac{2+tQoS_{ps_1ju}+t(1-QoS_{ps_1j(l-u+1)})}{2}ln\frac{2+tQoS_{ps_1ju}+t(1-QoS_{ps_1j(l-u+1)})}{2}\right)$$

Using the definition of Entropy, the entropy of Example1 will be:

$\mathrm{E}_{111}(\mathrm{h}_{\mathrm{s}_{11}}) = 0.841$

4.7.1.3 Model of Entropy weights

Generally, solving the service composition problem requires the knowledge of QoS and its importance as QoS weights. Different approaches ask the users to set their QoS preferences. However, these methods are inappropriate since the users don't have the necessary knowledge to attribute the weight for each QoS attribute.

In this approach, we leverage entropy theory to provide the QoS weights. The higher the weight, the more important the QoS criterion is.

$$\mathrm{w}_{\mathrm{p}}^{'} = \frac{1-\mathrm{E}_{\mathrm{p}}}{\mathrm{m}-\sum_{\mathrm{p}=1}^{\mathrm{r}}\mathrm{E}_{\mathrm{p}}} \tag{4.25}$$

Where

$$\mathrm{E}_{\mathrm{p}} = \frac{1}{\mathrm{m}}\sum_{\mathrm{i}=1}^{\mathrm{m}}\mathrm{E}_{\mathrm{h}_{\mathrm{ip}}}, \mathrm{p} = 1, 2, ..., \mathrm{r}. \tag{4.26}$$

$$\mathrm{E}_{\mathrm{h}_{\mathrm{ip}}} = 1 - \mathrm{C}(\mathrm{h}_{\mathrm{ip}}, \mathrm{h}_{\mathrm{ip}}^{\mathrm{c}}). \tag{4.27}$$

Based on the computed QoS weights, we propose an algorithm that allows to rank Web services.

Algorithm 6 Entropy Service Ranking (ESR) is presented below.

The description of algorithm 6, is presented as follows:

- We firstly start to compute the weight w_{p} of each attribute p according to Equation (4.25) (lines 1 to 3).

Algorithm 6 EntropyServiceRanking

Input: $< \mathrm{CL}_1, ..., \mathrm{CL}_\mathrm{n} >$,
$\mathrm{h}_\mathrm{p}^+ = 1$ // positive ideal solution,
$\mathrm{h}_\mathrm{p}^- = 0$ // negative ideal solution
Output: $\mathrm{RankedCl}_1, ..., \mathrm{RankedCl}_\mathrm{n}$
Begin

1: **for** p=1 to r **do**
2: $\mathrm{w_p}$ = compute-weight() // The weight is computed according to Equation (4.25)
3: **end for**
4: **for** j=1 to n **do**
5: **for** i=1 to m **do**
6: $\mathrm{C}^+(\mathrm{s_{ji}}) = \sum_{\mathrm{p}=1}^{\mathrm{r}}(\mathrm{w_p C}(\mathrm{h_{s_{ji}}(p)}, \mathrm{h}_\mathrm{p}^+))$
7: $\mathrm{C}^-(\mathrm{s_{ji}}) = \sum_{\mathrm{p}=1}^{\mathrm{r}}(\mathrm{w_p C}(\mathrm{h_{s_{ji}}(p)}, \mathrm{h}_\mathrm{p}^-))$
8: $\mathrm{c}(\mathrm{s_{ji}}) = \frac{\mathrm{C}^+(\mathrm{s_{ji}})}{\mathrm{C}^+(\mathrm{s_{ji}})+\mathrm{C}^-(\mathrm{s_{ji}})}$
9: **end for**
10: RankedCl$_j$=ascending-sort ($\mathrm{CL_j}, \mathrm{c}(\mathrm{s_{ji}})$)
11: **end for**
12: **return** $< \mathrm{RankedCl}_1, ..., \mathrm{RankedCl}_\mathrm{n} >$
13: **end**

- We compute the Cross-Entropy between the service $\mathrm{s_{ji}}$ and the positive-ideal solution h_p^+ which concerns positive attributes (resp the negative-ideal solution h_p^- which concerns the negative attributes) (lines 4 to 7).
- We compute the closeness degree between the service s_{ji} and the ideal solution (line 8).
- We sort the services of each class according to the values of $\mathrm{c}(\mathrm{s_{ji}})$ in ascending order(to explore new areas). The smaller the value of $\mathrm{c}(\mathrm{s_{ji}})$, the better the rank (line 10).
- We return the ranked classes in line 12.

4.7.2 Grey Wolf Optimizer

Grey Wolf Optimizer (GWO) is a recent swarm intelligence algorithm proposed in [Mirjalili et al., 2014]. This population-based meta-heuristic succeed to overcomes the shortcoming of the previous meta-heuristics. It enjoys a superiority due to its flexibility, simplicity of implementation with few control parameters, and fast convergence characteristics [Faris et al., 2018]. GWO represents the leadership and social hierarchy of the grey wolves (GW). Roughly speaking, this includes four main categories called: alpha (α), beta (β), delta (δ), and omega (ω). Additionally, GW groups follow a specific category order where the leadership starts from the upper level to the lower one (see Figure 4.5). Moreover, GW are characterized by their special hunting behavior which is realized in three phases: encircling, hunting, and attacking.

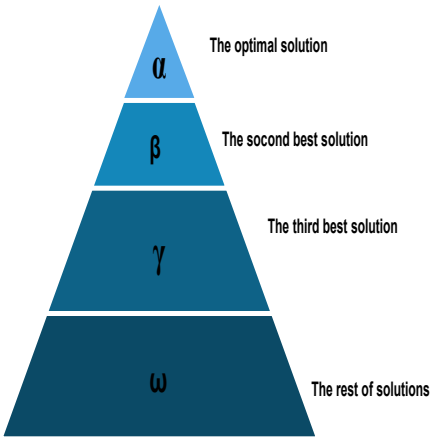

Figure 4.5: Social hierarchy of grey wolves.

Successfully, the GWO is adapted to several and recent problems (such as Support Vector Machine (SVM) Classification [Eswaramoorthy et al., 2016], Wireless Sensor Networks (WSNs) [Parsian et al., 2017], Path Planning [Zhang et al., 2016], Biomedical Research [Bian et al., 2017]). Therefore, we leveraged this meta-heuristic to handle the service composition problem. This approach considers that the α category is the most suitable solution, whereas β and δ represent the second and third best solutions; the Omega category (ω wolves) represents the weakest solutions.

In [Mirjalili et al., 2014], the researchers modeled mathematically the three main phases of the hunting behavior.

1. The encircling behavior is modeled as follows:

$$\overrightarrow{\mathrm{D}} = |\overrightarrow{\mathrm{C}}.\overrightarrow{\mathrm{X}}_{\mathrm{p}}(\mathrm{t}) - \overrightarrow{\mathrm{X}}(\mathrm{t})| \tag{4.28}$$

$$\overrightarrow{\mathrm{X}}(\mathrm{t}+1) = \overrightarrow{\mathrm{X}}_{\mathrm{p}}(\mathrm{t}) - \overrightarrow{\mathrm{A}}.\overrightarrow{\mathrm{D}} \tag{4.29}$$

Where

(a) t is the current iteration.

(b) $\overrightarrow{\mathrm{X}}_{\mathrm{p}}$ represents the prey's position.

(c) $\overrightarrow{\mathrm{X}}(\mathrm{t})$ is the current position of a wolf.

(d) $\overrightarrow{\mathrm{X}}(\mathrm{t}+1)$ is the next position.

(e) D represents the disturbed distance between the position of the current wolf X(t) and the position of the prey X_p (which represents the best wolf in the pack, such as alpha or even beta), the perturbation parameter C allows for exploration of new areas when its magnitude is greater than or equal to 1, otherwise, it encourages the exploitation of the current region.

(f) Equation 4.29 represents the actual encircling process, we notice that the distance D is scaled by the factor A which is comprised between -2a and 2a, and this enables the presence of wolves (i.e., X(t+1)) in the perimeter of the prey, especially when A belongs to [-1,1].

The random vectors $\overrightarrow{\mathrm{A}}$ and $\overrightarrow{\mathrm{C}}$ are calculated as follows [Mirjalili et al., 2014]:

$$\overrightarrow{\mathrm{A}} = 2.\overrightarrow{a}.\overrightarrow{\mathrm{r}}_1 - \overrightarrow{\mathrm{a}} \tag{4.30}$$

$$\overrightarrow{\mathrm{C}} = 2.\overrightarrow{\mathrm{r}}_2 \tag{4.31}$$

$$\overrightarrow{\mathrm{a}} = 2 - \frac{2.\mathrm{t}}{\mathrm{T}} \tag{4.32}$$

Where $\overrightarrow{\mathrm{a}}$ linearly decreases from 2 to 0 during the iterations; $\overrightarrow{\mathrm{r}}_1$, $\overrightarrow{\mathrm{r}}_2$ are random values in the range [0, 1]. Accordingly, the range of $\overrightarrow{\mathrm{A}}$ is [-2a,2a] and the range of $\overrightarrow{\mathrm{C}}$ is [0,2]. T is the maximum number of iterations.

The exploration and the exploitation processes are controlled in GWO through the vectors $\overrightarrow{\mathrm{A}}, \overrightarrow{\mathrm{C}}$. In the exploration process, the wolf discovers the promising areas and diverges from the prey (to explore new areas) when $\overrightarrow{\mathrm{A}} > 1$ or $\overrightarrow{\mathrm{A}} < -1$. In addition, the parameter $\overrightarrow{C}$ contributes to the exploration process since the wolf moves closer to the prey when $\overrightarrow{\mathrm{C}} > 1$.

Moreover, in the exploitation, the wolf makes an advance towards the prey when $-1 < \overrightarrow{A} < 1$. The vectors $\overrightarrow{\mathrm{A}}, \overrightarrow{\mathrm{C}}$ are used in the next equations, where $\overrightarrow{\mathrm{A}}_1, \overrightarrow{\mathrm{C}}_1$ are the vectors related to alpha (α) ($\overrightarrow{A_2}, \overrightarrow{C_2}$ are related to β and $\overrightarrow{\mathrm{A}}_3, \overrightarrow{\mathrm{C}}_3$ are related to δ).

2. Hunting is the second step after encircling the prey. In this step, only the three main leaders have knowledge about the position of the prey, therefore, all omega wolves update their positions according to alpha, beta, and delta positions'.

Figure 4.6 is a simple representation of the omega's relocation in 2d search space near the prey according to a specific distance D.

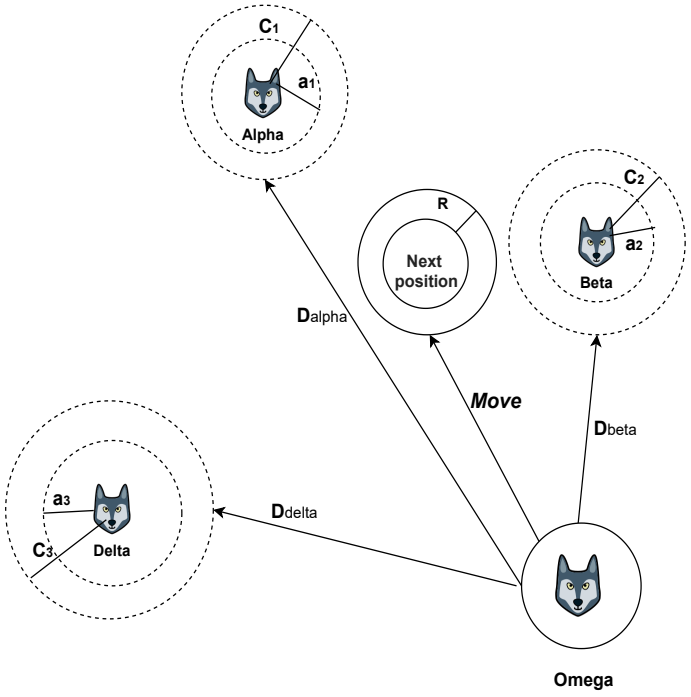

Figure 4.6: Position updating in GWO.

$\overrightarrow{\mathrm{D}}_{\alpha}$ ($\overrightarrow{\mathrm{D}}_{\beta}$ and $\overrightarrow{\mathrm{D}}_{\delta}$ respectively) is the disturbed distance between the position of the current omega wolf and the position of the alpha wolf (beta and delta respectively) (see Equations 4.34 - 4.36). Hunting is also modelled according to Equation 4.33.

Obviously, the computation of the distance D is controlled by the vectors $\overrightarrow{\mathrm{C}}$, and $\overrightarrow{\mathrm{A}}$, since the latter parameters assist candidate solutions to have hyper-spheres with different random radii [Mirjalili et al., 2014].

To improve the position of the current wolf (termed as $\overrightarrow{\mathrm{X}}(\mathrm{t})$), we calculate the next position ($\overrightarrow{\mathrm{X}}(\mathrm{t}+1)$) using Equation 4.33.

$$\overrightarrow{\mathrm{X}}(\mathrm{t}+1) = \frac{\overrightarrow{\mathrm{X}}_1 + \overrightarrow{\mathrm{X}}_2 + \overrightarrow{\mathrm{X}}_3}{3}. \tag{4.33}$$

Where

$$\overrightarrow{\mathrm{D}}_{\alpha} = \overrightarrow{\mathrm{C}}_1.\overrightarrow{\mathrm{X}}_{\alpha} - \overrightarrow{\mathrm{X}} \tag{4.34}$$

$$\overrightarrow{\mathrm{D}}_{\beta} = \overrightarrow{\mathrm{C}}_{2}.\overrightarrow{\mathrm{X}}_{\beta} - \overrightarrow{\mathrm{X}} \tag{4.35}$$

$$\overrightarrow{\mathrm{D}}_{\delta} = \overrightarrow{\mathrm{C}}_{3}.\overrightarrow{\mathrm{X}}_{\delta} - \overrightarrow{\mathrm{X}}. \tag{4.36}$$

Where $\overrightarrow{\mathrm{X}}_{\alpha}$, $\overrightarrow{\mathrm{X}}_{\beta}$, $\overrightarrow{\mathrm{X}}_{\delta}$ denote the positions of alpha, beta, and delta respectively.

$$\overrightarrow{\mathrm{X}}_{1} = \overrightarrow{\mathrm{X}}_{\alpha} - \overrightarrow{\mathrm{A}}_{1}.\overrightarrow{\mathrm{D}}_{\alpha} \tag{4.37}$$

$$\overrightarrow{\mathrm{X}}_{2} = \overrightarrow{\mathrm{X}}_{\beta} - \overrightarrow{\mathrm{A}}_{2}.\overrightarrow{\mathrm{D}}_{\beta} \tag{4.38}$$

$$\overrightarrow{\mathrm{X}}_{3} = \overrightarrow{\mathrm{X}}_{\delta} - \overrightarrow{\mathrm{A}}_{3}.\overrightarrow{\mathrm{D}}_{\delta} \tag{4.39}$$

3. The hunting behavior is ended through the attacking step. It is simply formulated by the value of a (see Equation 4.32) that implicitly controls the exploration and exploitation.

4.7.3 Discrete Grey Wolf Optimizer

Actually, the initial GWO address for the continuous optimization problem, nevertheless, the service composition problem in a discrete optimization problem. To handle this situation, we propose a recent approach called Grey Wolf-Based Composition (GWC).

In this approach, we update the previous Equations as follows:

$$\overrightarrow{\mathrm{X}}(\mathrm{t}+1) = \lfloor \frac{\overrightarrow{\mathrm{X}}_{1} + \overrightarrow{\mathrm{X}}_{2} + \overrightarrow{\mathrm{X}}_{3}}{3} \rfloor. \tag{4.40}$$

Where $\overrightarrow{\mathrm{X}}_{\alpha}$, $\overrightarrow{\mathrm{X}}_{\beta}$, $\overrightarrow{\mathrm{X}}_{\delta}$ denote the locations of the services alpha, beta, and delta respectively in the ranked classes.

$$\overrightarrow{\mathrm{X}}_{1} = \mathrm{Move}(\overrightarrow{\mathrm{X}}, \lfloor \overrightarrow{\mathrm{D}_{\alpha} \mod 3} \rfloor, \mathrm{A}_{1}) \tag{4.41}$$

$$\overrightarrow{\mathrm{X}}_{2} = \mathrm{Move}(\overrightarrow{\mathrm{X}}, \lfloor \overrightarrow{\mathrm{D}_{\beta} \mod 3} \rfloor, \mathrm{A}_{2}) \tag{4.42}$$

$$\overrightarrow{\mathrm{X}}_{3} = \mathrm{Move}(\overrightarrow{\mathrm{X}}, \lfloor \overrightarrow{\mathrm{D}_{\delta} \mod 3} \rfloor, \mathrm{A}_{3}) \tag{4.43}$$

$$\overrightarrow{\mathrm{D}}_{\alpha} = \overrightarrow{\mathrm{C}}_{1}.\overrightarrow{\mathrm{X}}_{\alpha} - \overrightarrow{\mathrm{X}} \tag{4.44}$$

$$\overrightarrow{\mathrm{D}}_{\beta} = \overrightarrow{\mathrm{C}}_{2}.\overrightarrow{\mathrm{X}}_{\beta} - \overrightarrow{\mathrm{X}} \tag{4.45}$$

$$\overrightarrow{\mathrm{D}}_{\delta} = \overrightarrow{\mathrm{C}}_{3}.\overrightarrow{\mathrm{X}}_{\delta} - \overrightarrow{\mathrm{X}} \tag{4.46}$$

$\mathrm{Move}(\overrightarrow{\mathrm{X}}, \lfloor\overrightarrow{\mathrm{D}_{\alpha} \mod 3}\rfloor, \mathrm{A}_1)$ means that the position of the next service regarding α, is the current position moved with $\lfloor\overrightarrow{\mathrm{D}_{\alpha} \mod 3}\rfloor$ steps.

As well, in this algorithm, we adopt the GQC (see Equation (4.14)) as the objective function.

Example

The following example is presented to explain the function called $\mathrm{Move}(\overrightarrow{\mathrm{X}}, \lfloor\overrightarrow{\mathrm{D} \mod 3}\rfloor, \mathrm{A})$. Let's assume that we have a dataset that contains 2 tasks and 10 services; the ranked classes based on Entropy and Cross-Entropy are presented in Table 4.3.

We generate the population (wolves) by setting k to 7 (7 pertinent services from each class), $\mathrm{A}_1 = \mathrm{A}_2 = \mathrm{A}_3$ = 0.5 (i.e., $-1 < \mathrm{A} < 1$) and $\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3$ =1.

Table 4.3: Ranked services based on Entropy and Cross-Entropy.

Ranked CL_1	s_6	s_7	s_8	s_4	s_{10}	s_2	s_9	s_5	s_3	s_1
Ranked CL_2	s_{10}	s_3	s_2	s_9	s_5	s_8	s_7	s_4	s_1	s_6

Table 4.4: Random wolves' population based on ranked services.

Task 1	Task 2	
s_8	s_{10}	Alpha
s_9	s_9	Beta
s_4	s_2	Delta
s_7	s_5	Omega 1
s_6	s_{10}	Omega 2
s_4	s_3	Omega 3

Using Equation 4.40, we update the omega's position (the fourth line of Table 4.4) as follows:

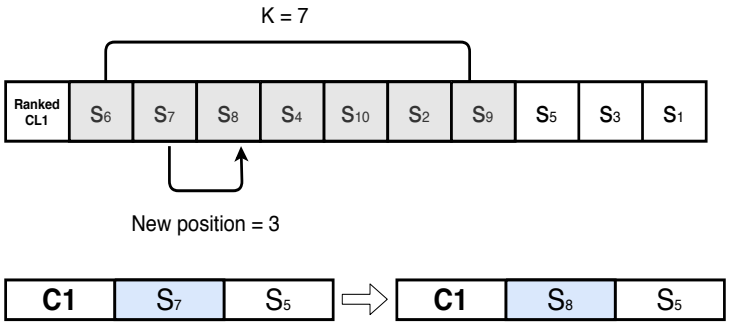

Figure 4.7: The move function for service s_7 .

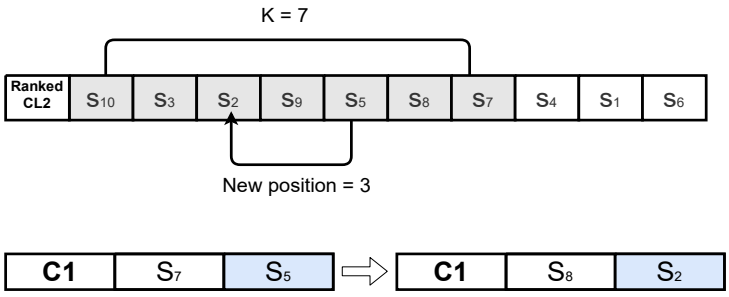

Figure 4.8: The move function for service s_5 .

- $d_1 = 1$, because the position of Alpha(s_8) equals 3 and the position of s_7 equals 2. Since the position of alpha is greater than the position of s_7, then Move ($s_7, \lfloor 1 \mod 3 \rfloor, A_1$) = 3: in fact, 2 (i.e., position of s_7) + 1 (i.e., $\lfloor 1 \mod 3 \rfloor$ steps) will provide 3 as a new position.
- $\text{newposition} = 3$.
- $d_2 = 5$ and Move ($s_7, \lfloor 5 \mod 3 \rfloor, A_2$) = 4 (According to beta).
- $\text{newposition} = 4$.
- $d_3 = 2$ and Move ($s_7, \lfloor 2 \mod 3 \rfloor, A_3$) = 4 (According to delta).
- $\text{newposition} = 4$.

New position = $\frac{\lfloor 3+4+4 \rfloor}{3}$ = 3.

- $d_1 = 4$, because the position of Alpha equals 1 and the position of s_5 equals 5. Since the position of alpha is less than position of s_5, then Move ($s_5, \lfloor 4 \mod 3 \rfloor, A_1$) = 4. In fact, 5 (i.e., position of s_5) - 1 (i.e., $\lfloor 4 \mod 3 \rfloor$ steps)= 4.

- $\text{newposition} = 4$.
- $\text{d}_2 = 1$ and Move ($\text{s}_5, \lfloor 1 \mod 3 \rfloor, \text{A}_2$) = 4 (According to beta).
- $\text{newposition} = 4$.
- $\text{d}_3 = 2$ and Move ($\text{s}_5, \lfloor 2 \mod 3 \rfloor, \text{A}_3$) = 3 (According to delta).
- $\text{newposition} = 3$.

New position = $\lfloor 4 + 4 + 3 \rfloor / 3$ = 3. New composition <s_8, s_2>.

The detailed grey wolf-based composition algorithm (GWC) is presented in follows.

In addition, we leverage GQC as an objective function (see Equation 4.14).

Algorithm 7 GWC

Input: $< \text{CL}_1, ..., \text{CL}_\text{n} >$, k , MaxIter , PopSize.

Output: TopKCompositions

Begin

1: $< \text{RankedCl}_1, ..., \text{RankedCl}_\text{n} >$= EntropyServiceRanking(CL)
2: **for** i=1 to PopSize **do**
3: Initialize the grey wolf population X_i = init($< \text{RankedCl}_1, ..., \text{RankedCl}_\text{n} >$, k) // Initialize the grey wolf population based on ($< \text{RankedCl}_1, ..., \text{RankedCl}_\text{n} >$) where ($\text{i} = 1, 2, ..., \text{PopSize}$).
4: Initialize a, A, and C;
5: $\text{FitnessX}_\text{i} = \text{ComputeGQC}(\text{X}_\text{i})$;
6: **end for**
7: Decreasing-sort($\text{X}_\text{i}, \text{FitnessX}_\text{i}$)
8: $\text{X}_\alpha = \text{X}_1$;
9: $\text{X}_\beta = \text{X}_2$;
10: $\text{X}_\delta = \text{X}_3$;
11: t=1
12: **while** ($\text{t} \leq \text{MaxIter}$) **do**
13: **for** i=1 to PopSize **do**
14: Update1 (X_i) //the position of the current wolf is updated using Equation (4.40)
15: $\text{FitnessX}_\text{i} = \text{ComputeGQC}(\text{X}_\text{i})$;
16: **end for**
17: Update2 (a, A, C) //We update A, C, a using Equations (4.30), (4.31), (4.32)
18: Decreasing-sort($\text{X}_\text{i}, \text{FitnessX}_\text{i}$)
19: Update3 ($\text{X}_\alpha, \text{X}_\beta, \text{X}_\delta$);
20: t=t+1;
21: **end while**
22: **return** (TopKCompositions)
23:
24: **end**

The implementation of GWC algorithm is explained as follows:

1. We call the ESR algorithm to obtain the ranked classes (line 1).
2. Based on the Top-k elements (Top-K services retrieved based on algorithm 7), we generate the grey wolf population (service compositions) (lines 2-3).
3. We initialize a, A, and C (line 4).
4. We evaluate each individual of the population (composition) with the GQC (line 5).
5. We design the leaders α, β, δ, based on a deceasing-sort function (lines 7 to 10).
6. For each iteration (lines 12 up to 21):
   - We call the function Update1 to relocate the wolves' positions according to Equation (4.40) (line 14) .
   - We evaluate the new wolves according to Equation (4.14) (line 15).
   - We invoke the function Update2 to update the vectors A, C using Equations (4.30), (4.31) (line 17).
   - We apply a selection mechanism between wolves to retrieve the best new wolves α, β, δ (i.e., Update3) after a decreasing sort (lines 18 to 19).
7. We return the Top k composition (α, β, δ) (Top3).

4.8 Complexity of the proposed approaches

In Table 4.5, we present the complexity of each proposed algorithm.

Table 4.5: Approaches complexity.

Approach	**Algorithm**	**Complexity**
DSR	DSR	$O(n.m^2.r.l^2 + n.mlogm)$
	Backtraking algorihm	$O(k^n(k.r.n + n + klogk))$
MJ-CP	MSR	$O(n.m^2.r.l^2 + n.mlogm)$
	MJ-CP	$O(k^n(k.r.n + n + klogk))$
GWC	ESR	$O(2.n.m.r.l)$
	GWC	$\mathrm{O(2.n.m.r.l + PopSize.MaxIter(n + r.l^n) + PopSize * log(PopSize))}$

4.9 Conclusion

In the aim of finding the best composition in a fluctuating environment, we proposed to reduce the search space by retrieving the relevant services according to their QoS. These services are combined in order to generate multiple compositions, which means that another selection process is performed to find the composition that respects the global constraints. Therefore, we present in this chapter the following approaches:

1. The first approach is mainly composed of two algorithms: Dominance Service Ranking (DSR) is leveraged for ranking the services. The ranking mechanism is implemented according to the probabilistic dominance relationship. The second algorithm is the Backtracking search. this algorithm is proposed to select the Top K compositions according to an improved objective function.

2. The second approach is termed Majority judgment and Constraints Programming approach (MJ-CP). MJ denotes the Majority service ranking algorithm. It is leveraged for local selection of services. The Constraint programming algorithm (CP) is adopted for global selection (i.e., composition selection).

3. The last approach is called Grey wolf-based composition. In fact, this is a combination of two algorithms: Entropy Service Ranking advanced for selecting the services with the consideration of the uncertain QoS (local search). Following that, the Grey wolf-based composition algorithm is proposed to select the Top K compositions according to the GQC.

Chapter 5

Implementation and experimental results

5.1 Introduction

In this chapter, we present the experiments carried out to evaluate the performance of our approaches. We initially introduce the detailed architecture of the selection module.

Next, we describe the datasets adopted in the experiments. Finally, we present the results of the experiments.

5.2 Case studies

We report the three case studies that we have proposed for the evaluation:

1. The first case concerns the e-health application presented in section 4.3. (see Figure 4.1).
2. The second case is the electronic product purchase composite service, adopted by [Hwang et al., 2015] (see Figure 5.1). This composite service is constituted of ElectronicProductFinder, SendSMS, SmartPayment, CreditCardPayment, and Delivery services.
3. The e-learning system on the cloud is the third case. This system includes two cloud services:
    - An Amazon EC2 Spot instance for storage management of the learning system (IaaS application).
    - An application server for handling the learning contents (SaaS application).

5.3 Dataset Description

The experiments have been conducted using both real and synthetic datasets to evaluate the proposed selection module. Each approach has its own datasets (It could include one or multiple

datasets).

In this section, we present both datasets, and configurations adopted in these experiments.

5.3.1 Dataset of Probabilistic dominance approach

The QoS of this dataset is generated by randomizing data values and leveraged for evaluating the Dominance Service ranking and Backtracking search algorithms.

The performed configuration is presented in Table 5.1. The number of tasks varies from 10 to 20, and the size of the service per task is up to 1100. The services are characterized by at least two QoS. Each QoS consist of 100 to 400 realizations (instances).

The proposed configuration is presented as follows:

Table 5.1: Configuration DSR approach.

Parameters Values	**Values**
Number of Tasks (n)	10, 15, 20
Number of Services (m)	100 to 1100
Number of QOS criteria (r)	2 to 10
Instances (l)	100 to 400
Size of the result (k)	2 ,6,10

5.3.2 MJ-CP approach's dataset

For the second experiment, we generate three types of datasets:

1. In order to evaluate our case study (see Figure 4.1), we generate a dataset according to a Gaussian distribution, the mean and the standard deviation of the response time and reliability are presented in Table 5.2. The fidelity is uniformly generated between 1 and 4. Furthermore, we fixed the global constraints of response time, reliability, and fidelity as 2600 ms, 0.035, and 3.

2. The second one is a standard dataset provided by Al-Masri and Q. H. Mahmoud [Al-Masri and Mahmoud, 2007]. It includes 365 rows that represent the Web services, and 13 columns, each one representing a QoS attribute. More specifically, we leverage this real-world dataset to analyze the example of Hwang et al.,2015 (see Figure 5.1). The response time and reliability values are generated according to a Gaussian distribution. The range of each task is presented in Table 5.3, where μ is the mean and σ is the standard deviation. Likewise, the fidelity is uniformly generated between 1 and 4. In addition, we set the global constraints of response time, reliability, and fidelity as 2400 ms, 0.025, and 3.

3. In order to emphasize the analysis, we adopt the dataset proposed by [Hadjila et al., 2020],

where the QoS values are generated according to a centered reduced Gaussian distribution (i.e., mean=0 and sigma=1).

For this dataset, we designate the following parameters:

- n=2.
- m=500.
- r=4 (i.e., response time, reputation, reliability, and throughput).
- l=21.
- b_j= 0.6*n for additive QoS attributes (such as reputation).
  0.6^n for multiplicative QoS attributes (such as reliability).
  0.6 for Max/Min QoS attributes(such as throughput).

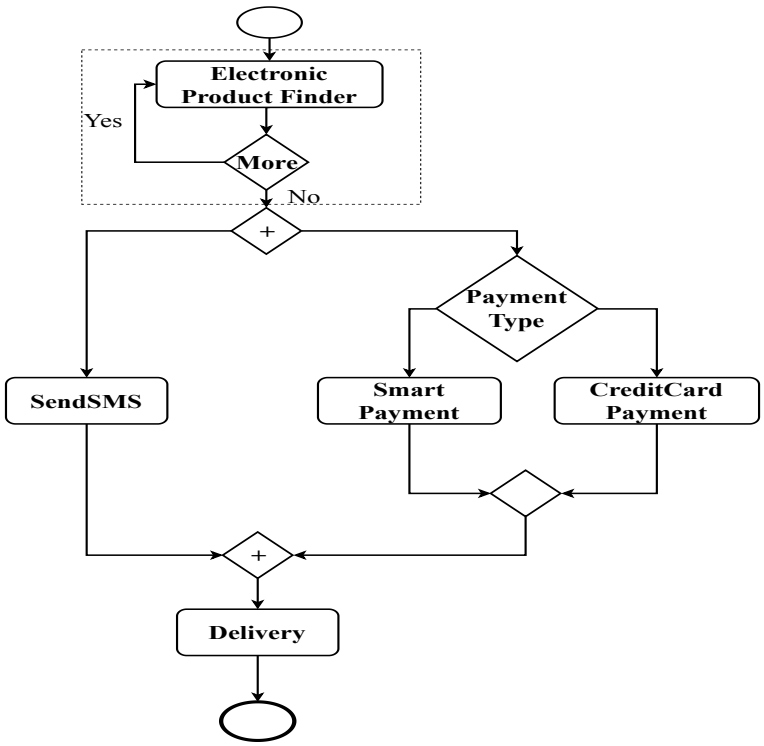

Figure 5.1: Electronic product purchase service (Case study 2) [Hwang et al., 2015].

5.3.3 Datasets of Grey Wolf based-Composition

For evaluating the third proposed approach, we adopt the same dataset of Al-Masri and Q. H. Mahmoud [Al-Masri and Mahmoud, 2007], for analyzing the second case study (see Figure 5.1, and Table 5.3).

Furthermore, in order to analyze the third case study, we use another synthetic dataset produced according to a centred reduced Gaussian distribution (i.e., mean=0 and sigma=1).

We employ the r4.large windows spot instance for one year, to generate the simulation requests, where the price configuration is: \$0.10 per hour for a virtual machine of 2 vcpu and 15 Gb for memory.

The capacity and/or price changes cause the revocation of the spot instances; consequently, this will influence the availability attribute. For instance, if the availability is equal to 70%, then the instance is available for approximately eight months (according to a request of one year).

Further, we propose the following configuration:

- n=2.
- m=500.
- r=3 (i.e., response time, availability, and fidelity).
- l=21.
- b_j= 0.6*n for additive QoS attributes (such as response time).
- 0.6^n for multiplicative QoS attributes (such as availability).
- 0.6 for Max/Min QoS attributes(such as throughput).
- k = 10 up to 25.
- PopSize=100 up to 500.
- MaxIter= 100 up to 500.

5.4 Experimental results and analysis

In this section, we present the results of the performed experiments, and we provide a detailed analysis of each result. It is worth mentioning that all algorithms were implemented using the Java programming language. Experiments were performed on a Windows 7 with an Intel I3 core 2.53GHz processor, 4 GB RAM.

5.4.1 Performance evaluation of the probabilistic dominance approach

In this experiment, we leverage a sequential workflow, and we treat only the QoS criteria of a composition that are aggregated according to the sum function (such as reputation). While, the multiplicative criteria are replaced with their log value and we handled them as additive criteria. Furthermore, the default value of each weight w_p equals to 1/r. We assign this equitable weight

Table 5.2: Configuration of Gaussian distributions (Case study 1).

Task	Response Time	Reliability
Authetication and access control for cloud storage service	$\mu \in$ [110,130] σ =50	$\mu \in$[0.5,0.95] σ =0.17
Annotation of medical images	$\mu \in$[1300,2000] σ = 300	$\mu \in$[0.60,0.90] σ =0.11
Medical image sharing	$\mu \in$[120,200] σ =37	$\mu \in$[0.40,0.95] σ =0.13
Medical diagnosis	$\mu \in$[100,150] σ =31	$\mu \in$[0.60,0.80] σ =0.27
Obtaining multimodal medical images	$\mu \in$[350,600] σ =53	$\mu \in$[0.53,0.65] σ =0.25
Medical scheduling service	$\mu \in$[800,900] σ =250	$\mu \in$[0.71,0.85] σ =0.33

Table 5.3: Configuration of Gaussian distributions (Case study 2).

Task	Response Time	Reliability
Electronic Product Finder	$\mu \in$ [78,117] $\sigma \in$[0,30]	$\mu \in$[0.63,0.95] $\sigma = \sqrt{\mu(1-\mu)}$
SendSMS	$\mu \in$[1046,1569] $\sigma \in$ [0,435]	$\mu \in$[0.45,0.80] $\sigma = \sqrt{\mu(1-\mu)}$
SmartPayment	$\mu \in$[117,175] $\sigma \in$[0,45]	$\mu \in$[0.60,0.90] $\sigma = \sqrt{\mu(1-\mu)}$
CreditCard Validator	$\mu \in$[254,48] $\sigma \in$[0,105]	$\mu \in$[0.48,0.71] $\sigma = \sqrt{\mu(1-\mu)}$
UPSTracking	$\mu \in$[74,111] $\sigma \in$[0,30]	$\mu \in$[0.62,0.94] $\sigma = \sqrt{\mu(1-\mu)}$

to each QoS criterion since we consider that the user attributes the same importance for all of them.

We analyze the computation time when retrieving 2, 6, and 10 services per class. Furthermore, we compare the DSR algorithm with Average Service Ranking (ASR) that rank the service according to the sum of average, based on the following Equation:

$$Rank(i,j) = \sum_{p=1}^{r} AVGQoS_{ps_iju} \tag{5.1}$$

Where

$AVGQoS_{ps_iju}$ is the average QoS computed over all the instances of $S_i \in Cl_j$.

Finally, we perform a comparison based on computational time, the utility function 4.6, and the ratio of respected user's constraints (p.s.g.c) through the inequalities 4.12, 4.13.

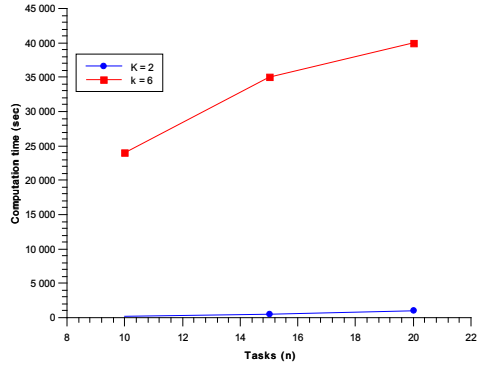

Figure 5.2: Computation time vs. n (r=3, m=50, l=10).

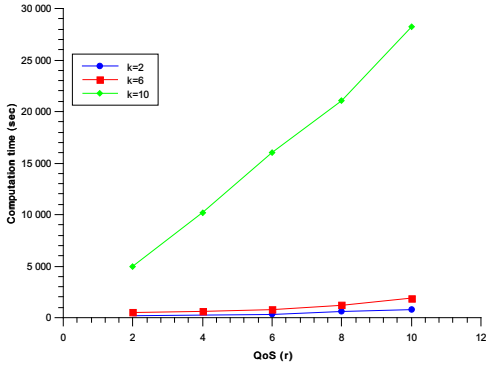

Figure 5.3: Computation time vs. r (n=5, m=200, l=100).

The performance evaluation in Figure 5.2 shows that the number of tasks influences the computational time, which follows an exponential growth. In fact, the execution time is acceptable when we select 2 services per class. However, when we select 6 or more, the computational time overhead is not tolerable for n $\geq$ 10, since the backtracking search algorithm checks at least 6^{10} possible compositions (k^n).

Figure 5.3 shows the sensitivity to QoS attributes (r) over the execution time. We can notice that all values of r, are tolerable for k=2 and k=6, however, for k=10 and r $\geq$4, the execution time will be unacceptable. The same observation is made for Figure 5.4, for k=2 and k=6, the execution time is tolerable, however for k=10 and l$\geq$ 200 the computational is not acceptable.

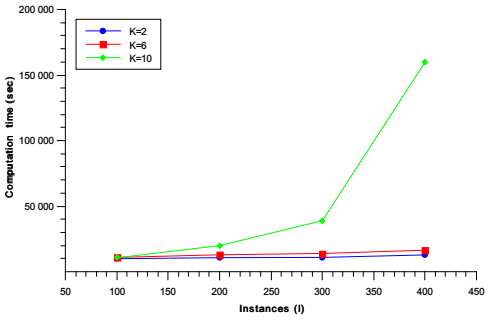

Figure 5.4: Computation time vs. l (n=5, r=3, m=200).

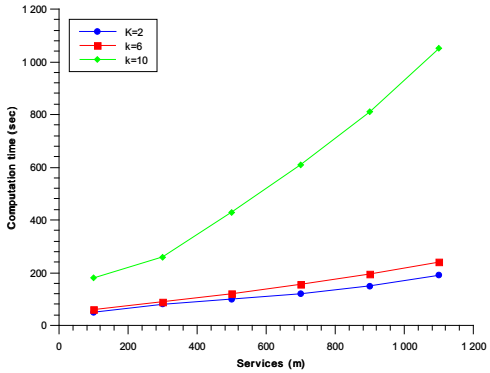

Figure 5.5: Computation time vs. m (n=5, r=3, l=10).

As shown in Figure 5.5, the number of services (m) has a slight impact on the global search (algorithm 2), this is mainly due to the fact that building the compositions depends on the number of relevant services (i.e., K).

We can notice from Figure 5.6, that the ASR approach is better than the DSR in terms of execution time. This is due to the fact that the ASR algorithm doesn't depend on m, unlike the

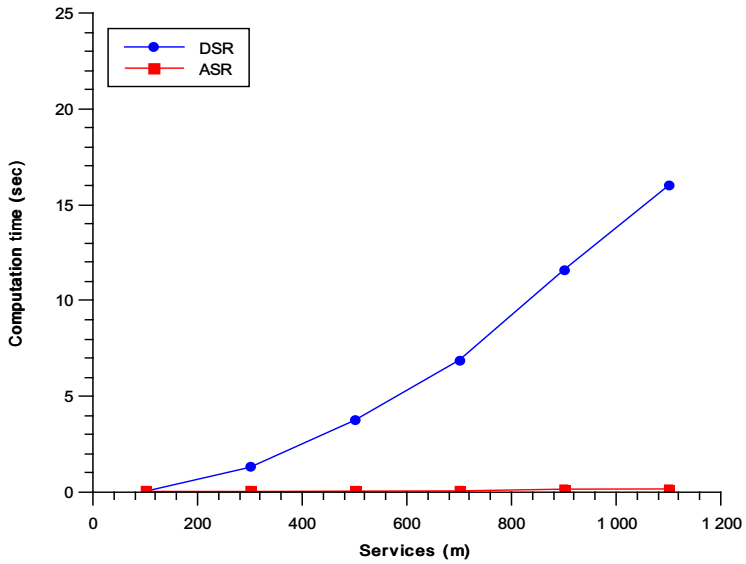

Figure 5.6: Computation time vs. m for DSR and ASR.

DSR algorithm. Furthermore, the complexity of Equation 5.1 is only $O(r.l)$, and the Equation 4.1 complexity is $O(r.l^2)$.

Table 5.4: Performance comparison between DSR and ASR.

Parameters	**Solutions**	**Algorithms**	**Utility** $U^{'}(.)$	**p.s.g.c (%)**
	Top2 Solutions	**DSR + Algo 2**	0.4063	100
			0.4053	100
		ASR + Algo 2	0.3823	100
			0.3813	100
	Top6 Solutions	**DSR + Algo 2**	0.4075	100
			0.4073	100
			0.407	100
n=5, r=3,			0.4068	100
m=200 ,l=300			0.4065	100
			0.4064	100
		ASR+ Algo2	0.3997	100
			0.3992	100
			0.3989	100
			0.3984	100
			0.3983	100
			0.3982	100

It is clearly seen in Table 5.4 that both approaches (ASR and DSR) have a superiority in terms of p.s.g.c. Meanwhile, we observe a slight fitness superiority (Equation 4.6) of the DSR compared to the ASR, for all values of K.

Regarding these experimental results, we can notice that the pairwise service comparison

(Dominance Service Ranking algorithm) leveraged for the service selection and the exhaustive search (backtracking search algorithm) is extremely time-consuming. Therefore, we propose the following approaches to solve the service composition problem.

5.4.2 Performance evaluation of the Majority judgment & Constraint programming approach

In the experiments related to the first case study, we extend the number of concurrent services from 100 to 500 and the number of QoS realizations from 10 to 60.

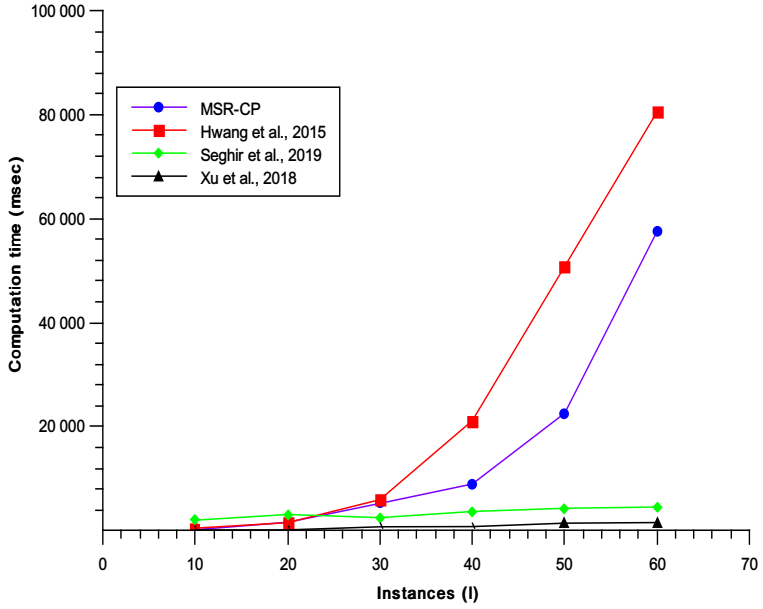

Figure 5.7: Computation time vs. l (Case Study 1).

In Table 5.5, we present the GQC and the utility function of each approach with respect to the number of candidate services (m) and instances (l) (our approach and [Hwang et al., 2015] approach use the GQC as a utility function). For IM_ABC [Seghir et al., 2019] approach, we present the interval-valued QoS of the best feasible solution. However, if the best solution is infeasible because of constraint violation, we compute the TCV function that represents the total constraint violation (expressed in terms of an unnormalized interval). For TGA [Xu et al., 2018] approach, the utility function is computed based on Equation 5.2, where $(\overline{d}_{ij})_h$ is the defuzzified QoS value of the h^{th} criterion, and the $(p_j)_h$ represents the QoS priority weight vector.

$$F_j = \sum_{h=1}^{k} (p_j)_h * (\overline{d}_{ij})_h \tag{5.2}$$

We can notice that the utility function of state-of-the-art methods is advanced (e.g., 0.86 for

Table 5.5: GQC of Top composition (Case Study 1).

		l=10		**l=20**		**l=30**		**l=40**	
		GQC	**Utility Function**	**GQC**	**Utility Function**	**GQC**	**Utility Function**	**GQC**	**Utility Function**
m=100	MSR-CP	0.95		0.87		0.86		0.84	
	Hwang et al.,2015.	0.79		0.72		0.71		0.73	
	Seghir et al.,2019.	0.62	TCV= [-0.56 , 0.040]	0.62	TCV= [-0.68,0.04]	0.68	TCV= [-0.83 , 0.044]	0.62	TCV= [-0.81 , 0.044]
	Xu et al.,2018.	0.63	0.86	0.80	0.84	0.81	0.82	0.73	0.78
m=300	MSR_CP	0.95		0.93		0.91		0.88	
	Hwang et al.,2015.	0.77		0.81		0.66		0.66	
	Seghir et al.,2019.	0.56	TCV= [-0.44 , 0.044]	0.63	TCV=0 $q_1 = [611.71, 1251.5]$ $q_2 = [0.04, 0.8031]$ $q_3 = [3, 4]$	0.62	TCV= [-0.88 , 0.044]	0.60	TCV= [-0.81 , 0.044]
	Xu et al.,2018.	0.50	0.52	0.50	0.48	0.48	0.46	0.49	0.45
m=500	MSR_CP	0.93		0.88		0.87		0.88	
	Hwang et al.,2015.	0.89		0.82		0.74		0.74	
	Seghir et al.,2019.	0.62	TCV= [0.025, 0.04]	0.57	TCV= [-0.84 , 0.044]	0.56	TCV=0 $q_1 = [574.43, 1383.6]$ $q_2 = [0.037, 0.8]$ $q_3 = [3, 3]$	0.56	TCV= [-0.80 , 0.044]
	Xu et al.,2018.	0.42	0.39	0.40	0.37	0.37	0.35	0.35	0.34

Xu et al.,2018 approach), however, MSR-CP ensures the highest GQC in all experiments presented in Table 5.5 (about 20% better than other methods). These results indicate that MSR-CP is more effective than the state-of-the-art methods. This is due to the fact that our approach retrieves the most pertinent services through the MajorityServiceRanking algorithm, and considers the global constraints in the selection of Top compositions. The curves shown in Figure 5.7 present the evolution of the computation time. We can notice that our approach is sensitive to the number of instances and slower than the approaches [Xu et al., 2018, Seghir et al., 2019], this is mainly due to the computation of GQC that is based on the number of instances l and number of abstract tasks n.

In the sequel, we present a comparison between the local selection' algorithms MSR, and DSR.

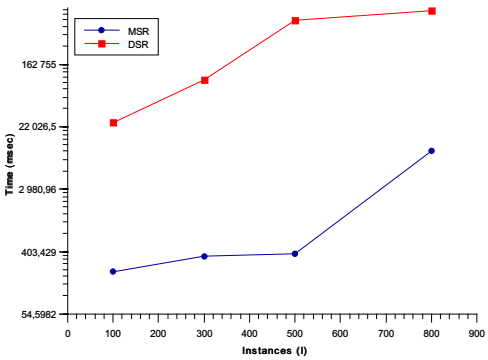

Figure 5.8: Computation time vs. l (n=5, r=3, m=100).

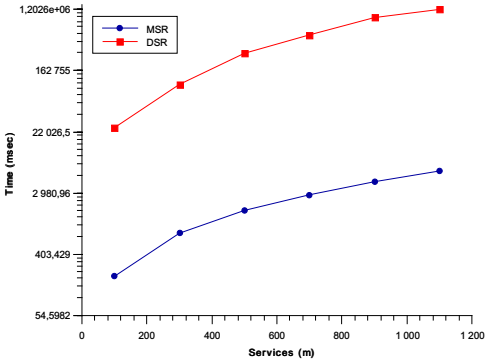

Figure 5.9: Computation time vs. m (n=5, r=3, l=100).

It can be seen from Figure 5.8, that the number of instances *l* has an impact on the computational time of DSR and MSR. Although the performance of MSR is better to that of DSR, it demonstrates a rising slope above 500 instances, this is mainly due to the sorting cost which will be prohibitive when the size of the QoS series is large.

From Figure 5.9, we can notice that time complexity is polynomial with respect to the number of services of both approaches. Nevertheless, there is a superiority of our ranking for all values of *m*. In fact, MSR can be improved to further reduce the execution time. In particular, we can use the transitivity property to prune the comparison steps; however, this pruning heuristic is not possible for DSR since we have to perform a pairwise comparison between the services to obtain the ranking score.

In the experiments of Table 5.6 , we fix the number of the services per class to 100 and vary the number of instances from 100 to 500. As depicted in Table 5.6, the GQC scores of Top

Table 5.6: GQC of Top compositions (Case study 2).

	l=100		**l=300**		**l=500**	
	k=1	**k=2**	**k=1**	**k=2**	**k=1**	**k=2**
MSR-CP	0,83	0,83	0,82	0,82	0,63	0,6328
[Hwang et al., 2015]	0,75	0,75	0,74	0,73	0,56	0,5616

composition of our approach are higher than those of [Hwang et al., 2015].

The third dataset [Hadjila et al., 2020] is adopted to compare our proposed model, termed MSR-CP with the following approaches:

- The probabilistic optimization of [Hwang et al., 2015].
- The Outlier-robust optimization of [Kim et al., 2016].
- The Probabilistic Skyline (PS) and Average Skyline (AS) heuristics are presented in [Hadjila et al., 2020].

Table 5.7: MSR-CP performance Vs. State-of-the-art methods.

	GQC	**p.s.g.c**
Probabilistic skyline of [Hadjila et al., 2020]	0.036	100%
Average skyline of [Hadjila et al., 2020]	0.025	75%
p-dominant skyline of [Hadjila et al., 2020]	0.037	75 %
[Hwang et al., 2015]	0.044	25%
[Kim et al., 2016]	0.001	25%
MSR-CP	0.044	100%

We can notice from Table 5.7 that MSR-CP is superior in terms of GQC and p.s.g.c with respect to all existing methods. Additionally, we can mention that even if MSR-CP ties with that of [Hwang et al., 2015], in terms of GQC, the p.s.g.c is still higher than that of [Hwang et al., 2015].

The modest results obtained by [Kim et al., 2016] and [Hwang et al., 2015] are mainly due to the low quality of the derived local thresholds. These experiments confirm that the optimization with global constraints is more promising than the optimization based on local thresholds.
The probabilistic skylines also show satisfying results (in terms of GQC and p.s.g.c).
Their main weakness lies in their sensitivity to outliers and noisy QoS data; therefore, the GQC of Top-K compositions will be affected. Regarding the average skyline heuristic, we observe that its performance is largely sensitive to the number of QoS attributes (r) and the sample size (l). More specifically, the larger the number of attributes, the lower the quality of results

(the curse of dimensionality), and the larger the sample size, the lower the performance (the averaging process entails an information loss).

5.4.3 Performance evaluation of the Grey Wolf based- Composition (GWC) approach

In these experiments, we compare the algorithms ESR, and DSR for the local selection of cloud services. We desire to identify the impact of the number of concurrent cloud services, and the number of instances on the cloud service selection.

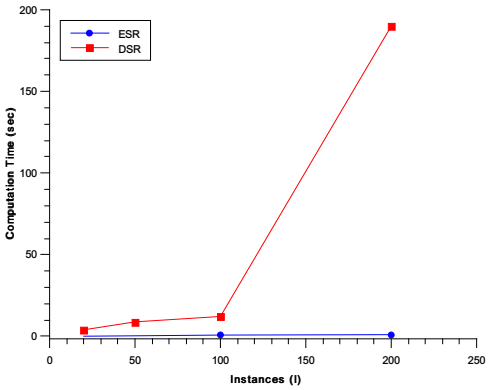

Figure 5.10: Computation time vs. l (n=10, r=3, m=300).

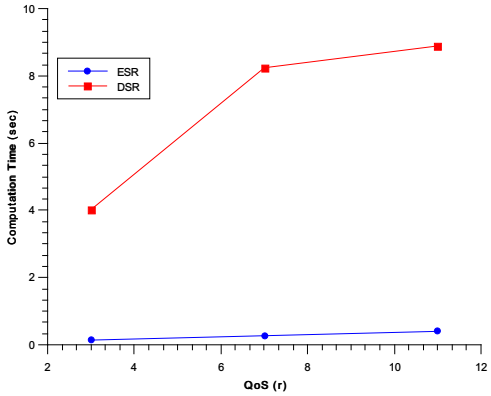

Figure 5.11: Computation time vs. r (n=10, m=500, l=20).

As shown in Figures 5.10 and 5.11, the computational time of ESR is much superior than that of DSR. It is mainly due to the pairwise comparison performed by DSR. Consequently, the complexity of DSR is $\mathrm{O}(\mathrm{nm}^2.\mathrm{r}.\mathrm{l}^2 + \mathrm{n}.\mathrm{mlogm})$, whereas the complexity of ESR is $\mathrm{O}(2.\mathrm{n}.\mathrm{m}.\mathrm{r}.\mathrm{l})$.

Afterward, we implement the experiments on the whole selection module (i.e., local selection, and global cloud service selection). To this end, We designate two meta-heuristics Particle Swarm Optimization (PSO) [Fethallah et al., 2012], and Genetic Algorithms (GA) [Canfora et al., 2005] to implement the comparison. Therefore, we diversify the number of iterations (i.e., MaxIter in algorithm 5) and fix the number of particles (i.e., $\mathrm{PopSize}$) to 100 (our search space contains 100 possible solutions for k=10 and 500^2 solutions for k=m). Following that, we set the number of iterations to 200, and we alternate the number of particles between 100 and 500 (our search space contains 25^2 possible solutions for k=25 and 500^2 solutions for k=m).

Tables 5.8, 5.9 show the GQC values according to different iterations and particles respectively. From these Tables, we deduce that the GQC value of the GWC approach with k=10 is more efficient than both GWC with k=m, PSO, and GA. This is mainly due to the local search that reduces the search space to (k).

In Figures 5.12 , 5.13 we give a comparison between our method (GWC), PSO, and GA

Table 5.8: GQC vs. number of iterations for GWC (k=10), GWC (k=m), PSO.

Iterations (MaxIter)	**GWC**		**PSO**	**GA**
	k=m	**k=10**		
100	0.22	0.29	0.17	0.15
200	0.24	0.30	0.18	0.15
300	0.29	0.31	0.20	0.14
400	0.28	0.32	0.21	0.13
500	0.28	0.32	0.22	0.16

Table 5.9: GQC vs. number of particles for GWC (k=25), GWC (k=m), PSO.

Particles (PopSize)	**GWC**		**PSO**	**GA**
	k=m	**k=25**		
100	0.23	0.28	0.18	0.15
200	0.23	0.30	0.20	0.16
300	0.28	0.30	0.21	0.15
400	0.24	0.32	0.23	0.17
500	0.28	0.32	0.24	0.16

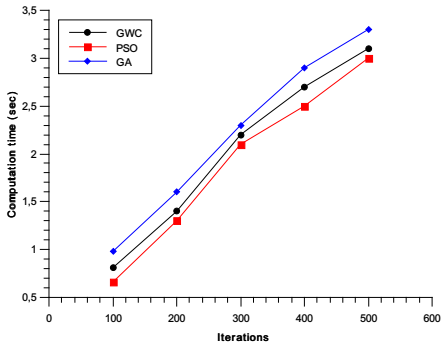

Figure 5.12: Computation time vs. number iterations.

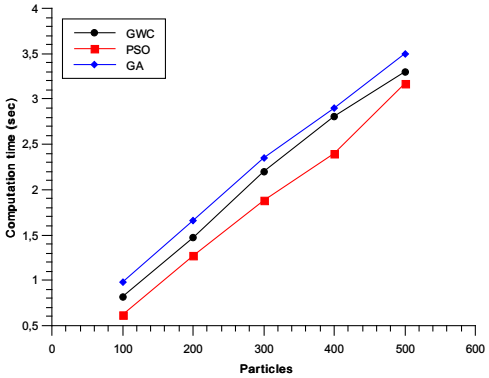

Figure 5.13: Computation time vs. number of particles.

in terms of the computation times. We can notice that the GWC approach is better than GA (about 0.4 sec) and slightly more time-consuming than PSO. The main reason is that the GWC algorithm is based on the ESR algorithm (having the complexity of $O(2.n.m.r.l)$); thus, the complexity of GWC is $O(2.n.m.r.l + PopSize.MaxIter(n + r.l^n) + PopSize * log(PopSize))$.

By analyzing the results of the first study, we notice that our approach is very encouraging.

Table 5.10: GQC and global constraint satisfiability vs. number of particles and number of services (m).

		PopSize = 100		**PopSize = 300**		**PopSize = 500**	
		GQC	**p.s.g.c**	**GQC**	**p.s.g.c**	**GQC**	**p.s.g.c**
m=100	GWC	0.92	100%	0.91	100%	0.94	100%
	IM_ABC . [Seghir et al., 2019]	0.68	66.7%	0.56	66.7%	0.45	100%
	TGA [Xu et al., 2018]	0.62	66.7%	0.67	66.7%	0.68	100%
m=300	GWC	0.92	100%	0.92	100%	0.95	100%
	IM_ABC [Seghir et al., 2019]	0.45	66.7%	0.67	66.7%	0.56	100%
	TGA [Xu et al., 2018]	0.63	66.7%	0.65	66.7%	0.66	66.7%
m=500	GWC	0.93	100%	0.95	100%	0.98	100%
	IM_ABC [Seghir et al., 2019]	0.67	66.7%	0.56	66.7%	0.58	66.7%
	TGA [Xu et al., 2018]	0.58	66.7%	0.62	66.7%	0.66	66.7%

Table 5.11: GQC and global constraint satisfiability vs. number of iterations (MaxIter) and number of services (m).

		MaxIter = 100		**MaxIter = 300**		**MaxIter = 500**	
		GQC	**p.s.g.c**	**GQC**	**p.s.g.c**	**GQC**	**p.s.g.c**
m=100	GWC	0.94	100%	0.92	100%	0.93	100%
	IM_ABC [Seghir et al., 2019]	0.66	66.7%	0.67	100%	0.67	100%
	TGA [Xu et al., 2018]	0.64	66.7%	0.54	66.7%	0.66	66.7%
m=300	GWC	0.92	100%	0.92	100%	0.93	100%
	IM_ABC [Seghir et al., 2019]	0.55	66.7%	0.57	66.7%	0.56	66.7%
	TGA [Xu et al., 2018].	0.67	66.7%	0.64	66.7%	0.64	100%
m=500	GWC	0.94	100%	0.97	100%	0.95	100%
	IM_ABC [Seghir et al., 2019]	0.65	66.7%	0.63	66.7%	0.67	66.7%
	TGA [Xu et al., 2018]	0.61	66.7%	0.57	100%	0.61	100%

Even if the execution time is slightly higher than PSO, it is still acceptable (less than 5 seconds). Furthermore, GWC shows better results in terms of GQC, since it considers all global constraints and nondeterministic QoS.

In Tables 5.10, and 5.11, we give a comparison between our proposal and TGA [Xu et al., 2018] and IM_ABC [Seghir et al., 2019] approaches according to the second case study in terms

of GQC and p.s.g.c.

From the above-mentioned tables, we observe that GWC is more effective compared to TGA and IM_ABC approaches in terms of GQC. Furthermore, we can notice that the overall global constraints are respected in our proposal according to the p.s.g.c values. We can conclude that the reduction of the search space through the elimination of irrelevant services through the Entropy Service Ranking algorithm provides significant improvement to the composition results.

5.5 Conclusion

In this chapter, we have briefly presented the implementation of our approaches including the datasets and the configurations. We also conducted a set of experiments to assess the efficiency of our approaches. The results indicate that our approaches are extremely promising for both Web service and cloud service composition.

Chapter 6

General conclusion and perspectives.

Web services constitute a key component to build distributed and complex systems. These latter systems needs multiple multiple interoperable and loosely coupled modules or services to create the desired functionality. The composition process must face several challenges, such as the exponential growth of Web services, the environment fluctuations, the QoS dynamicity and so on. The main objective of this thesis is to solve the Web service composition under uncertain QoS.

6.1 Summary

In the first chapter, we guided the reader to learn about the service oriented paradigm and its various implementations and challenges. We have presented the general principles of SOA and introduced the technology of Web services. We have also shown the two main implementations of web services, which are : the SOAP services and the REST services. Furthermore, we have addressed the cloud computing technology with a brief definition, then we highlighted the relationship between cloud computing and Web services. Thereafter, we presented a survey on the QoS-based service composition. We have detailed the main categories for solving this problem. After that, we addressed the problem of QoS uncertainty in service composition in a deeper way. we highlighted the main difficulties encountered in the related works of this issue. To handle the problem of service composition with QoS fluctuations, we firstly formulate it as a combinatorial optimization problem with global constraints, then a general framework based on constraint programming is proposed to solve the searching task. The key idea is to reduce the search space by eliminating the irrelevant services and selecting the optimal/near-optimal composition that satisfies the user's requirements (global constraints) and considering the nondeterministic nature of QoS.

In the first approach, we propose two algorithms. The first one allows for sorting the services according to the probabilistic dominance relationship, and the second one is adopted to explore the search space by using a backtracking algorithm. The second approach is a two-stage heuristic. The first one ranks the existing services based on the majority judgment heuristic. The second stage is leveraged to build up the Top-k compositions using constraint programming. The third approach is devoted to the Cloud service selection. We leverage the concept of Entropy and Cross-Entropy in a hesitant fuzzy set to sort and rank the cloud services. Afterward, we employ an improved meta-heuristic called Grey wolf-based composition GWC to select the best compositions. Our methods are objectively assessed using a collection of datasets ; these datasets follow multiple probability distributions and cover several types of workflows (see the fourth chapter). The experimental analysis and evaluation show that our approaches are effective in dealing with service composition under QoS uncertainty.

6.2 Perspectives

There are several research directions that we plan to investigate in future work.

- The service composition usually requires machine-understandable descriptions, hence, the semantic is an important concept to solve the service composition problem. Therefore, we intend to address the uncertain QoS-aware semantic Web service composition.
- In the real world, there a trade-offs between QoS attributes (they may conflict with each other). In the forthcoming works, we consider addressing the dependencies and correlations between both QoS user requirements and QoS of the web services in the composition process.
- In the context of cloud service composition, there are more specific QoS attributes and contextual aspects, that characterize the cloud services. In future work, we will consider other QoS attributes such as Disc storage performance, I/O Memory performance, and data center regions to refine the cloud composition.
- The service-oriented IoT is a recent and emerged research area. In fact, third parties such as enterprises and users intend to integrate other exposed IoT services which are usually implemented in electronic devices. Accordingly, we intend to investigate the potential of the proposed approaches in an IoT context. Likewise, we address the QoS of the

  IoT services, especially in the fluctuated environments.

Appendix A

Simulation of GQC calculation

Table A.1: Calculation of GQC

$u_1u_2u_3u_4u_5u_6$	**Instances**	**Aggregated QoS**	**Step(Aggregate ($s_{1u_11}, s_{2u_21}, s_{3u_31}, s_{4u_41}, s_{5u_51}, s_{6u_51}$), b_1)**
1 1 1 1 1 1	0.2, 0.1, 0.5, 0.2, 0.3 ,0.1	0.9	1
1 1 1 1 1 2	0.2, 0.1, 0.5, 0.2, 0.3, 0.4	1.2	1
1 1 1 1 2 1	0.2, 0.1, 0.5, 0.2, 0.6, 0.1	1	1
1 1 1 1 2 2	0.2, 0.1, 0.5, 0.2, 0.6, 0.4	1.3	1
1 1 1 2 1 1	0.2, 0.1, 0.5, 0.7, 0.3, 0.1	1.1	1
1 1 1 2 1 2	0.2 ,0.1, 0.5, 0.7, 0.3, 0.4	1.4	0
1 1 1 2 2 1	0.2, 0.1, 0.5, 0.7, 0.6, 0.1	1.1	1
1 1 1 2 2 2	0.2, 0.1, 0.5, 0.7, 0.6, 0.4	1.4	0
1 1 2 1 1 1	0.2, 0.1, 0.6, 0.2, 0.3, 0.1	1	1
1 1 2 1 1 2	0.2, 0.1, 0.6, 0.2, 0.3, 0.4	1.3	1
1 1 2 1 2 1	0.2, 0.1, 0.6, 0.2, 0.6, 0.1	1	1
1 1 2 1 2 2	0.2 ,0.1, 0.6, 0.2, 0.6, 0.4	1.3	1
1 1 2 2 1 1	0.2, 0.1, 0.6, 0.7, 0.3, 0.1	1.1	1
1 1 2 2 1 2	0.2, 0.1, 0.6, 0.7, 0.3, 0.4	1.4	0
1 1 2 2 2 1	0.2, 0.1, 0.6, 0.7, 0.6, 0.1	1.1	1
1 1 2 2 2 2	0.2, 0.1, 0.6, 0.7, 0.6, 0.4	1.4	0
1 2 1 1 1 1	0.2, 0.1, 0.5, 0.2, 0.3, 0.1	0.9	1
1 2 1 1 1 2	0.2, 0.1, 0.5, 0.2, 0.3, 0.4	1.2	1

1 2 1 1 2 1	0.2, 0.1, 0.5, 0.2, 0.6, 0.1	1	1
1 2 1 1 2 2	0.2 ,0.1, 0.5, 0.2, 0.6, 0.4	1.3	1
1 2 1 2 1 1	0.2, 0.1, 0.5, 0.7, 0.3, 0.1	1.1	1
1 2 1 2 1 2	0.2, 0.1, 0.5, 0.2, 0.3, 0.4	1.2	1
1 2 1 2 2 1	0.2, 0.1, 0.5, 0.7, 0.6, 0.1	1.1	1
1 2 1 2 2 2	0.2 ,0.1, 0.5, 0.7, 0.6, 0.4	1.4	0
1 2 2 1 1 1	0.2, 0.1, 0.6, 0.2, 0.3, 0.1	1	1
1 2 2 1 1 2	0.2, 0.1, 0.6, 0.2, 0.3, 0.4	1.3	1
1 2 2 1 2 1	0.2, 0.1, 0.6, 0.2, 0.6, 0.1	1	1
1 2 2 1 2 2	0.2 ,0.1, 0.6, 0.2, 0.6, 0.4	1.3	1
1 2 2 2 1 1	0.2, 0.1, 0.6, 0.7, 0.3, 0.1	1.1	1
1 2 2 2 1 2	0.2, 0.1, 0.6, 0.7, 0.3, 0.4	1.4	0
1 2 2 2 2 1	0.2, 0.1, 0.6, 0.7, 0.6, 0.1	1.1	1
1 2 2 2 2 2	0.2 ,0.1, 0.6, 0.7, 0.6, 0.4	1.4	0
2 1 1 1 1 1	0.5, 0.1, 0.5, 0.2, 0.3 ,0.1	1.2	1
2 1 1 1 1 2	0.5, 0.1, 0.5, 0.2, 0.3, 0.4	1.5	0
2 1 1 1 2 1	0.5, 0.1, 0.5, 0.2, 0.6, 0.1	1.3	1
2 1 1 1 2 2	0.5, 0.1, 0.5, 0.2, 0.6, 0.4	1.6	0
2 1 1 2 1 1	0.5, 0.1, 0.5, 0.7, 0.3, 0.1	1.4	0
2 1 1 2 1 2	0.5 ,0.1, 0.5, 0.7, 0.3, 0.4	1.7	0
2 1 1 2 2 1	0.5, 0.1, 0.5, 0.7, 0.6, 0.1	1.4	0
2 1 1 2 2 2	0.5, 0.1, 0.5, 0.7, 0.6, 0.4	1.7	0
2 1 2 1 1 1	0.5, 0.1, 0.6, 0.2, 0.3, 0.1	1.3	1
2 1 2 1 1 2	0.5, 0.1, 0.6, 0.2, 0.3, 0.4	1.6	0
2 1 2 1 2 1	0.5, 0.1, 0.6, 0.2, 0.6, 0.1	1.3	1
2 1 2 1 2 2	0.5 ,0.1, 0.6, 0.2, 0.6, 0.4	1.6	0
2 1 2 2 1 1	0.5, 0.1, 0.6, 0.7, 0.3, 0.1	1.4	0
2 1 2 2 1 2	0.5, 0.1, 0.6, 0.7, 0.3, 0.4	1.7	0
2 1 2 2 2 1	0.5, 0.1, 0.6, 0.7, 0.6, 0.1	1.4	0
2 1 2 2 2 2	0.5, 0.1, 0.6, 0.7, 0.6, 0.4	1.7	0

2 2 1 1 1 1	0.5, 0.1, 0.5, 0.2, 0.3, 0.1	1.2	1
2 2 1 1 1 2	0.5, 0.1, 0.5, 0.2, 0.3, 0.4	1.5	0
2 2 1 1 2 1	0.5, 0.1, 0.5, 0.2, 0.6, 0.1	1.3	1
2 2 1 1 2 2	0.5 ,0.1, 0.5, 0.2, 0.6, 0.4	1.6	0
2 2 1 2 1 1	0.5, 0.1, 0.5, 0.7, 0.3, 0.1	1.4	0
2 2 1 2 1 2	0.5, 0.1, 0.5, 0.7, 0.3, 0.4	1.7	0
2 2 1 2 2 1	0.5, 0.1, 0.5, 0.7, 0.6, 0.1	1.4	0
2 2 1 2 2 2	0.5 ,0.1, 0.5, 0.7, 0.6, 0.4	1.7	0
2 2 2 1 1 1	0.5, 0.1, 0.6, 0.2, 0.3, 0.1	1.3	1
2 2 2 1 1 2	0.5, 0.1, 0.6, 0.2, 0.3, 0.4	1.6	0
2 2 2 1 2 1	0.5, 0.1, 0.6, 0.2, 0.6, 0.1	1.3	1
2 2 2 1 2 2	0.5 ,0.1, 0.6, 0.2, 0.6, 0.4	1.6	0
2 2 2 2 1 1	0.5, 0.1, 0.6, 0.7, 0.3, 0.1	1.4	0
2 2 2 2 1 2	0.5, 0.1, 0.6, 0.7, 0.3, 0.4	1.7	0
2 2 2 2 2 1	0.5, 0.1, 0.6, 0.7, 0.6, 0.1	1.4	0
2 2 2 2 2 2	0.5 ,0.1, 0.6, 0.7, 0.6, 0.4	1.7	0

Appendix B

Academic Achievements

- **Journal Paper**
  - **Zeyneb Yasmina Remaci**, Fethallah Hadjila, and Fadoua Lahfa. Web service selection and composition based on uncertain quality of service. Concurrency and Computation: Practice and Experience, 34(1):e6531, 2022.
- **Conference Papers:**
  - **Zeyneb Yasmina Remaci**, Fethallah Hadjila, and Fedoua Didi. Selecting web service compositions under uncertain qos. In Computational Intelligence and Its Applications - 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings, volume 522 of IFIP Advances in Information and Communication Technology, pages 622–634. Springer, 2018.
  - **Zeyneb Yasmina Remaci**, Fethallah Hadjila , Abdelhak Echialli, and Mohammed Merzoug. Dynamic web service selection based on score voting. In International Conference on Computing Systems and Applications, pages 185–195. Springer, 2020.

Bibliography

[Yager, 1977] Yager, R. R. Multiple objective decision-making using fuzzy sets. *International Journal of Man-Machine Studies*, 9(4):375–382, 1977.

[Guttman, 1984] Guttman, A. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57. 1984. doi:10.1145/602259.602266.

[Mareschal et al., 1984] Mareschal, B., Brans, J. P., and Vincke, P. Promethee: a new family of outranking methods in multicriteria analysis. 1984.

[Schaffer, 1985] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, page 93–100. L. Erlbaum Associates Inc., USA, 1985. ISBN 0805804269.

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

[Hollingsworth and Hampshire, 1995] Hollingsworth, D. and Hampshire, U. Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19(16):224, 1995.

[Zadeh, 1996] Zadeh, L. A. Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific, 1996.

[Fielding, 2000] Fielding, R. T. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[Bochmann et al., 2001] Bochmann, G. v., Kerhervé, B., Lutfiyya, H., Salem, M.-V. M., and Ye, H. Introducing qos to electronic commerce applications. In *International Symposium on Electronic Commerce*, pages 138–147. Springer, 2001.

[Börzsönyi et al., 2001] Börzsönyi, S., Kossmann, D., and Stocker, K. The skyline operator. In *Proceedings 17th international conference on data engineering*, pages 421–430. IEEE, 2001. doi:10.1109/ICDE.2001.914855.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. Web services description language (wsdl) 1.1. 2001.

[Leymann, 2001] Leymann, F. *Web Services Flow Language (WSFL 1.0)*. 2001.

[Arkin, 2002] Arkin, A. Business process modeling language (bpml). *http://www. bpmi. org/bpml_prop. esp*, 2002.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[Fielding and Taylor, 2002] Fielding, R. T. and Taylor, R. N. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002. ISSN 1533-5399. doi:10.1145/514183.514185.

[Apt, 2003] Apt, K. *Principles of constraint programming*. Cambridge university press, 2003.

[Booth, 2003] Booth, D. Web services architecture. *http://www. w3. org/TR/2003/WD-ws-arch-20030808/# whatis*, 2003.

[Kalepu et al., 2003] Kalepu, S., Krishnaswamy, S., and Loke, S. W. Verity: a qos metric for selecting web services and providers. In *Fourth International Conference on Web Information Systems Engineering Workshops, 2003. Proceedings.*, pages 131–139. IEEE, 2003.

[KangChan et al., 2003] KangChan, L., JongHong, J., WonSeok, L., Seong-Ho, J., and Sang-Won, P. Qos for web services: Requirements and possible approaches. *W3C Working Group Note*, 2003.

[Kavantzas et al., 2003] Kavantzas, N., Olsson, G., Mischkinsky, J., and Chapman, M. Web services choreography description. *Language (WS-CDL)*, 1(2):3, 2003.

[Peltz, 2003] Peltz, C. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003. ISSN 0018-9162. doi:10.1109/MC.2003.1236471.

[Ran, 2003] Ran, S. A model for web services discovery with qos. *ACM Sigecom exchanges*, 4(1):1–10, 2003.

[Cardoso et al., 2004] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. Quality of service for workflows and web service processes. *Journal of web semantics*, 1(3):281–308, 2004.

[Electronic, 2004] Electronic, E. secure municipal administration for european citizens-emayor. *6th Framework Programme*, 2004.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. Qos computation and policing in dynamic web service selection. *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 66–73, 2004. doi:10.1145/1013367.1013379.

[Lopez-Alonso et al., 2004] Lopez-Alonso, V., Sanchez, J., Liebana, I., Hermosilla, I., and Martin-Sanchez, F. Inbiomed: a platform for the integration and sharing of genetic, clinical and epidemiological data oriented to biomedical research. In *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*, pages 222–226. 2004. doi: 10.1109/BIBE.2004.1317346.

[Rao and Su, 2004] Rao, J. and Su, X. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004.

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

[Liberty and Hurwitz, 2005] Liberty, J. and Hurwitz, D. *Programming ASP. NET: Building Web Applications and Services with ASP. NET 2.0*. " O'Reilly Media, Inc.", 2005.

[del Rey et al., 2005] del Rey, D. P., Crespo, J., Anguita, A., Ordóñez, J. L. P., Dorado, J., Bueno, G., Feliú, V., Estruch, A., and Heredia, J. A. Biomedical image processing integration through inbiomed: A web services-based platform. In *International Symposium on Biological and Medical Data Analysis*, pages 34–43. Springer, 2005.

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

[Al-Masri and Mahmoud, 2007] Al-Masri, E. and Mahmoud, Q. H. Discovering the best web service. *Proceedings of the 16th ACM International Conference on World Wide Web*, pages 1257–1258, 2007. doi:10.1145/1242572.1242795.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6):369–384, 2007.

[Balinski and Laraki, 2007] Balinski, M. and Laraki, R. A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences*, 104(21):8720–8725, 2007.

[Blanco et al., 2007] Blanco, P., Kotermanski, R., and Merson, P. Evaluating a service-oriented architecture. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2007.

[Cardellini et al., 2007] Cardellini, V., Casalicchio, E., Grassi, V., and Presti, F. L. Flow-based service selection forweb service composition supporting multiple qos classes. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 743–750. IEEE, 2007.

[Dehne and DiMare, 2007] Dehne, D. and DiMare, J. Service-oriented architecture-unlocking hidden value in insurance systems. *IBM Global Business Service*, 8(7), 2007.

[Papazoglou et al., 2007] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.

[Pei et al., 2007] Pei, J., Jiang, B., Lin, X., and Yuan, Y. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 15–26. Citeseer, 2007.

[Al-Masri and Mahmoud, 2008] Al-Masri, E. and Mahmoud, Q. H. Toward quality-driven web service discovery. *IT Professional*, 10(3):24–28, 2008.

[Branke et al., 2008] Branke, J., Branke, J., Deb, K., Miettinen, K., and Slowiński, R. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008.

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

[Marks and Bell, 2008] Marks, E. A. and Bell, M. *Service-oriented architecture: a planning and implementation guide for business and technology*. John Wiley & Sons, 2008.

[Papazoglou, 2008] Papazoglou, M. *Web services: principles and technology*. Pearson Education, 2008.

[Tong, 2008] Tong, K. I. K. *Developing Web Services with Apache Axis2*. Tip Tec Development, 2008.

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

[Georgakopoulos and Papazoglou, 2009] Georgakopoulos, D. and Papazoglou, M. *Service-oriented computing*. Sirsi) i9780262072960. 2009.

[Talbi, 2009] Talbi, E.-G. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[Torra and Narukawa, 2009] Torra, V. and Narukawa, Y. On hesitant fuzzy sets and decision. In *2009 IEEE international conference on fuzzy systems*, pages 1378–1382. IEEE, 2009.

[Wang and Deters, 2009] Wang, Q. and Deters, R. Soa's last mile-connecting smartphones to the service cloud. In *2009 IEEE International Conference on Cloud Computing*, pages 80–87. 2009. doi:10.1109/CLOUD.2009.73.

[Al-Shargabi et al., 2010] Al-Shargabi, B., Sabri, A., and El Sheikh, A. Web service composition survey: State of the art review. *Recent Patents on Computer Science*, 3(2):91–107, 2010.

[Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T. Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20. 2010. doi:10.1145/1772690.1772693.

[Liu et al., 2010] Liu, H., Zhong, F., Ouyang, B., and Wu, J. An approach for qos-aware web service composition based on improved genetic algorithm. In *2010 International conference on web information systems and mining*, volume 1, pages 123–128. IEEE, 2010.

[Luo et al., 2010] Luo, J., Li, W., Liu, B., Zheng, X., and Dong, F. Multi-agent coordination for service composition. In *Agent-based service-oriented computing*, pages 47–80. Springer, 2010.

[Torra, 2010] Torra, V. Hesitant fuzzy sets. *International Journal of Intelligent Systems*, 25(6):529–539, 2010.

[Yu and Bouguettaya, 2010] Yu, Q. and Bouguettaya, A. Computing service skyline from uncertain qows. *IEEE Transactions on Services Computing*, 3(1):16–29, 2010. doi: 10.1109/TSC.2010.7.

[Zheng et al., 2010] Zheng, Z., Zhang, Y., and Lyu, M. R. Distributed qos evaluation for real-world web services. In *2010 IEEE International Conference on Web Services*, pages 83–90. 2010. doi:10.1109/ICWS.2010.10.

[Ai et al., 2011] Ai, L., Tang, M., and Fidge, C. Resource allocation and scheduling of multiple composite web services in cloud computing using cooperative coevolution genetic algorithm. In *International Conference on Neural Information Processing*, volume 7063, pages 258–267. Springer, Berlin, Heidelberg, 2011. ISBN 978-3-642-24957-0. doi: 10.1007/978-3-642-24958-7_30.

[Benouaret et al., 2011] Benouaret, K., Benslimane, D., and Hadjali, A. On the use of fuzzy dominance for computing service skyline based on qos. In *2011 IEEE International Conference on Web Services*, pages 540–547. IEEE, 2011. doi:10.1109/ICWS.2011.93.

[Huang et al., 2011] Huang, L., Zhang, X., Huang, Y., Wang, G., and Wang, R. A qos optimization for intelligent and dynamic web service composition based on improved pso algorithm. In *2011 Second international conference on networking and distributed computing*, pages 214–217. IEEE, 2011.

[Jamal and Deters, 2011] Jamal, S. and Deters, R. Using a cloud-hosted proxy to support mobile consumers of restful services. *Procedia Computer Science*, 5:625–632, 2011.

[Karim et al., 2011] Karim, R., Ding, C., and Chi, C.-H. An enhanced promethee model for qos-based web service selection. In *2011 IEEE international conference on services computing*, pages 536–543. IEEE, 2011.

[Mell et al., 2011] Mell, P., Grance, T., et al. The nist definition of cloud computing. 2011.

[Xia and Xu, 2011] Xia, M. and Xu, Z. Hesitant fuzzy information aggregation in decision making. *International journal of approximate reasoning*, 52(3):395–407, 2011.

[Alrifai et al., 2012] Alrifai, M., Risse, T., and Nejdl, W. A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2):1–31, 2012. doi:10.1145/2180861.2180864.

[Benouaret et al., 2012] Benouaret, K., Benslimane, D., and Hadjali, A. Selecting skyline web services from uncertain qos. In *2012 IEEE Ninth International Conference on Services Computing*, pages 523–530. IEEE, 2012. doi:10.1109/SCC.2012.84.

[Fethallah et al., 2012] Fethallah, H., Chikh, M. A., Mohammed, M., and Zineb, K. Qos-aware service selection based on swarm particle optimization. In *2012 International Conference on Information Technology and e-Services*, pages 1–6. IEEE, 2012. doi:10.1109/ICITeS.2012.6216594.

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

[Nadanam and Rajmohan, 2012] Nadanam, P. and Rajmohan, R. Qos evaluation for web services in cloud computing. In *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*, pages 1–8. IEEE, 2012.

[Wagh and Thool, 2012] Wagh, K. and Thool, R. A comparative study of soap vs rest web services provisioning techniques for mobile host. *Journal of Information Engineering and Applications*, 2(5):12–16, 2012.

[Wang et al., 2012] Wang, L., Shen, J., and Yong, J. A survey on bio-inspired algorithms for web service composition. In *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 569–574. 2012. doi:10.1109/CSCWD.2012.6221875.

[Xu and Xia, 2012] Xu, Z. and Xia, M. Hesitant fuzzy entropy and cross-entropy and their use in multiattribute decision-making. *International Journal of Intelligent Systems*, 27(9):799–822, 2012.

[Zhao et al., 2012] Zhao, S., Wang, L., Ma, L., and Wen, Z. An improved ant colony optimization algorithm for qos-aware dynamic web service composition. *2012 International Conference on Industrial Control and Electronics Engineering*, pages 1998–2001, 2012.

[Barry and Dick, 2013] Barry, D. K. and Dick, D., editors. *Web Services, Service-Oriented Architectures, and Cloud Computing (Second Edition)*. The Savvy Manager's Guides. Morgan Kaufmann, Boston, second edition edition, 2013. ISBN 978-0-12-398357-2. doi:https://doi.org/10.1016/B978-0-12-398357-2.00019-1.

[Tan and Zhou, 2013] Tan, W. and Zhou, M. *Business and Scientific Workflows: A Web Service-Oriented Approach*, volume 5. John Wiley & Sons, 2013.

[Yu and Bouguettaya, 2013] Yu, Q. and Bouguettaya, A. Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):776–789, 2013. doi:10.1109/TKDE.2011.268.

[Bhattacharya and Chatterjee, 2014] Bhattacharya, A. and Chatterjee, T. *Goal Programming Based Multi-Objective Optimization Techniques of Task Allocation in Distributed Environment*. Lulu. com, 2014.

[Gabrel et al., 2014] Gabrel, V., Manouvrier, M., and Murat, C. Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming. In *International Conference on Service-Oriented Computing*, volume 8831, pages 108–122. Springer, Berlin, Heidelberg, 2014.

[Hadjila, 2014] Hadjila, F. Composition et interopération des services web sémantiques. *These*, 2014.

[Kyusakov, 2014] Kyusakov, R. *Efficient WEB services for end-to-end interoperability of embedded systems*. Ph.D. thesis, Luleå tekniska universitet, 2014.

[Merzoug et al., 2014] Merzoug, M., Chikh, M. A., and Fethallah, H. Qos-aware web service selection based on harmony search. In *2014 4th International Symposium ISKO-*

Maghreb: Concepts and Tools for knowledge Management (ISKO-Maghreb), pages 1–6. IEEE, 2014.

[Mirjalili et al., 2014] Mirjalili, S., Mirjalili, S. M., and Lewis, A. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[Moghaddam and Davis, 2014] Moghaddam, M. and Davis, J. G. Service selection in web service composition: A comparative review of existing approaches. *Web Services Foundations*, pages 321–346, 2014.

[Wen et al., 2014] Wen, S., Tang, C., Li, Q., Chiu, D. K. W., Liu, A., and Han, X. Probabilistic top-k dominating services composition with uncertain qos. *Service Oriented Computing and Applications*, 8(1):91–103, 2014. doi:10.1007/s11761-013-0152-4.

[Ampaporn and Gertphol, 2015] Ampaporn, P. and Gertphol, S. Performance measurement of simpledb apis for different data consistency models. In *2015 International Computer Science and Engineering Conference (ICSEC)*, pages 1–6. IEEE, 2015.

[Halfaoui et al., 2015] Halfaoui, A., Hadjila, F., and Didi, F. Qos-aware web services selection based on fuzzy dominance. In *IFIP International Conference on Computer Science and its Applications*, volume 456, pages 291–300. Springer, 2015. doi:10.1007/978-3-319-19578-0_24.

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

[Mostafa and Zhang, 2015] Mostafa, A. and Zhang, M. Multi-objective service composition in uncertain environments. *IEEE Transactions on Services Computing*, pages 1–1, 2015.

[Tripathy et al., 2015] Tripathy, A. K., Patra, M. R., and Pradhan, S. K. Dynamic qos requirement aware service composition and adaptation. In *Service-Oriented Computing-ICSOC 2014 Workshops*, pages 378–385. Springer, 2015.

[Baranwal and Vidyarthi, 2016] Baranwal, G. and Vidyarthi, D. P. A cloud service selection model using improved ranked voting method. *Concurrency and Computation: Practice and Experience*, 28(13):3540–3567, 2016. doi:10.1002/cpe.3740.

[Eswaramoorthy et al., 2016] Eswaramoorthy, S., Sivakumaran, N., and Sekaran, S. Grey wolf optimization based parameter selection for support vector machines. *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 35:1513–1523, 2016.

[Fekih et al., 2016] Fekih, H., Mtibaa, S., and Bouamama, S. User-centric web services composition approach based on swarm intelligence. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1087–1094. IEEE, 2016.

[Kim et al., 2016] Kim, M., Oh, B., Jung, J., and Lee, K. Outlier-robust web service selection based on a probabilistic qos model. *IJWGS*, 12(2):162–181, 2016. doi:10.1504/IJWGS.2016.076619.

[Serrai et al., 2016] Serrai, W., Abdelli, A., Mokdad, L., and Hammal, Y. An efficient approach for web service selection. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 167–172. IEEE, 2016. doi:10.1109/ISCC.2016.7543734.

[Sun et al., 2016] Sun, R., Zhang, B., and Liu, T. Ranking web service for high quality by applying improved entropy-topsis method. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 249–254. IEEE, 2016.

[Wang et al., 2016] Wang, Y., Song, Y., and Liang, M. A skyline-based efficient web service selection method supporting frequent requests. *20th IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2016, Nanchang, China, May 4-6, 2016*, pages 328–333, 2016. doi:10.1109/CSCWD.2016.7566009.

[Zhang et al., 2016] Zhang, S., Zhou, Y., Li, Z., and Pan, W. Grey wolf optimizer for unmanned combat aerial vehicle path planning. *Adv. Eng. Softw.*, 99:121–136, 2016. doi:10.1016/j.advengsoft.2016.05.015.

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

[Bian et al., 2017] Bian, X.-Q., Zhang, Q., Zhang, L., and Chen, J. A grey wolf optimizer-based support vector machine for the solubility of aromatic compounds in supercritical carbon dioxide. *Chemical Engineering Research and Design*, 123:284–294, 2017.

[Elsayed et al., 2017] Elsayed, D. H., Nasr, E. S., Alaa El Din, M., and Gheith, M. H. A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 537–546. Springer, 2017. doi:10.1007/978-3-319-64861-3_50.

[Paik et al., 2017] Paik, H.-Y., Lemos, A. L., Barukh, M. C., Benatallah, B., and Natarajan, A. *Web service implementation and composition techniques*, volume 256. Springer, 2017.

[Parsian et al., 2017] Parsian, A., Ramezani, M., and Ghadimi, N. A hybrid neural network-gray wolf optimization algorithm for melanoma detection. *Biomedical Research (0970-938X)*, 28(8), 2017.

[Wang et al., 2017] Wang, Y., He, Q., Ye, D., and Yang, Y. Service selection based on correlated qos requirements. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 241–248. 2017. doi:10.1109/SCC.2017.38.

[Fan et al., 2018] Fan, S.-L., Yang, Y.-B., and Wang, X.-X. Efficient web service composition via knapsack-variant algorithm. In *International Conference on Services Computing*, pages 51–66. Springer, 2018.

[Faris et al., 2018] Faris, H., Aljarah, I., Al-Betar, M. A., and Mirjalili, S. Grey wolf optimizer: a review of recent variants and applications. *Neural computing and applications*, 30(2):413–435, 2018.

[Jabbar et al., 2018] Jabbar, S., Khan, M., Silva, B. N., and Han, K. A rest-based industrial web of things' framework for smart warehousing. *The Journal of Supercomputing*, 74(9):4419–4433, 2018.

[Li et al., 2018] Li, Y., Hu, J., Wu, Z., Liu, C., Peng, F., and Zhang, Y. Research on qos service composition based on coevolutionary genetic algorithm. *Soft Computing*, 22(23):7865–7874, 2018.

[Ouadah et al., 2018] Ouadah, A., Hadjali, A., Nader, F., and Benouaret, K. Sefap: an efficient approach for ranking skyline web services. *Journal of Ambient Intelligence and Humanized Computing*, 10(2):709–725, 2018. ISSN 1868-5145.

[Permadi and Santoso, 2018] Permadi, V. A. and Santoso, B. J. Efficient skyline-based web service composition with qos-awareness and budget constraint. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 855–860. IEEE, 2018.

[Purohit and Kumar, 2018] Purohit, L. and Kumar, S. A classification based web service selection approach. *IEEE Transactions on Services Computing*, pages 1–1, 2018. ISSN 1939-1374. doi:10.1109/TSC.2018.2805352.

[Remaci et al., 2018] Remaci, Z. Y., Hadjila, F., and Fedoua, D. Selecting web service compositions under uncertain qos. In *Computational Intelligence and Its Applications - 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings*, volume 522 of *IFIP Advances in Information and Communication Technology*, pages 622–634. Springer, 2018. doi:10.1007/978-3-319-89743-1_53.

[Serrai et al., 2018] Serrai, W., Abdelli, A., Mokdad, L., and Serrai, A. How to deal with qos value constraints in mcdm based web service selection. *Concurrency and Computation: Practice and Experience*, e4512, 2018.

[Xu et al., 2018] Xu, J., Guo, L., Zhang, R., Hu, H., Wang, F., and Pei, Z. Qos-aware service composition using fuzzy set theory and genetic algorithm. *Wireless Personal Communications*, 102(2):1009–1028, 2018.

[Al-Faifi et al., 2019] Al-Faifi, A., Song, B., Hassan, M. M., Alamri, A., and Gumaei, A. A hybrid multi criteria decision method for cloud service selection from smart data. *Future Generation Computer Systems*, 93:43–57, 2019.

[Dahan et al., 2019] Dahan, F., Mathkour, H., and Arafah, M. Two-step artificial bee colony algorithm enhancement for qos-aware web service selection problem. *IEEE Access*, 7:21787–21794, 2019.

[Ghobaei-Arani and Souri, 2019] Ghobaei-Arani, M. and Souri, A. Lp-wsc: a linear programming approach for web service composition in geographically distributed cloud environments. *The Journal of Supercomputing*, 75(5):2603–2628, 2019.

[Jatoth et al., 2019] Jatoth, C., Gangadharan, G., Fiore, U., and Buyya, R. Selcloud: a hybrid multi-criteria decision-making model for selection of cloud services. *Soft Computing*, 23(13):4701–4715, 2019.

[Liang et al., 2019] Liang, X., Lu, Q., and Li, M. Research on web service selection based on improved skyline algorithm. In *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 1323–1328. IEEE, 2019.

[Naseri and Navimipour, 2019] Naseri, A. and Navimipour, N. J. A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 10(5):1851–1864, 2019.

[Seghir et al., 2019] Seghir, F., Khababa, A., and Semchedine, F. An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain qos. *The Journal of Supercomputing*, 75(9):5622–5666, 2019.

[Benomar et al., 2020] Benomar, Z., Longo, F., Merlino, G., and Puliafito, A. Enabling secure restful web services in iot using openstack. In *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 410–417. IEEE, 2020.

[Hadjila et al., 2020] Hadjila, F., Belabed, A., and Merzoug, M. Efficient web service selection with uncertain qos. *International Journal of Computational Science and Engineering*, 21(3):470–482, 2020.

[Haytamy and Omara, 2020] Haytamy, S. and Omara, F. Enhanced qos-based service composition approach in multi-cloud environment. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 33–38. IEEE, 2020.

[Hosseinzadeh et al., 2020] Hosseinzadeh, M., Tho, Q. T., Ali, S., Rahmani, A. M., Souri, A., Norouzi, M., and Huynh, B. A hybrid service selection and composition model for cloud-edge computing in the internet of things. *IEEE Access*, 8:85939–85949, 2020.

[Li et al., 2020] Li, C., Li, J., and Chen, H. A meta-heuristic-based approach for qos-aware service composition. *IEEE Access*, 8:69579–69592, 2020.

[Ranjan and Sahoo, 2020] Ranjan, A. and Sahoo, B. Web service selection mechanism in service-oriented architecture based on publish-subscribe pattern in fog environment. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 269–281. Springer, 2020.

[Remaci et al., 2020] Remaci, Z. Y., Hadjila, F., Echialli, A., and Merzoug, M. Dynamic web service selection based on score voting. In *International Conference on Computing Systems and Applications*, pages 185–195. Springer, 2020.

[Youssef, 2020] Youssef, A. E. An integrated mcdm approach for cloud service selection based on topsis and bwm. *IEEE Access*, 8:71851–71865, 2020.

[Barkat et al., 2021] Barkat, A., Kazar, O., and Seddiki, I. Framework for web service composition based on qos in the multi cloud environment. *International Journal of Information Technology*, 13(2):459–467, 2021.

[Dahan, 2021] Dahan, F. An effective multi-agent ant colony optimization algorithm for qos-aware cloud service composition. *IEEE Access*, 9:17196–17207, 2021. doi:10.1109/ACCESS.2021.3052907.

[Dahan et al., 2021] Dahan, F., El Hindi, K., Ghoneim, A., and Alsalman, H. An enhanced ant colony optimization based algorithm to solve qos-aware web service composition. *IEEE Access*, 9:34098–34111, 2021.

[Polska et al., 2021] Polska, O., Kudermetov, R., Alsayaydeh, J. A. J., and Shkarupylo, V. Qos-aware web-services ranking: Normalization techniques comparative analysis for lsp method. *ARPN Journal of Engineering and Applied Sciences*, 16(2):248–254, 2021.

[Prakash and Kumar, 2021] Prakash, J. and Kumar, T. V. A multi-objective approach for materialized view selection. In *Research Anthology on Multi-Industry Uses of Genetic Programming and Algorithms*, pages 512–533. IGI Global, 2021.

[Thangaraj and Balasubramanie, 2021] Thangaraj, P. and Balasubramanie, P. Meta heuristic qos based service composition for service computing. *Journal of Ambient Intelligence and Humanized Computing*, 12(5):5619–5625, 2021.

[Tiwari and Kumar, 2021] Tiwari, R. K. and Kumar, R. G-topsis: a cloud service selection framework using gaussian topsis for rank reversal problem. *The Journal of Supercomputing*, 77(1):523–562, 2021.

[Zeyneb Yasmina et al., 2022] Zeyneb Yasmina, R., Fethallah, H., and Fadoua, L. Web service selection and composition based on uncertain quality of service. *Concurrency and Computation: Practice and Experience*, 34(1):e6531, 2022.