OPTIMAL QOS-AWARE MULTIPLE PATHS WEB SERVICE COMPOSITION USING HEURISTIC ALGORITHMS AND DATA MINING TECHNIQUES

OSAMA KAYED TAHER QTAISH

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2014

Permission to Use

Abstrak

Matlamat penggubahan perkhidmatan sedar-QoS (kualiti perkhidmatan) adalah untuk menjana perkhidmatan gabungan yang memenuhi keperluan QoS yang ditetapkan oleh pelanggan. Walau bagaimanapun, adalah sukar untuk menjana satu perkhidmatan gabungan yang dapat mengoptimakan semua laluan yang terlibat dengan serentak apabila penggabungan tersebut mempunyai lebih daripada satu laluan pelaksanaan. Pada masa yang sama juga penggabungan itu mesti memenuhi keperluan QoS. Ini adalah masalah yang dikaji dalam penyelidikan ini, yang juga dikenali dengan masalah pengoptimuman. Cabaran lain ialah untuk menetapkan ciri QoS yang boleh dikelaskan sebagai kriteria pemilihan. Thesis ini mengusulkan kaedah penggubahan perkhidmatan sedar-QoS. Matlamatnya adalah untuk menyelesaikan masalah di atas melalui mekanisma pengoptimuman berdasarkan kombinasi kaedah jangkaan laluan masa larian dan algoritma heuristik. Mekanisma ini melibatkan dua langkah. Pertama, kaedah jangkaan laluan pelaksanaan yang menjangka laluan pelaksanaan yang mempunyai potensi untuk dilaksanakan, seketika sebelum pelaksanaan penggubahan sebenar dibuat. Kedua, prosedur konstruktif (CP) dan prosidur pelengkap (CCP) dalam algorithma heuristik digunakan untuk menghitung pengoptimuman dengan mengambil kira hanya laluan pelaksanaan yang telah dijangka oleh kaedah jangkaan laluan masa larian. Untuk kriteria pemilihan, lapan ciri QoS diusulkan selepas menganalisis hasil penyelidikan terdahulu. Seterusnya, diusulkan juga supaya kriteria terpilih tersebut disusun mengikut keutamaan bagi memudahkan pelanggan membuat pilihan. Ujikaji melalui alatan WEKA dan prototaip digunakan untuk membuat simulasi bertujuan menilai kedua-dua kaedah yang digunakan. Bagi kaedah jangkaan laluan masa larian, keputusan menunjukkan kaedah ini dapat mencapai tahap ketepatan jangkaan yang memberasangkan dan ketepatan tersebut pula tidak dipengaruhi oleh bilangan laluan yang terlibat dalam jangkaan. Bagi mekanisma pengoptimuman, penilaian dijalankan dengan membandingkan mekanisma ini dengan teknik pengotimuman yang relevan. Hasil simulasi menujukkan bahawa mekanisma pengoptimuman yang dicadangkan mengalahkan teknik lain kerana ia dapat (1) menjana penyelesaian nisbah QoS tertinggi, (2) menggunakan masa pengkomputeran yang terendah, dan (3) menghasilkan peratusan terkecil bagi bilangan kekakangan yang dicabuli.

Kata kunci: Penggubahan perkhidmatan web, QoS, Pemilihan perkhidmatan, Algorithma heuristik, Perlombongan data.

Abstract

The goal of QoS-aware service composition is to generate optimal composite services that satisfy the QoS requirements defined by clients. However, when compositions contain more than one execution path (i.e., multiple path's compositions), it is difficult to generate a composite service that simultaneously optimizes all the execution paths involved in the composite service at the same time while meeting the QoS requirements. This issue brings us to the challenge of solving the QoS-aware service composition problem, so called an optimization problem. A further research challenge is the determination of the QoS characteristics that can be considered as selection criteria. In this thesis, a smart QoS-aware service composition approach is proposed. The aim is to solve the above-mentioned problems via an optimization mechanism based upon the combination between runtime path prediction method and heuristic algorithms. This mechanism is performed in two steps. First, the runtime path prediction method predicts, at runtime, and just before the actual composition, execution, the execution path that will potentially be executed. Second, both the constructive procedure (CP) and the complementary procedure (CCP) heuristic algorithms computed the optimization considering only the execution path that has been predicted by the runtime path prediction method for criteria selection, eight QoS characteristics are suggested after investigating related works on the area of web service and web service composition. Furthermore, prioritizing the selected QoS criteria is suggested in order to assist clients when choosing the right criteria. Experiments via WEKA tool and simulation prototype were conducted to evaluate the methods used. For the runtime path prediction method, the results showed that the path prediction method achieved promising prediction accuracy, and the number of paths involved in the prediction did not affect the accuracy. For the optimization mechanism, the evaluation was conducted by comparing the mechanism with relevant optimization techniques. The simulation results showed that the proposed optimization mechanism outperforms the relevant optimization techniques by (1) generating the highest overall QoS ratio solutions, (2) consuming the smallest computation time, and (3) producing the lowest percentage of constraints violated number.

Keywords: Web service composition, QoS, Service selection, Heuristic algorithm, Data mining.

Acknowledgement

All thanks to Allah who gave me the ability to finish this work. Without the mercy of Allah, I could not achieve anything. I would like to gratefully acknowledge the enthusiastic supervision of my thesis supervisor, Professor Dr. Zulikha Bt Jamaludin and my Co-supervisor Dr. Massudi bin Mahmuddin. I could not have imagined having a better adviser and mentor for my Ph.D. Without their inspiration, stimulating suggestions, sound advice, guidance, and active participation throughout the process of work, I would never have finished.

I am truly and deeply indebted to so many people that there is no way to acknowledge them all or even any of them properly. I am grateful to my lovely wife, Dua'a. I'll forever be indebted to this great woman. Very special thanks to your practical and emotional support. Without your endless patience, understanding, support, and help, I would never have been able to finish this thesis. Your supreme trust is always the most efficient motivation to accomplish my ultimate goal. No word can describe what you have done for me. I love you. I would like to give special thanks to my cute son Yamn, who came to this life while I was studying for my PhD. His gorgeous smile blessed and encouraged me to continue my studies.

I would like to thank my Father, who passed away during my PhD study, for his support and encouragement to continue my study. I know his soul is very happy about my successful achievement. Then I would like to thank my beloved mother, who is always supporting me and praying for me to obtain this degree. I respect her deep faith, unconditional love, and support at each time of my life made me this man whom I am today. Furthermore, I would like to thank my stepfather Dr. Abdul Raheem Issa and stepmother for their support. Warm thanks go to my dearest brothers Mohammed and Emad, sisters Suhad, Sahar, Raghda, Wafa' and Abeer, and friends, especially Dr. Yousif Faza, Dr. Ammar Yasser, Dr. Mossab Al Hunaity, Wael Abo Rahma, Mohammed Eriqat, and Bashar Barakat for giving me their unequivocal support throughout, as always, for which my mere expression of thanks, likewise, does not suffice.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

ACO	Ant Colony Optimization
ARFF	Attribute Relation File Format
BPI	Business Process Intelligence
BPMS	Business Process Management System
CACO	Continuous Ant Colony Optimization
CIAC	Continuous Interacting Ant Colony
CCP	Complementary Constructive Procedure
CP	Constructive Procedure
CSV	Comma Separated Value
DAG	Directed Acyclic Graph
ESGA	Elitist Selection Genetic Algorithm
FN	False Negative
FP	False Positive
FS	Feasible State
GAELS	Genetic Algorithm Embedded Local Searching
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
HGA	Hybrid Genetic Algorithm
HR	Harmony Research
IDE	Integrated Development Environment
ILP	Integer Linear Programming
IP	Integer Programming
ISO	International Organization for Standardization
IT	Information Technology
MCDM	Multiple Criteria Decision Making
MCOP	Multi-Constraint Optimal Path
MILP	Mixed Integer Linear Programming
MMKP	Multi-dimensional Multi-choice Knapsack Problem
NB	Naïve Base
NP	Non-deterministic Polynomial-time
OASIS	Organization for the Advancement of Structured Information Standards

PACA	Particle-Ant Colony Algorithm
PAIS	Process-Aware Information System
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
QP	Quadratic Programming
QQDSGA	Quality of Experience (QoE)/Quality of Service (QoS) Driven Simulated Annealing-based Genetic Algorithm
SA	Simulated Annealing
SAW	Simple Additive Weight
SLA	Service Level Agreement
SMO	Sequential Minimal Optimization
SN	Solution
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
TS	Tabu Search
UDDI	Universal Description Discovery and Integration
URL	Uniform Resource Locator
US	Unfeasible State
W3C	World Wide Web Consortium
WEKA	Waikato Environment For Knowledge Analysis
WFMS	Workflow Management System
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Service Description Language
WSQM	Web Service Quality Model
XML	Extensible Markup Language

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Service Oriented Computing (SOC) recently has gained a considerable momentum from both industry and academia as a new emerging paradigm to develop rapid, low cost, and loosely coupled software systems. This vision is captured by Service Oriented Architecture (SOA) through the provision of an architectural style (Michlmayr, Rosenberg, Platzer, Treiber & Dustdar, 2006). SOA is "a way of designing a system so that it can provide services to end users and/or other applications in the network" (Baryannis et al., 2008).

The SOA model illustrated in Figure 1.1 consists of three core entities: service provider, service consumer (also called requester), and service registry. The service provider implements the web service and describes it using a standard format. And then it publishes the description in the service registry. The service consumer queries the registry about a specific web service. The service registry checks, whether the requested web service is available or not. If it is available, the registry returns descriptions of the matched web services back to the service consumer. The service consumer obtains the location of the selected web service from the returned descriptions. Finally, the service consumer binds and invokes the web service.

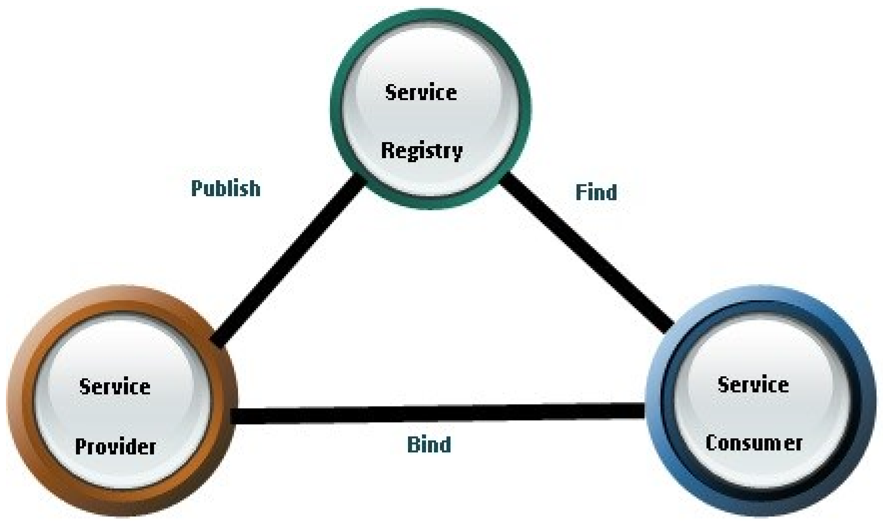


Figure 1.1. *SOA model (Source: Newcomer & Lomow, 2004)*

Web services are published by service providers (i.e., organizations that provide service descriptions and ensure service implementations), located, and invoked by clients (requesters). Web services refer to a special type of services which are provided by computer systems and are supposed to provide their functionality in computer networks such as the Internet. Clients can utilize web services through the Internet without the need to install it (Jaeger, 2007; Hilari, 2009). Many companies have deployed web services recently in order to provide them for individual customers or business which in turn can integrate these web services into their systems. Google, for example, provides many web services for customers who can simply integrate these web services into their applications. Google Maps API Web Services are an example of such services (https://developers.google.com/maps/documentation/webservices/).

The benefit gained from implementing SOA is the ability to compose new functionality out of existing outsourced web services into the so-called composite services. The process of creating such composite services is called a web service composition (Rosenberg, Leitner, Michlmayr, Celikovic & Dustdar, 2009). A travel agency web service is an example of a composite service that consists of many web services such as hotel reservation, airline booking, and car rental services.

Two applications are considered for web service composition technology; the first case targets the development of service-oriented software systems. In this case, complex software can be created by discovering and integrating individual web services. For this aim, the World Wide Web Consortium (W3C) recommends developing web services using a set of Extensible Markup Languages (XML) and Internet protocols. Specifically, the W3C provides a set of specifications required to develop web services which include the use of standard protocols such as Universal Description Discovery and Integration (UDDI) (Clement, Hately, Riegen & Rogers, 2004) for publishing the web services, Web Service Description Language (WSDL) (Booth & Liu, 2006) for descriptions of the services, and Simple Object Access Protocol (SOAP) (Mitra & Lafon, 2006) to exchange messages between the services. The W3C (Austin, Daniel, Ferris & Garg, 2004) defines the web service as:

> A software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

The second case targets the application of web service composition technology to build business processes that entirely performing within computer systems. The next subsection discussed this application of web service composition technology in detail.

1.1.1 Web Service Composition Technology for Building Business Processes

Workflow Management Systems (WFMSs) are employed currently by organizations to define, manage, and execute their business processes. These systems are often called Process-Aware Information System (PAIS) since they need to be aware of the processes in the context of their organizations (van der Aalst, 2009). Business process is a set of related tasks or activities that are designed to realize a specific organizational goal.

Nowadays, many existing organizations, that offer similar products and services, are operating in markets. In such markets, to remain competitive, organizations are required to perform their business processes effectively regarding cost and quality, and to quickly adapt to business's needs. In achieving this purpose, flexibility and performance play major roles. However, this flexibility can be hardly achieved within the current organization IT architectures, where programming languages, heterogeneous legacy systems, operating systems, and middleware platforms are predominating these architectures (Schuller, Eckert, Miede, Schulte & Steinmetz, 2010). SOA along with service composition technology has emerged as a solution for organizations seeking to increase the flexibility. In fact, they changed the way of building business processes. Rather than developing entirely new processes, SOA processes are developed by composing network available web services (Dustdar & Papazoglou, 2008). Each task (also referred to as an abstract web service) of such a business process can be accomplished by a single outsourced web service hosted by external partners. Under this scenario, complex applications are defined as business processes (hereafter used interchangeably with terms "composite service" and "composition") composed of abstract web services (an abstract web service is a description of a specific functionality in abstract fashion), and the service selection can be performed dynamically at runtime by selecting the best outsourced services that can accomplish the abstract services functionality (Ardagna & Pernici, 2006). This vision enables agile collaborations between several business partners, and thus, it decreases the cost of building business processes.

Creating business processes using web service composition can be performed statically at design time. In this approach, composition developers choose a set of available web services that is related to their business processes, and program the interaction between them by any low level programming languages such as Web Services Business Process Execution Language (WS-BPEL) (Jordan et al., 2007). Commercial orchestration engines can be used to execute WS-BPEL coded business processes (Ko, Kim & Kwon, 2008).

Although this ad hoc way of building business processes is supported by major IT companies like IBM and Microsoft, such an approach is inappropriate due to the dynamic and flexible nature of web services environments, where suddenly existing web services may be removed or other new web services become available (Dustdar & Schreiner, 2005). Therefore, outsourced web services should be automatically discovered, selected, and bound at runtime. To achieve this purpose, several approaches propose applying semantic web concepts to web services in order to enable dynamic, runtime discovery, selection, composition, and invocation of web services (Martin et al. 2005; Cardoso & Sheth, 2003; Burstein et al., 2005).

Several researches have been introduced to leverage network available web services to build workflows (Bolcer & Kaiser, 1999; Ganesarajah & Lupu, 2002; Hull, Benedikt, Christophides & Su, 2003; Patel, Supekar & Lee, 2004). These research works indicate the consensus that software engineers can utilize web service composition technology as technical foundations to implement workflows (Jaeger, 2007). The approach proposed can be associated and embedded to any workflow-based web service composition systems that rely on web service composition technology to build business processes such as the previously mentioned works.

1.1.2 QoS-Aware Service Composition

Recently, due to the continued proliferation of web services, hundreds of functionality equivalent web services are expected to exist, creating an issue on selection criteria, namely which service should be selected? And why? One of the most substantial selection factors that are used to distinguish between those equivalent services is the Quality of Service (QoS) criteria. QoS, which represents web service's non-functional characteristics such as cost, response time, availability, reliability, reputation, throughput, security, and composability, can serve as selection criteria. One may choose the lowest cost of web service, the fastest response time or a compromise between the two. QoS denotes how well services provide their functionality (Jaeger, 2007). Without QoS, no organizations want to rely on external web services to perform their business processes (Berbner, Heckmann & Steinmetz, 2005). When developing their business processes, organizations have the opportunity to select those outsourced web services that satisfy their QoS requirements. These requirements include QoS global constraints and preferences. QoS global constraints are constraints imposed by the clients in the whole business process. For example, a client could specify that the total cost of the composite service execution must be less than 2000 Dollars. At the same time, he/she could prefer the composite service with high security and/or low response time.

Generally, the process of composing services based on clients QoS requirements as illustrated in Figure 1.2 includes several phases: design, discover, select, bind, and execute/monitor.

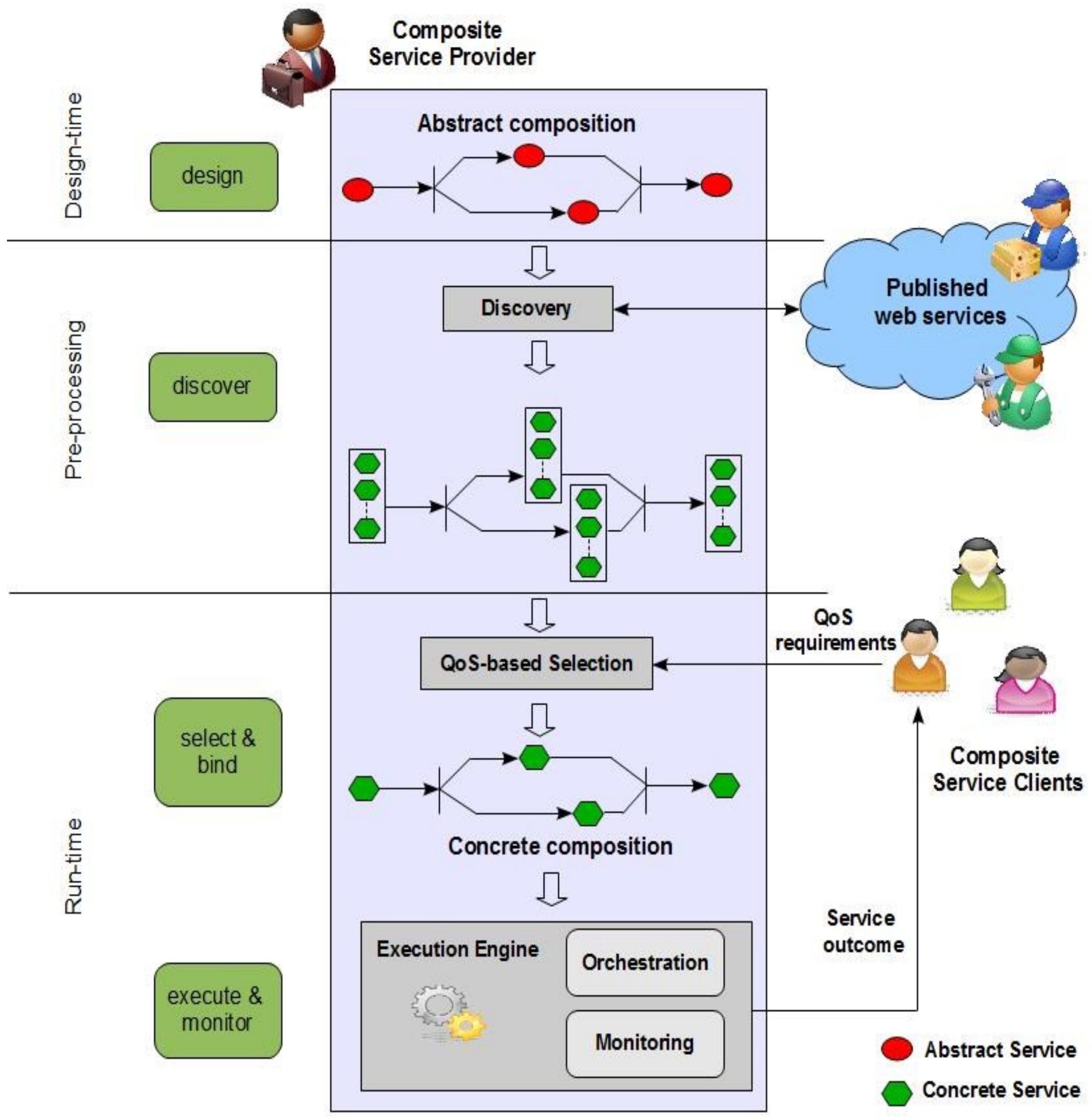


Figure 1.2. *Web service composition phases*

In the first phase, during the design time, a composition developer defines the composition by identifying and arranging the abstract services (e.g. invoking a credit card) that can be matched to the outsourced web services. Several flow languages are used for defining such an arrangement, for example, WS-BPEL (Jordan et al., 2007). In the second phase, based on the semantic descriptions of the abstract services, many functionality equivalent web services (called candidates or concrete) with different QoS characteristic values can be discovered for each abstract web service.

These candidate services are offered by different providers. Then the role of the selection phase is to select one candidate web service for replacing each abstract web service such that the entire QoS of the composition are optimized while client's QoS requirements are satisfied. This process is referred to as QoS-aware web service composition. After that, each selected outsourced service is assigned to its corresponding abstract service. This assignment, in turn, is saved in a flow description which is used by the system in order to execute the composition. Finally, the system tracks the composition executions and monitors the quality (Jaeger, Muhl & Golze, 2005; Jafarpour & Khayyambashi, 2010).

This thesis turns the attention to the selection phase and focuses mainly on the QoS-aware service composition process on the basis of workflow composition technology, and the proposed approach, as mentioned earlier, can be associated with any workflow-based service composition systems.

1.2 **Problem Background**

The goal of QoS-aware service composition process is to select one candidate web service for each abstract web service from its corresponding list of candidates such that the entire QoS of the composition is optimized while QoS requirements defined by clients are satisfied (Yu, Zhang & Lin, 2007; Canfora, Penta, Esposito & Villani, 2005; Zeng, Benatallah, Dumas, Kalagnanam & Sheng, 2003; Zeng et al., 2004; Alrifai, Risse, Dolog & Nejdl, 2009).

Composite services are defined using several composition structures such as sequential, parallel, loop and conditional structures. These structures are used to connect the abstract web services that constitute the compositions. Figure 1.3 illustrates a composition defined by using sequential and conditional structures. If a composition contains a conditional structure, it has multiple execution paths. These multiple paths are all represented by a single composition. The composite service illustrated in the Figure 1.3 has three different paths: $Path_1$, $path_2$, and $Path_3$. At the execution time, three different possible execution scenarios may occur.

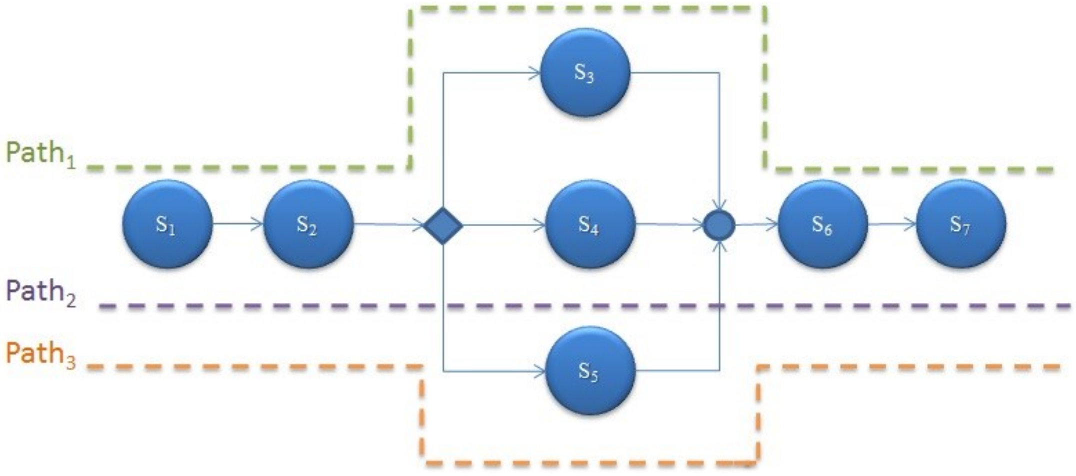


Figure 1.3. *An example of multiple paths composition*

On the other hand, an optimal composition solution is the solution that delivers the desired extreme value of the objective function while meeting client's QoS requirements. The objective function is a QoS statement of the value of any given composition solution (Ukor & Carpenter, 2008). Normally, the objective functions are defined as an aggregation of the QoS characteristics for each of the selected candidate services for all the abstract services participating in the composition. The QoS statement is used in the comparison of different composition solutions. The solution which has the maximum value of the objective functions is considered the optimal solution. Consequently, QoS statements are critical to be as precise as possible.

In the absence of conditional structure in a composition (i.e., there is only one path involved in the composition), it is certain that the only single path will be taken by all the composition instances during the execution (i.e., all web services involved in the composition will be executed by all composition instances). This certainly makes the QoS statements computed by the aggregation methods always precise. And then an algorithm can efficiently compare the QoS statements of the solutions. As a result, it is guaranteed that the generated solutions will be always optimal.

However, this is not the case in multiple paths compositions, where different paths (i.e., Subset of web services) can be executed because the multiple paths compositions are non-deterministic. For example, the composite service illustrated in Figure 1.3 can be executed many times, and several executions of the composite service can use different execution paths. It is impossible before the composition, executed to determine which path will be executed. Therefore, two techniques are proposed to handle this problem. The first technique considers all paths together for optimization. However, it's difficult for the objective functions provide precise QoS statements that represent only the subset of the services that will be executed. As a result, there is a possibility to generate suboptimal solutions for some execution paths.

In the second technique, the optimization is computed considering each execution path separately. If there is a conflict in service selection in some abstract services that are common to multiple execution paths, the system identifies the hot path for the considered web service. The hot path is defined as the path that has been most frequently used to execute the considered service. However, in the case that the actual execution of the composition is not following the hot path, the executed path may not have the best QoS ratio, worse than that, the executed path may violate QoS requirements.

1.3 **The Problem Statement**

The problem background section has explained the problem in detail. Based on this, the problem statement of this thesis is formulated as the following:

In multiple paths compositions, it is difficult for the optimization algorithms to generate a solution that simultaneously optimizes all the execution paths involved in the composition at the same time while meeting clients QoS requirements. Hence, the existing optimization techniques compute the optimization either by considering all execution paths together (Yu, Zhang & Lin, 2007; Canfora, Penta, Esposito & Villani, 2005; Jiang, Yang, Yin, Zhang & Cristoforo, 2011; Jaeger et al., 2004; Jafarpour & Khayyambashi, 2010; Parejo, Fernandez & Cort´es, 2008; Schuller, Polyvyanyy, García-Bañuelos & Schulte, 2011; Ko et al., 2008; Ukor & Carpenter, 2008, 2009; Singh, 2012) or by optimizing each path separately (Zeng et al., 2003, 2004; Zhang, Chang, Feng & Jiang, 2010; Liu, Wu & Liu, 2012). However, it is difficult for the optimization algorithms to generate a solution that simultaneously optimizes all the execution paths involved in the composition at the same time while meeting clients QoS requirements. Consequently, the solutions generated using the above-mentioned optimization techniques are suboptimal for some execution paths, worse than that; the generated solutions have high constraints violated number.

From the main problem mentioned above, the following sub problems are derived:

- **The Optimization Problem**

As mentioned earlier, QoS-aware service composition process aims to select one outsourced candidate web service for each abstract web service from its corresponding list of candidates such that the entire QoS of the composition is optimized while QoS requirements, defined by clients, are satisfied (Yu, Zhang & Lin, 2007; Canfora et al., 2005; Zeng et al., 2003, 2004; Alrifai et al., 2009). During this process, there is possibly a numerous number of composition plans (solutions) that could be generated. The objective is to choose the best (in terms of QoS) solution which leads to the best quality of the composition and meets clients' QoS requirements (Alrifai et al., 2009). Finding exact optimal solutions required a strategy based on evaluating all the possible combinations to find the optimal one. Such a straightforward strategy takes a significant amount of time and effort to find the optimal solution among a huge number of possible solutions. For example, for a composition with 5 abstract services and 50 candidates, the number of possible combinations to evaluate is 50^5. Furthermore, any increase in the number of candidates will dramatically increase the possible combinations to be evaluated (Jaeger, 2007). It is impractical and time consuming to evaluate all these combinations to find the optimal one. Such a straightforward strategy for finding optimal solutions is inappropriate for real time decision-making applications. Due to its high computational complexity, approaches that deliver exact optimal solutions are inappropriate for real time decision-making applications. Thus, heuristic represents a novel approach. Therefore, a new and powerful heuristic-based optimization strategy is needed to be applied to solve the optimization problem.

- **Prediction of Execution Paths**

In multiple paths compositions, it is impossible before the composition, executed to determine which path will be executed. Thus, there is a need to predict the path that will potentially be executed in order to focus only on the predicted path during the optimization process. If one can predict, just before the actual composition executions, with a certain degree of confidence the path that will be potentially executed, and then this useful information can be utilized by an optimization algorithm in order to optimize only the predicted path. Then, the precision of the QoS-statements can increase significantly. Consequently, it is guaranteed that the generated composition solutions are having the best possible QoS ratio. In addition, the constraints violated number will be significantly reduced.

- **QoS Characteristics for Web Service Composition**

Most of the existing research efforts in the area of web service composition are considering a set of general QoS characteristics (Zeng, et al., 2003, 2004; Alrifai, Skoutas & Risse, 2010; Canfora et al., 2005; Yu et al., 2007; Zhang et al., 2010; Jiang et al., 2011; Schuller et al., 2011; Singh, 2012). There is a need to analyze the QoS characteristics that were most commonly used by researchers' works in order to determine the relevant set of QoS characteristics that can be considered selection criteria when composing web services. The analysis should answer the following question: what QoS characteristics are appropriate for a web service composition? In this analysis, it is important to take into consideration, when determining the relevant set of QoS characteristics for composing web services, the features of composite services which are different from features of single web services. It is needed to derive the QoS characteristics from the special features of composite services.

Another issue is to assist clients (i.e., organizations) in choosing between multiple criteria. When multiple criteria are considered at once for optimization, clients might have difficulties in choosing the right criteria. Therefore, prioritizing the selected QoS criteria is needed to be suggested in order to assist clients when choosing the right criteria.

1.4 **The Motivation**

In global competitive markets, where organizations operate, the most important goal for organizations is to increase their competitive ability. To achieve this purpose, organizations rely on web service composition technology for developing their processes. In this way, business processes are developed by composing network available web services (Dustdar & Papazoglou, 2008). Each function of such processes can be accomplished by a single outsourced web service hosted by external partners. Web services can be automatically discovered and composed to create a more complex business process. This vision enables agile collaborations between several business partners.

However, the performance of business process is important due to the flexible and dynamic nature of the web service. It is important for organizations to build their business processes according to the QoS characteristics. For example, a bank loan process may need a quick response to a loan requester. On the other hand, critical business processes' functions may suffer from failures or loss without careful quality management. Failure or delay of even one outsourced web service that participates in the business process will result in failure or delay of the whole process, which directly will impact the success of the organization. Therefore, without QoS guarantee of the selected outsourced web services, no organizations want to rely on external web services to achieve their goals (Berbner et al., 2005).

When building business processes, organizations have the opportunity to choose those outsourced web services that satisfy their QoS requirements. In this case, it is essential for organizations to receive what they have requested by meeting their QoS requirements.

1.5 **Research Questions**

With regards to the problems delineated above, the following questions need to be probed further so that the objectives of the study can be achieved.

RQ1: How to derive an approach for multiple paths QoS-aware service composition in order to solve the multiple paths composition problem and the optimization problem?

RQ2: How an optimization mechanism be proposed to generate the best possible QoS ratios solutions within small computation time while significantly reducing the constraints violated of the generated solutions?

In specific, this research question comprises of two other sub questions that are related to proposing an optimization mechanism.

a. Can a runtime path prediction method be proposed to predict, at runtime, and just before the actual composition executions, the path that will be potentially executed based on the information provided by composition requesters?

b. Can heuristic optimization algorithms to be applied to efficiently solve the QoS-aware composition problem?

RQ3: Given 25 QoS characteristics highlighted in previous studies, which characteristics are appropriate for a web service composition?

RQ4: How can the new proposed approach be evaluated?

For this, broad question, two particular evaluations will be focused, i.e. the evaluation of the path prediction method and on the optimization mechanism. Thus, the sub questions are as follows:

a. Can the runtime path prediction method be evaluated in order to determine its accuracy and scalability?

b. Can the optimization mechanism be evaluated in terms of the QoS ratio, the constraints violated number, and the computation time?

1.6 **Research Objectives**

The main goal of this research is to develop an approach for QoS-aware service composition that is designed to solve the multiple paths composition problem and the optimization problem. This approach aims at generating a solution within a small computation time (i.e., 0.0022 s on average, based on the test experiments performed in this work) that delivers the best possible QoS ratio. Moreover, this approach aims at significantly reducing the constraints violated a number resulted from the generated solutions.

In order to achieve this, research goal, the following research objectives were formulated:

1. To propose an approach for multiple paths QoS-aware service composition in order to solve the multiple paths composition problem and the optimization problem.

2. To propose a new optimization mechanism based on the runtime path prediction method and heuristic optimization algorithms, in order to predict the path that will be executed at runtime and then optimize the predicted path.

3. To identify the appropriate QoS characteristics that can be considered as selection criteria for optimal service composition process.

4. To evaluate the proposed approach by evaluating (1) the path prediction method in terms of its accuracy and scalability using data mining tools, and

(2) the optimization mechanism in terms of the QoS ratio, the constraints violated number, and the computation time by conducting test experiments using the simulation prototype developed in chapter six.

1.7 **Research Scope**

This research focuses on the QoS-aware service composition process which is one of several processes required to create business processes based on clients QoS requirements (Jaeger et al., 2005; Jafarpour & Khayyambashi, 2010). More specifically, this work discusses methods and algorithms to perform the selection between the discovered candidate services based on QoS requirements imposed by clients (Baryannis, et al., 2008). In the discovery phase, many candidate services with different QoS characteristic values can be discovered for each abstract web service. Then the QoS-aware service composition process aims at selecting one candidate web service for replacing each abstract web service such that the entire QoS of the composition are optimized while client's QoS requirements are satisfied (Yu et al., 2007; Canfora et al., 2005; Zeng et al., 2003, 2004; Alrifai et al., 2009).

If more than one candidate service suites a particular abstract service, the selection between these candidate services should be based on preference criteria. This work considers the QoS characteristics as selection criteria to select between those competing candidate services (Zeng, et al., 2003, 2004; Alrifai, Skoutas & Risse, 2010; Canfora et al., 2005; Yu et al., 2007; Zhang et al., 2010; Jiang et al., 2011; Schuller et al., 2011; Singh, 2012).

Usually, business processes include the participation of human to perform tasks. This work, however, covers only business processes that do not required for human interferences, because web services represent primarily a technology that's aimed at the interoperation between software systems (Jaeger, 2007).

1.8 **Research Design**

Three main research steps had been conducted in doing this research, namely the analysis phase, the development phase, and the evaluation phase.

In the analysis phase, in-depth study was performed on the QoS–aware service composition approaches and the surrounding issues like the QoS characteristics considered in these approaches, the optimization strategy used to solve the optimization problem, and the techniques used to tackle multiple paths composition problem. Moreover, a review was conducted on the state of the art approaches proposed to solve the Multidimensional Multi-choice Knapsack Problem (MMKP) problem, the techniques and the methods used for path prediction. As a result, the drawbacks in the state of the art approaches were determined, the problems of the research were clearly defined, the core components that should be considered to develop an approach for QoS-aware service composition was identified, and the QoS characteristics that can be considered for web service composition were determined and prioritized. In addition, new optimization algorithms called Constructive Procedure (CP) and Complementary Constructive Procedure (CCP) for solving the optimization problem were identified. CP is applied to generate a feasible solution while CPP is applied to improve the solution generated from CP. Finally, data mining based techniques used for the runtime path prediction method were identified.

In the development phase, the core components which should be considered to develop the proposed approach were determined. The components include defining the problem and structure model, determining the selection criteria, describing the QoS computation for web service composition, defining the utility function, and finally developing a new optimization mechanism to solve the research problems. The development of the optimization mechanism includes proposing a runtime path prediction method, mapping the QoS-aware service composition problem to MMKP, and applying heuristic optimization algorithms to solve the selection problem. The MMKP is similar to the selection problem (Yu, Zhang & Lin, 2007; Alrifai et al., 2009). MMKP aims to pick exactly one item from each class in order to maximize the total profit value of the pick that is subject to resource constraints (Hifi et al., 2004) while the selection problem aims to select exactly one candidate from each service class, where the entire QoS value of the composition is optimized while QoS requirements defined by clients are satisfied.

In the evaluation phase, the evaluation process was divided into two parts. The first part aimed at evaluating the runtime path prediction. For this purpose, the data used for evaluation was collected and prepared. Beside the data, data mining tool, machine learning algorithms, evaluation measures, and a performance estimation method was identified. Finally, a set of test experiments was introduced. The second part aims at evaluating the optimization mechanism. For this purpose, the measures and the methods used for the evaluation were determined. A new simulation prototype was developed and the optimization mechanism was implemented. Finally, a set of test experiments was introduced.

1.9 **Thesis Layout**

The remainder of this thesis is organized as the following:

Chapter 2 gives background information about QoS characteristics and explains the need for QoS characteristics in the area of web service composition. It also provides an overview about the multiple paths composition and the QoS-aware service composition. A critical study and a survey of the relevant existing optimization techniques, that are used to handle with the multiple paths composition problem, are also given in this chapter. In addition, the chapter reviews the state of the art approaches proposed to solve the Multidimensional Multi-choice Knapsack Problem (MMKP) problem. Finally, the techniques and the approaches used in path mining are discussed in this chapter.

Chapter 3 describes the methodology used in the investigation. A description of the methods, algorithms, equations, and the simulators used in this research are given.

Chapter 4 reviews the QoS characteristics considered in the area of QoS for web services and SOA. The chapter also gives analyses of the QoS characteristics and suggests the QoS characteristics that can be considered as selection criteria for web service composition. Finally, it provides a priority for the suggested QoS criteria.

Chapter 5 describes the proposed approach for multiple paths QoS-aware service composition. The component used to develop the approach is discussed in detail in this chapter. Moreover, the runtime path prediction method and heuristic optimization algorithms are explained. Furthermore, this chapter covers the implementation of the proposed approach.

Chapter 6 presents the different test experiments used for the evaluation of the proposed approach. Results produced from the experiments are discussed and compared with other existing optimization techniques.

Chapter 7 summarizes the research work, highlights research contributions, and gives direction for future works related to this research.

CHAPTER TWO
RELATED WORKS

This chapter reviews the QoS-aware service composition approaches and the optimization techniques used to handle with the multiple paths composition problem. Section 2.1 begins by presenting background information about QoS characteristics. Section 2.2 explains the need for QoS characteristics in the area of web service composition. Section 2.3 provides an overview about the multiple paths composition while Section 2.4 introduces the related works and discusses the optimization strategies, the techniques used to handle multiple paths composition problem, the limitations, and the QoS computation methods. Section 2.5 reviews the state of the art approaches proposed to solve the MMKP. Section 2.6 reviews the techniques and the approaches used in path mining.

2.1 QoS for Web Service

Web services are designed to perform functionalities that describe what web services can do. These functionalities represent the functional behaviors of the web services. A flight booking service, for example, provides booking flight ticket functionality. However, non-functional behaviors of web services can also be considered in the description. These behaviors represent the way web services supply their functionality. The time needed for the flight web service to book a ticket is an example of non-functional property.

The International Organization for Standardization (ISO) provides a general definition for quality. ISO defines quality as “the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs" (ISO-9000:2005, 2005). Beside this definition, there are many definitions for QoS in literature. These definitions, however, vary according to the application scenarios. This research work uses the definition provided by Ran (2003) in order to describe the QoS for web service. He describes QoS as "a set of non-functional attributes that may impact the quality of the service offered by a web service" (Ran, 2003). QoS can be classified according to the domain into (1) domain-independent characteristics (i.e., A set of QoS characteristics that are applicable to all web service domains), for example, cost and response time characteristics, (2) domain-specific characteristics (i.e., A set of QoS characteristics that can be applied in a specific domain), for example, a precision is domain-specific characteristic of temperature web service.

In the past few years, QoS for web services has gained a considerable momentum. This is because QoS plays an important role in service automation tasks, especially in service discovery and selection. Imagine a scenario where many web services, that fulfill a user request and provide the same functionality, are discovered at runtime, and the selection among them is based on QoS characteristics like cost and response time (Toma & Foxvog, 2006). QoS characteristics can serve as selection criteria for selecting individual services. One may choose the lowest web service cost while the others may choose the fastest response time.

2.2 QoS for Web Service Composition

QoS characteristics become more crucial for web service composition because the general QoS performance of the composite services is determined by the QoS performance of its underlying web services. Choosing web services with poor quality will degrade the overall performance of the compositions. Clients who build composite services need objective QoS characteristics to distinguish between the competing web services (Liu, Ngu & Zeng, 2004). In this context, QoS characteristics play important roles to decide which web service must be selected to participate in compositions. Furthermore, clients can use these QoS characteristics to specify their QoS requirements. Thus, it guarantees that the clients' visions efficiently translate into composite services. Figure 2.1 illustrates an example of a composition that consists of two web services which are airline booking and hotel reservation. At runtime, two functionality equivalent web services, with different QoS characteristics, are discovered (service 1 and 2). Normally, the selection between them is based on clients' QoS requirements.

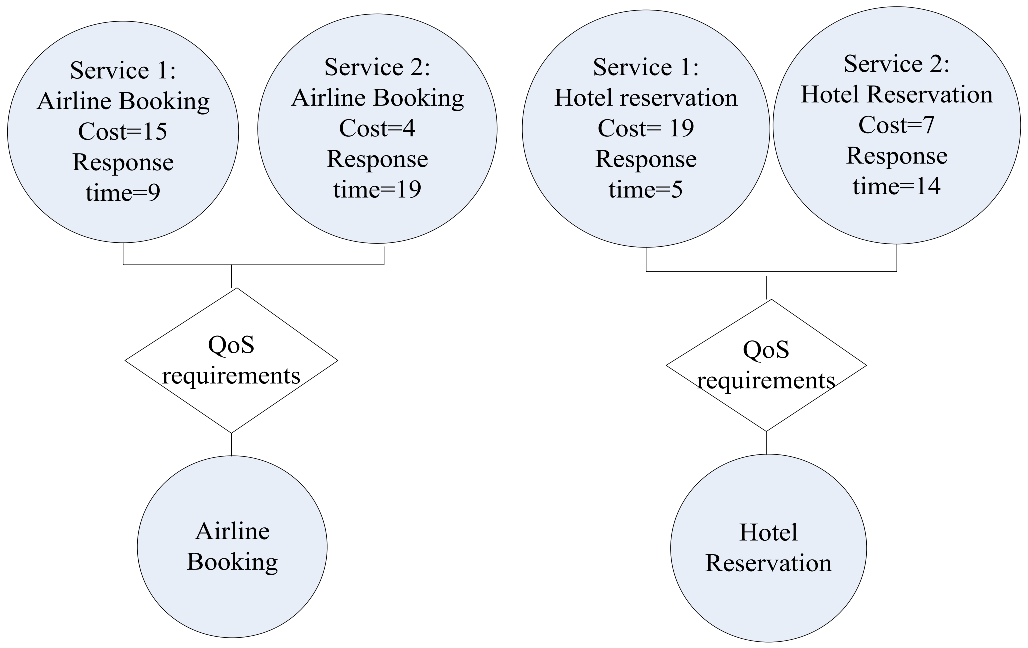


Figure 2.1. *Illustration of competing web services*

2.3 **Multiple Paths Composition**

To define compositions, sequential structures are used to connect the web services that constitute the compositions. These compositions are called single path compositions.

Compositions, however, are operating in highly dynamic environments which allow different possible scenarios to be occurring at runtime, making the real time compositions facing unanticipated changes. In the case that the composition is defined at design time, it is desirable to support the expectations that can be anticipated by composition developers.

Flexibility by configuration or flexibility by design refers to the structural properties of a composition which allows it to respond flexibly to different scenarios anticipated by composition engineers at design time (Ukor & Carpenter, 2008). Such flexibility can be achieved by the presence of a conditional structure in the composition definitions. The presence of the conditional structure makes it possible for multiple execution paths to be represented by a single composition, design (Schonenberg, Mans, Russell, Mulyar & Van der Aalst, 2008). This work referred to this kind of compositions as multiple paths compositions. The result is a distinctive set of composition paths where each path represents a scenario that can be taken during the execution of a composition instance (Ukor & Carpenter, 2008) (i.e., One path is selected and taken from multiple alternative composition paths).

2.4 **Optimization Approaches for QoS-Aware Service Composition**

Research community identified different approaches for a web service composition. Generally, these approaches can be classified under three main categories: (1) an automatic web service composition, (2) a model-driven web service composition, and (3) a QoS-aware web service composition. This section focused on the research works that are relevant to the approach proposed in this thesis. In particular, this section reviews the works that are related to the third category (i.e., A QoS-aware web service composition).

Although the QoS-aware web service composition approaches use different optimization techniques, they almost share the same methodology which can be summarized as the following (Baryannis, et al., 2008):

1. All approaches require defining an abstract composition (i.e., identifying and arranging the abstract services or tasks) and provide a desired functionality description for each abstract service.
2. For each abstract service in the composition, a QoS-aware service composition method is required to select optimal candidates based on QoS criteria without taking into account the entire QoS of the composition. This step, referred to as a local optimization and it does not necessarily meet the QoS global constraints.
3. With the presence of global QoS constraints, a global optimization method is required to generate an optimal composition plan that meets clients' (organization’s) global QoS requirements.

Different approaches have been proposed for QoS-aware service composition. In the following subsections, the most important and influenced approaches are discussed in detail. The approaches are divided into two groups in respect to the structure used to develop the composite services. The groups, namely, single path composition approaches and multiple paths composition approach. The subsections highlight the optimization strategies, the techniques used to handle multiple paths composition problem, the limitations, and the QoS computational methods used in these approaches.

2.4.1 Single Path Composition Approaches

This group of approaches optimizes compositions that accommodate only a single execution path (i.e., The compositions are defined by using a set of abstract service in a sequential order). Yu, Zhang, and Lin (2007) introduce several selection algorithms with end-to-end constraints. Two models study for the selection problem: a graph model that defines the problem as a Multi-Constraint Optimal Path (MCOP) and a combinatorial model that defines the selection problem as MMKP. A utility function is defined in both models. In the case of the graph model, the problem is defined as MCOP, and the single-source shortest paths based algorithm (MCSP) is proposed. However, the algorithm is very slow due to the huge number of the paths. Therefore, a heuristic algorithm called MCSP-K is proposed by modifying the MCSP algorithm. In the case of the combinatorial model, the problem is mapped as MMKP and two algorithms are used, namely the branch-and-bound algorithm (BBLP) that finds the optimal result, but with exponentially rising computational time, and the heuristic algorithm (WS-HUS) that finds a near-optimal solution in polynomial time. In order to improve the optimality of the WS-HUS algorithm, an algorithm named LASA-HEU is proposed recently by Sasikaladevi and Arockiam (2014).

An efficient approach to the problem of the QoS-aware composition is proposed by Alrifai and Risse (2009), and Alrifai, Skoutas, and Risse (2010). In their approach, the problem is solved by combining the local optimization approach with the global optimization approach to benefit from the advantages of both strategies. Their approach consists of two steps. First, the global QoS constraints are decomposed into a set of local constraints, where the satisfaction of these constraints guarantees the satisfaction of the global constraints. By doing so, the fulfillment of the global QoS constraints is guaranteed without enumerating all possible combinations. To find the optimal decomposition of global QoS constraints into local constraints, Mix Integer Linear Programming (MILP) solving techniques is used. Second, the best web services that satisfy the local constraints are selected by distributing the local selection. The results show that their approach is significantly outperforming the global optimization approach in terms of the computational time while computing near-to-optimal solutions.

In Tao, LaiLi, Xu and Zhang (2013), the service selection problem is converted to a graph searching problem to tackle the problem of finding an optimal composition in a large scale. Their approach is divided into two stages, namely, the run-up stage that deals with data preprocessing to parse service repository, and the composition stage that generates optimal top-k solutions.

Some authors employ Ant Colony Optimization (ACO) techniques to solve the QoS-aware service composition problem (Xia, Chen & Meng, 2008; Qiqing, Xiaoming, Qinghua & Yahui, 2009). As stated by Xia et al. (2008), ACO compared to Genetic Algorithm (GA) is simpler with fewer parameters. The Particle Swarm Optimization (PSO) is another optimization algorithm that is applied to solve the QoS-aware service composition problem with a lot of research work (Li & Yan-Xiang, 2010; Liao, Liu, Zhu, Wang & Qi, 2013; Wang, Zhu & Yang, 2014). PSO compared to GA is faster and easier to implement (Ming & Zhen-Wu, 2006). Zibanezhad, Zamanifar, Nematbakhsh, and Mardukhi (2009) use the Gravitational Search Algorithm (GSA), which is very much similar to PSO, to solve the problem. Apart from PSO, Berbner, Spahn, Repp, Heckmann, and Steinmetz (2006) propose to use heuristics in order to solve the QoS-aware service composition problem. Three heuristics are designed, implemented and evaluated. The performance of the heuristics is evaluated by comparing it with Integer Programming (IP). The results show that the three heuristics outperform the integer programming approach.

In Gao, Chen, Qiu, and Meng (2009), an algorithm named Quality of Experience (QoE) / Quality of Service (QoS) Driven Simulated Annealing-based Genetic Algorithm (QQDSGA) is introduced by combining GA and Simulated Annealing (SA). SA is used to avoid GA from falling into local optimal solution. The results show that QQDSGA is better than SA and GA individually. In order to evaluate the composite service, a model based on QoE and QoS is proposed. However, the model evaluates the solutions relying on customer feedbacks which are unreliable and vulnerable to malicious customers' manipulation. In addition, the model evaluates the solutions without taking into account the dependencies between the web services that participate in the composite services.

A hybrid algorithm based on the combination between ACO and GA is introduced by Lou, Tao, Wang, and Yue (2009). The idea is to avoid the limitations of the standard GA and ACO algorithms by combining them together in order to benefit from the advantages of both algorithms. The results show that the new algorithm is efficient in terms of the speed and the computational time. Another GA and ACO combination algorithm is introduced by Yang, Shang, Liu and Zhao (2010). The authors aim to improve the low efficiency issue of the selection algorithms in large size solution space. The selection problem is transformed into a problem of finding an optimal path that meets users' QoS requirements in the weighted directed acyclic graph. An ant colony algorithm is used to find the optimal path that has the maximum sum of the QoS values from the start point (which represents the ant nest) to the target point (which represents the food source) in the weighted directed acyclic graph. Since the selection of the ant colony algorithm parameters has a great effect on the performance of the algorithm, GA is employed to set the parameters. Recently, an algorithm called Particle-Ant Colony Algorithm (PACA) is proposed by Pei, Shi, and Hu (2014). The algorithm transforms the selection problem into shortest path problem. In their approach, PSO is applied to (1) find suboptimal paths, and (2) initialize the pheromones of these paths, then ACO is used to find the optimal solution.

The aforementioned approaches simplify the problem of the QoS-aware service composition by presuming that a composition can be represented by a single execution path i.e., A sequential order of abstract web services. Nevertheless, a composition can also contain multiple execution paths which allow it to respond flexibly to different scenarios anticipated by composition engineers at design time.

2.4.2 Multiple Paths Composition Approaches

This group of approaches optimizes compositions that accommodate multiple execution paths. In these approaches, it is difficult for the optimization algorithms to generate a solution that simultaneously optimizes all the execution paths involved in the composition at the same time while meeting clients' QoS requirements. Consequently, different optimization techniques were proposed to solve this issue. This research work divides these approaches into two categories with respect to the optimization techniques used to handle the multiple paths composition problem highlighted in this work. The categories are: (1) a separate path optimization technique and (2) all paths optimization technique.

2.4.2.1 A Separate Path Optimization Technique

In this technique, a composite service is decomposed into execution paths in order to optimize each path separately. Then after the completion of optimal solution computation, the execution paths are aggregated into an overall composition that consists of all paths. If there is a common abstract service that belongs to more than one path, the system identifies the hot path for the considered web service. The hot path is defined as the path that has been most frequently used to execute the considered service.

Zeng et al. (2003, 2004), as one of the first, introduce the idea of composite service decomposition. The authors introduce a framework that covers several aspects of developing a web service composition taking into consideration QoS as criteria for selection. In their work, they propose a simple QoS model which has been adopted by a large number of subsequent approaches in this field. The model consists of five general QoS characteristics that are applicable to all web service domains. These characteristics are cost, duration (response time), the success rate (reliability), availability, and reputation. In Zeng et al. (2003, 2004), a state chart is used to represent a composition. It is assumed to be acyclic; if it is not, a technique is used for unfolding it. The state chart is then divided into multiple execution paths. Each path is represented as a Directed Acyclic Graph (DAG). Furthermore, execution plans are defined for each execution path. An execution plan is a set of pairs, where each pair consists of a task and a web service that implements the task operation. Zeng et al. (2003, 2004) perform the selection by optimizing each path separately. And then the optimized paths merge into an overall composition that consists of all paths. If there is a common task that belongs to more than one path, the system identifies the hot path for the considered task. The hot path is the path that has been most frequently used to execute the considered task.

Zeng et al. (2003, 2004) have identified two strategies for optimizations: local optimization strategies and global optimization ones. For local optimization, the system selects the optimal candidate for each task that participates in a composition without considering the overall QoS of a composition. To explain in detail, when executing a task, the system collects the QoS information for all candidates of this task. And then it computes quality vectors for these candidates. Based on these vectors, the selection is done by applying Multiple Criteria Decision Making (MCDM) technique that computes a quality score for each candidate and the system selects the candidate with the maximum quality score to execute the task.

Beside Zeng et al. (2003, 2004) work, there are many existing approaches adopted the locally optimization strategy (Dimitrios, Hans, Andrzej & Donald, 1999; Casati, Ilnicki, Jin, Krishnamoorthy & Shan, 2000; Benatallah, Dumas, Sheng & Ngu, 2002). Although the local approach is very efficient in terms of computation time, it does not guarantee satisfying the global QoS constraints; however, it can only satisfy local QoS constraints (i.e., constraints on a task, part of a composition). An example of local constraints is the response time of a single web service that must be less than 5 seconds. Therefore, Zeng et al. (2003, 2004) introduce a global optimization strategy. They first introduce a naïve approach for global planning. In this approach, for each execution path, all possible execution plans are generated. To select the optimal plan, the system computes the global quality score for each plan and relies again on MCDM to select the plan with the highest score. The time complexity is the major disadvantage of this approach. Such a straightforward strategy results in a combinatorial problem and the computational complexity to find a solution for this problem NP-hard. Therefore, the authors propose an approach based on Integer Programming (IP) that performs the optimization without generating all possible plans. An IP has variables, constraints, and an objective functioning as inputs. Constraints and an objective function must be linear. Given the three inputs, a solution can be found using IP solver by adjusting the values of the variables according to the constraints in order to maximize or minimize the objective function's value. Similar to this approach, Ardagna and Pernici (2006) provide an approach based on Mixed Integer Linear Programming (MILP) that addresses both local and global constraints. They introduce loops peeling which improves the unfolding loop techniques introduced by Zeng et al. (2003, 2004).

Zhang et al. (2010) apply the divide and conquer strategy to decompose the general flow structures into several sequential structures due to the general flow structures that are hard to handle. In the approach of Zhang et al. (2010), the problem is modeled as a multi-objective optimization problem and ACO techniques to solve the QoS-aware service composition problem. Their experiments show that the ACO is efficient and scalable. Nonetheless, global QoS constraints cannot be imposed in their approach.

Liu, Wu, and Liu (2012) propose an approach based on path decomposition. The entire plan is decomposed into fine-grained fragments and then stored in the Case Library. The optimization process is performed in two steps: (1) adjusting the path by retrieving and reusing the plans stored in the Case Library, and (2) applying GA for the service selected. The QoS model defined in their approach consists of cost, response time, reputation, and reliability.

The aforementioned approaches have several issues:

1. The optimization is computed considering each execution path separately. If there is a conflict in service selection in some abstract services that are common to multiple execution paths, the system identifies the hot path for the considered web service. However, in the case that the actual execution of

the composition is not following the hot path, the executed path may not have the best QoS ratio, worse than that, the executed path may violate the QoS requirements.

2. Although IP approaches are very efficient when the problem size is small, they are inappropriate for runtime selection. This is because the computation time is rising exponentially with the increasing problem size (Alrifai & Risse, 2009). Moreover, IP approaches consider the linearity of the constraints and the objective functions.
3. The QoS characteristics considered in their model are a small set of general QoS characteristics.

The above issues are resolved by the proposed approach. In the proposed approach, there is no conflict in service selection in some abstract services that are common to multiple execution paths. This is because only one path will be considered in the optimization i.e., the path that will be most likely taken by a composition instance. By this strategy of optimization, it is expected that the resulted solutions deliver the best possible QoS ratio and, at the same time, it meets the QoS requirements. Moreover, instead of using IP for optimization, heuristic algorithms are applied to solve the optimization problem. Heuristics are efficient in reducing the computation time making the proposed approach be used in any problem size. In addition, the proposed approach does not impose the linearization of the constraints. This permits the use of the proposed approach for all possible QoS characteristics without the need for linearization. Finally, the QoS characteristics considered in the proposed approach are determined after investigating and analyzing the related works in the area of web service and SOA. Furthermore, these QoS characteristics are derived from the special features of composite services which differ from the features of single web services.

2.4.2.2 All Paths Optimization Technique

In this technique, the optimization is computed assuming that a certain path will be more likely executed than others according to the probability of path execution. The assumptions are based on stochastic information indicating the probability of paths being executed at runtime. Estimation of the paths probability of executions is estimated either by inspecting the system logs or being specified by the composition engineers. All the approaches that will be introduced in this subsection share the above-mentioned technique for handling multiple paths composition problem.

In Yu, Zhang, and Lin (2007), the composition is decomposed into two kinds of subgraphs: (1) the execution route that includes one branch in each conditional structure and all branches are in parallel, (2) a sequential path that includes one branch in both conditional and parallel structures. Each subgraph has a probability indicating its probability to be executed. In the case of the combinatorial model, similar to Zeng et al. (2003, 2004), the problem is mapped to 0-1 IP problem. Their model, compared to Zeng et al. (2003, 2004), ensures that the generated solutions always meet the QoS requirements. Two algorithms are presented, namely WS_IP algorithm, to find an optimal solution, but with exponentially rising computational time, and WFlow algorithm as a heuristic algorithm that finds a near-optimal solution in polynomial time.

GA is applied first by Canfora et al. (2005) for solving the QoS-aware service composition problem. The motivations behind the application of GA to solve the selection problem are demonstrated in the following points. First, not like IP approaches, GAs can handle the non-linear functions, making GA-based approaches able to handle all possible QoS characteristics. Second, GAs, compared with IP approaches, are able to scale up when the problem size is very big. Their approach aims to quickly find a set of concrete (candidate) services to be bound to abstract services that participate in the composition. Such a set needs to satisfy the global QoS constraints imposed by Service Level Agreement (SLA), and at the same time, it optimizes the overall QoS of composition.

In the approach of Canfora et al. (2005), the QoS-aware service composition problem is encoded by a genome that is represented by an integer array with a number of items equals to the number of distinct abstract services participating in the composition. The crossover operator is the standard two-point crossover while the mutation operator randomly selects an abstract service and randomly replaces the corresponding concrete service with another one.

The QoS-aware composition problem is modeled by a fitness function that aims to maximize some QoS attributes (e.g. availability) while minimizing others (e.g. cost). In addition, the individuals who do not meet the global QoS constraints must be penalized by the fitness function. The fitness function has a static penalty; if its weight is high, there is a risk that individuals also violate the constraints, but being “close" to a good solution could be discarded. Therefore, the authors define an alternative dynamic function that may allow considering some individuals violating the constraints. The authors evaluate the approach by comparing it with the well-known IP methods. The results show that GA provides better scalability and performance when the number of candidate services is large. However, IP is preferable instead of GA when the number of candidate services is small.

There are some other GA-based proposed approaches varying either on the fitness function, the encoding schema or on the genetic operators i.e., crossover operator, mutation operator, and selection operator. In Zhang, Li, Chao, and Chang (2003), binary strings of chromosomes are designed to represent a solution. In a chromosome, each abstract web service is represented by a cluster and each cluster, in turn, consists of genes representing the candidate services. There are two possible values of a gene: 0 if the service is not selected and 1 if it is selected. However, when the number of candidates is very big, it results in a very long chromosome. This kind of manner results in poor readability. Moreover, any change in the number of candidate services could influence the length of chromosome which results in poor stability of a chromosome length. In comparison with the one dimensional coding proposed by Canfora et al. (2005), one dimensional coding is shorter and shows better stability since it is not influenced by the changing number of candidates. Du, Wang, Ai, and Li (2012) propose a penalty-based genetic algorithm for selecting the appropriate services under temporal constraints. In their approach, re-planed solution process is performed at runtime to resolve the constraint violation.

In Jiang et al. (2011), the author proposed to use the variable length chromosome to represent the different composition plan. Zhang, Su and Chen (2006a, 2006b, 2006) design GA with a relational matrix coding scheme of chromosomes and a population diversity, handling mechanism to solve the QoS-aware service composition problem. The population diversity, handling mechanism is introduced to avoid the prematurity convergence phenomenon of standard GA, but it is in contrast with the one dimensional coding scheme designed by Canfora et al. (2005) which can represent only one path of a composition, the introduced relational matrix coding scheme can express all the composition paths at one time. Thus, the proposed GA is only needed to run once in order to generate the optimal plan. The experiments show that GA with relational matrix coding can generate an excellent composition plan more than the standard GA can do. Furthermore, as the experiments show, the adopted initial population policy and mutation policy improve the fitness of GA.

In Wu, Xiong, Ying, Jin and Yu (2011), the selection problem is modeled as an Objective Multi-Constraints optimization problem and a new algorithm named GAELS (Genetic Algorithm Embedded Local Searching) is proposed to solve the optimization problem. The algorithm uses the strategies of enhanced initial population and mutation with local searching, to speed up the convergence. Their experiment results showed that the algorithm generates the non-inferior solution more quickly than simple genetic algorithm in large-scale web service composition.

In Dong and Dong (2009), the authors use the Elitist Selection Genetic Algorithm (ESGA) to solve the QoS-aware service composition problem. They consider only the two QoS characteristics, cost and execution time, which should be minimized. This represents a simplification that does not cover all web service composition problems. This is because the solutions obtained from their approach could negatively affect other QoS characteristics such as reliability and availability which both should be maximized. In contrast to this, various QoS characteristics are addressed. Some of them should be maximized while others should be minimized. Lécué (2009) studied QoS-aware semantic web service composition in a context of how to effectively compute optimal compositions of QoS-aware web services by considering their semantic links. They address the optimization problem by using GA-based approach. In general, the GA-based approaches are scalable and efficient when the problem size is large. However, one well-known drawback of GA is that it can easily fall in local optima.

In Jafarpour and Khayyambashi (2010), the recently developed Harmony Research (HR) algorithm, which is inspired by the musical process for searching of the best harmony, is applied to find a solution for the QoS-aware composition problem. The HR algorithm is characterized as simple and easy to implement, and it needs a few parameters. Also, it needs a few mathematical requirements. In their approach, users can define both local and global QoS constraints and the algorithm must find the solution that has the optimal QoS while meeting these constraints. To evaluate the performance of the proposed approach, the execution time and the optimality results of the HR approach are compared with GA-based approaches' results. The results show that their approach introduces lower execution time and best QoS solutions compared to GA-based approaches.

Tabu Search (TS) is another meta-heuristic optimization technique used to solve the problem of QoS-aware service composition. Parejo et al. (2008) used TS and Hybrid Genetic Algorithm (HGA) to solve the problem. In order to apply TS technique, neighborhood of solutions is defined by changing the selected web service for a given task. After applying this simple movement, a fixed size tabu list is used as the implementation of the recent memory strategy. Such a strategy is used to escape the trap of the local optimality by preventing the heuristic from the reversal of the recent moves. In addition, an aspiration condition is incorporated in the search algorithm that allows the reverse move if the resulted value of the objective function is better than the old one. The results show that TS performs better than both HGA and the standard GA only if the problem size is small. However, if the problem size is medium, TS performs badly due to the size of the neighborhood of solutions. In Bahadori, Kafi, Far, and Khayyambashi (2009), a hybrid GA-Tabu Search approach is proposed for the QoS-aware service composition problem. The encoding schema of the chromosomes, the crossover, and the mutation operators are similar to that proposed by Canfora et al. (2005), whereas the fitness function is adopted from Zhang et al. (2006b). Integrating TS with GA leads to increase in the population diversity and escapes the trap of local optimality.

In Zheng, Zhao, Yang, and Bouguettaya (2013), a systematic QoS computation approach is presented. The approach is capable of providing comprehensive QoS information for a composite service with complex structures include: sequential, parallel, loop, and conditional structures. The approach processes the conditional structures by transforming the service graph into a rooted tree, and computes the QoS of the web service composition as the probability weighted sum of the QoS of the paths.

Jaeger, Rojec-Goldmann, and Muhl (2004) propose a method to evaluate the performance of the QoS-based selection algorithms; it checks whether a set of selected services for composition satisfies the global QoS requirements or not. The method drives the overall QoS of a composition by aggregating the QoS of individual services. It is based on a well-known workflow pattern by Van der Aalst, Hofstede, Kiepuszewski, and Barros (2003). The workflow patterns are analyzed and seven patterns (called composition patterns) are chosen, namely sequence, loop, XOR split followed by XOR join, AND split followed by AND join, AND split followed by m-out-of-n join, OR spilt followed by OR join and OR spilt followed by m-out-of-n join. Then a composition model that derived from these patterns is defined. In addition, they define a QoS model as consisting of response time, cost, encryption grade, throughput, reputation, availability, and reliability. For each combination of composition patterns and QoS characteristics, a QoS formula is defined. Building on these QoS formulas, the method calculates the overall QoS of a composition by identifying a composition patterns in a graph that represents the composition structure calculating the QoS for each pattern according to the QoS formulas until one single node remains. This process is referred to as a stepwise graph collapse. The QoS statements resulted from the method are used in the comparison between the solutions, and the solution with the maximum QoS utility is considered as the optimal one. For selecting the candidates to optimize the overall QoS for a composition, Jaeger, Muhl and Golze (2005) propose four heuristic algorithms, namely the greedy selection, the discarding subsets, the bottom-up approximation, and the pattern-wise selection.

Schuller et al. (2011) address the problem of multiple paths composition in their work. They propose to perform average-case analysis according to the probability of execution paths. For solving the optimization problem, the problem is transformed as a nonlinear optimization problem and transforms it into a linear one. Then applying Integer Linear Programming (ILP) techniques are applied to solve it.

Ukor and Carpenter (2008) survey the literature approaches on the QoS-aware service composition problem and review the effects of the presence of multiple paths. These are involved in compositions on the ability of the optimization algorithms to simultaneously generate exact optimal plans for all execution paths contained in such compositions. In brief, within the context of multiple paths composition, it is difficult for the optimization algorithms to generate a solution that simultaneously optimizes all the execution paths in the composition at the same time. Their work highlighted the main problem, where this thesis aims to solve. In a subsequent paper, Ukor and Carpenter (2009) address this problem by presenting an approach for QoS-aware service composition that enables users to bias the optimizations using a set of meta-metrics. The approach aims to find an approximation solution for each path involved in the composition. A trade-off between the paths is made choosing a path to favor by using a set of meta-metrics. These meta-metrics include execution probability of an activity, previous execution history of each activity, and probability of occurrence. For each path, the meta-metrics are computed as the weighted average of the aggregate values of the meta-metrics. Then the problem is formulated as IP problem and both the constraints and the objective function are defined.

Ko et al. (2008) modeled the QoS-aware service composition problem as a constraint satisfaction problem. Based on the problem model, a hybrid algorithm that combines TS and SA techniques is proposed. To implement the algorithm, the authors suggest a QoS-oriented service composition planning architecture designed to support the automatic generation of the QoS-aware service composition plan in an optimal way. A comparison between the proposed service composition algorithm and the IP approach proposed by Zeng et al. (2003, 2004) is performed to evaluate the computation time of the proposed algorithm. The results show that the average execution time for the proposed algorithm does not exceed 3.5 second in its worst case, whereas the average execution time needed for the IP to find a solution is 12.04 minutes. This makes IP approach inappropriate in real-time application scenario. In a case where a composition is connected with XOR parallel pattern (i.e., multiple paths composition problem), a worse case strategy is used to evaluate such a pattern. Worse case strategy is also used by Singh (2012) for estimating the conditional structure. However, such a strategy for computing the entire QoS of composite services may result in non-precise QoS statements.

The aforementioned approaches consider all execution paths together in computing the optimal solution. However, the following drawbacks are identified in those approaches:

1. Considering all paths together in computing the optimal solution may result in a suboptimal solution for some execution paths.
2. If the composition execution follows the path with the less probability, global QoS constraints may be violated.
3. Applying IP for solving the optimization problem results in high computation time.

4. For Jaeger et al. (2004) works, in the case of conditional patterns, the aggregation for the majority of QoS characteristics are given as the maximum of the aggregation values of all paths contained in the pattern. Such a strategy for computing the QoS may not provide a precise QoS statement that represents the QoS of the entire composition, making it possible for the optimization algorithm to generate a suboptimal composition plan.
5. For Ukor and Carpenter (2008, 2009) works, the meta-metrics are based on assumptions assigned either by the composition developers or estimated from the log trace records. These assumptions may be false. Consequently, the optimal solutions obtained from this approach may prove to be suboptimal for some execution paths. Even worse, the QoS requirements may violate.
6. For the works of Ko et al. (2008) and Singh (2012), similar to Jaeger et al. (2004), the worst case strategy of evaluation may not result in a precise QoS statement that represents the QoS of the entire composition.

Instead of considering all execution paths together in computing the optimal solution, the proposed approach considers one single path in computing the optimal solution i.e., the path that will be potentially executed. For QoS computation, this work does not utilize any aggregation formulas for the conditional structures because the QoS computation considers only one path. Thus, the resulted QoS statements are expected to be always precise.

Table 2.1 summarizes the state of the art approaches proposed to solve the multiple paths composition problem.

Table 2.1

A Summary of the State of the Art Approaches Proposed to Handle the Multiple Paths Composition Problem

Research groups	Optimization strategy for solving optimization problems	Optimization techniques to handle multiple paths composition problem	Considered QoS characteristics	Weaknesses
Zeng et al. (2003, 2004)	IP	• separate path optimization technique • decompose/merge • hot path	cost, response time, reliability, availability, and reputation	• small set of general QoS characteristics • conflict in service selection • high computation time
Yu et al. (2007)	WS_IP and WFlow heuristic algorithms	• all paths optimization technique • probability of execution paths	cost, response time, and availability	• small set of general QoS characteristics • non-precise QoS statements
Canfora et al. (2005)/ Jiang et al. (2011)	GA	• all paths optimization technique • probability of execution paths	cost, response time, reliability, and availability	• small set of general QoS characteristics • non-precise QoS statements
Jaeger et al.,(2004, (2005)	greedy selection, discarding subsets, bottom-up approximation, and the pattern-wise selection	• all paths optimization technique • maximum QoS value of all execution paths	Throughput, response time, cost, reliability, availability, reputation, and encryption grade	• non-precise QoS statements
Liu, Wu, and Liu (2012)	GA	• separate path optimization technique • path decomposition /path adjustment	cost, response time, reputation, and availability	• small set of general QoS characteristics • conflict in service selection
Schuller et al. (2011)	ILP	• all paths optimization technique • average-case analysis • probability of execution paths	response time, reliability, throughput, and cost	• small set of general QoS characteristics • non-precise QoS statements • high computation time
Ukor and Carpenter (2008, 2009)	IP	• all paths optimization technique • meta-metrics to bias the optimization	cost, response time, reliability, and availability	• small set of general QoS characteristics. • conflict in service selection • high computation time
Singh (2012)	combining the local optimization approach with the global optimization approach	• all paths optimization technique • worse case strategy	cost, response time, availability, and reliability	• small set of general QoS characteristics • non-precise QoS statements
Zhang et al. (2010)	ACO	• separate path optimization	cost, response time, reliability,	• small set of general QoS characteristics

		technique • divide and conquer strategy for decomposition	and availability	• conflict in service selection • cannot impose global QoS constraints
Ko et al. (2008)	TS-SA	• all paths optimization technique • worse case strategy	cost, response time, reliability, availability, reputation, and frequency	• small set of general QoS characteristics • non-precise QoS statements

2.5 Solutions for Multidimensional Multi-choice Knapsack Problem (MMKP)

The MMKP is similar to the selection problem (Yu, Zhang & Lin, 2007; Alrifai et al., 2009). MMKP aims to pick exactly one item from each class in order to maximize the total profit value of the pick that is subject to resource constraints (Hifi et al., 2004). On the other hand, the selection problem aims to select exactly one candidate from each service class, where the entire QoS value of the composition is optimized while QoS requirements defined by clients are satisfied. The following is a review on the state of the art approaches proposed to solve the MMKP problem with the purpose of identifying new optimization algorithms to be applied to solve the optimization problem resulted from the QoS-aware service composition.

According to Mostofa Akbar, Sohel Rahman, Kaykobad, Mannin, and Shoja (2006), solutions for MMKP problem are divided into two types: (1) exact optimal algorithms that generate the optimal solutions within reasonable computational time, and (2) heuristic algorithms that generate near-optimal solutions within small computational time.

For exact optimal algorithms, some existing algorithms are based on the branch-and-bound paradigm (Balas & Zemel, 1980; Martello & Toth, 1988; Sbihi, 2006; Razzazi & Ghasemi, 2009). As mentioned earlier, the branch-and-bound paradigm is applied by Yu, Zhang, and Lin (2007) to solve the QoS-aware service composition problem. Dynamic programming is another paradigm used to solve the MMKP problem (Pisinger, 1996). Martello, Pisinger, and Toth (1999) proposed a hybrid approach that combines dynamic programming and branch-and-bound paradigm.

However, MMKP is known as NP-hard (Martello & Toth, 1986). Due to its high computational complexity, approaches that deliver exact optimal solutions are inappropriate for real time decision-making applications. This can be explained, especially in our scenario, where a quick response for a workflow instance is very important. Thus, heuristic represents a novel approach.

On the other hand, several heuristic algorithms are proposed to solve the MMKP problem. Moser, Jovanovich, and Shiratori (1996) propose a heuristic based on the method of the Lagrange multiplier. It is started with computing an unfeasible solution and iteratively replacing the items to reduce the unfeasibility of the solution. Khan, Manning, and Akbar (2002) propose a heuristic called HEU that used the iteratively improvement procedure based on the concept of aggregate resource by Toyoda (1965). As it starts, an initial feasible solution is computed. Items are then selected to be picked based on the concept of aggregate resource by Toyoda (1965). Finally, exchanges of picked items are used in order to improve the initial solution.

A constructive and complementary search approach is developed by Hifi, Michrafy, and Sbihi (2004) for solving the MMKP. In this approach, the constructive procedure (CP) is applied to generate a feasible solution while the complementary CP (CCP) is used to improve the quality of the solution generated from CP. A comparison between their approach and the approaches of Moser et al. (1996) and Khan et al. (2002) is conducted. The results showed that their approach led to better results than those obtained by the approaches adopted by Moser et al. (1996) and Khan et al. (2002). The experiment results show that the algorithms generate high-quality solutions within small computing times. Based on these results, their approach is applied here to solve the optimization problem resulted from the QoS-aware service composition. The approach is chosen because of its ability to generate quality solutions and reduce the computational efforts resulted from the problem. Also, it can be easily applied to solve the selection problem.

2.6 **The Path Prediction**

In order to determine the execution path that will potentially be executed at runtime, stochastic analysis and data mining techniques can be used (Cardoso & Lenic, 2006). In stochastic analysis, the process of determining that a particular path will be more likely executed than another is according to the probability of path execution. The assumptions are based on stochastic information indicating the probability of paths being executed at runtime. Estimation of the paths probability of executions is estimated either by inspecting the system logs or being specified by the business engineers. When the business process has never been executed before, the process engineer initially sets the values of probabilities. These values are re-estimated at runtime based on executing instance's data. Relying on business engineers and past instances executions are not enough to produce accurate results for composition executions. There are high chances that global QoS constraints may violate when performing optimization based on this technique. Constraint violation can occur if the composition execution follows the path with less probability. Composition scenarios require a dynamic strategy to identify the path for each composition instance. One of the many possible ways is using data mining approach.

In data mining area, the majority of research works focuses on process mining (Agrawal, Gunopulos, Leymann, 1998; van der Aalst, Weijters & Maruster, 2002; Rozinat & van der Aalst, 2006; van der Werf, van Dongen, Hurkens & Serebrenik, 2008; van der Aalst et al., 2012). The idea of process mining is to extract information about processes from logs. A set of real business process executions (i.e., Process logs) can be taken by data mining techniques to discover, monitor, and improve the process (van der Aalst et al., 2012). A little work has discussed path mining (Rozinat & van der Aalst, 2006; Grigori et al., 2004; Cardoso, 2008). In Rozinat and van der Aalst (2006), their work aims at analyzing the process logs in order to detect data dependencies that affect choices made in the process. Their work focuses only on extracting knowledge from logs about the rules controlling the path that follows at runtime. Grigori et al. (2004) describe a set of integrated Business Process Intelligence (BPI) tool suite to support business managers and IT. The work outlines the use of data mining techniques for process behavior analysis in a broader scope (Rozinat & van der Aalst, 2006). Cardoso (2005, 2008) and Cardoso and Lenic (2006) work on business process quality and emphasizes on the importance of QoS management for workflows and organizations. The authors propose a method, based on data mining techniques, that allows predicting with high level of accuracy the QoS of workflows. The method consists of three phases. First, data mining algorithms are applied on a process log to mine the activities that will potentially be executed at runtime. Second, for each predicted activity, a QoS activity model is built, including information about the activity behavior at runtime. Finally, the QoS of workflows estimation is computed. This approach will be extended and refined for the purpose of runtime path prediction based on the information provided by composition requesters.

2.7 Chapter Summary

This chapter gives background information about the QoS characteristics and explains the need for QoS characteristics in the area of web service composition. Then the most important and influenced approaches, that are related to this research work, have been discussed in detail. These approaches have been grouped into two categories, namely single path composition approaches and multiple paths composition approaches. Single path approaches optimize compositions that accommodate only a single execution path. However, these approaches simplify the problem by presuming that a composition can be represented by a single execution path. Nevertheless, a composition can also contain multiple execution paths. For multiple paths composition approaches, a composition can be represented by multiple paths. This research work divided these approaches into two categories, namely a separate path optimization technique and all paths optimization technique. In the first technique, a composite service is divided into execution paths to optimize each path separately. Then after the optimization is completed, the execution paths are aggregated into an overall composition. The hot path is identified for the service that belongs to more than one path. However, in the case that the actual execution of the composition is not following the hot path, the executed path may not have the best QoS ratio, worse than that, the executed path may violate the QoS requirements. All paths optimization technique computes the optimization assuming that a certain path will be more likely executed than another one according to the probability of path execution. However, this technique may produce suboptimal solution for some execution paths. In addition, global QoS constraints may violate if the composition execution follows the path with less probability.

The second part reviewed the state of the art approaches proposed to solve the MMKP problem. This in order to identify new optimization algorithms to be applied to solve the optimization problem resulted from the QoS-aware service composition. These solutions are divided into exact optimal algorithms that generate the optimal solutions within reasonable computational time and heuristic algorithms that generate near-optimal solutions within small computational time. Exact optimal solutions are inappropriate for real time decision-making applications. For heuristics, reviewing the near-optimal solutions result in applying the CP and CCP algorithms to solve the optimization problem.

The final part presented the techniques used to determine the execution path that will potentially be executed at runtime. There are two techniques: stochastic analysis and data mining techniques. Determining the potential executed path in stochastic analysis is according to path probability which is estimated either by inspecting the system logs or being specified by the business engineers. However, using such a strategy of estimation is not enough to produce accurate results. There are high chances that global QoS constraints may violate. On the other hand, little work has discussed path prediction in the data mining area. One approach proposed by Cardoso (2005, 2008) and Cardoso and Lenic (2006) shows how to apply data mining for path prediction process. This approach will be extended and refined for the purpose of runtime path prediction based on the information provided by composition requesters.

CHAPTER THREE
RESEARCH METHODOLOGY

In this chapter, the methodology that is used in doing this research work is presented in detail. Section 3.1 presents the research phases. Then, each phase is discussed separately in different sections. In Section 3.2, the analysis phase is discussed while Section 3.3 presents the development phase with a description of the models, methods, algorithms, tools, and equations that have been used to develop the approach. The evaluation phase is discussed in Section 3.4 with a description of the experimental procedure, data, tools, algorithms, evaluation measures, evaluation methods, simulation prototype, and different experiments used for the purpose of evaluating the proposed approach.

3.1 **Introduction**

The core activities in the methodology used in conducting this research and fulfilling the objectives of the thesis are shown in Table 3.1. As seen in the table, the core activities are divided into three steps and discussed in detail throughout this chapter.

1. The first step aims at performing an in-depth study on QoS–aware service composition approaches and the surrounding issues like the QoS characteristics considered in these approaches, the optimization strategy used to solve the optimization problem, and the techniques used to tackle multiple paths composition problem. Also, a review on the state of the art approaches proposed to solve the MMKP problem, the techniques and the methods used for path prediction are conducted in this step.

Table 3.1

Research Methodology

	Aim	Steps	Deliverables
1	Analyzing the research problem	• Reviewing the literature	• Criticize the current works • Determine the major drawbacks in these works • Define the problems of the research • Determine the QoS characteristics for web service composition • Prioritize the selected QoS characteristics • Identify the components needed to develop multiple paths QoS-aware service composition approach • Identify the techniques exist for handling the multiple paths composition problem • Identify heuristic algorithms to solve the optimization problem • Identify the techniques and the methods that can be used for the runtime path prediction method
2	Developing a QoS-aware service composition approach	• Defining problem model • Defining composition structure model • Describing the selection criteria • Describing the QoS computation for web service composition • Defining the utility function • Developing new optimization mechanism: • Predicting the execution path • Mapping the QoS-aware service composition problem to multi-dimensional multi-choice knapsack problem (MMKP) • Computing the optimization	• Problem model • Composition structure model • Selection criteria • Aggregation functions • Utility function **New optimization mechanism:** • Runtime path prediction method based on data mining techniques • CP and CCP heuristic optimization algorithms
3	Evaluating the approach performance	**Evaluation of runtime path prediction method:** • Identifying a data mining tool • Preparing datasets • Identifying the machine learning algorithms • Identifying the evaluation measures • Identifying the performance estimation method • Conducting a set of experiments **Evaluation of runtime path prediction method:** • Identifying the evaluation measures • Identifying the evaluation methods • Developing a new prototype software • Implementing the optimization mechanism • Conducting a set of experiments	• WEKA tool • 10 datasets • J48, NB, and SMO algorithms • Prediction accuracy, precision, recall, and number of correctly/incorrectly classified instances. • 10-fold cross validation method • Evaluate the accuracy and scalability of the runtime path prediction method • The QoS ratio, the constraints violated number, and the computation time • Separate path optimization technique and all paths optimization technique • New simulation prototype • Finalize the algorithms • Evaluate the optimization mechanism

2. The second step aims at developing the multiple paths QoS-aware service composition approach to solve the problems identified in the reviewed approaches. The development includes defining the problem and the structure model, determining the selection criteria, describing the QoS computation for web service composition, defining the utility function, and developing a new optimization mechanism to solve the research problems. The development of optimization mechanisms includes proposing a runtime path prediction method which is based on data mining techniques, mapping the QoS-aware service composition problem to MMKP, and applying CP and CCP heuristic optimization algorithms.

3. The third step aims at conducting performance evaluation of the approach. The evaluation is divided into two parts. The first part aims at evaluating the runtime path prediction method. For this purpose, the data used for evaluation were collected and prepared. Beside the data, data mining tool, machine learning algorithms, evaluation measures, and a performance estimation method were identified. Finally, a set of test experiments was introduced. The second part aims at evaluating the optimization mechanism. For this purpose, the measures and the methods used for the evaluation were determined. A new simulation prototype was developed and the optimization mechanism was implemented. Finally, a set of test experiments was introduced.

The details of each step of the research methodology are described in the next sections.

3.2 **Analyzing the Research Problem**

The first step was studying the different methods used to develop the composite services. The focus was particularly on the QoS-aware service composition method. This is due to the rapid growth of the number of the available functionality equivalent web services over the Internet which results in the need for QoS as selection criteria to differentiate between those competing services.

This step also involved understanding the QoS-aware service composition problem and determining the state of the art approaches that were proposed to solve it. In these approaches, special focus was given to study and analyze the QoS characteristics considered in these works, the optimization algorithms adopted to solve the optimization problem, and the strategies used to tackle multiple paths compositions (i.e., the cases when conditional structures are accommodated in compositions). In this step, the current works were criticized and the weaknesses were determined in order to illustrate and frame the gaps. Based on the analysis, the major drawbacks in the state of the art approaches were determined and the problems of the research were clearly defined. In addition, the components needed to develop the approach for multiple paths QoS-aware service composition were identified. The QoS characteristics that can be considered for web service composition are determined and prioritized.

Also, in this step a review was conducted on the state of the art approaches proposed to solve the MMKP problem. This led to identify new algorithms which solved the optimization problem. In addition, a review was conducted on the techniques and the approaches used for path prediction. Based on this study, new optimization algorithms called CP and CCP were identified to solve the optimization problem, and data mining based technique was identified to develop the runtime path prediction method. A detailed outcome of this process has been presented in chapter two.

3.3 **Multiple Paths QoS-Aware Service Composition Approach**

This section presents a description of the models, methods, algorithms, tools, and equations used to develop the multiple paths QoS service composition approach.

3.3.1 The Problem and Composition Structure Model

The development of the approach was launched with formulating the problem model of the QoS-aware service composition. The model allows for mapping the problem for multi-dimensional, multi-choice knapsack problem (MMKP). In this model, it is assumed that there is a set of abstract service classes. For each class, there is a set of functionality equivalent candidate service that can execute the abstract service. For each candidate, a QoS vector is assigned. Finally, a vector is used to represent the global QoS constraints imposed by clients.

Also, the definition of the composition structure model includes the definition of the execution path and predicted path concepts. The structural model is based on the sequential and conditional structures. The model allows performing the path prediction method.

A detailed explanation about the definition of the problem and the structure models is given in Section 5.2 and Section 5.3 respectively.

3.3.2 The Selection Criteria

By reviewing the QoS characteristics that are considered in the field of web services and SOA, it is seen that there is no standard or formal QoS model used for web service composition. Therefore, the QoS characteristics that were most commonly used in these approaches were investigated and analyzed in order to determine the relevant set of QoS characteristics that can be considered as selection criteria when composing web services. QoS characteristics for a web service composition should be derived from the features of composite services which are different from a single service. The selection is made by counting the frequency of the characteristics which have been considered in the related works in the field of web services and SOA putting into consideration their implicit importance despite of being scarcely included in these studies. As a result, eight characteristics were suggested, namely cost, response time, availability, reliability, throughput, security, reputation, and composability.

Furthermore, when multiple criteria are considered at once for optimization, clients might have difficulties in choosing the right criteria. In order to assist clients when assigning weights, prioritizing the selected characteristics is suggested. A detailed explanation is presented in chapter four.

3.3.3 QoS Computation for Web Service Composition

This step of the approach's development includes the definition of the aggregation functions that are used to compute the overall QoS of a composition. Aggregating the overall QoS of a composition is needed to compute the optimization. The aggregation functions which were used are similar to those proposed by Zeng et al. (2004), Jaeger et al. (2004), and Guoping et al. (2009). The aggregation functions are described in detail in Section 5.5.

3.3.4 The Utility Function

An aggregated goal function is required to consider the different QoS characteristics that are subject to optimization. The function is used to compare between the services when an optimization algorithm tries to solve the optimization problem. The defined function based on a Simple Additive Weight (SAW) method which was introduced in the context of Multiple Criteria Decision Making (MCDM) (Yoon & Hwang, 1995). The method is performed in two phases, namely a scaling phase and a weighing phase. In the scaling phase, the values of different QoS characteristics are scaled to a range from 0 to 1, where the 0 value indicates a worse quality while 1 value indicates a better one. In the weighting phase, all QoS characteristics are weighted by their importance. A detailed explanation about SAW method and utility function is given is Section 5.6.

3.3.5 New Optimization Mechanism

In the proposed approach, the optimization mechanism consists of a runtime path prediction method for the purpose of predicting the execution path and optimization algorithms for optimizing the predicted path.

3.3.5.1 Prediction of Execution Path

The existing techniques used for predicting the execution path have been reviewed in Section 2.6. The section suggested using data mining techniques for the purpose of prediction.

In order to perform prediction using data mining, a composition log is needed. In workflow-based web service composition systems, the data generated from the execution of business processes are usually recorded in the so-called execution logs. Logs store data that are rich with hidden information. One useful and important piece of knowledge that can be extracted from these logs is the path that will potentially be executed at runtime. During the execution of a business process (i.e., a composite service), workflow-based web service composition systems store data in logs, including real time information describing the execution and the behavior of the composite service, web services, and instances.

However, such data are not enough to perform prediction. Cardoso (2005, 2008) and Cardoso and Lenic (2006), who propose a method based on data mining that allows predicting the QoS of workflows, suggest extending the log in order to record extra data that are required to perform prediction. These data are runtime generated information indicating the input (output) values parameters passed (received) to (from) web services and their types. These values are generated at runtime during the execution of composition instances. Each ‘parameter/value’ entry is a data type, a name, and a value, (for example, int loannumber=12323). Furthermore, an extra field needs to be added to the log in order to store execution path information which describes the path that has been taken during the execution of a composite service (i.e., the set of web services that has been executed). The path field is associated in order to determine whether the paths that have been taken might be influenced by the runtime information or not.

Table 3.2 illustrates an example of an extended composition log which includes the parameter/value and path as extra fields. This work adopted the idea of extending the logs in order to perform runtime path prediction based on the data provided by the composite service requester as discussed later in chapter five.

Table 3.2

An Example of Extended Composition Log

.. Instance	Web service	Instance	Parameter/value	Path
.. LA112	RejectHome Loan	RHL01	Int loannumber= 1232; string email=’ ali@yahoo.com’ string loantype= ’home-loan’	FillLoanRequest, CheckLoanType, CheckHomeLoan, RejectHomeLoan
.. LA112	Archive Application	NU22	string tel= ’1626354’; string email= ’ali@hotmail.com ’	FillLoanRequest, CheckLoanType, CheckHomeLoan, RejectHomeLoan, NotifyHomeLoan Client, ArcheiveApplication
. ….	….	….	….	….

(Source: Cardoso, 2008)

Using these data, datasets can be created and applied to machine learning algorithms to perform prediction. The learning algorithms are described in detail in Section 3.4.15.

3.3.5.2 The Computation of Optimization

To solve the selection problem, the solution is to apply heuristic algorithms. To do this, the approach maps the selection problem to MMKP due to the similarity between these two problems (Yu, Zhang & Lin, 2007; Alrifai et al., 2009) as explained in the next subsection. Then it selects algorithms that are known to be efficient for solving MMKP and applies it to solve the selection problem.

Mapping to Multidimensional Multi-choice Knapsack Problem (MMKP)

Definition 1: (MMKP) (Hifi et al., 2004):

Suppose there are n classes J_i of items, each class J_i, $i=1,...,n$, has r_i items. Each item j, $j=1,...,r_i$, of class J_i has the non-negative profit value v_{ij}, and requires resources given by the weight vector $W_{ij} = (W_{ij}^{1}, W_{ij}^{2}, ..., W_{ij}^{m})$, where each weight component W_{ij}^{k}, $k=1,...,m$, also is a non-negative value. The amounts of available resources are given by a vector $C = (C^1, C^2, ..., C^m)$.

The aim of the MMKP is to select exactly one item from each class in order to put them into a knapsack. Each item has a profit value, a weight, and the knapsack has a limited amount of resources. The amount of resources for the knapsack does not allow taking all items. Thus, it is required to perform a selection to identify the optimal items which maximize the total profit value that is subject to resource constraints. On the other hand, the selection problem aims to select exactly one candidate from each service class, where the entire QoS value of the composition is optimized while QoS requirements defined by clients are satisfied.

Building on the similarity between these two problems, the selection problem can be mapped to MMKP as in the following 1-6 steps (Yu et al., 2006; Alrifai, Risse, Dolog & Nejdl, 2009). A detailed explanation about mapping the selection problem to MMKP is given in Section 5.8.2.1.

1. The knapsack is represented by the composition.
2. Each service class represents a class or an object group.
3. Each candidate in a service class represents one item in a class.
4. Each utility function u_{ij} represents a non-negative profit value v_{ij} and can be calculated using Equation 5.3.
5. The QoS characteristics q_{ij} of a candidate $\boldsymbol{s_j}$ represents the required resource W_{ij} of the item.
6. The QoS global constraints GS is considered the resources available in the knapsack C.

Heuristic Algorithms for the Selection Problem

Section 2.5 discusses the solution types for MMKP problem. It concludes that heuristic represents a novel solution. Section 2.5 also reviews and analyzes the existing heuristic approaches proposed to solve the MMKP problem. In addition, it suggests the constructive and complementary search approach by Hifi et al. (2004) for solving the selection problem. In this approach, the constructive procedure (CP) is applied to generate a feasible solution while the complementary CP (CCP) is used to improve the quality of the solution generated from CP. The approach was selected to be applied because of its ability in generating quality solutions and reducing the computational efforts resulted from the problem. Also, it can be easily applied to solve the selection problem. The followings are the two algorithms (Hifi et al., 2004).

1. The Constructive Procedure (CP)

This algorithm is a greedy procedure used to generate an initial feasible solution for the MMKP problem. Two phases are included in this algorithm, namely DROP phase and ADD phase. Moreover, two states are distinguished, namely a feasible state (FS), if the current solution does not violate the amount of available constraints, and an unfeasible state (US), if the current solution violates at least one constraint. Figure 3.1 shows the flow of the algorithm.

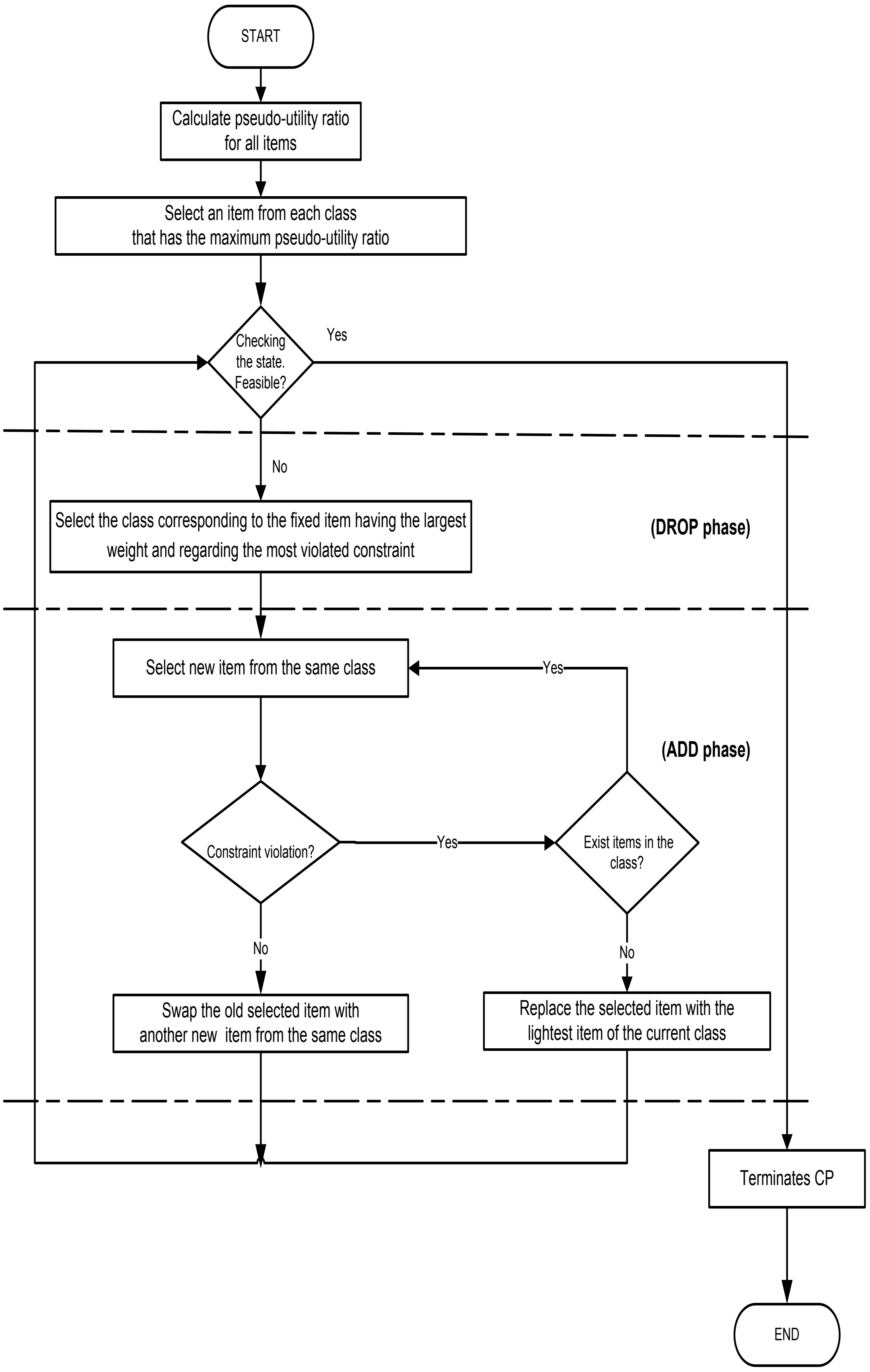

START
Calculate pseudo-utility ratio for all items
Select an item from each class that has the maximum pseudo-utility ratio
Checking the state. Feasible?
Yes
No
Select the class corresponding to the fixed item having the largest weight and regarding the most violated constraint
(DROP phase)
Select new item from the same class
Yes
(ADD phase)
Constraint violation?
Yes
Exist items in the class?
No
No
Swap the old selected item with another new item from the same class
Replace the selected item with the lightest item of the current class
Terminates CP
END

Figure 3.1. *Flowchart for CP*

As seen in Figure 3.1, the algorithm starts by calculating for each item the pseudo-utility ratio $u_{ij} = v_{ij} / \left\langle C, W_{ij} \right\rangle$; v_{ij} is the item profit, C the amounts of available resources, W_{ij} is the required resources, and $\langle \cdot, \cdot \rangle$ is the scalar product in R^m. Then it selects the item which has the maximum u_{ij} from each class as an initial solution. After that, the state of the obtained solution is checked; if it is a feasible solution, CP terminates. Else (DROP Phase) it determines the most violate constraint. With respect to the most violated constraint, the algorithm selects the class corresponding to the fixed item which has the largest weight all over the fixed items. (ADD Phase) The selected item is swapped with another item from the same class. Then the feasibility of the new obtained solution is checked; if is not feasible, it selects the lightest item of the current class which in turn is considered the new selected item. Finally, the algorithm iterates until a feasible solution or the smallest infeasibility amount is obtained.

2. **The Complementary CP (CCP)**

CCP is used to iteratively improve the initial feasible solution obtained by CP. The algorithm applies a local swap strategy for the selected item (called old item), and a replacement stage, that replaced the old item with another new one, called a new item, is selected from the same class. Each replacement between an old item and a new one is authorized if, and only if, the solution newly obtained realizes a FS and its value is better. Figure 3.2 shows the flow of the algorithm.

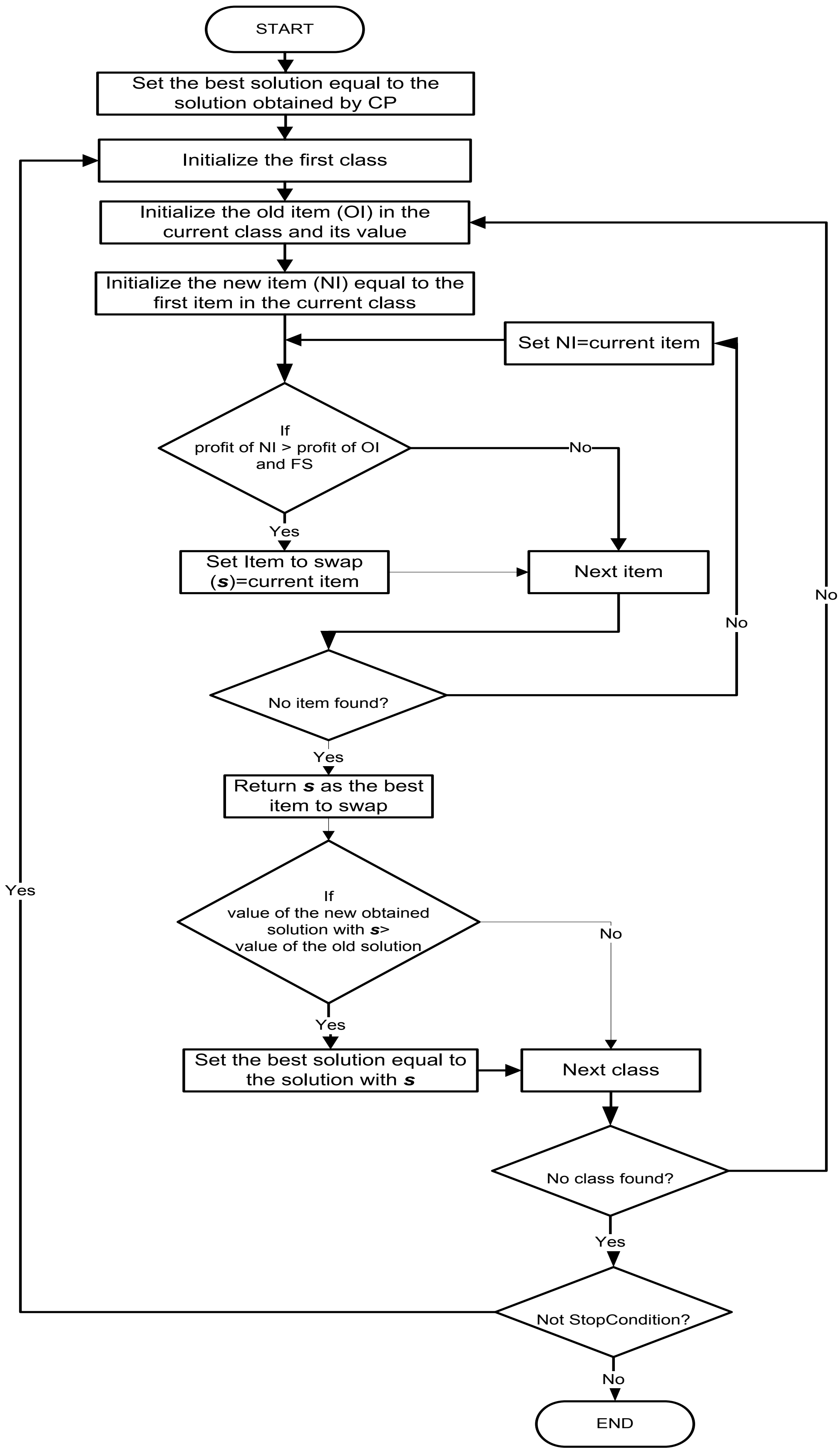

START
Set the best solution equal to the solution obtained by CP
Initialize the first class
Initialize the old item (OI) in the current class and its value
Initialize the new item (NI) equal to the first item in the current class
Set NI=current item
If profit of NI > profit of OI and FS
No
Yes
Set Item to swap (s)=current item
Next item
No
No
No item found?
Yes
Return s as the best item to swap
Yes
If value of the new obtained solution with s> value of the old solution
No
Yes
Set the best solution equal to the solution with s
Next class
No class found?
Yes
Not StopCondition?
No
END

Figure 3.2. *Flowchart for CCP*

As seen in Figure 3.2, the algorithm starts by initializing the best solution that is equal to the solution obtained by CP. Then the loop starts by performing a local swap search strategy procedure in order to improve the initial solution. If the obtained solution realizes a better solution value compared to the initial one, then the algorithm sets the best current solution that is equal to the obtained one. The loop is repeated until no more classes remain. This process is iterated by using a stopping condition.

3.4 **Evaluation of the Proposed Approach**

In order to evaluate the performance of the proposed approach, the evaluation process is divided into two parts. The first part aims at evaluating the runtime path prediction method while the second one aims at evaluating the new optimization mechanism.

3.4.1 Evaluation of Runtime Path Prediction Method

The following sections present the experimental procedure, data and dataset preparation, data mining tool, machine learning algorithms, evaluation measures, evaluation method, and different experiments used for the purpose of evaluating the runtime path prediction method.

3.4.1.1 An Experiment Design

The experimental procedure for evaluating the runtime path prediction method is presented in Figure 3.3. As seen in Figure 3.3, runtime data about instances for auto insurance and bank loan processes were collected. Based on this data, 10 datasets were created. Using WEKA, data preprocessing was conducted for data reduction (i.e., “Select attributes” technique for identifying the most important attributes in a dataset) and data transformation (i.e., covert from string to nominal) (Hall et al., 2009). Then a 10-fold cross-validation method was used to train and test the machine learning algorithms. Finally, the performance measures (i.e., accuracy, correctly classified instances, etc.) were collected.

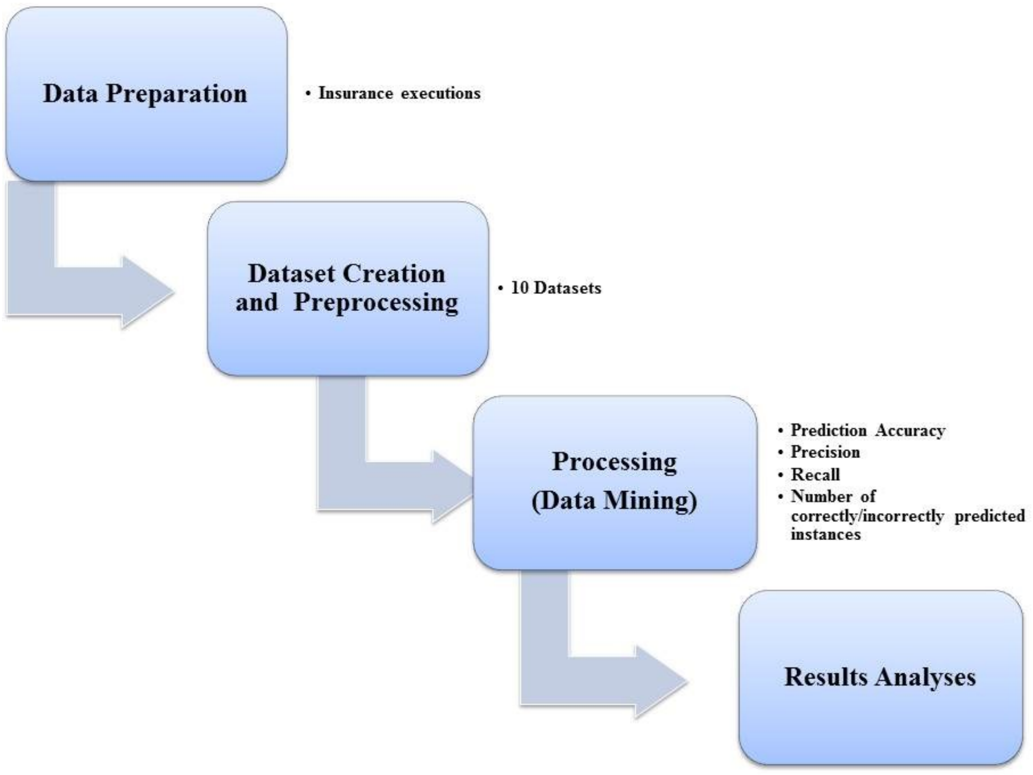
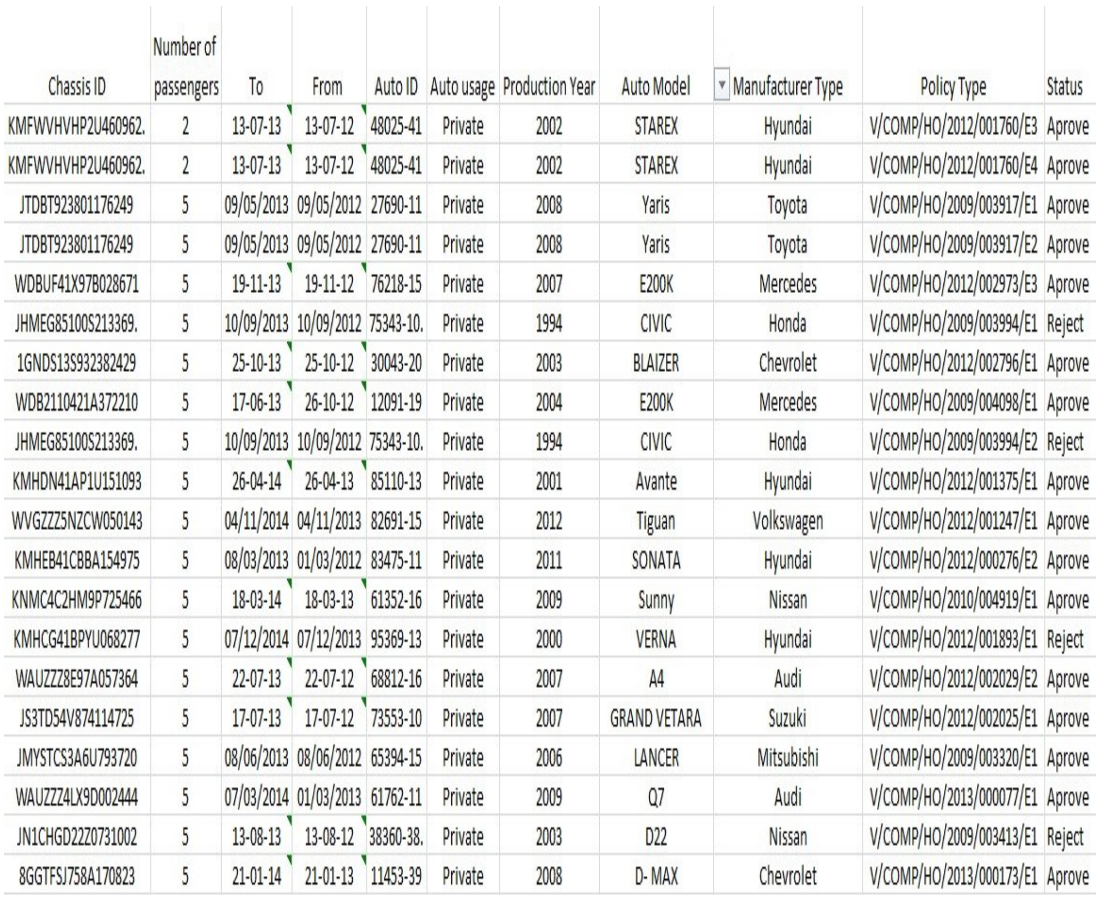


Figure 3.3. *The main steps for the experimental procedure used for evaluating the runtime path prediction method*

3.4.1.2 Data Preparation

This work considers two processes for evaluating the accuracy and scalability of the proposed method, namely auto insurance and bank loan. Furthermore, in order to perform and evaluate the runtime path prediction method, it is required to have logs which include data about the process instances. A process instance represents one specific instance of a process that is currently executing. It contains all runtime data related to that instance.

In auto insurance process, a set of 826 instances (i.e., all runtime data related to the auto insurance requesters) was collected from the First Insurance Company. The company was established in Jordan in 2006, with a paid up capital of JOD 24 million (the second highest capitalized insurer in Jordan), as a General Insurance Company providing Insurance Services that are based on Islamic principles (Takaful) (http://www.firstinsurance.jo/index.html). The data include chassis ID, number of passengers, insurance period (from/to), auto ID, an auto usage, a production year, an auto model, a manufacture type, a policy type, and the decision that has been made for each instance when evaluating the request for particular policy (i.e., reject or approve). Figure 3.4 shows a fragment of the collected insurance data.



Chassis ID	Number of passengers	To	From	Auto ID	Auto usage	Production Year	Auto Model	Manufacturer Type	Policy Type	Status
KMFWVHVHP2U460962.	2	13-07-13	13-07-12	48025-41	Private	2002	STAREX	Hyundai	V/COMP/HO/2012/001760/E3	Aprove
KMFWVHVHP2U460962.	2	13-07-13	13-07-12	48025-41	Private	2002	STAREX	Hyundai	V/COMP/HO/2012/001760/E4	Aprove
JTDBT923801176249	5	09/05/2013	09/05/2012	27690-11	Private	2008	Yaris	Toyota	V/COMP/HO/2009/003917/E1	Aprove
JTDBT923801176249	5	09/05/2013	09/05/2012	27690-11	Private	2008	Yaris	Toyota	V/COMP/HO/2009/003917/E2	Aprove
WDBUF41X97B028671	5	19-11-13	19-11-12	76218-15	Private	2007	E200K	Mercedes	V/COMP/HO/2012/002973/E3	Aprove
JHMEG85100S213369.	5	10/09/2013	10/09/2012	75343-10.	Private	1994	CIVIC	Honda	V/COMP/HO/2009/003994/E1	Reject
1GNDS13S932382429	5	25-10-13	25-10-12	30043-20	Private	2003	BLAIZER	Chevrolet	V/COMP/HO/2012/002796/E1	Aprove
WDB2110421A372210	5	17-06-13	26-10-12	12091-19	Private	2004	E200K	Mercedes	V/COMP/HO/2009/004098/E1	Aprove
JHMEG85100S213369.	5	10/09/2013	10/09/2012	75343-10.	Private	1994	CIVIC	Honda	V/COMP/HO/2009/003994/E2	Reject
KMHDN41AP1U151093	5	26-04-14	26-04-13	85110-13	Private	2001	Avante	Hyundai	V/COMP/HO/2012/001375/E1	Aprove
WVGZZZ5NZCW050143	5	04/11/2014	04/11/2013	82691-15	Private	2012	Tiguan	Volkswagen	V/COMP/HO/2012/001247/E1	Aprove
KMHEB41CBBA154975	5	08/03/2013	01/03/2012	83475-11	Private	2011	SONATA	Hyundai	V/COMP/HO/2012/000276/E2	Aprove
KNMC4C2HM9P725466	5	18-03-14	18-03-13	61352-16	Private	2009	Sunny	Nissan	V/COMP/HO/2010/004919/E1	Aprove
KMHCG41BPYU068277	5	07/12/2014	07/12/2013	95369-13	Private	2000	VERNA	Hyundai	V/COMP/HO/2012/001893/E1	Reject
WAUZZZ8E97A057364	5	22-07-13	22-07-12	68812-16	Private	2007	A4	Audi	V/COMP/HO/2012/002029/E2	Aprove
JS3TD54V874114725	5	17-07-13	17-07-12	73553-10	Private	2007	GRAND VETARA	Suzuki	V/COMP/HO/2012/002025/E1	Aprove
JMYSTCS3A6U793720	5	08/06/2013	08/06/2012	65394-15	Private	2006	LANCER	Mitsubishi	V/COMP/HO/2009/003320/E1	Aprove
WAUZZZ4LX9D002444	5	07/03/2014	01/03/2013	61762-11	Private	2009	Q7	Audi	V/COMP/HO/2013/000077/E1	Aprove
JN1CHGD22Z0731002	5	13-08-13	13-08-12	38360-38.	Private	2003	D22	Nissan	V/COMP/HO/2009/003413/E1	Reject
8GGTFSJ758A170823	5	21-01-14	21-01-13	11453-39	Private	2008	D- MAX	Chevrolet	V/COMP/HO/2013/000173/E1	Aprove

Figure 3.4. *A fragment of the auto insurance data collected from First Insurance Company*

The prediction accuracy of the path prediction method is an important measure. Therefore, it is important to evaluate the accuracy using a real data to be confident about the results. Thus, the auto insurance data (i.e., real data) is used for evaluating the accuracy of the path prediction method.

For bank loan data, the first choice was to contact different Jordanian banks requesting for runtime data about instances for the loan processes. Unfortunately, because the data is confidential, the request was rejected; therefore, the intention was turned to generate the data. The data needed to be generated describes personal information about the requesters and the loan being requested. In addition, the decision that has been made by the bank i.e., either rejected or approved the loan requests. Specifically, the data needed to be generated include income, loan type, loan amount, loan year, and the decision that has been made by the bank. The generated data are used for the purpose of studying how the prediction method will scale with a rising number of involved execution paths.

The value range of income is generated based on a study by Bayt (2013), which is the leading job site in the Gulf and Middle East. The study shows that the salary range in Jordan is between 155 up to 15024 Dollars with an average of 1033 Dollars. Regarding the loan type, loan amount, and loan year, these attributes can vary between a bank and another and depend on the bank loan policy. The value ranges of these attributes were obtained from Arab Bank (http://www.arabbank.jo/). Arab bank is one of the largest financial institutions in the Middle East and is the largest global Arab banking network with over 600 branches in 30 countries spanning five continents. The value ranges are presented in Table 3.3. The value ranges of loan amount were converted from Jordan Dinner to Dollars.

Table 3.3
Value Ranges of Loan Type, Loan Amount, and Loan Year Attributes

Loan type	Loan amount		Loan year
New car	1130<amount<141203	Dollars	1-6
Used car	3530<amount<98842	Dollars	1-5
Home	7060<amount<988421	Dollars	1-30
Education	1412<amount<1994	Dollars	1-4
Personal	1130<amount<98842	Dollars	½-8

Regarding the bank decision, every bank has their own loan policies used to determine if a loan request is approved or rejected. The credit approval of Arab Bank policy is based on maintaining the (DBR), and minimum required salary at manageable levels (Arab Bank, 2012). For simplicity, it is assumed that the decision is preliminary, and based on the income, loan interest, a set of standard formulas, and rules which were collected from Arab Bank. For example, the formulas used for calculating the monthly payment and DBR are:

$$monthly\ payment = (rate + \frac{rate}{((1+rate)^{month} - 1)\ )}) \times loan\ amount \tag{3.1}$$

where,

$$rate = \frac{interest\ rate}{12} \tag{3.2}$$

$$DBR = \frac{income}{2} \tag{3.3}$$

The DBR should be equal or less than the monthly payment. Using the collected value ranges and formulas, much data about instances for bank loan process was generated.

The collected/generated data about auto insurance and bank loan processes was used for creating several datasets for the purpose of evaluating the path prediction method. Section 6.1.2 explains the process of creating the datasets based on the collected data.

3.4.1.3 The Data Mining Tool

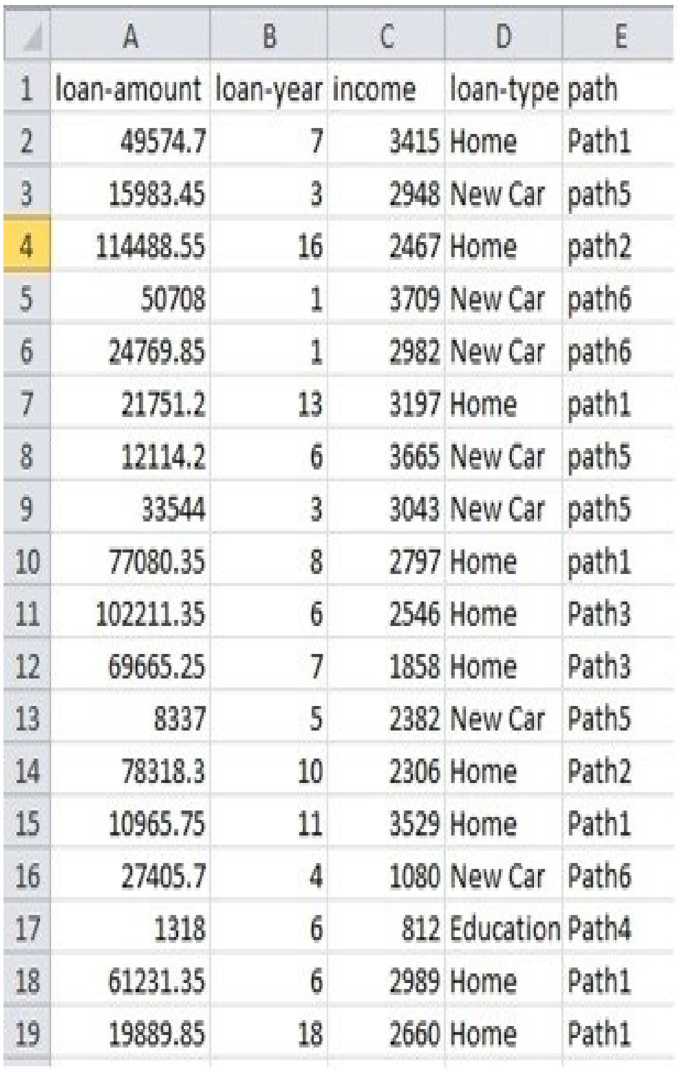
The experiments were conducted utilizing Waikato Environment for Knowledge Analysis (WEKA) (http://www.cs.waikato.ac.nz/ml/weka/). WEKA is open source software developed at the University of Waikato in New Zealand. The system is written in Java Language to support several data mining tasks. WEKA includes a collection of tools and the state of the art machine learning algorithms for data analysis and predictive modeling (Witten & Frank, 2005).

Academically, WEKA becomes a widely used tool for data mining research (Hall et al., 2009). It provides implementations for J48, NB, and SMO as well as wide varieties of learning algorithms (Witten & Frank, 2005). Thus, there is no need to manually write the algorithms' code. In a simple way, the algorithms are easily applied to the datasets created from the log data. WEKA is also used for analyzing and statistically evaluating the result obtained from applying the learning algorithms to the datasets, making it easy to compare between the performances of the learning algorithms (Witten & Frank, 2005).

3.4.1.4 Datasets Preparation for WEKA

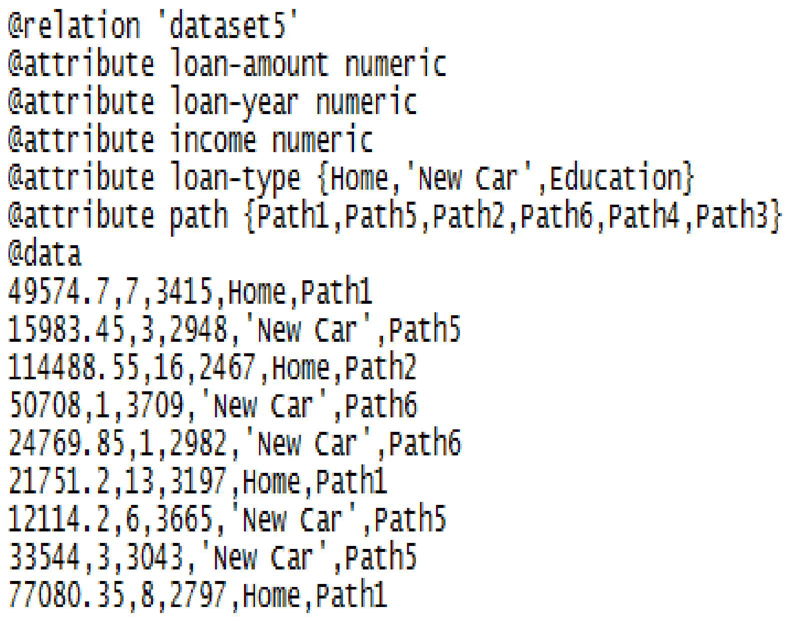
As the WEKA data mining tool is used for experiments, an attribute-relation file format (ARFF) is used in WEKA to represent datasets. An ARFF file consists of a list of the independent instances, and the attribute values for each instance are separated by commas.

In order to create datasets in ARFF format, a Microsoft Excel was used to generate the instances and the attributes. Figure 3.5 shows a fragment of dataset5 which was created by using Microsoft and contained a number of instances. Then the files were saved as a comma-separated value (CSV) format. Once datasets were saved into CSV format, it can be easily converted into the ARFF format by loading the files into a Microsoft Word; add the dataset name using the @relation tag, the attribute information using @attribute, and a @data line, and save the file as raw text (Witten & Frank, 2005). Figure 3.6 shows dataset5 after converting it to ARFF format.



	A	B	C	D	E
1	loan-amount	loan-year	income	loan-type	path
2	49574.7	7	3415	Home	Path1
3	15983.45	3	2948	New Car	path5
4	114488.55	16	2467	Home	path2
5	50708	1	3709	New Car	path6
6	24769.85	1	2982	New Car	path6
7	21751.2	13	3197	Home	path1
8	12114.2	6	3665	New Car	path5
9	33544	3	3043	New Car	path5
10	77080.35	8	2797	Home	path1
11	102211.35	6	2546	Home	Path3
12	69665.25	7	1858	Home	Path3
13	8337	5	2382	New Car	Path5
14	78318.3	10	2306	Home	Path2
15	10965.75	11	3529	Home	Path1
16	27405.7	4	1080	New Car	Path6
17	1318	6	812	Education	Path4
18	61231.35	6	2989	Home	Path1
19	19889.85	18	2660	Home	Path1

Figure 3.5. *An illustration of dataset5 created by using a spreadsheet application*



```
@relation 'dataset5'
@attribute loan-amount numeric
@attribute loan-year numeric
@attribute income numeric
@attribute loan-type {Home,'New Car',Education}
@attribute path {Path1,Path5,Path2,Path6,Path4,Path3}
@data
49574.7,7,3415,Home,Path1
15983.45,3,2948,'New Car',Path5
114488.55,16,2467,Home,Path2
50708,1,3709,'New Car',Path6
24769.85,1,2982,'New Car',Path6
21751.2,13,3197,Home,Path1
12114.2,6,3665,'New Car',Path5
33544,3,3043,'New Car',Path5
77080.35,8,2797,Home,Path1
```

Figure 3.6. *A fragment of dataset5 in ARFF format*

3.4.1.5 Machine Learning Algorithms

Different supervised learning algorithms can be used to carry out path prediction. Among these algorithms, Naïve Base (NB), J48 which is Weka's (2004) implementation of the C4.5 (Quinlan, 1993) decision tree learner, and Sequential Minimal Optimization (SMO) (i.e., an improved training algorithm for Support Vector Machines (SVM)) methods were selected to be experimented. These algorithms are among the most influential and the best-known algorithms in the data mining community (Kotsiantis, 2006; Wu et al., 2008).

J48 algorithm is Weka's (2004) implementation of the C4.5 (Quinlan, 1993) decision tree learner. Instances to be classified, a decision tree sorts them on the basis of feature values. In a decision tree, each node represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values (Kotsiantis, 2006). It uses a heuristic approach to generate suboptimal decision trees because finding an ‘optimal’ solution tree is a multi-objective problem. NB classifier technique is based on the so-called Bayesian theorem. It is done by analyzing the relationship between the dependent variable and the independent variable, and for each relationship, a conditional probability is derived. SMO (Platt, 1999) is an improved training algorithm for SVM (Cortes & Vapnik, 1995). A very large quadratic programming (QP) problem of the solution is usually required to train SVM. SMO breaks down a large QP problem into a series of smaller QP problems. SMO improves its scaling and computation time significantly because the utilization of the smallest possible QP problems is solved quickly.

3.4.1.6 The Performance Evaluation

The runtime path prediction method is considered successful if the prediction accuracy is high. The prediction accuracy of classifiers is usually the most important evaluation measure (Masseglia, Poncelet & Teisseire, 2008). It is established to determine how accurate a classifier is in the prediction. In this work, the prediction accuracy is the primary measure for evaluating the prediction method. Beside the accuracy measure, the precision and recall criteria and the number of correctly/incorrectly classified instances are also considered.

Classifiers' accuracy is defined as the probability of correctly classifying a randomly selected instance. Precision is a measure of the accuracy provided that a specific class has been predicted (Norinder, Lidén & Boström, 2006). Recall is a measure of the ability of a prediction model to select instances of a certain class from a dataset (Jacobsson, Lidén, Stjernschantz, Boström & Norinder, 2003).

These measures are derived from the confusion matrix. The results on a test datasets are usually displayed as a confusion matrix of rows and columns (Witten & Frank, 2005). The rows correspond to the known class while the columns correspond to the predictions made by the classifier. Table 3.4 shows a confusion matrix for a problem of two classes.

Table 3.4

A Confusion Matrix of Two Classes

		Predicted Class	
		Positive	Negative
Known Class	Positive	True Positive (TP)	False Negative (TN)
	Negative	False Positive (FP)	True Negative (TN)

In relation to the confusion matrix, the accuracy, precision, and recall measures are calculated as in the following:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3.4}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{3.5}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{3.6}$$

Concerning accuracy’s estimation, a review of estimation methods was conducted by Kohavi (1995). The author recommends a 10-fold cross-validation method for accuracy’s estimation. The method proved to be statistically good enough to evaluate the performance of the classifiers (Witten & Frank, 2005).

In this method, the dataset is split into 10 mutually executive subsets of approximately equal size. A machine learning algorithm is trained and tested 10 times; at each time it is tested on 1 of the 10 subsets and trained using the 9 remaining subsets (i.e., each subsets being once the test set and reaming subsets being the training set). The iteration is necessary to ensure that all instances in the dataset are part of the test and train subsets. The 10 results are then averaged to give the overall result (Prekopcsák et al., 2010). Figure 3.7 graphically illustrates the 10-fold cross-validation using a dataset which consists of 1000 instances; each subset is divided into equal size of instances i.e., 100.

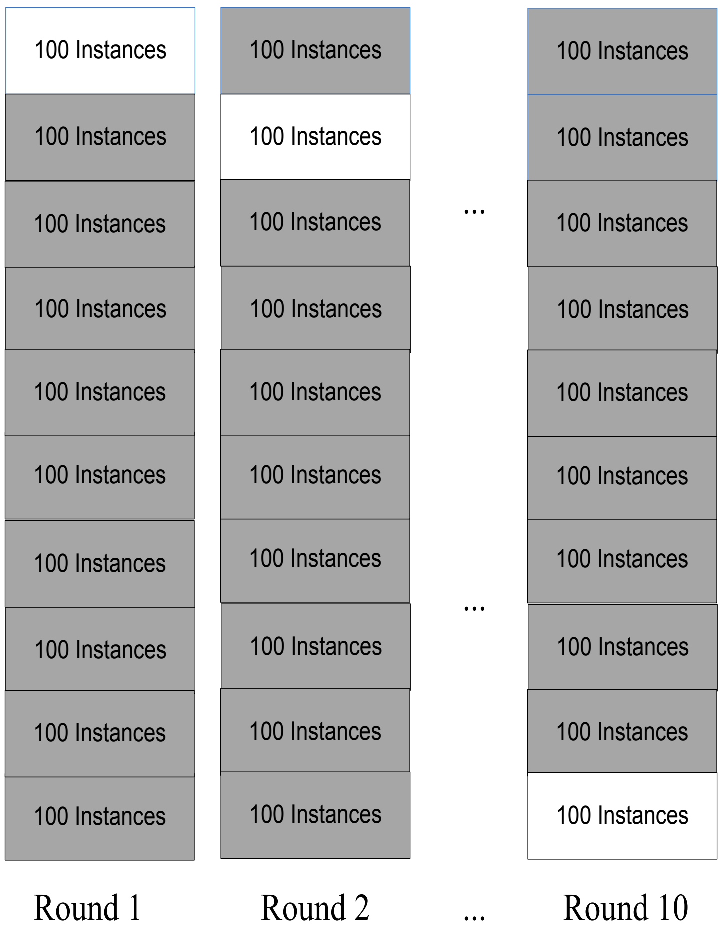


Figure 3.7. *A graphical illustration of 10-fold cross-validation method using a dataset which consists of 1000 instances, test subsets (gray), train subsets (white)*

3.4.1.7 Experiments

Different test experiments were conducted in order to evaluate the runtime path prediction method. The first test experiment aimed at validating the accuracy of path prediction. The experiment has been conducted using three selected learning algorithms, namely J48, NB, and SMO. These algorithms are applied to the auto insurance dataset which contains 826 instances. The second experiment aimed at studying how the prediction method will scale with a rising number of involved execution paths. For this purpose, 9 datasets of the loan process were used. Each dataset contained an equal number of instances i.e., 1000 representing a loan process that involved execution paths ranging from 2 up to 10 paths. J48, NB, and SMO algorithms are applied on these 9 datasets.

3.4.2 Evaluation of the Optimization Mechanism

The following sections present the experiment procedure, simulation prototype, evaluation measures, evaluation methods, and different test experiments used to evaluate the optimization mechanism.

3.4.2.1 An Experimental Design

Performing a simulation test experiment is divided into three main steps as outlined in Figure 3.8:

1. Generate problem instances.
2. Let the implementations of the three algorithm versions solve these instances.
3. Evaluate the resulting aggregated QoS ratio, the calculated constraints violated number, and the computation time.

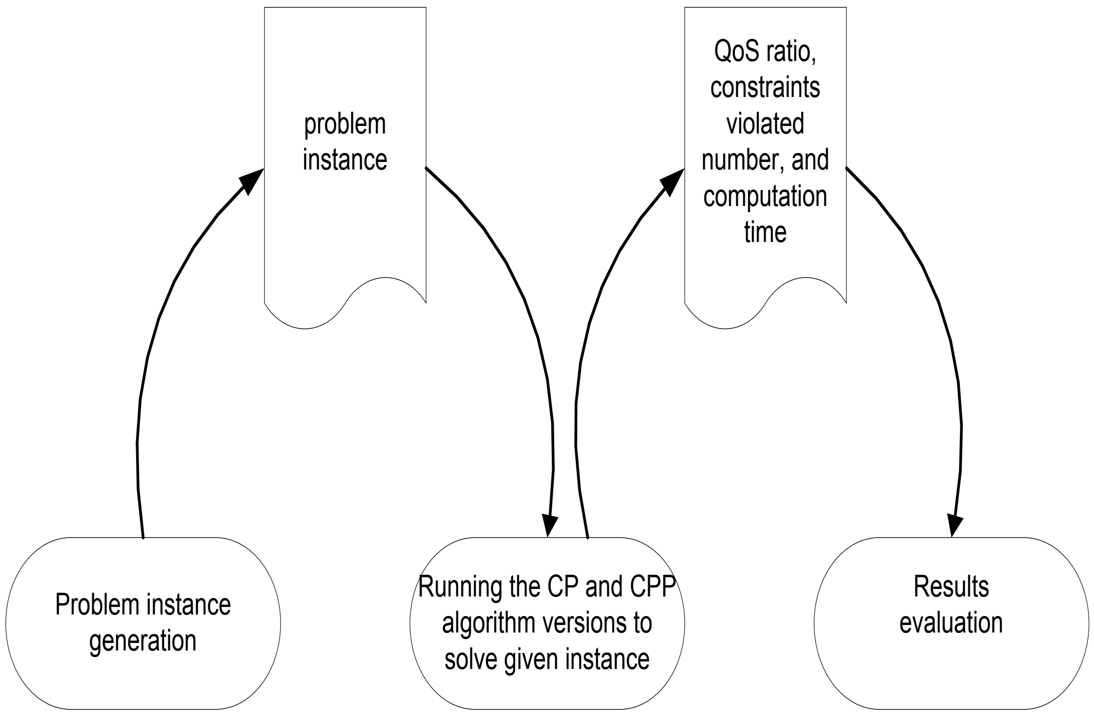

problem instance
QoS ratio, constraints violated number, and computation time
Problem instance generation
Running the CP and CPP algorithm versions to solve given instance
Results evaluation

Figure 3.8. *The main steps for the experimental procedure used for evaluating the optimization mechanism*

3.4.2.2 Simulation Prototype

For evaluating the performance of the proposed optimization mechanism, the goal was to compare the proposed optimization mechanism, which combined a path prediction method and heuristic algorithms, with the current existing optimization techniques that has been proposed to solve the problem of multiple paths composition. To do so, a new simulation prototype that simulates the proposed approach for multiple paths QoS-aware service composition has been developed. The simulation prototype provides the implementations for the proposed mechanism and the current techniques. Section 6.2.1 discussed the simulation prototype in detail.

3.4.2.3 The Evaluation Measure

As explained in the section of the problem background, this work claims that the current optimization techniques, that are proposed to handle the problem of multiple execution paths composition, have several drawbacks. First, they generate solutions that may not have the best possible QoS ratio. Second, the generated solutions which use these techniques may have high constraints violated number. The work also claims that by using the proposed optimization mechanism, it is expected to generate solutions that deliver the best possible QoS ratio. At the same time, it minimizes the constraints violated number. Based on the above mentioned, it was logic to verify these claims by conducting a comparison between the current optimization techniques, which will be introduced in the next section, and the proposed optimization mechanism in terms of (1) the aggregated QoS resulting from a solution and (2) the calculated constraint violations number.

On the other hand, one can claim that the proposed mechanism may have high computation time compared to current optimization techniques. This is because of the combination between the runtime path prediction method and the optimization algorithms. Thus, one extra test experiment was required to compare all the techniques in terms of the computation time consumed to generate solutions.

Briefly, for the purpose of comparison, the performance of the optimization techniques covers three measures: (1) the QoS ratio, (2) the constraints violated number, and (3) the computation time.

3.4.2.4 Evaluation Methods

Three different optimization techniques were selected for the performance of the evaluation. The first technique represents the optimization mechanism proposed in this thesis. The other two techniques are determined after reviewing the state of the art approaches. These approaches were discussed in detail in Section 2.4.2. The techniques are:

All Paths Optimization Technique: In this technique, the optimization is computed assuming that a certain path will be more likely executed than others according to the probability of path execution. The assumptions are based on stochastic information indicating the probability of paths being executed at runtime. Estimation of the paths probability of executions is estimated either by inspecting the system logs or by being specified by the composition engineers.

A Separate Path Optimization Technique: In this technique, a composite service is decomposed into execution paths in order to optimize each path separately. Then after completing the optimization process, the execution paths are aggregated into an overall composition that consists of all paths. If there is a common abstract service that belongs to more than one path, the system identifies the hot path for the considered web service. The hot path is defined as the path that has been most frequently used to execute the considered service.

For fair comparisons, it is preferable to conduct the comparisons using the same optimization approach; therefore, CP and CCP algorithms are used for this purpose. The simulations implement three different versions of the CP and CCP algorithms. Each version represents a particular optimization technique for handling multiple paths composition problem. Based on these techniques, three different versions of the CP and CCP algorithms are implemented, namely CP1 & CCP1, CP2 & CCP2, and CP3 & CCP3.

CP1 & CCP1 algorithms: this version represents the optimization mechanism proposed in this work. The main ideas of these algorithms are using the runtime path prediction method to predict the execution path that will potentially be executed and running the CP and CCP algorithms to optimize the predicted execution path.

CP2 & CCP2 algorithms: this version represents all paths optimization technique. The main idea of these algorithms is to optimize all execution paths based on their probability of executions.

CP2 & CCP2 have a similar structure as CP1 & CCP1 by finding a feasible solution at first, and then trying to improve the solution by using a local swap strategy and a replacement stage. However, CP2 & CCP2 are different from CP1 & CCP1 in the following aspects:

- **Solution feasibility check**: CP2 checks the feasibility of all the execution paths together while CP1 only checks the feasibility of the predicted path (EP_{pred}).
- **QoS computation**: in CP1 & CCP1, the QoS value of a solution *SN* is decided by aggregating the QoS value of the services that belong only to the predicted path. It is computed by using the aggregation functions presented in Table 5.2. However, in CP2 & CCP2, the QoS value of *SN* is computed by aggregating the QoS value of all the services that belong to all paths. For the

computation, each path $Path_i$ has a probability P_i, where the sum of all *Pi* must equal 1. The values of probability P_i are estimated using the composition log as in the following (Canfora et al., 2005; Yu, Zhang & Lin, 2007; Jafarpour & Khayyambashi, 2010; Jiang et al., 2011):

$$P_i = \frac{number\ of\ times\ execution\ \ for\ path_i}{total\ number\ of\ times\ execution\ \ for\ all\ \ paths} \tag{3.7}$$

The QoS value of a *SN* is computed using these probabilities. For example, for a composition containing two execution paths, with costs C_1 and C_2 and probabilities P_1 and P_2, the overall cost is computed in this technique as in the following way (Canfora et al., 2005; Yu, Zhang & Lin, 2007; Jafarpour & Khayyambashi, 2010; Jiang et al., 2011):

$$Q(SN^k) = P_1 \times C_1 + P \times_2 C_2$$

- **Solution improvement**: CCP2 improves the *SN* obtained by CP1 by applying a local swap strategy and a replacement stage for all the services that belong to all paths while CCP1 improves the services that belong only to the predicted path.

CP3 & CCP3 algorithms: This version represents a separate path optimization technique. The main idea of these algorithms is to optimize each path separately by decomposing the composition into execution paths.

CP3 & CCP3 have a similar structure as CP1 & CCP1 by finding a feasible solution at first, and then trying to improve the solution using a local swap strategy and a replacement stage; however, CP3 & CCP3 are different from CP1 & CCP1 in the following aspects:

- **Solution feasibility check**: CP1 only checks the feasibility of the predicted path (EP_{pred}) while CP3 checks the feasibility of each execution path separately. If there is a common abstract service that belongs to more than one path, the hot path is identified for the service and the feasibility of the hot path is only checked. The hot path is identified using the composition log as the path has been most frequently used to execute the common services in past instances of composite services. For example, in the multiple paths composite service illustrated in Figure 5.2, service S_1 be,s,ex.on paths $Path_1$, $Path_2$, and $Path_3$. Assume that the composite service has been executed 10 times. Also assume that in 6 times the execution follows the $Path_1$, and in 3 times the execution follows $Path_2$, and in 1 times the execution follows $Path_3$. This indicates that the execution path $Path_1$ is the hot path for the service S_1 since it has been more frequently used to execute the service (Zeng et al., 2003, 2004).

- **QoS computation**: In CP1 & CCP1, as it has been mentioned above, the QoS value of a solution *SN* is computed considering only the predicted path. On the other hand, in CP3 & CCP3, the QoS value is computed considering each path separately using the aggregation functions presented in Table 5.2.

- **Solution improvement**: CCP3 improves all services that belong to all paths separately while CCP1 improves the services that belong only to the predicted path.

Table 3.5 shows a summary and a comparison between the three algorithms implemented for performance evaluation.

Table 3.5

A Comparison Between the Algorithms Used for the Preformance Evaluation

Algorithm	Optimization technique	Solution feasibility check	QoS computation for composition	Solution improvement
CP1 & CCP1	optimize the predicted path by using runtime path prediction method	check the feasibility of the predicted path	composition QoS value is decided by aggregating the QoS value of the services that belong only to the predicted path	improve the services that belong only to the predicted path
CP2 & CCP2	optimize all paths together based on their probability of executions	check the feasibility of all paths together	composition QoS value is decided by aggregating the QoS value of all services belonging to all paths together by using a probability of execution value for each path	improve the services that belong to all paths
CP3 & CCP3	optimize each path separately by decomposing the composition into execution paths	check the feasibility of each execution path separately	composition QoS value is decided by aggregating the QoS value of each path separately	improve the services that belong to all paths separately

3.4.2.5 Experiments

Three different test experiments were conducted in order to evaluate the optimization mechanism. The goal of the first test experiment was to evaluate the proposed mechanism in terms of the resulting QoS ratio (i.e., the resulting utility of the generated solution). Quantitative statements that represent the scores of the QoS resulting from each technique can be achieved by using the SAW method that has been introduced in Section 5.6. The different QoS characteristics are aggregated considering equal weights. For the purpose of comparison, the total QoS ratio, the total average of QoS ratio, and the standard deviation of all competing techniques (i.e., three versions of CP and CCP algorithms) were captured.

The second test experiment aimed to evaluate the proposed mechanism in terms of the constraints violated number (i.e., the total number of constraints that have been violated for each optimization technique). Therefore, the constraint violated numbers for all competing techniques were calculated in order to compare between them. For the purpose of calculation, the aggregation functions presented in Table 5.2 were used to aggregate the value of the considered constraint characteristic. After that, the aggregated value and the imposed constraint value were compared to determine whether the constraint was violated or not (i.e., if the aggregated value is greater than the imposed constraint value, the constraint is violated. Otherwise, it is not violated). Finally, count if there is a violation.

The third test experiment aimed to evaluate the proposed optimization mechanism in terms of the computation time. Two different computation times were captured by the simulation, namely:

1. Computation time for the proposed optimization mechanism (CP1 & CCP1) which represents the time needed for computing the optimization plus the time needed for path prediction.

2. Computation time for the rest optimization techniques (CP2 & CCP2 and CP3 & CCP3) which represents only the time needed for computing the optimization.

The computation time is calculated using the operation *System.nanoTime( )* which is a precise measurement of time provided by Java EE 5 platform.

The experiment environments as well as the technical details of the simulation prototype were discussed in detail in Section 6.2.1 and Section 6.2.2.

3.5 **Chapter Summary**

This chapter described the approach used in ensuring that the research objectives are fulfilled, verified, and validated. Three main research steps had been explored in this chapter, namely the analysis phase, the development phase, and the evaluation phase.

In the analysis phase, the problems of the research were clearly defined, and the core components that should be considered to develop an approach for QoS-aware service composition were identified. This step also reviewed the state of the art approaches proposed to solve the MMKP problem as well as the techniques and the approaches used for path prediction. In the development phase, the core components which should be considered to develop the proposed approach were presented. In this phase, the optimization mechanism was developed which includes proposing a runtime path prediction method, mapping the QoS-aware service composition problem to MMKP, and applying heuristic optimization algorithms to solve the selection problem. In the evaluation phase, the evaluation process was divided into two parts. The first part aimed at evaluating the runtime path prediction method while the second part aimed at evaluating the new optimization mechanism.

CHAPTER FOUR
QUALITY OF SERVICE FOR WEB SERVICE COMPOSITION

Section 4.1 reviews the QoS characteristics considered in the existing research work in the area of QoS for web services and SOA. Based on this review, Section 4.2 analyzes these QoS characteristics and suggests the QoS characteristics that can be considered as selection criteria in this work. The section also provides definitions for the suggested characteristics. Section 4.3 provides prioritization for the suggested QoS criteria.

4.1 QoS Models and QoS Characteristics

The ISO QoS Framework (ISO/IEC, 1998) introduces the term of QoS characteristics which represent the fundamental term to express QoS. A QoS characteristic, as defined by the framework, is "a quantifiable aspect of QoS, which is defined independently of the means by which it is represented or controlled". The ISO ISO/IEC 25012:2008 (ISO/IEC 25012:2008, 2008) defines a quality model for evaluating software products. The goal of quality models in general is to define a set of characteristics and their relationships. The ISO quality model is illustrated in Figure 4.1.

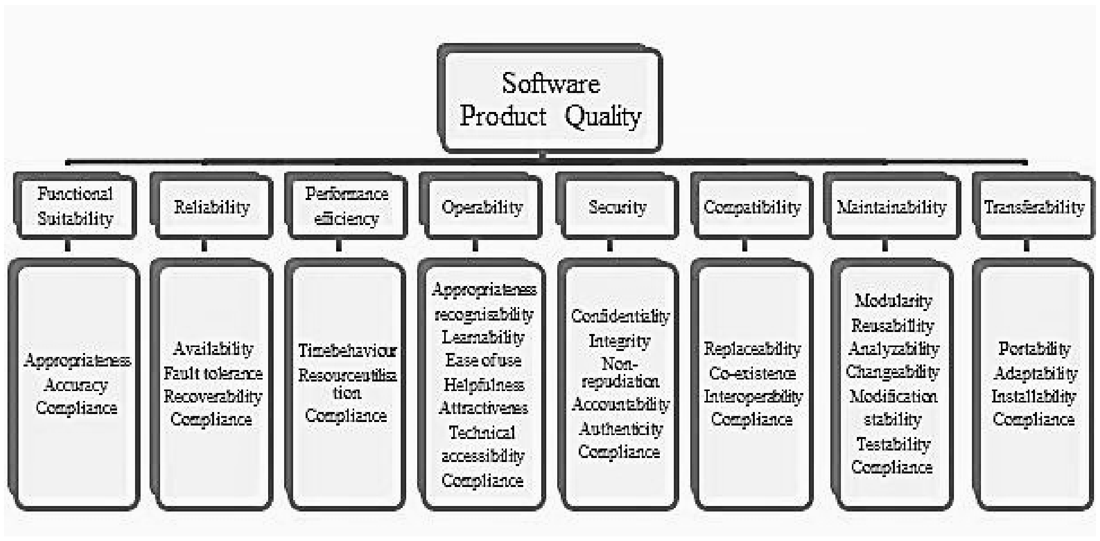

Software Product Quality
Functional Suitability
Reliability
Performance efficiency
Operability
Security
Compatibility
Maintainability
Transferability
Appropriateness Accuracy Compliance
Availability Fault tolerance Recoverability Compliance
Timebehaviour Resourceutilisation Compliance
Appropriateness recognizability Learnability Ease of use Helpfulness Attractiveness Technical accessibility Compliance
Confidentiality Integrity Non-repudiation Accountability Authenticity Compliance
Replaceability Co-existence Interoperability Compliance
Modularity Reusability Analyzability Changeability Modification stability Testability Compliance
Portability Adaptability Installability Compliance

Figure 4.1. *Software product quality model (source: ISO/IEC 25012:2008, 2008)*

Beside the ISO model, different quality models have been proposed (Parasuraman, Zeithaml & Berry, 1988; Dromey, 1995; Fitzpatrick, 1996; Botella, Burgués, Carvallo, Franch & Quer, 2002). However, a study by Behkamal, Kahani, and Akbari (2009) shows that the ISO/IEC 9126 quality model is more complete and free of shortcoming than these models.

Although the ISO/IEC 9126 quality model seems to be complete, it is not used in the web service domain (Hilari, 2009). This is because web services show different characteristics from traditional software (i.e., installation-based software) because of their service-oriented nature, such as loosely coupled binding and platform independent characteristics. Consequently, the quality of the traditional software is different from the quality of web services (Kim et al., 2011). For example, the quality of web services plays an important role in their usage. Clients could change their services if the services do not satisfy the client requirements. Therefore, not all the quality characteristics/sub-characteristics are applicable to web services.

Recently, the Organization for the Advancement of Structured Information Standards (OASIS) develops a quality model for web service named Web Service Quality Model (WSQM) (Kim, Kang, Lee & McRae, 2005). In the OASIS's final draft written in July 2011 (Kim et al., 2011), the web service quality factors are composed of a business value quality, a service level measurement quality, an interoperability quality, a business processing quality, a manageability quality, and a security quality.

Although the OASIS's Quality Factor model is designed specifically for web service, most of the research works regarding QoS for web services do not use this model. The reason is because this model is still a working draft and has not been widely known (Hilari, 2009). Instead, most of the existing research efforts in the area of web service composition consider a small set of general QoS characteristics that are applicable to all domains (i.e., domain-independent characteristics) (Menasce, 2002; Patel, Supekar & Lee, 2003; Zeng et al., 2003, 2004; Canfora et al., 2005; Yu et al., 2007; Alrifai et al., 2010; Zheng et al., 2013; Leitner, Hummer & Dustdar, 2013; Rajeswari et al., 2014; Yu, Li & Yin, 2014).

Zeng et al. (2003, 2004) introduce a simple QoS model which consists of multiple general QoS characteristics that include cost, duration (response time), the success rate (reliability), availability, and reputation. This model is adopted by many researches such as (Ukor & Carpenter, 2008, 2009; Zhang et al., 2010; Jiang et al., 2011; Schuller et al., 2011; Singh, 2012).

The authors have justified their choice of these characteristics; they are general and applicable to all web service domains. Menasce (2002, 2004) discusses QoS issues in web services in different publications. The author considers response time, cost, availability, and security as the relevant QoS characteristics for web service and web service composition. Patel et al. (2003) propose a QoS oriented framework for adaptive web service-based workflow. The goal of this framework is to enable service selection, dynamic binding, and execution of web services for the underlying workflow. In order to achieve this, a QoS model is designed with respect to web service and workflow features. The model is divided into three categories. The first category consists of latency, throughput, reliability, and cost. The second one consists of availability, security, accessibility, and regularity. The last one defines separate QoS characteristics named task-specific. Yu et al. (2005) present a broker-based framework for dynamic and adaptive QoS-aware service composition with end to end QoS constraints. In their work, they mention the relevant characteristic response time, cost, availability, and reliability. Alrifai et al. (2010) propose an approach for QoS-aware service selection that is based on the notion of skyline. In their work, they consider quantitative general QoS characteristics which include cost, response time, availability, reliability, reputation, and throughput. Some works consider a few characteristics as the relevant QoS characteristics for web service and web service composition. For example, Cardellini, Di Valerio, Grassi, Iannucci, and Presti (2011) consider only cost, response time, and availability as selection criteria, in Li and Chen (2010), cost and response time is considered, while cost is the only characteristics considered in the work by Ivanovic, Carro, and Hermenegildo (2010).

In summary, there is no standard, formal or a complete QoS model for a web service. Therefore, most of the aforementioned research efforts consider a set of general QoS characteristics that are applicable to all domains (i.e., domain-independent characteristics). However, there is no agreement between them about a specific general set. Therefore, there is a need to analyze the QoS characteristics that were most commonly used to evaluate the web services in order to determine the relevant set of QoS characteristics which can be considered as selection criteria when composing web services. In this analysis, it is important to take into consideration, when determining the relevant set of QoS characteristics for composing web services, the features of composite services which are different from features of single web services. It is needed to derive the QoS characteristics from the special features of composite services. The analysis should answer the following question: what QoS characteristics are appropriate for a web service composition?

4.2 **Analysis of QoS Selection Criteria**

The related works in the field of web services and SOA were reviewed and analyzed. As a result, the characteristics, which have been addressed, can be grouped into 25 major items. Table 4.1 itemizes the QoS characteristics that were considered by those researchers. Based on the percentage, there are seven QoS characteristics which have the highest score. There was a sharp drop in the percentages after the seventh characteristic (the security). The figures and trends give an indication about the importance of those characteristics.

The highest seven QoS characteristics are cost (100%), response time (100%), reliability (76%), availability (74%), reputation (44%), throughput (21%), and security aspects (21%). Surprisingly, eighteen other characteristics (starting from item 8 till item 25) are not really emphasized by those research groups.

Table 4.1

A Summary of the Considered QoS Characteristics in the Domain of Web Services

	QoS characteristics / researchers/ research groups	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		Cost	Response time	Reliability	Availability	Reputation	Throughput	Security aspects	Latency	Regulatory	Accessibility	Guaranteed messaging	Accuracy	Capacity	Compensation rate	Completeness	Composability	Reliable messaging	Integrity	Penalty rate	Scalability	Robustness/flexibility	Supported standard	Stability/change cycle	Task specific	Transaction
1	Menasce (2002, 2004)	x	x		x		x	x																		
2	Ran (2003)	x	x	x	x		x	x	x	x		x	x	x		x			x		x	x	x	x		
3	Patel et al. (2003)	x	x	x	x		x	x		x	x														x	
4	Zeng et al. (2003, 2004)	x	x	x	x	x																				
5	Cardoso et al. (2004)	x	x	x																						
6	Degwekar et al. (2004)	x	x		x																					
7	Liu et al. (2004)	x	x	x	x	x									x					x						x
8	Canfora et al. (2005)	x	x	x	x																					
9	Cardellini et al. (2006)	x	x			x																				
10	Tong et al. (2006)	x	x	x		x		x																		
11	Yang et al. (2006)	x	x	x	x	x																				
12	Zhang et al. (2006)	x	x	x																						
13	Yu et al. (2007)	x	x	x	x																					
14	Jin et al. (2008)	x	x	x	x																					
15	Wan et al. (2008)	x	x	x	x																					
16	Wang et al. (2008)	x	x	x																						
17	Gao et al. (2009)	x	x	x	x																					
18	Guoping et al. (2009)	x	x	x	x	x											x									
19	Huang et al. (2009)	x	x	x	x																					
20	Rosenberg et al. (2009)	x	x		x		x	x	x									x								
21	Shen et al. (2009)	x	x		x	x																				
22	Wang et al. (2009)	x	x	x																						
23	Liu et al. (2009a)	x	x			x																				
24	Liu et al. (2009b)	x	x	x	x	x																				
25	Alrifai et al. (2010)	x	x	x	x	x	x																			
26	Ardagna et al. (2010)	x	x			x																				
27	Luo et al. (2011)	x	x	x	x	x	x	x																		
28	Missaoui et al. (2010)	x	x	x	x		x	x			x															
29	Tang et al. (2010)	x	x	x	x	x																				
30	Wang et al. (2010)	x	x	x																						
31	Zhang et al. (2010)	x	x	x	x	x																				
32	Jiang et al. (2011)	x	x	x	x																					
33	Liu et al. (2012)	x	x		x	x																				
34	Singh (2012)	x	x	x	x																					
	PERCENTANGE	**100**	**100**	**76**	**74**	**44**	**21**	**21**	**5.9**	**5.9**	**5.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**
	Relative importance	20	20	15	15	8.9	4.1	4.1	1.2	1.2	1.2	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

A sharp drop is observed after security aspect characteristic. Obviously, major drops happen at three points: after response time, after reputation, and after security characteristics. Such drops clearly indicate that a few criteria can be grouped together as having a similar weight.

The result indicates that cost and response time are critical and compulsory. These two criteria are commonly used to evaluate web services and there is consensus among the research works about the importance of these characteristics. As a result, these characteristics can be considered mandatory for selection criteria.

Characteristics like reliability and availability are frequently used having rather a high score. Reputation has its own class of importance, whereas throughput and security fall in the same category of importance. These five characteristics, in spite of having different scores, are still important to be considered due to their impact on building an optimal web service composition.

It is strange enough when latency, regulatory, accessibility, guaranteed messaging, accuracy, capacity, compensation rate, completeness, composability, reliable messaging, integrity, penalty rate, scalability, robustness, flexibility, supported standard, stability, task-specific, and transaction features are less studied and rarely considered by the researchers. On average, these features score less than 4% in terms of coverage in previous studies. Some of these ‘less important’ features are actually crucial. For example, composability is significant for compositions because a high composability score guarantees a well successfully composed web services that contribute to a successful execution of compositions.

Based on the findings, it is valid to suggest these criteria as crucial: cost, response time, reliability, availability, security aspects, throughput, reputation and composability. Figure 4.2 illustrates the suggested QoS criteria. The first seven criteria are taken from item 1-7 demonstrated in Table 4.1 above. An extra characteristic, i.e., composability, is taken from criteria ranked 16 in the Table. It is the only 'out of range item' considered crucial for selection criteria. This feature, representing the probability that the service is executed as a member of the composition service (Guoping, Huijuan & Zhibin, 2009), is suggested because every day, new and more sophisticated web services are being programmed. With such proliferation of web services, new and interesting applications can be developed by composing several web services. Composability reflects the interoperable data exchange between web services. Also, a good composability characteristic would imply guarantee availability of services (because the web services have a better degree of automating the matching algorithm) yielding better response time and increasing throughput.

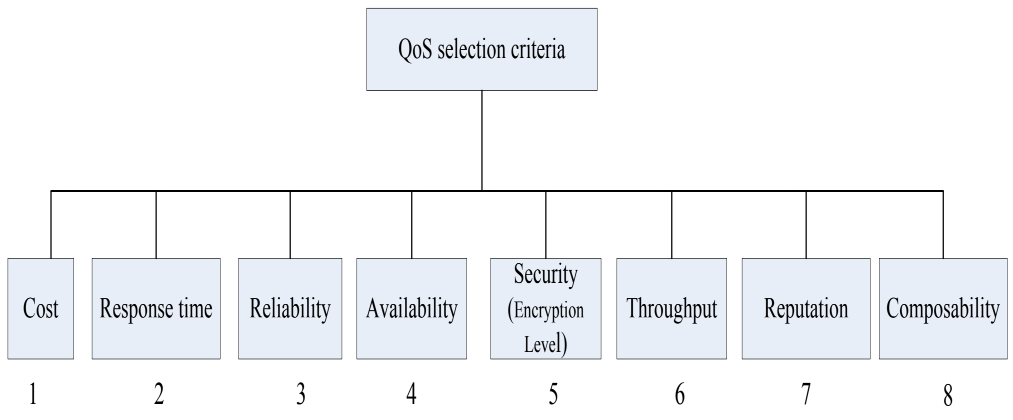


Figure 4.2. *Suggested QoS criteria for web service selection*

Cost or price represents the amount of money that a service requester has to pay for a service provider as a result of using its service (Zeng et al., 2004). Response time is a typical measure of performance that represents the total time required to complete a service request which can be defined by the sum of the time a service needs to process a request on the provider's side (processing time) and the time needed to send a request and receive a response over a network (Lee, Jeon, Lee, Jeong & Park, 2003). Concerning reliability, it represents the degree that a service is able to correctly respond to a request in a specified time interval. The number of service failures in minutes, days or months describes its reliability (Mani & Nagarajan, 2002). Reliability may include another aspect, i.e., the reliability of the messages sent and received between the web services and the applications (O’Brien, Bass & Merson, 2005). Availability of a web service represents the probability that the service is ready for access when required for immediate use (Zeng et al., 2004; Mani & Nagarajan, 2002). Security for a web service can include numerous aspects. It means providing confidentiality, authentication, authorization, encrypting data and non-repudiation. These security aspects can be provided at a different level of policy by service providers (Mani & Nagarajan, 2002; Lee et al., 2003). The only aspect that can be described with numerical value is the encryption level, which determines the encryption key length. Regarding throughput, it is a measure of service productivity. It can be defined as the number of requests that the service provider can process in a given time period (Mani & Nagarajan, 2002; Lee et al., 2003). Reputation represents a ranking that is provided by service users based on their experience of using a service. It measures the trustworthiness of a service (Zeng et al., 2004). Finally, Composability represents the probability that the service is executed as a member of the composition service (Guoping et al., 2009).

The suggested criteria, however, are not mutually exclusive, i.e., it is not a kind of if-then-else selection. Services that fall into the intersection of the eight listed criteria are the best candidate for composition. The considered QoS characteristics have the following characteristics; (1) they are commonly used to evaluate web services, (2) they have been emphasized in the research works, (3) they are applicable to all web service domains, and (4) they are derived from the special features of composite services.

4.3 **Priority for QoS Criteria**

The web service selection criteria always depend on clients' preferences which range from functional to non-functional objectives. When multiple criteria are considered for optimization at once, clients might have difficulties in choosing the right criteria. It is suggested that they express their preferences among these criteria by assigning particular weight values which represent the priority of importance. In order to assist clients when assigning weights, prioritizing the selected criteria is suggested. The prioritization is derived from the last row in Table 4.1, which indicates the relative importance of the QoS characteristics being addressed, and also by considering the significance of the QoS characteristics in building optimal web service compositions. Table 4.2 suggests the priority and its justification for each criterion**.**

Table 4.2

The Priority of QoS Criteria and Justifications

Priority	Criteria	Justification
1st	Cost	Clients can build compositions according to their budgets. However, the cost usually depends on quality, i.e. cheapest web services is often with a poor quality; therefore, clients should find a counter balanced solution to this trade-off created from cost versus any other QoS metrics (Jaeger, Muhl & Golze, 2005).
2nd	Response time	A shorter composition execution time allows for a faster response to customers' needs, thus, ensuring their satisfaction (Cardoso, Miller, Sheth & Arnold, 2004)
3rd	Reliability	As web services operate in dynamic, flexible and unreliable environments like the Internet, they are vulnerable to failure during their executions. Failure of even one service involved in a composition will result in a failure of the whole composition.
4th	Availability	In a composition that consists of several web services, if at least one service becomes unavailable, then the whole composition will be unavailable. Services must be available to have successful execution (Choi, Her & Kim, 2006).
5th	Security aspects	Web services interoperate by sending and receiving messages which may contain confidential information. Therefore, a secure communication in such a scenario has to be ensured. Must consider basic security aspects such as authentication, authorization and non-repudiation.
6th	Throughput	Some composite services, for example travel booking, received intensive requests in a short period of time. It is critical to ensure that the compositions can process the volume of expected requests; this is done by considering throughput characteristic when building the compositions.
7th	Reputation	The internet is an open, dynamic and untrustworthy environment. Web services are strange and unknown to each other. Dealing with high reputation web services is needed to avoid the trustworthiness problems that emerged in such an environment. In compositions, where previously unknown web services are discovered and bound automatically; reputation plays an important role.
8th	Composability	One of the factors that contribute to have successful execution of compositions is composability. High composability web services mean that these web services are capable to compose well as a member of compositions. Moreover, high composability increases the service reusability (Choi et al., 2006)

4.4 **Chapter Summary**

A review of the QoS characteristics that are used in the area of web service composition was given in this chapter. It concludes that there is no standard, formal or a complete QoS model for a web service. Therefore, most of the aforementioned research efforts consider a set of general QoS characteristics that are applicable to all domains (i.e., domain-independent characteristics). However, there is no agreement between them about a specific general set. Moreover, composite services have features different from single web services. Consequently, there is a need to analyze the QoS characteristics that were most commonly used to evaluate web services in order to determine the relevant set of QoS characteristics. These can be considered as selection criteria when composing web services. The analysis should answer the following question: what QoS characteristics are appropriate for a web service composition? As a result, eight QoS characteristics were suggested, namely cost, response time, reliability, availability, security, throughput, reputation, and composability. The considered QoS characteristics have the following characteristics; (1) they are commonly used to evaluate web services, (2) they have been emphasized in the research works, (3) they are applicable to all web service domains, and (4) they are derived from the special features of composite services. The selection is made with respect to web service composition features and by counting the frequency of the characteristic been considered in the related works in the field of web services and SOA as well as by considering their implicit importance despite of being scarcely included in these studies.

In assisting clients when assigning weights, prioritizing the selected QoS criteria was suggested. The prioritization is based on the relative importance of these criteria in building optimal web service compositions.

CHAPTER FIVE
A SMART QOS-AWARE SERVICE COMPOSITION APPROACH

In this chapter, the approach for QoS-aware service composition is presented in detail. The components needed to develop the approach are presented. The composite service scenarios that simulate multiple paths composite service are introduced. The proposed optimization mechanism, which consists of runtime path prediction methods and heuristic optimization algorithms, is explained. The implementation of the proposed approach is also covered in this chapter.

5.1 The Proposed Approach for QoS-Aware Service Composition

As it's explained in the introduction chapter, QoS-aware web service composition phase is one of several phases required to create composite services. Prior to this phase, there are two phases, namely design and discovery phases. In this work, it is presumed that the design of the composition has already been done, and the discovery phase has discovered a set of candidate services for each abstract web service along with their QoS values.

As illustrated in Figure 5.1, QoS-aware composition approach has four inputs, namely abstract composition i.e., a set of abstract services connected using composition structures, a list of outsourced candidate web services discovered for each abstract service and their QoS characteristic values, and a client's (organization's) global QoS requirements. The desired output is optimal solutions.

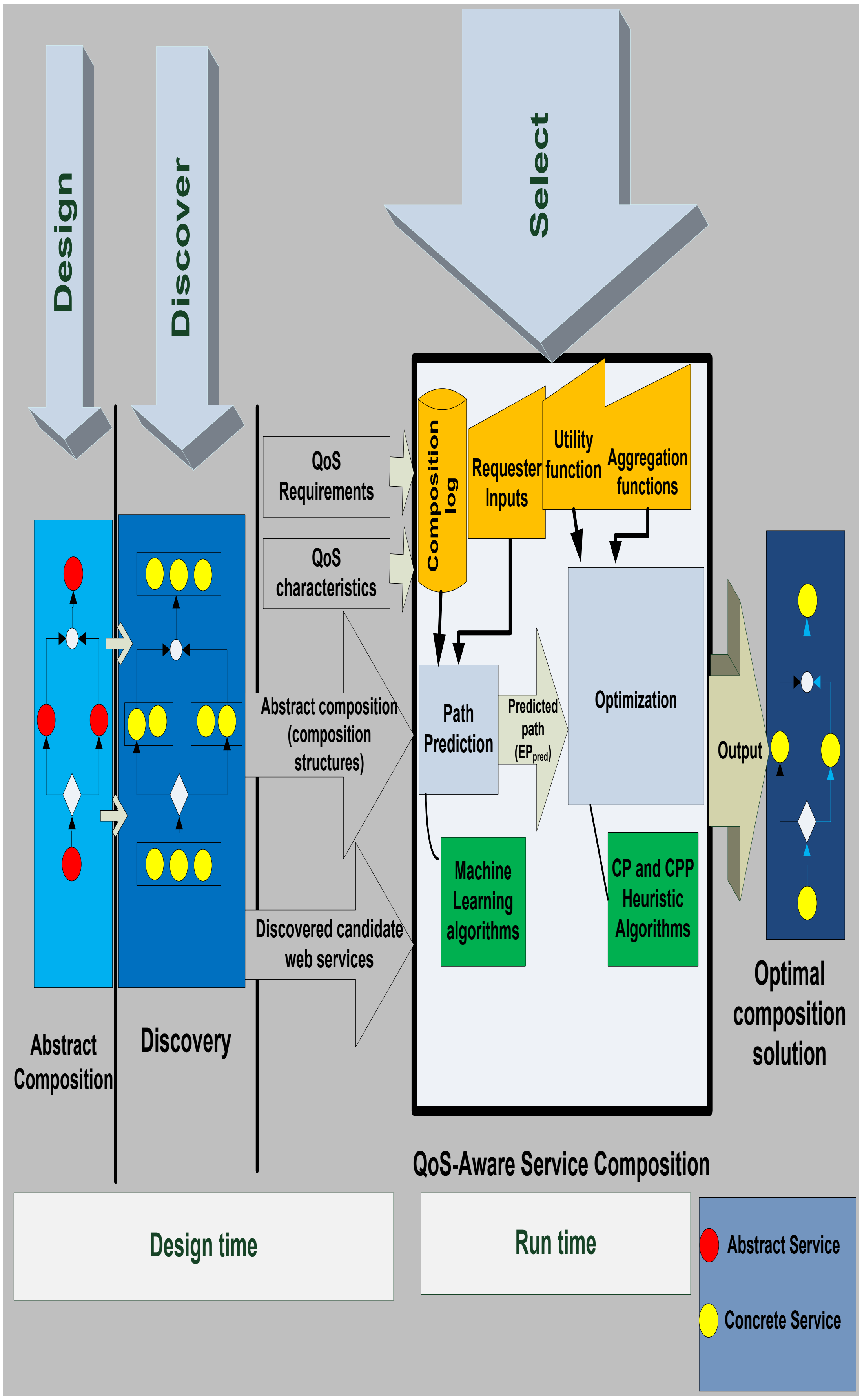


Figure 5.1. *The proposed approach*

In this work, the QoS-aware compositions process is performed dynamically on instance by instance basis. It starts by predicting, at runtime, and just before the actual execution of compositions, the path that will potentially be followed during the execution of a composition. A runtime path prediction method is proposed for this purpose. In this method, a machine learning algorithm is applied to the composition log to learn how to classify the unknown classes. This step is carried out offline; consequently, it does not affect the performance of the proposed approach. After training the machine learning algorithm, it will be ready to predict the path based on the data provided by the composite service requester. At runtime, a client (i.e., a service requester) is required to fill online form for requesting a service. Then the data needed for prediction is collected. After that, the data feed into the classifier to classify into target classes, i.e., composition paths. The output of the runtime prediction method is the prediction of a certain execution path (EP_{pred}) representing the path that will potentially be followed during composition execution. EP_{pred} is then utilized by two heuristic algorithms called CP and CCP to compute the optimization by considering only the predicted path. In the optimization process, a utility function is used to consider all the QoS characteristics that are subject to optimization. This function is used by the heuristic algorithms in the comparisons of the candidate services. In addition, aggregation functions are used to compute the overall QoS of a composition. Finally, the output of the proposed approach is solutions that are expected to deliver the best possible QoS ratios, and at the same time, reduce the constraints violated number, while consuming small computation time.

5.2 The Problem Model

In this section, the problem model of the QoS aware-service composition is defined. The model allows for mapping the problem to MMKP. The problem model is formulated in the following way:

- Assume that there is a set of service classes $S = \{S_1, S_2, ..., S_a\}$ representing the abstract services, where $i = 1, ..., a$, and a represents the total number of abstract services involved in a composition.
- For each service class S_i, there is a set of functionality equivalent candidate service (also called concrete service) $S_i = \{s_1, s_2, ..., s_{b_i}\}$, that can execute the abstract service S_i, where $j = 1, ..., b_i$, and the variable b_i represents the number of candidates found for abstract service S_i.
- A number from 1 to n is used to identify the QoS characteristics that are considered for optimization. The variable n denotes the total number of QoS characteristics.
- A QoS vector q_{ij} is assigned for each candidate s_j. The vector contains the different QoS values represented by the index k, $q_{ij} = [q_{ij}^{1}, ..., q_{ij}^{n}]$, where $k = 1, ..., n$.
- A vector of global QoS constraints imposed by clients $GS = [GS^1, ..., GS^n]$, where $k = 1, ..., n$.

A valid solution, i.e., a composition plan, can be obtained by assigning candidate services s_j to each abstract service S_i such that $s_j \in S_i$ for which the aggregated

QoS values meet the given global QoS constraints, and the overall QoS value of the composite service is maximized. Table 5.1 represents the system model notations.

Table 5.1

System Notations

Notation	Meaning
Service class S_i	Is a collection of candidate services with a common functionality but different QoS characteristics value.
Candidate service s_j	Candidate service j from service class S_i.
QoS vector $q_{ij} = [q_{ij}^{1},...,q_{ij}^{n}]$	QoS vector represents the different QoS values of candidate service s_j, where $s_j \in S_i$.
QoS global constraints $GS = [GS^{1},...,GS^{n}]$	Is a vector of all QoS global constrains imposed by the client on the whole composition.

5.3 Modeling the Composition Structures

To define a composition, different structures (such as sequential, loop) are used to connect the services. In this work, we focus on compositions that were defined by using sequential and conditional structures. Other structures such as loop, for example, may be reduced to sequential as in Zeng et al. (2003, 2004).

The structure model allows the path prediction process to be performed. In the following, two concepts used in this research work are defined.

Definition 2: Execution path (*Path*): If a composition contains conditional structures, it has multiple sequential execution paths. Each execution path $Path_i$ represents a sequence of services $\{S_1,...,S_i,...,S_a\}$. Each $Path_i$ takes only one path in each conditional structure. For example, there are 3 execution paths in Figure 5.2:

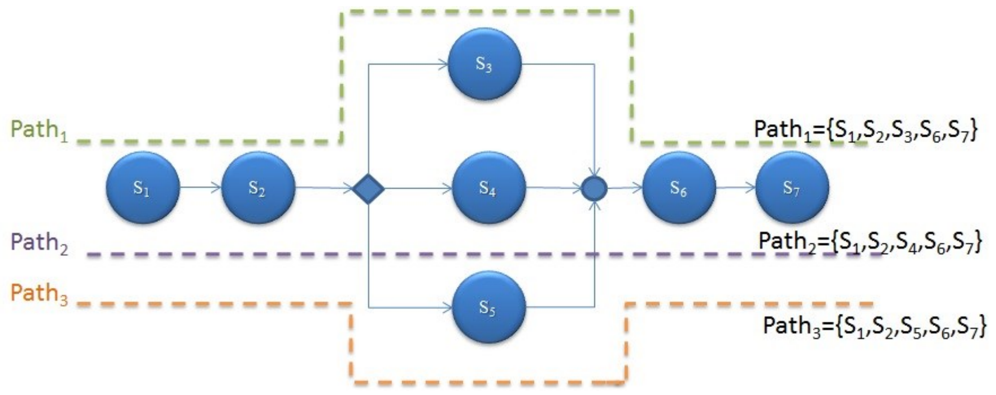

Path$_1$
Path$_2$
Path$_3$
S$_1$
S$_2$
S$_3$
S$_4$
S$_5$
S$_6$
S$_7$
Path$_1$={S$_1$,S$_2$,S$_3$,S$_6$,S$_7$}
Path$_2$={S$_1$,S$_2$,S$_4$,S$_6$,S$_7$}
Path$_3$={S$_1$,S$_2$,S$_5$,S$_6$,S$_7$}

Figure 5.2. *A multiple execution paths composition*

$Path_1 = \{S_1, S_2, S_3, S_6, S_7\}$

$Path_2 = \{S_1, S_2, S_4, S_6, S_7\}$

$Path_3 = \{S_1, S_2, S_5, S_6, S_7\}$

where: $Path_1, Path_2, Path_3 \subset S$.

Definition 3: Predicted path (EP_{pred}): is the execution path that will potentially be followed during the execution of a composition. It can be one of the sequential execution paths defined above i.e., $EP_{pred} \in \{Path_1, Path_2, Path_3\}$. EP_{pred} is identified using the runtime path prediction method which will be explained later.

5.4 Selection Criteria

In the proposed approach, QoS characteristics can serve as selection criteria to distinguish between candidate services. Eight QoS characteristics are considered as selection criteria, namely cost, response time, reliability, availability, security (encryption level), throughput, reputation and composability.

5.5 QoS Computation for Web Service Composition

The aggregation of overall QoS of a composition is needed to compute the optimization. The QoS value of a composite service is aggregated from its constituent web services. The QoS value of a composite service SN is defined by the vector Q. It contains the aggregated QoS values of a composite service (i.e., a solution) represented by the index k, $Q = [Q(SN^1),...,Q(SN^n)]$, where $Q(SN^k)$ is the estimated k^{th} QoS characteristic of the composite service SN.

Table 5.2 presents the aggregation functions that are used to compute the overall QoS of a composition. The aggregation functions are similar to those proposed by Zeng et al. (2003, 2004), Jaeger et al. (2004), and Guoping, Huijuan, and Zhibinet (2009).

Table 5.2
QoS Aggregation Functions

QoS characteristic	Aggregation function
Cost	$Q(SN^k) = \sum_{i \in EP_{pred}} q_{ij}{}^k \times x_{ij}$
Response time	
Reliability	$Q(SN^k) = \prod_{i \in EP_{pred}} q_{ij}{}^k \times x_{ij}$
Availability	
Encryption level	

Throughput	$Q(SN^k) = min(q_{ij}{}^k \, x_{ij}), \forall i \in EP_{pred}$
Reputation	$Q(SN^k) = \frac{\sum_{i \in EP_{pred}} q_{ij}{}^k \times x_{ij}}{a}$
Composability	$Q(SN^k) = \frac{\sum_{i \in EP_{pred}} q_{ij}{}^k \times x_{ij}}{a}$

In Table 5.2, the variable a represents the number inservices involved for the computation. The variable x_{ij} represents a selection variable. Note that the web services that only belong to the predicted path are considered for computation i.e. $i \in EP_{pred}$.

5.6 A Utility Function

If more than one QoS characteristics are subject to optimization, an aggregated goal function is required to consider all the QoS characteristics. The function is used for comparisons between the candidate services when an algorithm tries to solve the optimization problem. Each candidate service $s_j \in S_i$ is associated with a utility function u_{ij}. Similar to Zeng et al. (2003, 2004), Jaeger, Muhl and Golze (2005), and Alrifai and Risse (2009), the Simple Additive Weight (SAW) method, which was introduced in the context of Multiple Criteria Decision Making (MCDM) (Yoon & Hwang, 1995), is applied to compute the utility function. This method is carried out in two phases:

Scaling phase: all the considered QoS characteristics have different units of measurements. For example, reliability is a probability ratio and varies between 0 and 1 while response time is expressed in milliseconds by a positive number. In this phase, the values of different QoS characteristics are scaled to a range from 0 to 1, where the 0 value indicates a worse quality while 1 value indicates a better value. For scaling, a QoS vector q_{ij} with individual QoS values $q_{ij}{}^{k}$ is considered. Then each value is replaced by the scale value $v_{ij}{}^{k}$. Some of the characteristics could be negative. This means that the higher value denotes a worse quality such as cost and response time. Other QoS characteristics are positive which means that the higher value denotes a better quality such as availability and reliability. Based on SAW, Equation 5.1 is used to scale the positive characteristics, i.e., reliability, availability, throughput, security, reputation, and composability while Equations 5.2 is used for negative, i.e., cost and response time.

$$v_{ij}{}^{k} = \begin{cases} \dfrac{q_{ij}{}^{k} - min(q_{i1}{}^{k},...,q_{ib_i}{}^{k})}{max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) - min(q_{i1}{}^{k},...,q_{ib_i}{}^{k})} & If\ max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \neq min(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \\ 1 & If\ max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) = min(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \end{cases} \tag{5.1}$$

$$v_{ij}{}^{k} = \begin{cases} \dfrac{max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) - q_{ij}{}^{k}}{max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) - min(q_{i1}{}^{k},...,q_{ib_i}{}^{k})} & If\ max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \neq min(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \\ 1 & If\ max(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) = min(q_{i1}{}^{k},...,q_{ib_i}{}^{k}) \end{cases} \tag{5.2}$$

The interval $q_{i1}{}^{k},...,q_{ib_i}{}^{k}$ refers to all values from the considered QoS vectors referring to the relevant QoS characteristic ***k***.

Weighting phase: based on the scale values, a utility function u_{ij} can be applied to each candidate and is defined as:

$$u_{ij} = \frac{1}{n}\sum_{k=1}^{n} W_k \times v_{ij}{}^{k} \tag{5.3}$$

Where n represents the amount of the considered QoS characteristics. In this function, all QoS characteristics are weighted by their importance. W_k is the weight assigned for each QoS characteristics which are defined by clients such that $W_k \in [0,1]$ and $\sum_{k=1}^{n} W_k = 1$.

In order to assist clients when assigning weights, prioritizing the criteria is suggested in Table 4.2.

5.7 **Multiple Paths Composite Service Scenarios**

For the purpose of implementing and evaluating the proposed approach, a composite service scenario that simulates multiple paths composite service is needed to be considered. In this work, two different scenarios are used to show how generable is the proposed approach. In the next subsections, these scenarios are introduced and discussed in detail.

5.7.1 An Auto Insurance Composite Service

Auto insurance is one of several insurance types sold by insurance companies. A typical auto insurance composite service which represents a multiple paths composite service is illustrated in Figure 5.3. As seen in Figure 5.3, the service is composed of 11 web services and represents a multiple paths composite service scenario. It includes 4 different execution paths. The service sells two policies of auto insurance, namely comprehensive and third party. Comprehensive is the most complete protections for vehicles; it covers the client's vehicle, other vehicles. However, third party insurance covers only the damages that clients may cause for other vehicles. The service requesters are required to fill and apply application forms requesting for auto insurance. Then the information provided by clients is forwarded to Check Policy to determine the requested insurance policy. Based on the policy, the request is forwarded either to Evaluate Comprehensive or Evaluate Third Party services. The request can be either approved or rejected. Approved Comprehensive and Approved Third Party are the services responsible for approving comprehensive/third party insurances. In contrast, Rejected Comprehensive and Rejected Third Party are the web services responsible for rejecting comprehensive/third party insurances. The result of the auto insurance request is then e-mailed to the client. Finally, auto insurance application data is stored in a database by the Archive Application web service.

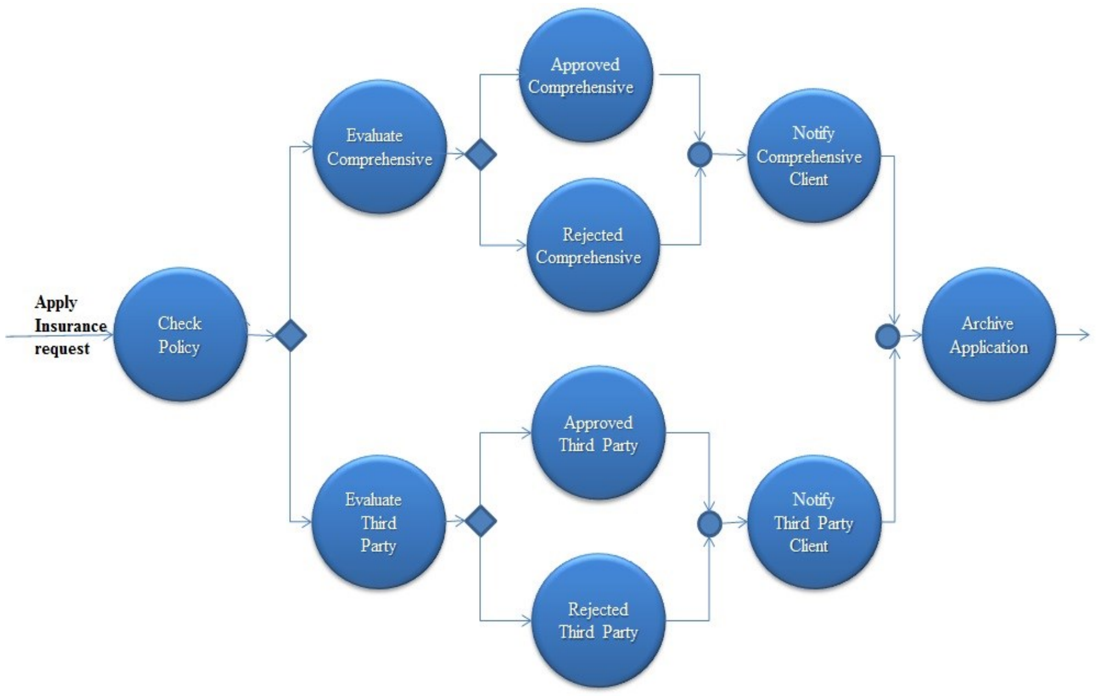


Figure 5.3. *A typical auto insurance composite service scenario*

5.7.2 A Bank Loan Composite Service

The loan composite service is one of several services supplied by banks. A typical bank loan service which represents a multiple paths composite service is illustrated in Figure 5.4. It is composed of 22 web services and contains 10 different execution paths. The service offers various types of loans that are most commonly used, namely a new car loan, a used car loan, an education loan, a home loan, and a personal loan.

In this composite service, a client (i.e., a bank loan requester) is required to fill the application form for requesting a loan. The information provided by the client is forwarded to Check Loan Type web service to determine the loan types. Based on its type, the request is then forwarded to one of the five services: Check Home Loan, Check Educational Loan, Check New Car Loan, Check Personal Loan, or Check

Used Car Loan. The request can be accepted, rejected or approved conditionally in the case of a home loan. Approve /Reject Home Loan, Approve /Reject Educational Loan, Approve/Reject New Car Loan, Approve/Reject Personal Loan, and Approve/Reject Used Car Loan are the web services responsible for accepting /rejecting a loan request. The result of the loan request is then e-mailed to the client. Finally, the loan application data is stored in a database by the Archive Application web service.

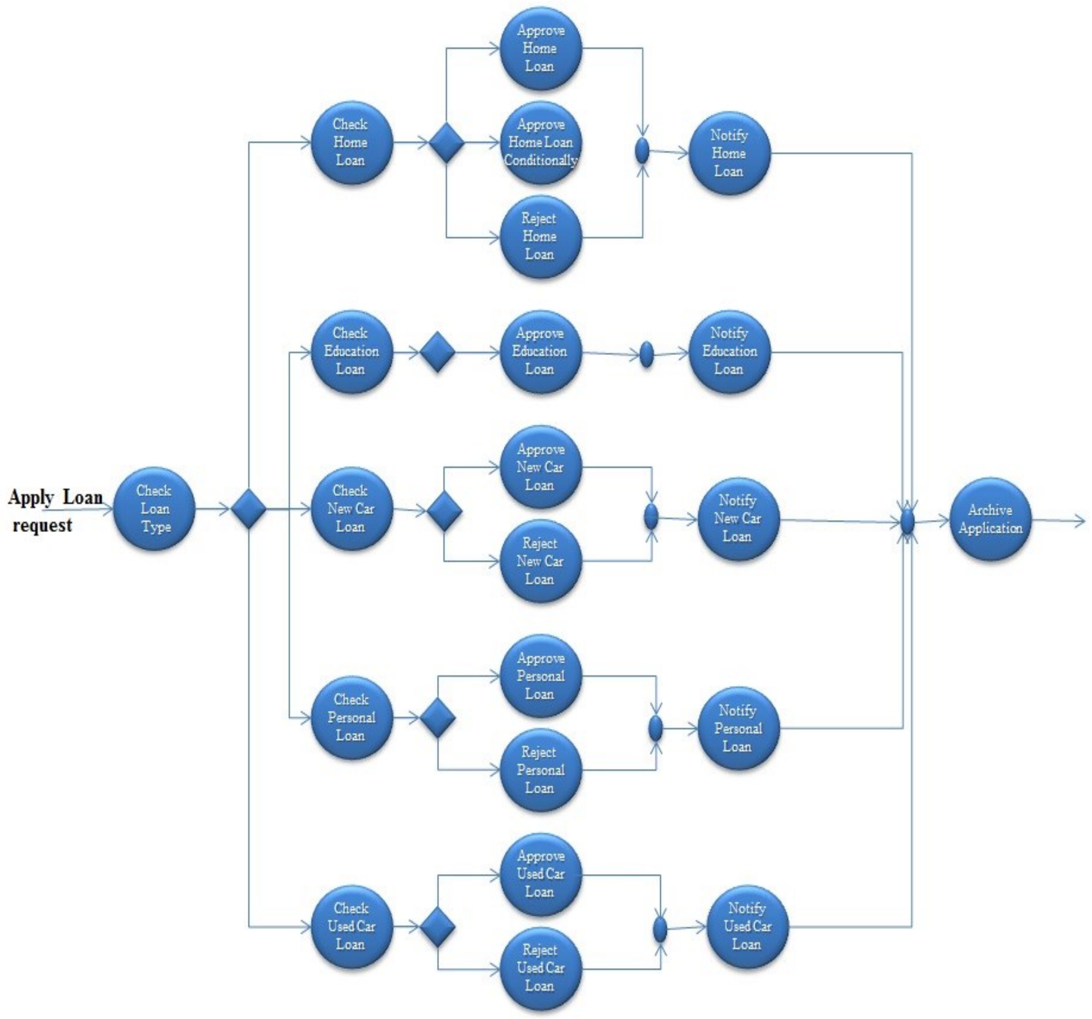


Figure 5.4. *A typical bank loan composite service scenario*

5.8 New Optimization Mechanism

The new optimization mechanism is performed in two steps: (1) predict the execution path that will potentially be executed; (2) compute the optimization for the predicted path. The mechanism is started by predicting, at runtime, and just before the actual execution of compositions, the path that will potentially be followed during the execution of a composition. A runtime path prediction method is proposed for this purpose. In this method, a machine learning algorithm is applied to the composition log to learn how to classify the unknown classes. After building the classifier, it will be ready to predict the path based on the data provided by the composite service requester. At runtime, a client (i.e., a service requester) is required to fill online form for requesting a service (for example, a loan or auto insurance services). Then the data needed for prediction (i.e., personal data and the data describes the condition of the service being requested) is collected. After that, the data feed into the classifier to classify into target classes, i.e., execution paths. The output of the runtime prediction method is the prediction of a certain execution path (EP_{pred}) representing the path that will potentially be followed during composition execution. EP_{pred} is then utilized by two heuristic algorithms called CP and CCP to compute the optimization by considering only the predicted path (EP_{pred}).

5.8.1 Predicting the Execution Path

Workflow management systems store the data generated from the execution of workflows in logs. The data stored in logs are rich with concealed information that can be used for making intelligent business decisions (Sumathi & Esakkirajan, 2007). One possible way to reveal this valuable information is by applying data mining algorithms on these logs. Based on historical data contained in logs, data mining can be used to predict the value of a particular target class. If the class is discrete, this process is referred to as a classification which includes assigning a class label to a set of unclassified instances. If the set of possible classes is known in advance, the process is referred to as supervised learning (Gutiérrez-Peña, 2004). An application example includes predicting an insurance claim as fraudulent or not.

5.8.1.1 A Runtime Path Prediction Method

The path prediction process by Cardoso (2005, 2008) and Cardoso and Lenic (2006) needed to be extended and refined for the purpose of runtime path prediction based on the information provided by the service requesters. The following limitations are identified to be addressed in this work:

1. The mentioned work performs the prediction at design time on information indicating the input (output) values parameters passed (received) to (from) activates. The prediction in this work is performed at runtime based on the information provided by the composite service requester when filling an online application.
2. In the mentioned work, it is not necessary that all attribute values are stored in logs i.e., there may be some missing information. This is because some activities may not have been invoked by the workflow management system when path mining is started. Using datasets with missing values to train classifiers can affect the prediction quality of classifiers (Acuna & Rodriguez, 2004; Liu, Lei & Wu, 2005; Blomberg & Ruiz, 2013). In contrast to the mentioned work, in the proposed prediction method, there are no missing attribute values in the datasets

because the stored value parameters are the kinds of must enter attributes. These attributes represent personal information and information describing the condition of the service being requested. For example, a policy-type and an auto-model are examples of information that must be provided when requesting auto insurance.

3. In the mentioned work, profiles for each process instance are needed to be constructed for training the algorithms. In this work, the training dataset is created in the form of a relational table.
4. In contrast to the mentioned work, detailed information about the implementation of the path prediction method is given.
5. In the mentioned work, experiments were conducted using one dataset that represents one process scenario. In this work, 10 datasets were used which represent two process scenarios. The datasets represent different business process domains, i.e., auto insurance and bank loan processes.

In the following discussions, the proposed runtime path prediction method is discussed in detail. The auto insurance composite service scenario is used to explain and illustrate the method. The method consists of four phases:

The Log Preparation Phase

This phase is adopted from Cardoso (2005, 2008) and Cardoso and Lenic (2006). It includes extending the logs to store information indicating the input (output) value parameters passed (received) to (from) web services and their types. These values are generated at runtime during the execution of composition instances. Each 'parameter/value' entry as a data type, a name, and a value, (for example, int production-year=3). In addition, the class path is an extra field needs to be added to the log to store path information. It indicates the path that has been taken by a particular composition instance when the parameters have been assigned to a specific value set. The class path is associated in order to analyze the choices that have been made (i.e., the paths that have been executed) in the past execution of a composition, and to determine whether the paths that have been taken might be influenced by the information provided by compositions instances.

Preparation of Training Dataset Phase

This phase aims at using the runtime data about instance contained in the logs as a training dataset for machine learning algorithms. The training dataset is typically in the form of a relational table in which each row represents one composition instance extracted from logs. Each instance in the training dataset is characterized by the value parameters of a composition requester. In addition, it is labeled with a class indicating the path that has been taken when the parameters have been assigned to a specific value set. In this way, a set of classified data is taken by a learning schema to learn a way of classifying unseen instances. For example, Table 5.3 shows the structures of training datasets for the auto insurance composite service scenario presented in the previous section. As seen in Table 5.3, each instance consists of four parameters, namely a policy-type, a manufacture-type, an auto-model, and a production-year. These are associated with a class, namely path indicating the path that has been executed when these parameters have been assigned to a specific value set. A detailed description of datasets and attributes is given in Section 6.1.1. In order to determine the attributes, WEKA's "Select attributes" technique, which gives access to a wide variety of algorithms and evaluation criteria for identifying the most important attributes in a dataset, is used (Hall et al., 2009). Table 5.4 shows an example of training datasets for the auto insurance composite service. As mentioned earlier, there are no missing attribute values in the datasets because the stored value parameters are the kinds of must enter attributes.

In a case that commercial workflow management systems are used, the process of extracting the training dataset from logs can be performed manually. Another solution is to use workflow management systems support tools. There are different existing tools proposed to support workflow management systems. By using these tools such as BPI tool suite proposed by (Grigori et al., 2004), the processes of extracting training dataset can be performed automatically. For example, in PBI tool, log data are periodically extracted and loaded into warehouses for analyzing purposes.

Table 5.3

Training Dataset Structure for Auto Insurance Problem

Dataset structure				Class
Policy-type	Manufacture-type	Auto-model	Production-year	Path

Table 5.4

Example of Auto Insurance Training Dataset

Policy-type	Manufacture-type	Auto-model	Production-year	Path
Comprehensive	Kia	Rio	2004	$Path_1$
Third Party	Volkswagen	Golf	2000	$Path_3$
Comprehensive	Fiat	Punto	1996	$Path_2$

The Learning Phase

This phase aims at building classifiers. Path prediction is treated as a classification problem. Since the class path of each instance is provided, supervised learning is used (Sumathi & Esakkirajan, 2007). Once storing enough information in logs, machine learning algorithms can be used to establish a relationship between the value parameters and the paths taken at runtime.

It is recommended for a learning process to be iteratively refined when the process execution proceeds and more information about composite service execution becomes available. More data yields to build more accurate prediction classifiers.

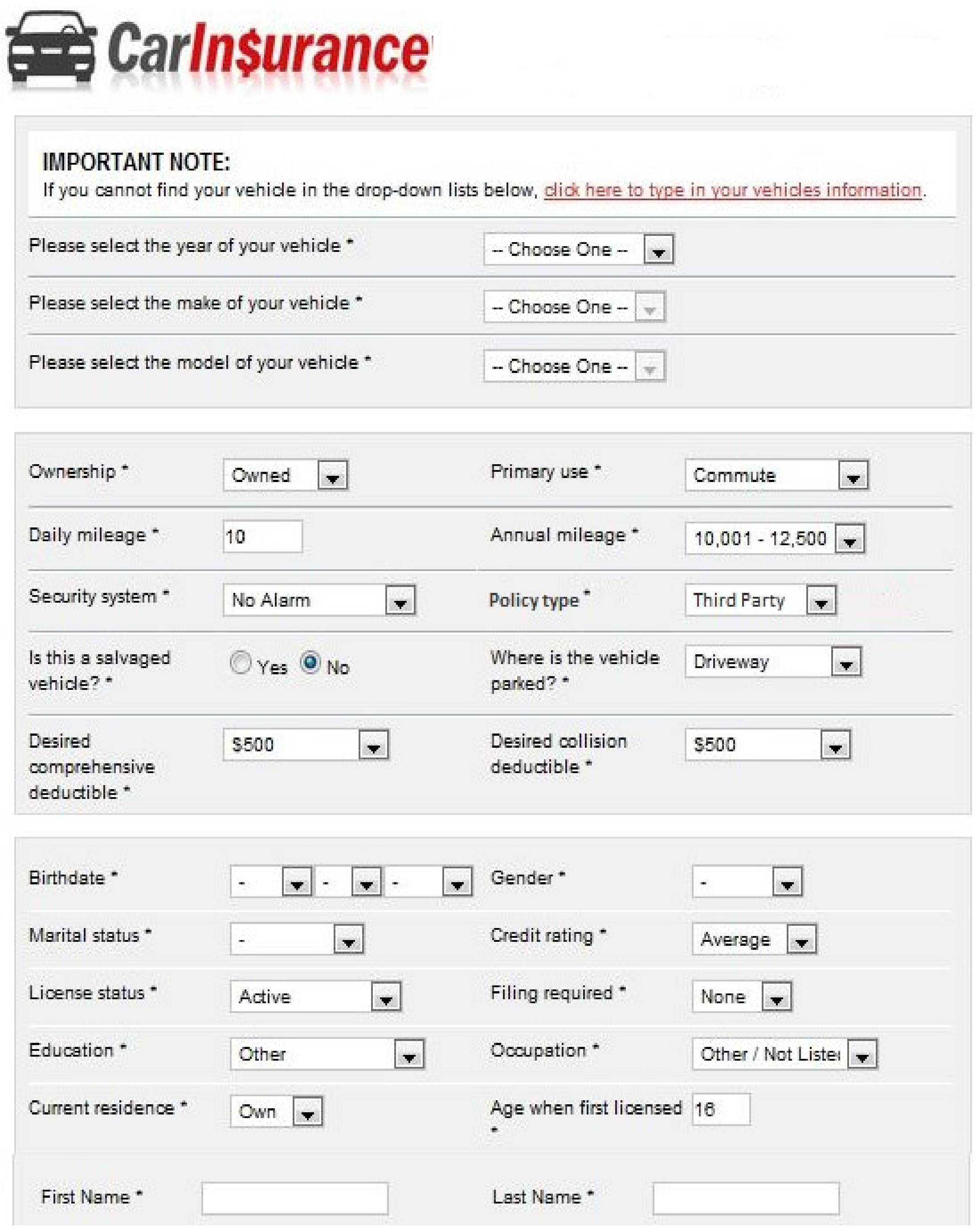
The output of this phase is classifiers. A classifier is a function used to map unlabeled instance to a labeled (Kohavi, 1995) by producing a set of classification rules. For example, if the requested policy-type is comprehensive, the manufacture-model is Fiat, and the production-year is less than 2004 then $Path_2$ i.e., rejected comprehensive insurance. In this approach, classifiers are built offline, so the computation time consumed for building a classifier does not affect the overall performance of the approach.

Again, in a case that commercial workflow management systems are used, the process of executing the machine learning algorithms on the generated training dataset can be performed manually. By using workflow management systems support tools, it can be performed automatically using an engine to execute it.

The Runtime Path Prediction Phase

This phase aims at performing runtime path prediction based on the information provided by a composition requester. The classifier is now ready for classifying unknown classes, i.e., predict the path that is followed during the execution.

At runtime, a client (i.e., a service requester) for auto insurance is required to fill an application form and apply it to request insurance. The form represents personal data and the data describes the condition of the service being requested. For example, a policy-type, a manufacture-type, an auto-model, and a production-year are examples of such data. Figure 5.5 illustrates an example of a typical application form for auto insurance request.



CarIn$urance

IMPORTANT NOTE:
If you cannot find your vehicle in the drop-down lists below, click here to type in your vehicles information.

Please select the year of your vehicle *	-- Choose One --
Please select the make of your vehicle *	-- Choose One --
Please select the model of your vehicle *	-- Choose One --

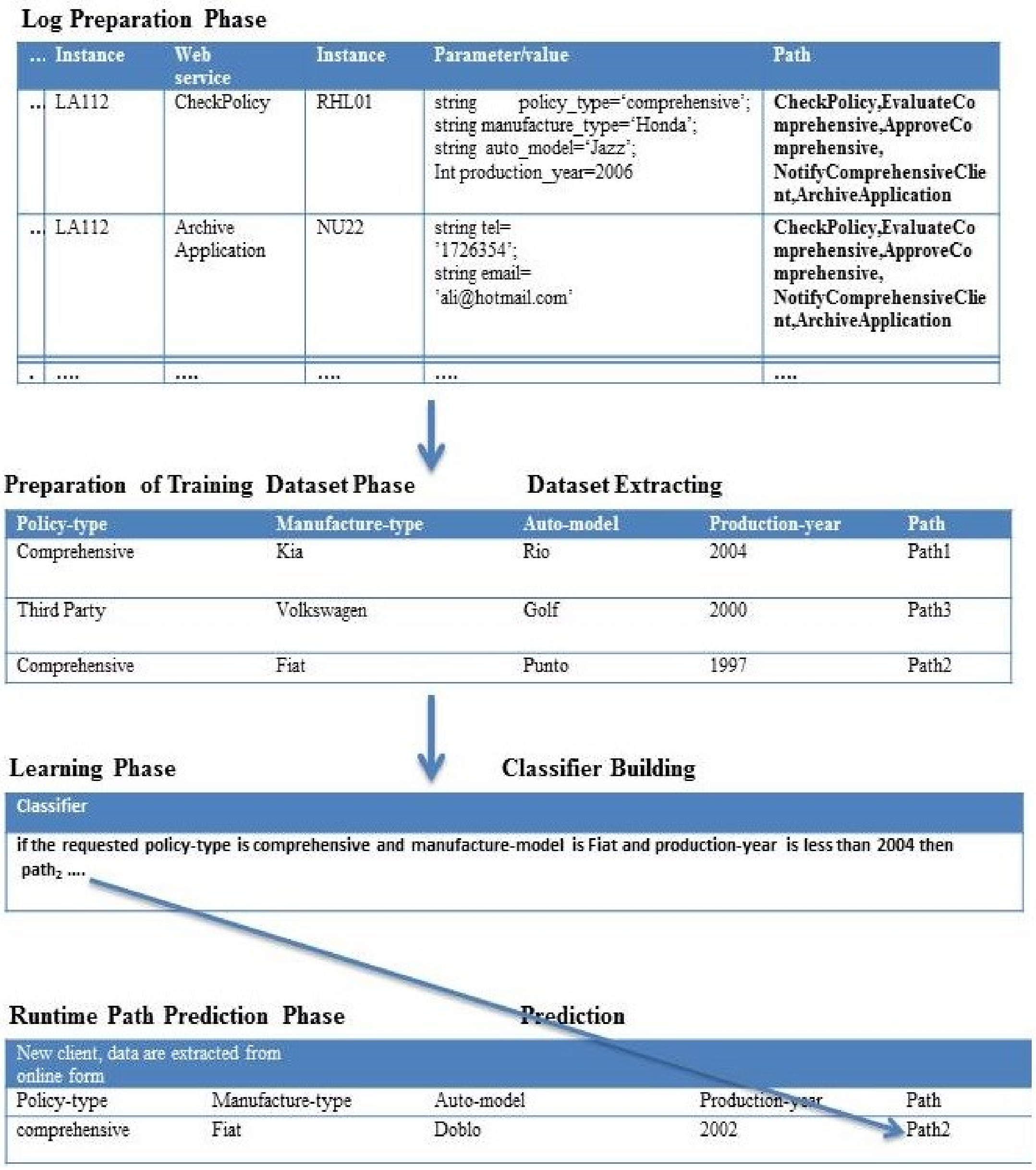
Ownership *	Owned	Primary use *	Commute
Daily mileage *	10	Annual mileage *	10,001 - 12,500
Security system *	No Alarm	Policy type *	Third Party
Is this a salvaged vehicle? *	Yes [✓ No]	Where is the vehicle parked? *	Driveway
Desired comprehensive deductible *	$500	Desired collision deductible *	$500

Birthdate *	- - -	Gender *	-
Marital status *	-	Credit rating *	Average
License status *	Active	Filing required *	None
Education *	Other	Occupation *	Other / Not Listed
Current residence *	Own	Age when first licensed *	16
First Name *		Last Name *	

Figure 5.5. *A typical online application form for requesting auto insurance*

The data needed for prediction i.e., a policy-type, a manufacture-type, an auto-model, and a production-year are then collected and fed to a classifier to be classified into target classes, i.e., execution paths.

The output of this phase is the prediction of a certain execution path (EP_{pred}) representing the path that will potentially be followed during the execution of the bank loan composite service. This important information, i.e., EP_{pred} is utilized by the optimization algorithms in order to optimize the predicted path. The runtime path prediction method is illustrated in Figure 5.6.



...	Instance	Web service	Instance	Parameter/value	Path
..	LA112	CheckPolicy	RHL01	string policy_type='comprehensive'; string manufacture_type='Honda'; string auto_model='Jazz'; Int production_year=2006	**CheckPolicy,EvaluateComprehensive,ApproveComprehensive, NotifyComprehensiveClient,ArchiveApplication**
..	LA112	Archive Application	NU22	string tel= '1726354'; string email= 'ali@hotmail.com'	**CheckPolicy,EvaluateComprehensive,ApproveComprehensive, NotifyComprehensiveClient,ArchiveApplication**
.	....	....	....	....	....

Policy-type	Manufacture-type	Auto-model	Production-year	Path
Comprehensive	Kia	Rio	2004	Path1
Third Party	Volkswagen	Golf	2000	Path3
Comprehensive	Fiat	Punto	1997	Path2

Classifier
if the requested policy-type is comprehensive and manufacture-model is Fiat and production-year is less than 2004 then path$_2$ ....

New client, data are extracted from online form				
Policy-type	Manufacture-type	Auto-model	Production-year	Path
comprehensive	Fiat	Doblo	2002	Path2

Figure 5.6. *The runtime path prediction method*

5.8.2 Computing the Optimization

As mentioned earlier, QoS-aware service composition problem is known as NP-hard (Martello & Toth, 1986). The computational complexity for solving such a problem is high and exact optimal algorithms are inappropriate for scenarios where the quick response to a composition instance is very important. This represents the motivation to apply heuristic algorithms although they produce near-optimal solutions but with small computational time. To do so, the selection problem is mapped to MMKP due to the similarity between these two problems (Yu, Zhang & Lin, 2007; Alrifai et al., 2009). Then the approach adapts heuristics that are known to be efficient for solving MMKP, and applies it to solve the selection problem.

5.8.2.1 Mapping the Selection Problem to Multidimensional Multi-choice Knapsack Problem (MMKP)

The aim of the MMKP is to select exactly one item from each class in order to put them into a knapsack. As stated by Hifi et al. (2004), each item has a profit value, a weight, and the knapsack has a limited amount of resources. The amount of resources for the knapsack does not allow taking all items. Thus, it is needed to perform a selection to identify the optimal items which maximize the total profit value that is subject to resource constraints. The MMKP is formulated in the following way:

$$maximize \quad \sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij} \times x_{ij}$$

$$subject\ to\ resource\ constraint: \quad \sum_{i-1}^{n}\sum_{j=1}^{r_i} W_{ij}{}^{k} \times x_{ij} \le C^k, \quad k \in \{1,..,m\}$$

$$while\ keeping: \sum_{j=1}^{r_i} x_{ij} = 1, i \in \{1,...,n\}$$

$$where: \ x_{ij} \in \{0,1\},\ i \in \{1,...,n\}, j \in \{1,...,r_i\}, and$$

$$x_{ij} = \begin{cases} 1, & if \quad j \quad is \quad selected \quad for \quad class \quad J_i \\ 0, & if \quad j \quad is \quad not \quad selected \end{cases}$$

As explained in Section 3.3.5.2, v_{ij} represents the non-negative profit value of the item j in the class J_i. The variable x_{ij} is either equal to 1, indicating that item j of the J_i class is selected, or equal to 0, indicating that item j of the class J_i is not selected. $W_{ij}{}^{k}$ represents the requires resources for each item and C^k represents the amounts of available resources.

On the other hand, the selection problem aims to select exactly one candidate from each service class such that the entire QoS value of the composition is optimized while QoS requirements defined by clients are satisfied.

Based on the problem model and the utility function, the selection problem is modeled in the following way:

$$\begin{aligned}
&maximize \quad \sum_{i=1}^{a}\sum_{j=1}^{b_i} u_{ij} \times x_{ij} \\
&subject\ to\ global\ constraint: \\
&Q(SN^k) \le GS^k,\ for\ negative\ QoS\ characteristics\ (cost\ and\ response\ time) \\
&Q(SN^k) \ge GS^k,\ for\ positive\ QoS\ characteristics\ (rest\ of\ characteristics) \\
&while\ keeping: \sum_{j=1}^{b_i} x_{ij} = 1, i \in \{1,...,a\} \\
&where:\ x_{ij} \in \{0,1\}, i \in \{1,...,a\}, j \in \{1,...,b_i\}, and \\
&x_{ij} = \begin{cases} 1, & if\ s_j\ is\ selected\ for\ class\ S_i \\ 0, & if\ s_j\ is\ not\ selected \end{cases}
\end{aligned}$$

Where u_{ij} represents the utility function calculated using Equation 5.3. The selection variables x_{ij} is used to determine whether a candidate service is selected for optimal composition or not. The value of x_{ij} is either equal to 0 or 1. The value of x_{ij} is equal to 1, if the candidate service s_j is selected for the class S_i, or equal to 0, if the candidate service s_j is not selected for the class S_i. There is exactly one candidate service selected for each class S_i i.e., $\forall i, 1 \le i \le a, x_{i1} + x_{i2} + ... + x_{ib_i} = 1$. $Q(SN^k)$ is the estimated k^{th} QoS characteristic of the composite service SN calculated by using the aggregation function defined in Table 5.2, and GS^k is the k^{th} global QoS constraints imposed by clients.

The MMKP requires that the total resources are less than the resource available. However, in the selection problem, the total QoS characteristics are required to be either less (for negative characteristics) or greater (for positive) than the global QoS constraints. To map the selection problem into MMKP, positive characteristics are needed to be transformed into negative. To do so, the values of positive characteristics are multiplied by -1. Then the service selection problem is formulated mathematically:

$$\begin{aligned}
&\textit{maximize} \quad \sum_{i=1}^{a}\sum_{j=1}^{b_i} u_{ij} \times x_{ij} \\
&\textit{subject to global constraint :} \\
&Q(SN^k) \le GS^k, k \in \{1,..,n\} \\
&\textit{while keeping :} \sum_{j=1}^{b_i} x_{ij} = 1, i \in \{1,...,a\} \\
&\textit{where :} \quad x_{ij} \in \{0,1\}, i \in \{1,...,a\}, j \in \{1,...,b_i\}, \textit{and} \\
&x_{ij} = \begin{cases} 1, & \textit{if} \quad s_j \quad \textit{is selected for class} \quad S_i \\ 0, & \textit{if} \quad s_j \quad \textit{is not selected} \end{cases}
\end{aligned}$$

On the basis of the similarity between these two problems, the selection problem is mapped to MMKP as in the following (Yu, Zhang & Lin, 2007; Alrifai et al., 2009):

1. The knapsack is represented by the composition.
2. Each service class represents a class or an object group.
3. Each candidate in a service class represents one item in a class.
4. Each utility function u_{ij} represents a non-negative profit value v_{ij} and can be calculated using Equation 5.3.
5. The QoS characteristics q_{ij} of a candidate s_j represent the required resource W_{ij} of the item.

6. The QoS global constraints GS are considered the resources available in the knapsack C.

5.8.2.2 Initial Feasible Solution of Constructive Procedure

Mapping the selection problem to MMKP allows selecting heuristics that are known to be efficient for solving MMKP, and applies it to solve the selection problem. As mentioned earlier, the constructive and complementary search approach by Hifi et al. (2004) is selected for solving the optimization problem. In this approach, the constructive procedure (CP) is applied to generate a feasible solution while the complementary CP (CCP) is used to improve the quality of the solution generated from CP.

A solution (*SN*) for the selection problem is represented as it illustrated in Figure 5.7.

$$\begin{array}{lc}
\text{Service class} \rightarrow & \\
\text{Service candidate} \rightarrow & \overbrace{1\quad 2\quad 3}^{S_1}\quad \overbrace{1\quad 2}^{S_2}\ \ldots\ \overbrace{1\quad 2\quad 3\quad 4}^{S_a} \\
\text{Selection variable } x_{ij} \rightarrow & 0\quad 1\quad 0\quad 0\quad 1\ \ldots\ 0\quad 0\quad 1\quad 0
\end{array}$$

Figure 5.7. *Representation for a solution*

For each service class S_i, one and only one candidate service s_j is selected, i.e., $x_{ij} = 1$ if the j^{th} candidate service s of the i^{th} service class S_i has been selected; otherwise $x_{ij} = 0$. A feasible solution is:

$$\forall k \in \{1,..,n\}, \sum_{i=1}^{a} \sum_{j=1}^{bi} q_{ij}{}^{k} \times x_{ij} \leq GS^k, \forall i \in EP_{pred}$$

Note that the service selection is performed for all services contained in all execution paths. However, the feasibility of the predicted execution path is only checked i.e., $\forall i \in EP_{pred}$.

For the solution *SN*, there are two distinguished states: feasible state (FS); if the *SN* does not violate the amount of available global QoS constraints, and unfeasible state (US); if the *SN* violates at least one or more global QoS constraints.

The initial feasible solution is obtained using the CP. The CP is a greedy approach with DROP and ADD phases to generate a feasible solution. Prior performing the CP algorithm, a machine learning algorithm should be trained in order to be ready for predicting the path that will potentially be executed (EP_{pred}). Then, the CP algorithm starts by calling a classifier in order to predict the execution path (EP_{pred}). The data needed for prediction are collected and fed to a classifier to be classified into execution paths. The proposed runtime path prediction method is used to predict the execution path (EP_{pred}). EP_{pred} is then utilized by the CP to compute the optimization by considering only the predicted path. To the best of our knowledge, this work is one of the first that combines machine learning algorithms with optimization algorithms in order to optimize the multiple paths composition. The steps involved in the CP are introduced in the following points:

1. Call a classifier and fed it with the attribute values provided by a service requester. For example, a policy-type, a manufacture-type, an auto-model, and a production-year.

2. Update EP_{pred} with the predicted path.

3. Calculate the utility function u_{ij} using the Equation 5.3.

4. Select the candidate s from each service class S_i, $i \in \{1,...,a\}$ which has the maximum utility ratio u_{ij}. By this step, service selection is performed for all execution paths.

5. Check the state of the obtained solution *SN*, only for the classes which belong to the predicted execution path. If a state is feasible (FS), then CP terminates; else (DROP phase), it determines the most violate constraint GS^{k_o}, with respect to GS^{k_o}, it selects the service class S_{i_o} corresponding to the fixed candidate service S_{i_o} having the largest QoS value $q_{i_o j_{i_o}}{}^{ko}$ all over the fixed candidate services.

6. (ADD phase) Swap the selected candidate service with another candidate s from the same service class S_{i_o}.

7. Check the feasibility of new obtained *SN*, if the state is unfeasible (US), select the lightest candidate s'_{i_o} of the current service class S_{i_o} which in turn is considered the new selected candidate service.

8. Iterate until an FS or the smallest infeasibility amount is obtained.

9. Call the CCP algorithm.

Figure 5.8 describes the CP.

Input: an instance of the selection problem

Output: a feasible solution *SN* with value *O(SN)*

```
1. $EP_{pred}$ = **Call** a classifier (policy-type, manufacture-type, auto-model, production-year)
2. Calculate utility function for every candidate service s of service class $S_i$ as:
   $u_{ij} = \frac{1}{n}\sum_{k=1}^{n} W_k \times v_{ij}^{k}$
3. **For** every service class $S_{i,} i = 1,...,a$
4. $u_{iji} = max\{u_{ij}, j = 1,...,b_i\}$
5. $S_i \leftarrow s_i$
6. $p[i] = s_i$ /* $p[i]$ represents the $j^{th}$ position of the selected candidate */
7. $x_{ip[i]} = 1$
8. **End For**
9. Solution vector is $SN = (S_1,...,S_a)$
10. **For** every $S_i \in EP_{pred}$
11. $R^k = Q(SN^k), \forall k \in \{1,...,n\}$ /* $R^k$ the total QoS characteristic values for the constraint k, $Q(SN^k)$ is the aggregation function for the $k^{th}$ QoS characteristics in Table 5.2 */
12. **End for**
13. **While**$(R^k > GS^k$, for $k = 1,...,n)$/*DROP Phase*/
14. $k_o \leftarrow \underset{1 \le k \le n}{argmax}\{R^k\}$/* Determine the value of k for which $R^k$ attains its largest value i.e., considers the most violate constraint*/
15. $i_o \leftarrow \underset{1 \le i \le a, iff: \forall i \in EP_{pred}}{argmax}\{q_{ip[i]}^{k_o}\}$/* determine the service class that has the maximum $k^{th}$ QoS values, for all classes that belong only to the predicted path*/
16. $p[i_o] = s_{i_o}$
17. $x_{i_o p[i_o]} = 0$
```

```
18. $R^k \leftarrow Drop_k(R^k, q^k_{i_o p[i_o]})$ for $k = 1,...,n$ /*calculate the new $R^k$ value after
    applying the Drop function */
19. For $j = 1,...,b_{i_o}$ /*ADD phase*/
20. If ($\exists s \neq s_{i_o}$ and $Add_k(R^k, q^k_{i_o j}) < GS^k$ for $k = 1,...,n)$ then
21. $x_{i_o j} = 1$ /* Swap the selected candidate with another from the same service
    class $S_{i_o}$ */
22. $s_{i_o} = s$
23. $p[i_o] = s_{i_o}$
24. $R^k \leftarrow Add_k(R^k, q^k_{i_o p[i_o]})$ for $k = 1,...,n$
25. $SN = (p[i_0]; p[i], \forall i \neq i_o, i = 1,...,a)$ /* is a feasible solution*/
26. Exit with SN vector
27. End if
28. End for
29. $s'_{i_o} \leftarrow \underset{1 \leq j \leq r_{i_o}}{argmin}\{ q^{k_o}_{i_o j} \}$ /* if the obtained solution is not feasible */
30. $s_{i_o} = s'$
31. $p[i_o] = s_{i_o}$
32. $x_{i_o p[i_o]} = 1$
33. End While
34. Call CCP(SN, O(SN)) /* the QoS value of SN (O(SN)) is the utility
    summation of every class that belongs to $EP_{pred}$ */
```

Return solution *SN* with QoS value *O(SN)*, *O(SN)* only computes the QoS value for every class in the predicted execution path (EP_{pred}) i.e., $\forall S_i \in EP_{pred}$

Figure 5.8. *The constructive procedure (CP)*

5.8.2.3 Using CCP to Improve the Initial Feasible Solution

To improve the QoS values of the initial feasible solution *SN* obtained by CP, CCP algorithm is applied. It tries to iteratively improve *SN* by applying:

1. A local swap strategy for selected candidate services that belongs to *SN*, called old candidates.
2. A replacement stage that replaced the old candidate with another new one, called a new candidate, is selected from the same service class. Each replacement between an old candidate and a new one is authorized if, and only if, *SN* realizes a FS, i.e., maintains the feasibility of *SN*.

The followings are the steps of the Local swap search procedure:

Step 1: Initialize the best candidate service to swap:

1.1 $value \leftarrow u_{iS_i}$, where u_{iS_i} is the utility of the old selected candidate s_i in the *i*th service class S_i to be swapped.

1.2 $k_i \leftarrow S_i$, where k_i is a selected candidate service in S_i service class to be swapped.

Step 2: Perform the exchange if it is authorized:

2.1 perform the exchange if there is a new candidate service that has larger QoS value than the old candidate, and at the same time, it realizes a FS.

2.2 return the best candidate service k_i to be swapped.

The steps involved in the CCP are introduced as in the following:

1. Apply CP to obtain an initial feasible solution.
2. Initially, set the best solution equal to the solution obtained by CP.
3. Start the loop (i.e., all service classes which belong to the predicted execution path) by performing a local swap search strategy procedure in order to improve the initial solution.
4. If the obtained solution (obtained after performing the local swap strategy) realizes a better solution value compared to the initial one, then set the best current solution equal to the obtained one.
5. Repeat the loop until no more classes remain.

The CCP is described in Figure 5.9.

Input: a feasible solution *SN* with value *O(SN)*
Output: an improved SN^* with value $O(SN^*)$

```
1. $SN = (S_1,...,S_a) \leftarrow CP()$
2. $SN* \leftarrow SN$
3. **For** every $S_i \in EP_{pred}$
4. $value \leftarrow u_{iS_i}$
5. $k_i \leftarrow S_i$
6. **For** ( $j = 1,...,b_i$ and $s \neq S_i$ ) do
7. **If** ( $u_{ij} > value$ ) then
8. $Drop_k(R^k, \overset{k}{q_{iS_i}})$
9. **If**( $Add(R^k, {q_{ij}}^k) \leq GS^k, \forall k = 1,...,n$ ) then
10. $value \leftarrow u_{ij}$
11. $k_i \leftarrow s$
12. **End if**
13. **End if**
14. **End for**
15. $s_i' \leftarrow k_i$
16. $SN_i \leftarrow s_i'$
17. $SN \leftarrow (S_1,...,s_i',...,S_a)$
18. **If**( $O(S_1,...,s_i',...,S_a) > O(SN^*)$ ) then
19. $SN^* \leftarrow (S_1,...,s_i',...,S_a)$
20. **End if**
21. **End for**
```

Return solution SN^* with QoS value $O(SN^*)$, $O(SN^*)$ only computes the QoS value for every class in the predicted execution path (EP_{pred}) i.e., $\forall SN_i \in EP_{pred}$

Figure 5.9. *The Complementary Constructive Procedure (CCP)*

5.9 Chapter Summary

A smart QoS-aware service composition approach is proposed for multiple paths compositions. In this approach, the new optimization mechanism is proposed which computes the optimization by considering only the path that will potentially be followed during the execution of a composition. The optimization mechanism is performed in two steps: (1) predict the execution path that will potentially be executed and (2) compute the optimization for the predicted path. To do so, a runtime path prediction method, which is based on data mining techniques, is proposed. The method is composed of four phases, namely log preparation phase, preparation of training dataset phase, learning phase, and runtime path prediction phase. To compute the optimization; first, due to the similarity between the selection problem and the MMKP problem, the QoS-aware service composition problem is mapped to MMKP. Second, heuristic optimization algorithms called CP and CCP are applied to solve the selection problem. CP is used to generate an initial solution. Then CCP is used to iteratively improve the initial solution.

CHAPTER SIX
PERFORMANCE EVALUATION OF THE PROPOSED APPROACH

In order to evaluate the proposed approach, the evaluation process is divided into two parts. The first part aimed to evaluate the runtime path prediction method. With the aim of validating and studying the accuracy and the scalability of the prediction method, this chapter presents the datasets description and the different test experiments with their results. The second part aimed to evaluate the optimization mechanism. For this purpose, this chapter presents the simulation prototype and its setup. Based on the simulation prototype, different test experiments are introduced and their results are analyzed. These test experiments are designed to evaluate particular aspects of the optimization mechanism.

6.1 **Evaluation of Runtime Path Prediction Method**

In order to evaluate the runtime path prediction method, two different composite service scenarios, namely auto insurance and bank loan, are used to create several datasets. The scenarios are used to show how generable is the proposed approach. The following sections present the datasets and the different experiments used for the purpose of evaluating the runtime path prediction method.

6.1.1 Datasets Description

Section 3.4.1.2 has described the process of collecting data about auto insurance and bank loan processes. The data were used to create 10 datasets representing the auto insurance and bank loan process problems. The datasets were used for the purpose of evaluating the path prediction method.

The first dataset represented an auto insurance problem and characterized by four attributes, namely a policy-type, a manufacture-type, an auto-model, and a production-year. The attributes policy-type, manufacture-type, and auto-model are nominal while the production-year is numeric. For example, a policy-type attribute can take comprehensive and third party values.

Beside the auto insurance dataset, several datasets were required for evaluating the scalability of the path prediction method. Each dataset should include different numbers of involved paths. For this purpose, a bank loan problem was used to create 9 datasets representing variable numbers of paths ranging from 2 up to 10 paths. Paths are identified based on the bank loan composite service illustrated in Figure 5.4 which included 10 paths. To effectively compare between the learning algorithms when evaluating the scalability, each dataset has an equal number of service instances, i.e., 1000 instances; therefore, an equal size of subsets (i.e., 100 instances) can be obtained in each iteration of the 10-fold cross-validation method. Table 6.1 lists the 9 datasets used for evaluating the scalability of the path prediction method. The loan datasets are characterized by four attributes, namely income, loan-amount, a loan-type, and a loan year. The attribute income, loan-amount, and loan-years are numeric whereas the attribute loan-type is nominal. The attribute loan-type can take the finite set of values: a home loan, an education loan, a new car loan, a personal loan and a used car loan. These types are the most common loan types (Choen, 2012).

For all created datasets i.e., 10 datasets, the most informative attributes were selected for each dataset which was determined using WEKA's "Select attributes" technique (Hall et al., 2009). In addition, a class, namely path was added as an extra field for each instance in the datasets for the purpose of path prediction. It indicates that the path has been followed by each instance. The class path of the auto insurance dataset can take a finite set of values: $Path_1$, $Path_2$, $Path_3$, and $Path_4$. These four paths are contained in the auto insurance composite service as seen from Figure 5.3. A detailed description of each path is presented in Table 6.2. On the other hand, the class path in loan datasets can take a finite set of values: $Path_1$, $Path_2$... $Path_{10}$ as seen in Figure 5.4. A detailed description of each path is presented in Table 6.3.

The class path is labeled based on the instance data and the decision that has been made when evaluating the instance (i.e., either approve or reject a process request). For example, assume that third party insurance has been requested by the auto insurance's requester, and the request is rejected. Then, as seen in Table 6.2, the class path is labeled as $Path_4$.

Table 6.1

Datasets Description for Bank Loan Composite Service

Dataset	No. of paths	No. of instances	Loan type	Included classes(paths)
Dataset1	2 paths	1000	New Car	$Path_5, Path_6$
Dataset2	3 paths	1000	New Car Education	$Path_4, Path_5, Path_6$
Dataset3	4 paths	1000	New Car Personal	$Path_5, Path_6, Path_7, Path_8$
Dataset4	5 paths	1000	New Car Home	$Path_1, Path_2, Path_3, Path_5, Path_6$
Dataset5	6 paths	1000	New Car Home Education	$Path_1, Path_2, Path_3, Path_4, Path_5, Path_6$
Dataset6	7 paths	1000	New Car Home Personal	$Path_1, Path_2, Path_3, Path_5, Path_6, Path_7, Path_8$
Dataset6	8 paths	1000	New Car Home Education Personal	$Path_1, Path_2, Path_3, Path_4, Path_5, Path_6, Path_7, Path_8$
Dataset8	9 paths	1000	New Car Home Personal Used Car	$Path_1, Path_2, Path_3, Path_5, Path_6, Path_7, Path_8, Path_9, Path_{10}$
Dataset9	10 paths	1000	New Car Home Education Personal Used Car	$Path_1, Path_2, Path_3, Path_4, Path_5, Path_6, Path_7, Path_8, Path_9, Path_{10}$

Table 6.2

Path Description for Auto Insurance Composite Service

Path	Policy Type /Decision	Path description
$Path_1$	Comprehensive Approved	CheckPolicy, EvaluateComprehensive, RejectComprehensive, NotifyComprehensiveClient, ArchiveApplication
$Path_2$	Comprehensive Rejected	CheckPolicy, EvaluateComprehensive, ApproveComprehensive, NotifyComprehensiveClient, ArchiveApplication
$Path_3$	Third Party Approved	CheckPolicy, EvaluateThirdParty, ApproveThirdParty, NotifyThirdPartyClient, ArchiveApplication
$Path_4$	Third Party Rejected	CheckPolicy, EvaluateThirdParty, RejectThirdParty, NotifyThirdPartyClient, ArchiveApplication

Table 6.3

Path Description for Bank Loan Composite Service

Path	Loan Type /Decision	Description
$Path_1$	Home Approved	CheckLoanType, CheckHomeLoan, ApproveHomeLoan, NotifyHomeLoanClient, ArchiveApplication
$Path_2$	Home Approved Conditionally	CheckLoanType, CheckHomeLoan, RejectHomeLoan, NotifyHomeLoanClient, ArchiveApplication
$Path_3$	Home Rejected	CheckLoanType, CheckHomeLoan, ApproveHomeLoanConditionaly, NotifyHomeLoanClient, ArchiveApplication
$Path_4$	Education Approved	CheckLoanType, CheckEducationLoan, ApproveEducationLoan, NotifyEducationLoanClient, ArchiveApplication
$Path_5$	New Car Approved	CheckLoanType, CheckNewCarLoan, ApproveNewCarLoan, NotifyNewCarLoanClient, ArchiveApplication
$Path_6$	New Car Rejected	CheckLoanType, CheckNewCarLoan, RejectNewCarLoan, NotifyNewCarLoanClient, ArchiveApplication
$Path_7$	Personal Approved	CheckLoanType, CheckPersonalLoan, ApprovePersonalLoan, NotifyPersonalLoanClient, ArchiveApplication
$Path_8$	Personal Rejected	CheckLoanType, CheckPersonalLoan, RejectPersonalLoan, NotifyPersonalLoanClient, ArchiveApplication
$Path_9$	Used Car Approved	CheckLoanType, CheckUsedCarLoan, ApproveUsedCarLoan, NotifyUsedCarLoanClient, ArchiveApplication
$Path_{10}$	Used Car Rejected	CheckLoanType, CheckUsedCarLoan, RejectUsedCarLoan, NotifyUsedCarLoanClient, ArchiveApplication

6.1.2 Experiments and Results

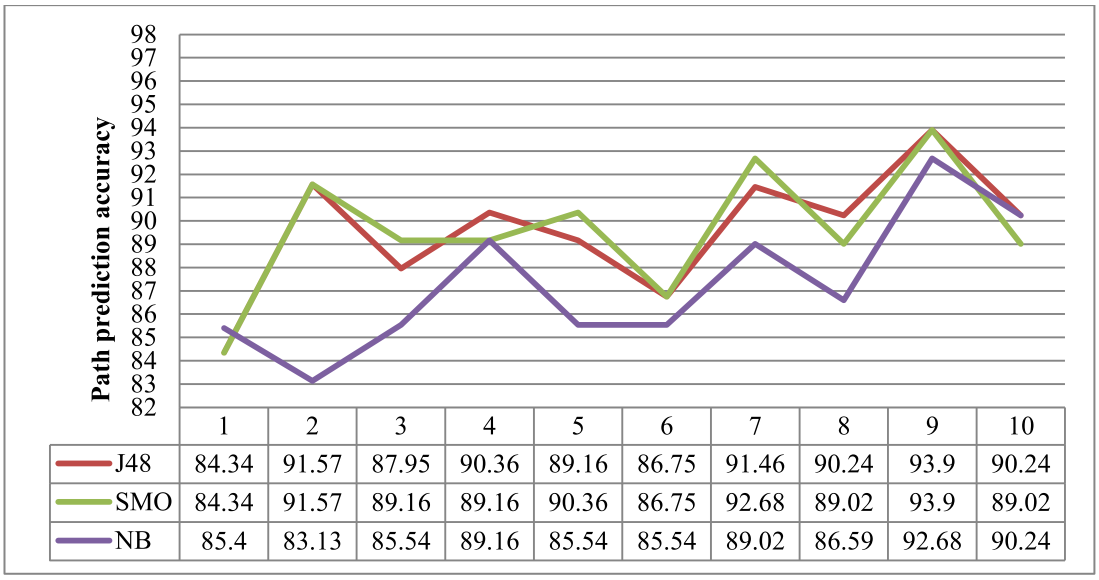
In order to evaluate the runtime path prediction method, different test experiments with their results were presented in this section. These experiments aimed to validate and study the accuracy and the scalability of the prediction method.

6.1.2.1 The Accuracy of Path Prediction

In the proposed approach, it is crucial to have high prediction accuracy when predicting the execution paths because the optimization process depends on the predicted path. Any false prediction means that the optimization results in solutions that may have a low QoS ratio or may violate the global constraints. Therefore, the first experiment aimed at validating the accuracy of the path prediction.

The experiments were conducted using three selected learning algorithms, namely J48, NB, and SMO. Details about these algorithms are presented in Section 3.4.1.5. These algorithms are applied to the auto insurance dataset which contains 826 instances.

As mentioned in Section 3.4.1.6, by using the10-fold cross validation method to train and test learning algorithms, the algorithms train and test 10 times, meaning that there are 10 prediction accuracy results were produced by this procedure. These results, as they are presented in Figure 6.1. Table 6.4 depicts the average results obtained for the various measures used for evaluating the selected classifiers.



	1	2	3	4	5	6	7	8	9	10
🟥 J48	84.34	91.57	87.95	90.36	89.16	86.75	91.46	90.24	93.9	90.24
🟩 SMO	84.34	91.57	89.16	89.16	90.36	86.75	92.68	89.02	93.9	89.02
🟪 NB	85.4	83.13	85.54	89.16	85.54	85.54	89.02	86.59	92.68	90.24

Figure 6.1. *The prediction accuracy per each fold achieved using J48, NB, and SMO classifiers when applied to the auto insurance dataset*

Table 6.4

Evaluation Criteria Results Achieved by Using the J48, NB, and SMO Classifiers When Applied to Auto Insurance Dataset

Evaluation Criteria	Classifiers		
	J48	SMO	NB
Prediction Accuracy	89.60	89.60	87.30
Precision	0.90	0.90	0.88
Recall	0.94	0.95	0.95

The results presented in Table 6.4 indicate that all the selected classifiers achieved promising accuracy rates ranging from 87.30 % to 89.60 % compared with the path prediction method by Cardoso (2005, 2008) and Cardoso and Lenic (2006), which is the most comparable method to the proposed method in this work. This is expected because learning algorithms in the proposed method are trained on the most informative attributes of instances executions. This allows classifiers for better learning and consequently improves the prediction quality. Furthermore, in the proposed method, the attributes used for learning are a kind of must entered attributes that are provided by service requesters i.e., no missing attribute values. Using datasets with missing values to train classifiers can affect the prediction quality of classifiers (Acuna & Rodriguez, 2004; Liu, Lei & Wu, 2005; Blomberg & Ruiz, 2013).

As presented in Table 6.4, both J48 and SMO classifiers achieve the highest accuracy prediction, i.e. 89.60. The lowest accuracy is achieved by using NB classifiers i.e., 87.30. It is observed from Figure 6.1 that both J48 and SMO produce prediction models with the best accuracies in 9 out of 10 tests than NB. NB outperforms both J48 and SMO in the first test only.

Comparing the precision and recall results of all classifiers, higher precision results of J48 and SMO indicate that both have a high proportion of the true positives against all the positive results. For recall results, the results indicate that SMO and NB have a high proportion of actual positives which are correctly identified as such.

In the form of a bar diagram, Figure 6.2 illustrates the number of correctly/incorrectly classified instances using all classifiers. It is seen that the total number of instances, i.e., 826 is equal in the three cases since the same dataset is used in the experiment. As seen in Figure 6.2, both J48 and SMO classifiers are able to correctly classify 740 instances out of 826 instances. Only 86 instances are incorrectly classified by these classifiers. However, 721 instances are correctly classified by NB and 105 instances are incorrectly classified.

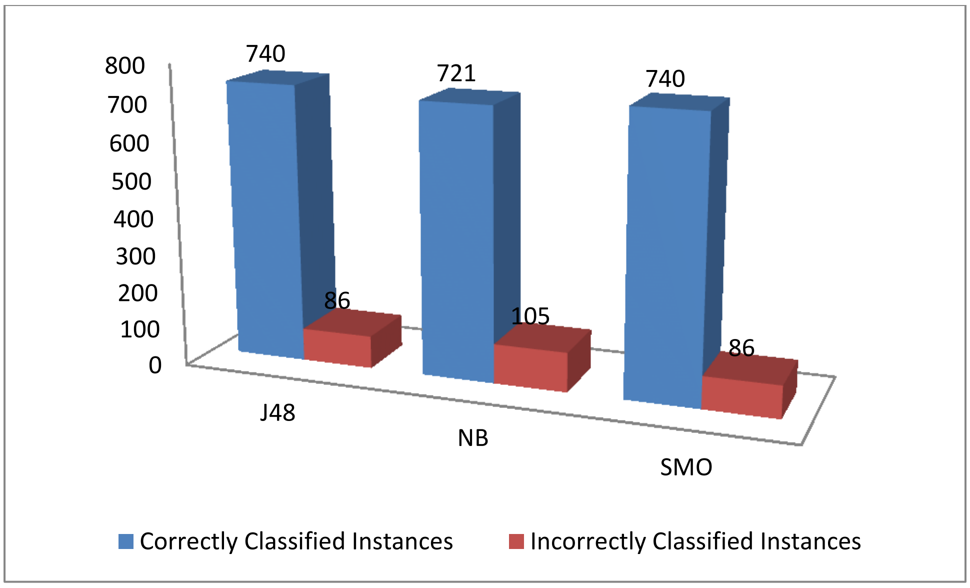


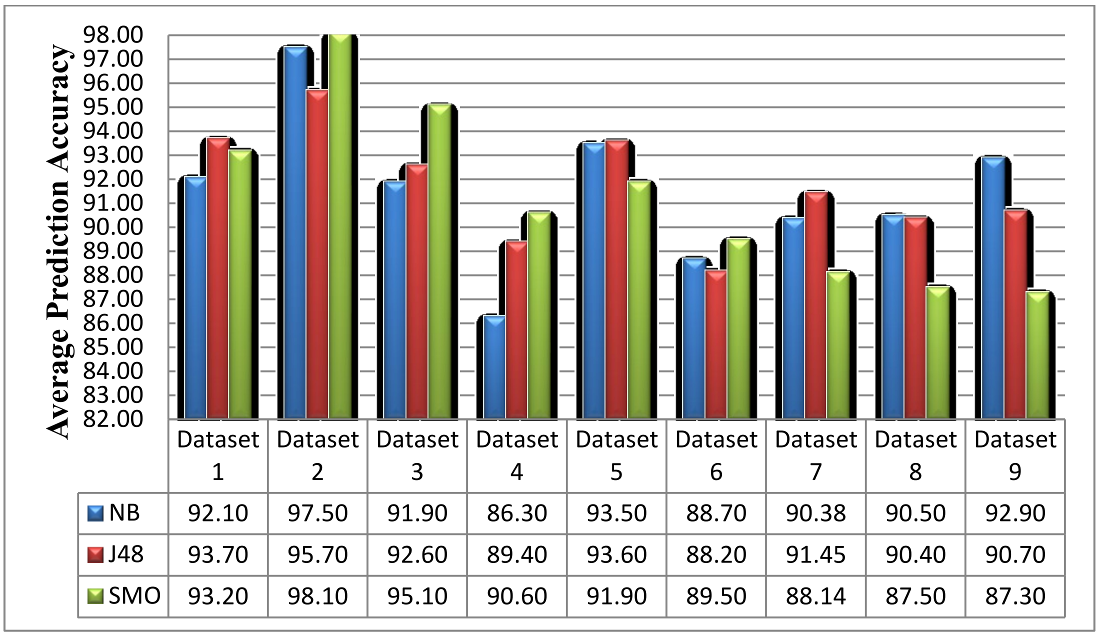
Figure 6.2. *A number of correctly/incorrectly classified instances achieved by using J48, NB, and SMO classifiers when applied to the auto insurance dataset*

Having such encouraging results of prediction accuracy contributes to the generation of high QoS ratio solutions and minimizes the constraints violated a number of the generated solutions.

6.1.2.2 Scalability of the Prediction Method

The number of execution paths involved in compositions varies between one composite service and another one making us wonder about the prediction method's ability to accurately predict the paths when having a growing number of involved paths. Therefore, the second experiment aimed at studying how the prediction method scaled with a rising number of involved execution paths. For this purpose, 9 datasets representing the bank loan process were used for this experiment. Table 6.1 shows the datasets. For fair comparison, each dataset contained 1000 instances. The instances represented a bank loan process that involved execution paths ranging from 2 up to10 paths. The experiments were conducted using J48, NB, and SMO.

The average of the prediction accuracy for the three classifiers when applied to the 9 datasets is illustrated in Figure 6.3.



	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7	Dataset 8	Dataset 9
🟦 NB	92.10	97.50	91.90	86.30	93.50	88.70	90.38	90.50	92.90
🟥 J48	93.70	95.70	92.60	89.40	93.60	88.20	91.45	90.40	90.70
🟩 SMO	93.20	98.10	95.10	90.60	91.90	89.50	88.14	87.50	87.30

Figure 6.3. *An average prediction accuracy achieved by using NB, J48, and SMO classifiers when applied to 9 different datasets*

As illustrated in Figure 6.3, it can be observed that the accuracy results for all classifiers are varied, which make us wonder about the reasons behind the variance in the prediction accuracy results. Do the results of the prediction accuracy depend on the number of paths involved in a dataset (i.e., composite service)? To answer this question, take Dataset1, Dataset5, Dataset4, and Dataset9 as examples. Table 6.5 presents the average accuracy and the number of involved paths for these datasets.

Table 6.5

Average Accuracy and Number of Involved Paths for Dataset1, Dataset5, Dataset4, and Dataset9

Dataset	Number of involved paths	Average accuracy
Dataset1	2 paths	93
Dataset5	6 paths	93
Dataset4	5 paths	88.80
Dataset9	10 paths	90.30

As seen from Table 6.5, Dataset1 involves 2 paths while Dataset5 involves 6 paths. The average prediction accuracy of all classifiers when applied to Dataset1 is 93 which is equal to the average prediction accuracies for all classifiers when applied to Dataset5 which is 93. Furthermore, take Dataset4, which involves 5 paths, and Dataset9 which involves 10 paths, the average prediction accuracy of all classifiers when applied to Dataset4 i.e., 88.80 is less than Dataset9 i.e., 90.30. Even that Dataset9 includes 10 paths; it has a higher average prediction accuracy than Dataset4 which includes 5 paths. Based on these findings, it is valid to conclude that the rising number of classes involved in the prediction does not affect the prediction accuracy of the classifiers. In other words, there is no relationship between the number of classes involved in the classification process and the prediction accuracy of the classifier.

What causes the variation in the accuracy results is a question that is still needed to be answered? To answer this question, a comparison between the Dataset2, which has the maximum average accuracy, i.e., 96.10, and the Dataset4, which has the minimum i.e., 88.80, is needed to be conducted. An extra test experiment was conducted and aimed at studying these datasets. SMO classifier was chosen to be applied on these datasets since it had the maximum prediction accuracy when applied to these datasets. Having got the experiment results, it was noticed that the precision and the recall results for some classes (i.e., paths) were either very low or very high in comparison with other classes in the same dataset. Table 6.6 presents the precision and the recall results for Dataset2 while Table 6.7 presents the precision and the recall results for Dataset4. As seen in Table 6.6, the precision and the recall results for *path$_4$* are very high i.e., 1. It indicates that there are no incorrectly classification for this class i.e., $Path_4$. In comparison with $Path_2$ in the Dataset4, the precision and the recall results for $Path_2$ are low 0.67 and 0.69 respectively as presented in Table 6.7.

Table 6.6

Precision and Recall Results for Dataset2

Class	Precision	Recall
Path4	1	1
Path5	0.94	0.92
Path6	0.89	0.92

Table 6.7

Precision and Recall Results for Dataset4

Class	Precision	Recall
Path1	0.93	0.97
Path2	0.67	0.69
Path3	0.95	0.92
Path5	0.92	0.91
Path6	0.85	0.86

As seen in Table 6.1, Dataset2 represents two loan types, namely a new car, which involves 2 paths (i.e., $Path_5$ and $Path_6$), and education, which involves 1 path (i.e., $Path_4$). It is clear that the simple structure of education (i.e., only one class belongs to the education loan type) is the reason behind the very high results of precision and recall for this class. These high results contribute in achieving high accuracy results compared with the complex home loan structure, where $Path_2$, which consists of 3 paths, belongs to the home loan type. Investigating other datasets, which have a low average prediction accuracy such as Dataset6 88.80, and Dataset8 89.50, showed that the presence of home loan type (i.e., $Path_2$) is the reason behind the low average accuracy results. Based on these findings, it can be concluded that the accuracy highly depends on the structure of compositions.

Based on the results of the this test experiment, it is valid to say that the proposed approach is suitable for any compositions regardless of the number of execution paths involved in a composition. However, the structure of the business process plays an important role in the results of prediction accuracy.

6.2 Evaluation of Optimization Mechanism

Based on the model of the multiple paths QoS-aware service composition problem, the algorithms, and the methods which were given in the previous chapters, the proposed approach was implemented and evaluated by using simulation prototype. In the following, the simulation prototype is described in detail. Based on the simulation and its setup, different test experiments are introduced and their results are analyzed. These test experiments are designed to evaluate particular aspects of the optimization mechanism.

6.2.1 A Simulation Prototype

The lack of general simulation that can be used in the area of QoS in service composition makes the researcher implement their own simulation. Therefore, a simulation prototype is developed for the purpose of evaluating the new optimization mechanism for handling multiple paths composition problem. The proposed optimization mechanism was implemented and evaluated by using simulation prototype.

In order to evaluate the new optimization mechanism, a comparative evaluation experiment between the proposed optimization mechanism and the current techniques existing in the state of the art approaches (Zeng et al., 2003; Zeng et al., 2004; Yu, Zhang & Lin, 2007; Canfora, Penta, Esposito & Villani, 2005; Jaeger et al., 2004; Zhang et al., 2010; Parejo, Fernandez & Cort´es, 2008; Ko et al., 2008; Ukor & Carpenter, 2008, 2009) was conducted. For fair comparisons, it is preferable to conduct the comparisons using the same optimization approach. Therefore, CP and CCP algorithms were used for this purpose. The simulations implement three different versions of the CP and CCP algorithms. Each version represents a particular optimization technique for handling multiple paths composition problem. Based on these techniques, three different versions of the CP and CCP algorithms were implemented, namely CP1 & CCP1, CP2 & CCP2, and CP3 & CCP3. Section 3.4.2.4 has explained the versions of the CP and CPP algorithms in detail.

In order to compare the optimization techniques (i.e., three versions of CP and CCP algorithms), the simulation was divided into three test experiments which were designed to compare all the techniques from particular aspects. Different measures are captured by the simulation, namely: the resulting aggregated QoS, the imposed constraint values, the aggregated QoS value relative to the considered constraint characteristic, and finally the computation time. Section 3.4.2.3 explains the evaluation measures in detail.

The main idea of the simulation was to generate problem instances, and then let the implementations of the three algorithm versions solve these instances. The generations of the elements that constitute a problem instance are in the following way:

Abstract and candidate services: the amount of abstract service depends on the considered composite service scenario. For candidate services, the amount of candidate is set to fix the value. The next subsection discussed these parameters in detail.

QoS values of the candidates: the values were generated stochastically as explained in the next subsection.

Optimization goal: in the entire simulation, the optimization goal remains the same; the goal was to optimize four QoS characteristics. The next subsection discussed these characteristics.

Constraints: the amount of the constraints is either increased or set to a fixed value. The range of constraints amount is between 1 to 4 constraints.

Structures: the structure of a composition is generated as in a bank loan composite service, illustrated in Figure 5.4, which consists of 10 execution paths.

6.2.1.1 Parameters

The first parameter that needs to set is the amount of abstract services i.e., composition size. This parameter depends on the considered composite service scenario which is the bank loan composite service. The service illustrated in Figure 5.4 is composed of 22 abstract services. Thus, the parameter is set to 22.

Beside the composition size parameter, there are many parameters which need to be set. These parameters are number of candidate services, cost, response time, reputation, and availability parameters. For setting these parameters, this work follows Jaeger (2007) who studied the QoS aware service composition problem, and discusses different optimization algorithms as solutions. The author proposes simulation software called SENECA to evaluate these algorithms. For the simulation, Jaeger (2007) studied and analyzed different parameters in detail in order to set these parameters. In this work, the value ranges of the parameters: number of candidate services, response time, reputation, and availability parameter are based on Jaeger (2007). Regarding the value range of cost parameter, the author mentioned that this parameter is individually set based on the payment model and the considered currency. This work sets this parameter based on the pay-per-use model which is according to Weinhardt, Anandasivam, Blau, and Stosser (2009), is the most commonly model used in Business Process Management System (BPMS). For currency, any currency can be used for an amount of a particular currency could be transferred into another currency. In this work, Dollar currency was used. Finally, one more parameter needed to be set is the probability of path execution values. This parameter was used only in all paths optimization technique to compute the optimization. This work sets the parameter by counting the frequency of each path being executed and divides the results on the total number of all instances (Canfora et al., 2005; Yu, Zhang & Lin, 2007; Ukor & Carpenter, 2008, 2009; Jiang et al., 2011; Singh, 2012). The following formula is used for the probability computation:

$$P_i = \frac{the\ frequency\ of\ exeuction\ of\ path_i}{1000} \tag{6.1}$$

Where P_i is the probability execution of $Path_i$ such that $P_i \in [0,1]$ and $\sum_{i=1}^{k} P_i = 1$, and k is the total number of paths. Table 6.8 summarizes the value ranges of the parameters.

Table 6.8
Parameter Value Ranges of the Simulation

Parameter	Value
Number of abstract services	22
Number of candidate services	50
Cost	[0 … 10] Dollar
Response time	[150 … 9999] milliseconds
Reputation	[0 … 10]
Availability	[0.9650 … 0.9999]
Probability	[0 …1]

6.2.1.2 Implementations

The simulation applies each of the different optimization techniques to solve the generated problem instances. The generation of the problem instances includes the generation of composition structures, candidates QoS values, and constraint values.

As mentioned earlier, the composition structure is generated as in the bank loan composite service which consists of sequential and conditional structures. For candidates QoS values and constraints values, the generation is as Jaeger (2007):

- Four QoS characteristics are considered for the simulation: cost, response time, availability, and reputation. The simulation generates candidate services with random QoS values. To ensure realistic QoS variance, the simulation randomly assigns for each abstract service optimal cost and response time from the intervals given in Table 6.9.

Table 6.9

Cost and Response Time Intervals

QoS characteristics	Value range
Cost	[0 … 5]
Response time	[150 … 2000]

The formula used to generate a QoS values for cost and response time is as the following (Jaeger, 2007):

$$Q = optimal\ value \times (1 + x), 0 \leq x \leq 1 \qquad (6.2)$$

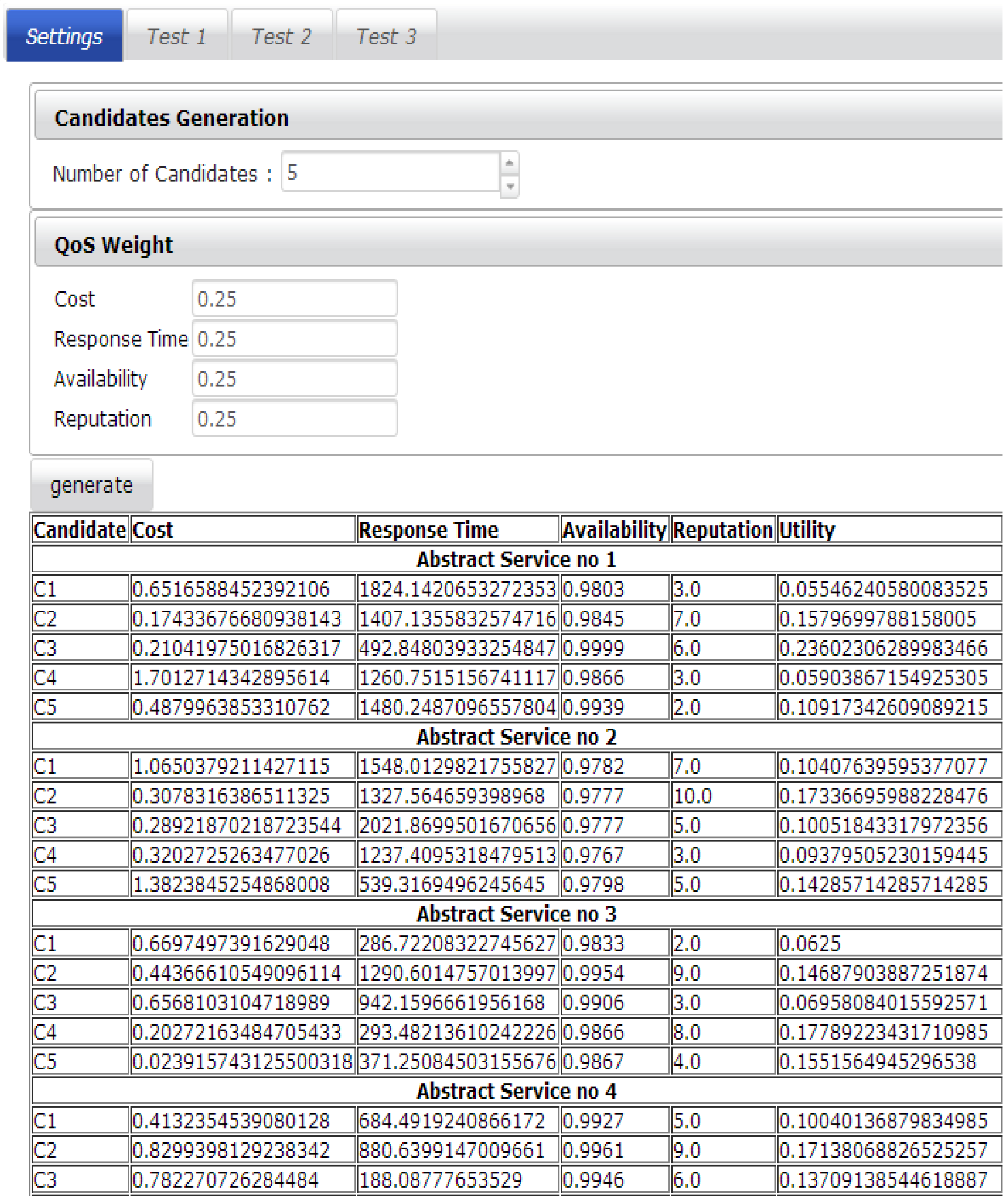
The variable x is a random determined percentage between 0 and 100. A trade-off couple between cost and response time is needed to be formed. The better the cost is, the worse response time and vice versa. To do so, the value x added to the optimal cost is taken to calculate x_1 the value added to the optimal response time, with $x + x_1 = 1$.

Regarding the availability and reputation, the simulation chooses, with uniform distribution, a random value between their intervals as given in Table 6.8. Figure 6.4 shows an example of four QoS characteristics values generation of five candidate services. Figure 6.4 also shows the utility value for each candidate. The utility value is computed using the Equation 5.3, and considering equal weights.

- After determining the composition structure and the candidate QoS values, the constraints needed to be determined. Cost is the considered constraint characteristic for the resulting QoS ratio and the computation time test experiments. Regarding the calculated constraints violated number test, it aimed to compare the three techniques in terms of the constraint violation. Thus, it was necessary to consider the four constraint characteristics, namely cost, response time, availability, and reputation. The values of the constraints were determined after running the following algorithm:

  1. Considering the constraint QoS characteristic; calculate the average of the candidates QoS values for each abstract service.
  2. Calculate the total average of the candidates QoS values for all the abstract services.
  3. Increase the total average by a set of percentage (for example by 30%).

After the generation of the problem instances, the simulation runs the three test experiments. For the first test, the simulation captures the resulting aggregated QoS. In the second test, the simulation captures the considered constraint values as well as the aggregated QoS value related to the constraint characteristic. While in the third test, the simulation captures the computation time in a microsecond. The computation time is the time taken by the algorithm to compute a solution. Thus, the time needed for the generation of the problem instance is not captured by the measurements.



Candidates Generation

Number of Candidates : 5

QoS Weight

Cost	0.25
Response Time	0.25
Availability	0.25
Reputation	0.25

Candidate	Cost	Response Time	Availability	Reputation	Utility
		Abstract Service no 1			
C1	0.6516588452392106	1824.1420653272353	0.9803	3.0	0.05546240580083525
C2	0.17433676680938143	1407.1355832574716	0.9845	7.0	0.1579699788158005
C3	0.21041975016826317	492.84803933254847	0.9999	6.0	0.23602306289983466
C4	1.7012714342895614	1260.7515156741117	0.9866	3.0	0.05903867154925305
C5	0.4879963853310762	1480.2487096557804	0.9939	2.0	0.10917342609089215
		Abstract Service no 2			
C1	1.0650379211427115	1548.0129821755827	0.9782	7.0	0.10407639595377077
C2	0.3078316386511325	1327.564659398968	0.9777	10.0	0.17336695988228476
C3	0.28921870218723544	2021.8699501670656	0.9777	5.0	0.10051843317972356
C4	0.3202725263477026	1237.4095318479513	0.9767	3.0	0.09379505230159445
C5	1.3823845254868008	539.3169496245645	0.9798	5.0	0.14285714285714285
		Abstract Service no 3			
C1	0.6697497391629048	286.72208322745627	0.9833	2.0	0.0625
C2	0.44366610549096114	1290.6014757013997	0.9954	9.0	0.14687903887251874
C3	0.6568103104718989	942.1596661956168	0.9906	3.0	0.06958084015592571
C4	0.20272163484705433	293.48213610242226	0.9866	8.0	0.17789223431710985
C5	0.023915743125500318	371.25084503155676	0.9867	4.0	0.1551564945296538
		Abstract Service no 4			
C1	0.4132354539080128	684.4919240866172	0.9927	5.0	0.10040136879834985
C2	0.8299398129238342	880.6399147009661	0.9961	9.0	0.17138068826525257
C3	0.782270726284484	188.08777653529	0.9946	6.0	0.13709138544618887

Figure 6.4. *A screen capture of generation of 5 candidate services and their QoS characteristics values and utility*

6.2.1.3 Software and Hardware Simulation

For computation time comparison, only the operating system and the software environment were installed on the computer that hosted the simulation prototype. Thus, no other processes running in parallel affect the measurements.

The simulation prototype is a web application implemented in Java language, so it can be run on different platforms and operating systems. In addition, it can be accessed from a wide range of researchers via web browsers.

One extra library, i.e., WEKA was added to the simulation prototype for the purpose of performing path prediction using WEKA built in machine learning algorithms. The software is developed using the Java Enterprise Edition (EE) 5 platform and the NetBeans IDE (Integrated Development Environment) (V6.8). NetBeans IDE is an open source IDE for developing an application. NetBeans IDE supports the Java EE 5 platform. Java EE 5 platform provides the operating *System.nanoTime( )* which is a precise measurement of time.

Concerning the host computer, a standard hardware with Windows 6 Professional operating system was used. The processor is an AMD Turion (tm) X2 Dual-Core Mobile RM-62 2.10 GHz. Memory 2.00 GB RAM.

6.2.2 Experiments and Results

In the following, three test experiments were discussed and their results were analyzed. These test experiments have a goal to evaluate the proposed mechanism by comparing it with the two previously mentioned techniques. The comparisons were conducted using the CP1 & CCP1, CP2 & CCP2, CP3 & CCP3 algorithms which were implemented based on these techniques. Regarding the number of runs, from a preliminary test, a few rounds of testing show that the optimal results can be obtained after 20 number of runs. It is shown that for this number of runs, a test experiment results in almost similar statistical results when run again.

One general issue has been applied to the resulting QoS ratio and the constraints violated number test experiments. This issue is related to the computation of the resulting QoS ratio and the calculated constraints violated number when comparing all techniques. The proposed mechanism computes the optimization considering one path, i.e., the predicted path. Consequently, it is logic to compute the QoS ratio and the constraints violated a number of the services that belong only to the predicted path. On the other hand, the other techniques compute the optimization for all services involved in a composition, and it is logic to compute the QoS ratio and the constraints violated number for all services that belong to this composition. The comparison between all techniques is not fair because the results obtained from the proposed mechanism (represent only one path) are different from the results obtained from the other techniques (represent all paths). Therefore, the comparisons are conducted on the assumption that the predicted path is the path that is executed at runtime, and for all the techniques involved in comparisons, the resulting QoS ratios and the calculated constraints violated numbers are computed considering only the services that belong only to the predicted path. This assumption is valid because the accuracy of path prediction has been evaluated in Section 6.1.2.1, and the results are promising.

However, one can argue that comparing one single path with another is not enough to evaluate the existing techniques. Therefore, the comparisons between all 10 paths involved in the considered composite service scenario were conducted to cover all of them, i.e., each simulation run is repeated 10 times to cover all the 10 paths. In each time, the simulation considers a particular path from the 10 paths as the predicted path.

6.2.2.1 QoS Ratio

This test experiment has a goal to evaluate the proposed mechanism in terms of the resulting QoS ratio. Quantitative statements that represent the scores of the QoS resulting from each technique can be achieved by using the SAW method introduced in Section 5.6. The different QoS characteristics are aggregated considering equal weights. Then the 10 QoS ratio results for each run (i.e., 1 QoS ratio result for each path) were averaged to give the overall result. The test that used the setup is listed in Table 6.10.

Table 6.10

Setup for the Resulting QoS Ratio Test

Setup	Value
Number of abstract services	22
Number of candidate service	50
QoS characteristics of the candidate	As given in Table 6.8
Constraint	Cost characteristic
Algorithms for comparison	CP1 & CCP1, CP2 & CCP2, CP3 & CCP3

Simulation Results

The results of this test experiment are shown in Figure 6.5. Figure 6.5 shows the resulting average QoS ratio comparison of the different algorithms in each run. In addition, Table 6.11 lists the total QoS ratio, average of the total QoS ratio, and standard deviation.

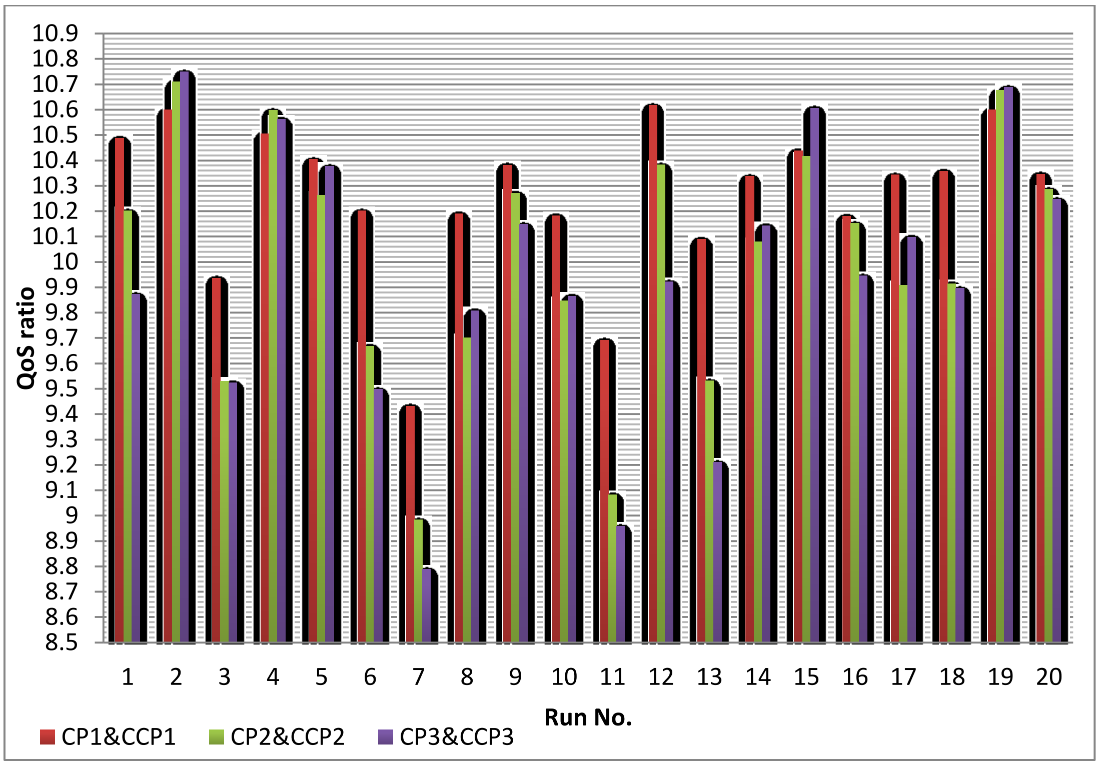


Figure 6.5. *Average QoS ratios of the different algorithms in each run*

Table 6.11

The Results of QoS Ratio Test

Algorithm	Total QoS ratio	Average of total QoS ratio	Standard Deviation
CP1 & CCP1	205.318	10.266	0.302
CP2 & CCP2	200.196	10.010	0.486
CP3 & CCP3	198.898	9.945	0.546

The best algorithm is the algorithm that has high overall QoS ratio and high average of the total QoS ratio. As shown in Figure 6.5, CP1 & CCP1 algorithms, which represent the proposed optimization mechanism, produce average QoS ratios that are higher than the other algorithms in 16 out of 20 runs. Both CP2 & CCP2 and CP3 &

CCP3 outperform the CP1 & CCP1 only in 4 runs, i.e., run number 2, 4, 15, and 19. In these runs, the values of the imposed constraints are lower than the values in the rest of the runs. This means that there is a high possibility to violate such constraints. The total QoS ratio of a solution that violates a constraint may be higher than a solution that satisfies a constraint. For example, in run number 4, no constraints were violated using CP1 & CCP1 algorithms, 5 constraints were violated using CP2 & CCP2 algorithms, and 6 using CP3 & CCP3 algorithms. Such high numbers of constraints violated are the reason that these algorithms produced high values of the QoS ratio at considered runs. Compared to CP2 & CCP2 and CP3 & CCP3, no constrains were violated using the CP1 & CCP1 in all conducted runs. However, constraints were violated using CP2 & CCP2 and CP3 & CCP3 in all conducted runs except for run number 7. As mentioned earlier, this test imposed only one constraint, i.e., cost. The next test evaluated the constraints violated number when more than one constraint was considered.

As the Table 6.11 shows, CP1 & CCP1 algorithms show results QoS performances that outperform the existing algorithms by achieving the highest total QoS ratio and average of the total QoS ratio 205.318, 10.266 respectively. CP1 & CCP1 algorithms also show a small standard deviation 0.302. Regarding CP2 & CCP2 and CP3 & CCP3 algorithms, both algorithms show close resulting QoS performances by achieving total QoS ratios 200.196, 198.897 and the average total QoS ratio 10.010, 9.945 respectively.

These good results are expected because of the special behavior of CP1 & CCP1 algorithms which focus on optimizing only one execution path, i.e., the path will potentially be executed at runtime. Therefore, solutions produced by the new optimization mechanism have high QoS ratios compared with the solutions produced by other techniques which focus on all execution paths when performing optimization whereas only one path is executed at runtime.

6.2.2.2 Constraints Violated Number

This test experiment has a goal to evaluate the proposed mechanism in terms of the calculated constraints violated number. The numbers of constraints violated for all techniques were calculated in order to compare between them. For the calculation, with respect to the considered constraint characteristic, the aggregation functions presented in Table 5.2 were used to aggregate the value of constraint characteristics. Then a comparison between the aggregated value and the imposed constraint value was conducted to determine whether the constraint was violated or not (i.e., if the aggregated value is greater than the imposed constraint value, then the constraint is violated; otherwise, it is not violated). Finally, count if there is a violation. The test that uses the setup is listed in Table 6.12.

Table 6.12

Setup for the Constraints Violated Number Test

Setup	Value
Number of abstract services	22
Number of candidate service	50
QoS characteristics of the candidate	As given in Table 6.8, randomly set, uniformly distributed
Constraint	Cost, response time, availability, and reputation characteristics
Algorithms for comparison	CP1 & CCP1, CP2 & CCP2, CP3 & CCP3

Simulation Results

The test results are shown in Table 6.13. The table lists the average of the aggregated QoS characteristics produced by each technique in each run and the average of the constraint imposed for each considered QoS characteristic.

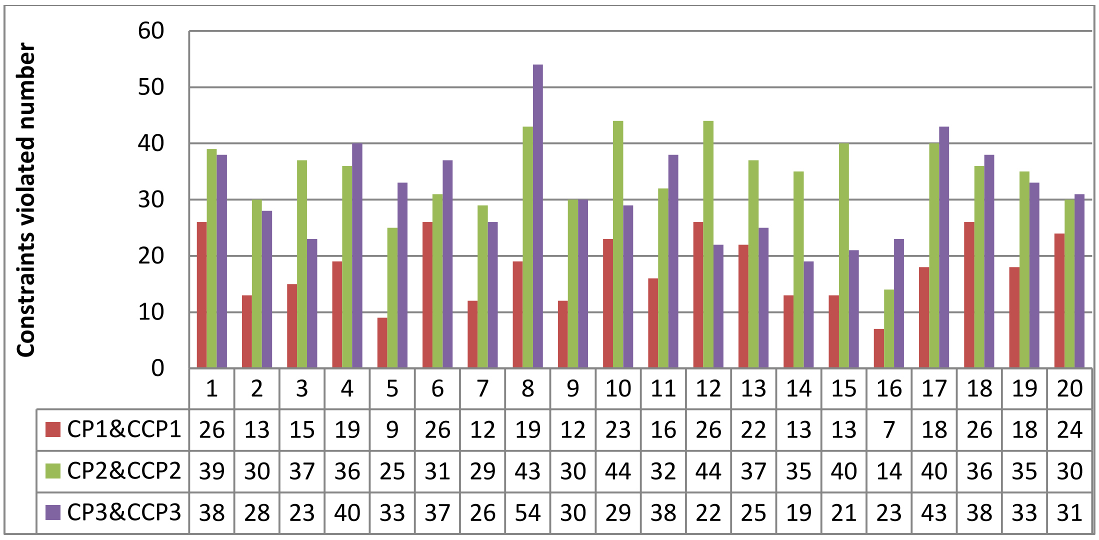
Figure 6.6 shows the calculated constraints violated numbers of the different considered algorithms in each run.

Table 6.13

The Average of the Aggregated QoS Characteristics in Each Run and the Average of the Constraint Imposed for Each Considered QoS Characteristic

Run no.	Technique	Aggr. cost	Cost const.	Aggr. Response T.	Response T. const.	Aggr. avail.	Avail. const.	Aggr. reput.	Reput. const.
1	CP1 & CCP1	1.112	1.163	3502.760	3925.232	0.975	0.968	7	5
1	CP2 & CCP2	1.383	1.163	3219.350	3925.232	0.977	0.968	8	5
1	CP3 & CCP3	1.299	1.163	3424.987	3925.232	0.974	0.968	7	5
2	CP1 & CCP1	0.868	1.135	3442.893	4079.841	0.972	0.968	7	7
2	CP2 & CCP2	1.031	1.135	3539.047	4079.841	0.977	0.968	8	7
2	CP3 & CCP3	1.012	1.135	3649.915	4079.841	0.975	0.968	7	7
3	CP3 & CCP3	1.434	1.143	3058.102	3983.068	0.974	0.968	7	6
3	CP1 & CCP1	0.926	1.143	2693.539	3983.068	0.974	0.968	8	6
3	CP2 & CCP2	1.427	1.143	2763.231	3983.068	0.984	0.968	9	6
4	CP2 & CCP2	1.195	1.171	2554.920	4025.645	0.972	0.967	8	5
4	CP3 & CCP3	1.254	1.171	2666.321	4025.645	0.969	0.967	6	5
4	CP1 & CCP1	0.953	1.171	2977.520	4025.645	0.968	0.967	7	5
5	CP1 & CCP1	1.103	1.193	2699.259	3992.783	0.975	0.967	5	4
5	CP2 & CCP2	1.212	1.193	2369.185	3992.783	0.977	0.967	8	4
5	CP3 & CCP3	1.251	1.193	2372.292	3992.783	0.972	0.967	6	4
6	CP3 & CCP3	1.239	1.146	2616.797	4073.224	0.968	0.968	7	6
6	CP1 & CCP1	1.161	1.146	2655.725	4073.224	0.967	0.968	8	6
6	CP2 & CCP2	1.229	1.146	2565.839	4073.224	0.973	0.968	8	6
7	CP2 & CCP2	1.17	1.166	2578.629	4018.266	0.974	0.968	8	6
7	CP3 & CCP3	1.156	1.166	2603.173	4018.266	0.973	0.968	7	6
7	CP1 & CCP1	1.018	1.166	2619.112	4018.266	0.974	0.968	7	6
8	CP1 & CCP1	1.16	1.169	3322.541	3999.051	0.974	0.968	6	5
8	CP2 & CCP2	1.621	1.169	3607.647	3999.051	0.982	0.968	8	5
8	CP3 & CCP3	1.424	1.169	4198.726	3999.051	0.967	0.968	6	5
9	CP3 & CCP3	1.129	1.162	3224.782	4129.261	0.981	0.968	7	6

9	CP1 & CCP1	1.129	1.162	2935.226	4129.261	0.978	0.968	7	6
9	CP2 & CCP2	1.152	1.162	2959.911	4129.261	0.988	0.968	8	6
10	CP2 & CCP2	2.087	1.157	2837.357	3985.524	0.984	0.968	9	6
10	CP3 & CCP3	1.661	1.157	2972.175	3985.524	0.978	0.968	7	6
10	CP1 & CCP1	1.1	1.157	2868.605	3985.524	0.969	0.968	7	6
11	CP1 & CCP1	1.173	1.196	3154.710	4057.039	0.978	0.968	5	5
11	CP2 & CCP2	1.325	1.196	3025.303	4057.039	0.987	0.968	9	5
11	CP3 & CCP3	1.563	1.196	2767.655	4057.039	0.981	0.968	5	5
12	CP3 & CCP3	1.286	1.198	2651.360	3982.601	0.969	0.968	7	5
12	CP1 & CCP1	1.151	1.198	2412.339	3982.601	0.969	0.968	7	5
12	CP2 & CCP2	1.213	1.198	2734.947	3982.601	0.972	0.968	9	5
13	CP2 & CCP2	1.783	1.204	2973.581	4007.617	0.98	0.968	8	8
13	CP3 & CCP3	1.22	1.204	2875.877	4007.617	0.973	0.968	6	8
13	CP1 & CCP1	1.093	1.204	2996.492	4007.617	0.969	0.968	7	8
14	CP2 & CCP2	1.18	1.134	2695.256	3946.578	0.984	0.968	8	5
14	CP2 & CCP2	1.18	1.134	2695.256	3946.578	0.984	0.968	8	5
14	CP3 & CCP3	1.253	1.134	3025.148	3946.578	0.976	0.968	6	5
15	CP1 & CCP1	0.979	1.19	2601.090	3899.377	0.973	0.968	7	6
15	CP1 & CCP1	0.979	1.19	2601.090	3899.377	0.973	0.968	7	6
15	CP2 & CCP2	1.312	1.19	2307.166	3899.377	0.979	0.968	9	6
16	CP3 & CCP3	0.996	1.213	2863.730	3962.515	0.974	0.968	6	4
16	CP3 & CCP3	0.996	1.213	2863.730	3962.515	0.974	0.968	6	4
16	CP1 & CCP1	0.951	1.213	2857.301	3962.515	0.974	0.968	6	4
17	CP2 & CCP2	1.436	1.149	3424.056	3998.478	0.987	0.968	8	5
17	CP2 & CCP2	1.436	1.149	3424.056	3998.478	0.987	0.968	8	5
17	CP3 & CCP3	1.34	1.149	3609.378	3998.478	0.98	0.968	6	5
18	CP1 & CCP1	1.006	1.204	2380.563	4066.163	0.967	0.968	7	7
18	CP2 & CCP2	1.097	1.204	2384.762	4066.163	0.967	0.968	8	7
18	CP2 & CCP2	1.097	1.204	2384.762	4066.163	0.967	0.968	8	7
19	CP3 & CCP3	1.364	1.173	2760.271	4014.478	0.973	0.968	7	5
19	CP1 & CCP1	0.915	1.173	2437.098	4014.478	0.968	0.968	7	5
19	CP1 & CCP1	0.915	1.173	2437.098	4014.478	0.968	0.968	7	5
20	CP2 & CCP2	0.896	1.174	3886.594	3997.709	0.982	0.968	8	5
20	CP3 & CCP3	0.918	1.174	3635.701	3997.709	0.979	0.968	6	5
20	CP3 & CCP3	0.918	1.174	3635.701	3997.709	0.979	0.968	6	5



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
🟥 CP1&CCP1	26	13	15	19	9	26	12	19	12	23	16	26	22	13	13	7	18	26	18	24
🟩 CP2&CCP2	39	30	37	36	25	31	29	43	30	44	32	44	37	35	40	14	40	36	35	30
🟪 CP3&CCP3	38	28	23	40	33	37	26	54	30	29	38	22	25	19	21	23	43	38	33	31

Figure 6.6. *Constraints violated numbers of the different algorithms in each run*

In order to show the percentage of the constraints violated number, the percentage values are computed by using the following equation:

$$Percentage = \frac{constraints\ violated\ number}{total\ number\ of\ considerd\ constraints} \times 100 \tag{6.3}$$

Table 6.14 lists the total number of the considered constraints in all runs, the constraints violated number, and the percentage values of constraint violation for all existing algorithms.

Table 6.14

The Results of the Constraints Violated Number Test

Algorithm	Total number of considered constraints	Constraints violated number	Percent of constraints violation
CP1 & CCP1	2000	357	17.850
CP2 & CCP2	2000	687	34.350
CP3 & CCP3	2000	631	31.550

The best algorithm is the algorithm that has less number of constraints violated. As shown in Figure 6.6, CP1 & CCP1 algorithms perform the best among the existing algorithms. In all runs, CP1 & CCP1 algorithms show the lowest constraints violated number compared with other algorithms. As Table 6.14 shows, the lowest percentage of the constraints violated number 17.850 is achieved when executing CP1 & CCP1 algorithms. Out of a total of 2000 considered constraints, only 357 constraints are violated when executing CP1 & CCP1 algorithms.

Compared with CP2 & CCP2, CP3 & CCP3 algorithms performed better by achieving a percentage of the constraints violated number 31.550, compared with 34.350 achieved by CP2 & CCP2. These results are expected because CP3 & CCP3 check the feasibility for each path separately by decomposing the composition into execution paths. Such a technique for optimization leads to reduce the constraint violation. On the other hand, CP2 & CCP2 check the feasibility of all paths together using probability of paths. As mentioned earlier, the paths probability is estimated either by inspecting the system logs or being specified by the composition engineers. If the composition execution follows the path with the less probability, there are high chances of constraint violation.

The simulation results indicate that by using the proposed optimization mechanism, the constraints violated number is significantly reduced while achieving the highest overall QoS ratio. It is expected to have such excellent results because CP1 & CCP1 algorithms check only the feasibility of the path that will potentially be executed regardless of other paths (i.e., focus on one path). On the other hand, the solutions resulted from checking the feasibility of all paths are not necessarily satisfy the imposed constraints for all paths at the same time.

6.2.2.3 Computation Time

This test has a goal to evaluate the proposed optimization mechanism in terms of the computation time. Two different computation times are captured by the simulation:

1. Computation time for the proposed optimization mechanism (CP1 & CCP1). As mentioned earlier, the proposed mechanism is a combination between the runtime path prediction method and the optimization algorithms. The data mining algorithm in the proposed mechanism is trained offline. Thus, the overall computation time for executing the mechanism is not affected by the computation time needed for training the algorithm. Therefore, the time needed for training the algorithm is not taken into account when calculating the overall computation time for this mechanism. However, the time needed for the path prediction is taken into account when calculating the overall computation time because the path prediction phase is carried out at runtime. In summary, the overall computation time captured by the simulation prototype is calculated as in the following:

$$\begin{aligned}&\mathit{Overall\ computation\ time\ for\ CP1\ \&\ CPP1} =\\&\ \mathit{prediction\ time} + \mathit{optimization\ time}\end{aligned} \tag{6.4}$$

2. Computation time for the rest techniques (CP2 & CCP2, and CP3CCP3): the computation time represents only the time needed for computing the optimization. It is calculated as:

$$\textit{Overall computation time for CP2 \& CPP2 and CP3 \& CPP3} = \textit{optimization time} \tag{6.5}$$

In this test, CP1 & CCP1 algorithm calls the path prediction process to determine the path that will potentially be executed at runtime. The time needed for path prediction is captured to calculate the total computation time using Equation 6.4. The computation time for CP2 & CCP2 and CP3 & CCP3 is calculated using Equation 6.5. The test using the setup is listed in Table 6.15.

Table 6.15

A Setup for the Computation Time Test

Setup	Value
Number of abstract services	22
Number of candidate service	50
QoS characteristics of the candidate	As given in Table 6.8
Constraint	Cost characteristic
Algorithms for comparison	CP1 & CCP1, CP2 & CCP2, CP3 & CCP3

Simulation Results

The results of this test experiment are shown in Figure 6.7. Figure 6.7 shows the computation times (in microseconds) of the different algorithms each run. In addition, Table 6.16 lists the average computation times and the standard deviation of the different algorithms.

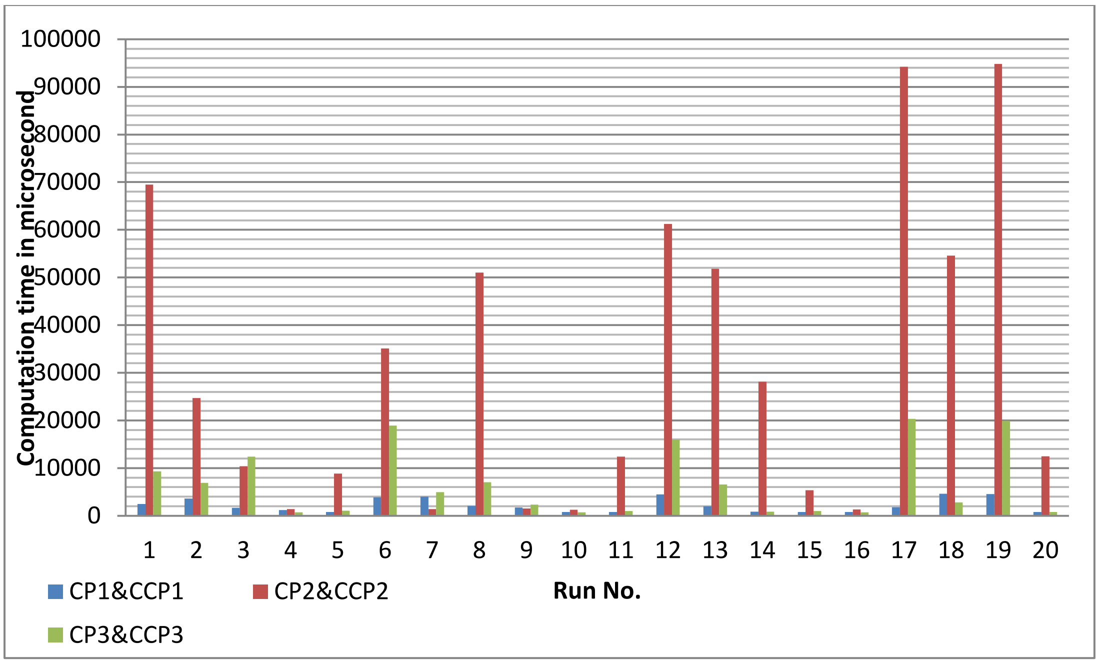


Figure 6.7. *Computation times of the different algorithms each run*

Table 6.16

The Results of Computation Time Test

Algorithm	Average of computation time	Standard Deviation
CP1 & CCP1	2186.200 µs	1449.891
CP2 & CCP2	31078.700 µs	1448.166
CP3 & CCP3	6720.350 µs	1446.417

The best algorithm is the algorithm that has less computation time. As shown in Figure 6.7, the computation times consumed by the different algorithms vary between runs. For example, in run number 1, 2, and 3, the algorithms consumed longer computation times than in 4, 10, and 16. The variance in computation times between the algorithms is because of the initial solution computed by the algorithm. In the case that the initial solution is feasible, the algorithms consume less computation time than the opposite case. It is also the reason that lies behind the large standard deviation showed by the algorithms.

As seen in Figure 6.7, by comparing it with CP2 & CCP2 and CP3 & CCP3, CP1 & CCP1 show computation times that are less than in 13 out of 20 runs. Table 6.16 shows that CP1 & CCP1 produced low average computation time, i.e., 2186.200 µs. Although CP1 & CCP1 perform path prediction process for predicting the path, it produced the lowest average computation time compared with other algorithms. Two reasons lie behind that. First, optimizing one path consumed less computation time than optimizing all paths either separately or altogether. Second, the computation time consumed by the path prediction process is very low, i.e., 96.520 µs on average.

The results of CP3 & CCP3 show that the algorithms have good average computation time, i.e., 6720.350 µs compared with CP2 & CCP2 which have the highest average of computation time 31078.700 µs compared with other algorithms. As seen in Figure 6.7, in the case that the initial solution computed by CP2 is feasible, the CP2 & CCP2 consumes very close computation time compared with other algorithms. In run number 4, 9, and 10 is an example. However, in the case of infeasibility, the algorithms consumed higher than the both algorithms because of the optimization strategy of the algorithms which consider all paths together when computing the optimization. Such a strategy of optimization leads to consume more computation time searching for feasible solutions compared to a strategy like CP1 & CCP1 which considers one path when computing the optimization.

As seen in Table 6.16, all algorithms showed large standard deviation. This is because the algorithms consume less time (i.e., 1178.736 µs on average) if the initial solution is feasible whereas it consume long time (i.e., 6618.852 µs on average) if it is not feasible.

The simulation results indicate that the computation time needed for executing the proposed optimization mechanism is small. This is due to the nature of the CP1 & CCP1 algorithms which check only the feasibility of the path that will potentially be executed regardless of other paths (i.e., focus on one path). These excellent results make the proposed approach suitable for real time decision-making applications, especially in a scenario like ours, where a quick response for a composition instance is very important.

6.2.2.4 Practical Composite Service Scenario

An extra experiment is required to demonstrate the need for organizations, which intend to increase their business processes performances as well as reduce the developments cost and time, to outsource web services. To do so, a travel agency business process is introduced as the practical scenario that represents a group of web services, such as hotel and airline reservation web services, that can be integrated into a travel agency business process. A typical travel agency composite service is illustrated in Figure 6.8.

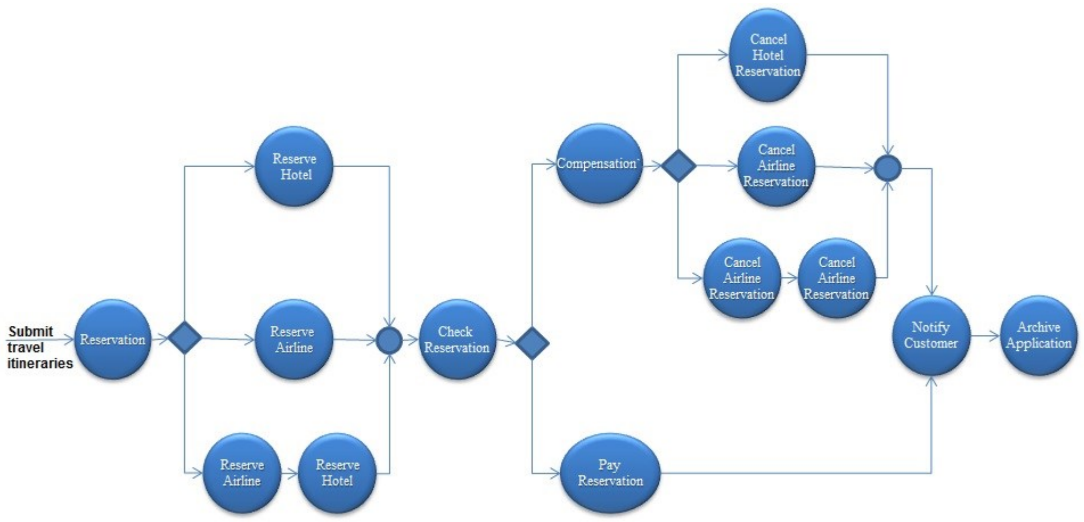


Figure 6.8. *A typical travel agency scenario*

In this scenario, customers have the opportunity to plan and reserve travel arrangements using the travel agency composite service. A customer submits travel eternity and payment information using an online form. According to the customer's itinerary, the travel agency composite service reserves either a hotel or an airline or both. After completing the reservation operation, either payment or compensation operations are performed. In the case of an itinerary failure, the composite service performs compensation operation for canceling itinerary. The service automatically notifies the customer of either confirmation or failure of reservation. Finally, the service stores the travel eternity data in a database.

As seen in Figure 6.8, the service is composed of 14 web services and represents a multiple paths composite service scenario. It includes 6 different execution paths. This test experiment has a goal to evaluate the proposed mechanism by comparing it with the two previously mentioned techniques, the comparisons were conducted using the CP1 & CCP1, CP2 & CCP2, CP3 & CCP3 algorithms which were implemented based on these techniques. These techniques are evaluated in terms of the resulting QoS ratio as well as the calculated constraints violated number. The test that used the setup is listed in Table 6.17.

Table 6.17

Setup for Test Experiment

Setup	Value
Number of abstract services	14
Number of candidate service	50
QoS characteristics of the candidate	As given in Table 6.8
Constraint	Cost characteristic
Algorithms for comparison	CP1 & CCP1, CP2 & CCP2, CP3 & CCP3

The results of this test experiment are shown in Table 6.18. Table 6.18 shows the resulting QoS ratio, i.e., total utility, the total cost value aggregated from the generated solutions, and the value of the specified cost constraint.

Table 6.18

Results for Test Experiment

Technique	Path	Path(Abstract Service name - Selected Candidate Name - Selected Candidate)	Total Utility	Total Cost	Cost Constraint
CP1&CCP1	Path1	C1_42 - 0.2179494921040104, C2_39 - 0.1688407841273432, C6_30 - 0.19698753455987147, C7_41 - 0.1463123664451505, C9_15 - 0.2128389756901315, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.379	0.724	0.733
CP2&CCP2	Path1	C1_42 - 0.2179494921040104, C2_39 - 0.1688407841273432, C6_29 - 0.1037053358968876, C7_41 - 0.1463123664451505, C9_15 - 0.2128389756901315, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.285	0.897	0.733
CP3&CCP3	Path1	C1_42 - 0.2179494921040104, C2_27 - 0.2081498089173867, C6_31 - 0.14908644475757019, C7_41 - 0.1463123664451505, C9_15 - 0.2128389756901315, C13_18 - 0.2344213934257782, C14_34 - 0.13173786918068925	1.300	0.801	0.733
CP1&CCP1	Path2	C1_42 - 0.2179494921040104, C2_39 - 0.1688407841273432, C6_31 - 0.14908644475757019, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834,	1.189	0.717	0.733
CP2&CCP2	Path2	C1_42 - 0.2179494921040104, C2_39 - 0.1688407841273432, C6_29 - 0.1037053358968876, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834,	1.143	0.975	0.733
CP3&CCP3	Path2	C1_42 - 0.2179494921040104, C2_27 - 0.2081498089173867, C6_31 - 0.14908644475757019, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_1 - 0.15919030478608132,	1.186	0.930	0.733
CP1&CCP1	Path3	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998, C6_31 - 0.14908644475757019, C7_34 - 0.17909229909159627, C10_21 - 0.19970691438136834, C13_18 - 0.2344213934257782, C14_23 - 0.18489977292121432	1.396	0.687	0.733
CP2&CCP2	Path3	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998,	1.351	1.328	0.733

		C6_13 - 0.12138347308782538, C7_41 - 0.1463123664451505, C10_21 - 0.19970691438136834, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834			
CP3&CCP3	Path3	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998, C6_31 - 0.14908644475757019, C7_41 - 0.1463123664451505, C10_21 - 0.19970691438136834, C13_18 - 0.2344213934257782, C14_1 - 0.15919030478608132	1.337	0.689	0.733
CP1&CCP1	Path4	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998, C6_30 - 0.19698753455987147, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_23 - 0.18489977292121432	1.065	0.466	0.733
CP2&CCP2	Path4	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998, C6_31 - 0.14908644475757019, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.033	0.751	0.733
CP3&CCP3	Path4	C1_42 - 0.2179494921040104, C3_1 - 0.23048148109032998, C6_31 - 0.14908644475757019, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_23 - 0.18489977292121432	1.017	0.380	0.733
CP1&CCP1	Path5	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_31 - 0.14908644475757019, C7_41 - 0.1463123664451505, C11_41 - 0.13150274936823553, C12_26 - 0.16816058978351534, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.577	0.728	0.733
CP2&CCP2	Path5	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_26 - 0.16837895241208095, C7_41 - 0.1463123664451505, C11_44 - 0.21963708909923335, C12_26 - 0.16816058978351534, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.684	1.233	0.733
CP3&CCP3	Path5	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_31 - 0.14908644475757019, C7_41 - 0.1463123664451505, C11_44 - 0.21963708909923335, C12_44 - 0.16631762759196728, C13_18 - 0.2344213934257782, C14_20 - 0.16743334904654786	1.630	0.831	0.733
CP1&CCP1	Path6	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_50 - 0.2174268637195497, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.120	0.700	0.733
CP2&CCP2	Path6	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_29 - 0.1037053358968876, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_37 - 0.20120580065087834	1.120	0.700	0.733
CP3&CCP3	Path6	C1_42 - 0.2179494921040104, C4_7 - 0.12497517831801178, C5_12 - 0.20337401261661384, C6_31 - 0.14908644475757019, C8_32 - 0.2172718205473536, C13_18 - 0.2344213934257782, C14_18 - 0.08712708692762687	1.085	0.557	0.733

Table 6.19 shows the average QoS ratios, the total number of considered constraints, and the total constraints violated numbers for the three algorithms.

Table 6.19

Average QoS ratio, Total Number of Considered Constraint, and Constraints Violated Number

Algorithm	Average QoS ratio	Total number of considered constraints	Constraints violated number
CP1 & CCP1	0.390	6	0
CP2 & CCP2	0.384	6	5
CP3 & CCP3	0.378	6	3

As seen from Table 6.19, the proposed optimization mechanism, i.e., the CP1 & CCP1 algorithms, shows resulting QoS performances that outperform the existing algorithms by achieving the highest average QoS ratio 0.390 with 0 constraints violated number. These results are matched with the results obtained from the QoS ratio and constraints violated number test experiments.

The proposed approach has been implemented and evaluated using three different business processes, namely, auto insurance, bank loan, and travel agency business processes, and the results were promising. This makes the proposed approach suitable for any multiple paths business processes.

6.3 **Conclusions**

The approach evaluation has been divided into two parts. The first part evaluated the runtime path prediction method while the second one evaluated the optimization mechanism.

For evaluation of runtime path prediction method, two test experiments have been conducted by utilizing the WEKA data mining tool. The first test experiment aimed at validating the accuracy of the path prediction using three different learning algorithms, including J48, NB, and SMO. These algorithms are applied to the auto insurance dataset. The results indicate that all the selected classifiers achieved promising accuracy prediction when predicting the execution paths. Having got such encouraging results of prediction accuracy contributes to the generation of high QoS ratio solutions and minimizes the constraints violated a number of the generated solutions. The second experiment aimed at studying how the prediction method scales with a rising number of involved execution paths. For this purpose, J48, SMO, and NB algorithms are applied to 9 different datasets representing bank loan process. Each dataset contains 1000 instances representing a loan process that involves execution paths ranging from 2 up to10 paths. The results showed that the rising number of classes involved in the prediction process does not affect the prediction accuracy of the classifier. The structure of the business process plays an important role in the prediction accuracy results. These results make the proposed approach suitable for any compositions regardless of the number of involved execution paths.

For the evaluation of the optimization mechanism, three test experiments have been conducted using new simulation prototype developed for this purpose. The evaluation was conducted by comparing the performance of the CP1 & CCP1 algorithms, which represented the proposed optimization mechanism, with the performance of the CP2 & CCP2 and the CP3 & CCP3 algorithms, which represented the relevant optimization techniques.

The first test experiment has evaluated the optimization techniques in terms of the resulting QoS ratio. The results showed that the CP1 & CCP1 algorithms achieved the highest total QoS ratio and average of total QoS ratio 205.318, 10.266

respectively. The CP1 & CCP1 algorithms also show a small standard deviation. Regarding CP2 & CCP2 and CP3 & CCP3 algorithms, both algorithms show close resulting QoS performances by achieving total QoS ratios 200.196, 198.897 and the average total QoS ratio 10.010, 9.945 respectively.

The second test has evaluated the optimization techniques in terms of the calculated constraints violated number. The results showed that the lowest percentage of the constraints violated number is 17.850 achieved when executing CP1 & CCP1 algorithms. Out of a total of 2000 considered constraints, only 357 constraints are violated when executing CP1 & CCP1 algorithms. Compared with CP2 & CCP2, CP3 & CCP3 algorithms performed better by achieving a percentage of the constraints violated number 31.550, compared to 34.350 achieved by CP2 & CCP2.

The third test has evaluated the optimization techniques in terms of the computation time. The results showed that CP1 & CCP1 produced the lowest average computation time 2186.200 µs. The results of CP3 & CCP3 showed that the algorithms have reasonable average computation time, i.e., 6720.350 µs compared to CP2 & CCP2 which have the highest average of computation time 31078.700 µs. This is because the algorithms consume more computation time searching for feasible solutions since it considers all paths together when checking the feasibility. All algorithms showed large standard deviation. This is because the algorithms consume less time if the initial solution is feasible whereas it consume long time if it is not feasible.

6.4 Chapter Summary

The evaluation results of the path prediction method indicated that the prediction method achieved promising prediction accuracy which is not affected by the number of paths involved in the prediction process. However, the structure of the business process plays an important role in the prediction accuracy results. These results make the proposed approach suitable for any compositions regardless of the number of involved execution paths. The evaluation results of the proposed optimization mechanism showed that the proposed optimization mechanism outperforms the relevant optimization techniques in terms of the resulting QoS ratio, the calculated constraint violation number, and the computation time. These promising results make the optimization mechanism able to generate high overall QoS ratio solutions, and significantly reduce the constraints violated number, while consuming small computation time.

CHAPTER SEVEN
CONCLUSION AND FUTURE RESEARCH WORK

In this chapter, the achievements of the research work are summarized, research contributions are highlighted, research limitations are introduced, and directions for future works related to this research are given.

7.1 **Conclusion of the Research**

QoS-aware service composition process aims to select one outsourced candidate web service for each abstract web service from its corresponding list of candidates such that the entire QoS of the composition is optimized while QoS requirements, defined by clients, are satisfied. Finding exact optimal solutions required a strategy based on evaluating all the possible combinations to find the optimal one. It is impractical and time consuming to evaluate all these combinations to find the optimal one. Thus, solutions based on heuristic algorithms, although they deliver near-to-optimal solutions, represent a novel approach (Jaeger, 2007). Furthermore, in multiple execution paths composition, generating solutions that simultaneously optimize all the execution paths while meeting global QoS constraints imposed by the clients is very difficult.

A QoS aware service composition approach has been designed and developed to solve the optimization problem and the multiple paths composition problem mentioned above. The idea for solving the optimization problem was to apply heuristic algorithms. For multiple paths composition problem, the strategy of the proposed work was, rather than considered all execution paths in optimization; the path that will potentially be executed at runtime is the only path that is optimized. Therefore, in the proposed approach, the new optimization mechanism has been proposed based on the combination between CP and CPP heuristic algorithms and runtime path prediction method. The runtime path prediction method has been proposed for the purpose of predicting at runtime the execution path that will potentially be executed based on the information provided by a composition requester. CP and CPP algorithms have been applied to solve the optimization problem by considering only the path that has been predicted by the prediction method. CP has been applied to generate a feasible solution while CCP to improve the quality of the solution generated from CP.

The approach influenced the direction of the research on the QoS in web service composition. The question needed to be answered was what QoS characteristics are appropriate for service selection. A review of the QoS characteristics that were used in the area of web service composition has been given. It has been concluded that there is no standard, formal or a complete QoS model for a web service and most of the research effort considers a set of general QoS characteristics that are applicable to all domains. Therefore, it was needed to analyze the QoS characteristics that are most commonly used to evaluate web services in order to determine the relevant set of QoS characteristics. These can be considered as selection criteria when composing web services. As a result, eight QoS characteristics have been suggested and identified after investigating and analyzing the related works in the area of web service and SOA. The suggested criteria were cost, response time, reliability, availability, security, throughput, reputation, and composability. The mandatory obtained criteria were cost and response time. In assisting clients when assigning weights, prioritizing the selected QoS criteria has been suggested. The prioritization was based on the relative importance of these criteria in building optimal web service compositions. The suggested priority were: (1) cost, (2) response time, (3) reliability, (4) availability, (5) security, (6) throughput, (6) reputation, and (8) composability.

Finally, the proposed approach has been evaluated. The evaluation was divided into two parts. The first part has been evaluated the runtime path prediction method. The evaluation has been performed by utilizing WEKA data mining tools and using three different learning algorithms, including J48, NB, and SMO. The goal of the evaluation was to validate the accuracy of the path prediction and to study how the prediction method scales with a rising number of involved execution paths. In summary, the evaluation has been revealed the following points:

- All the selected classifiers achieved promising accuracy prediction when predicting the execution paths. The promising results yield the generation of high QoS ratio solutions, and minimizing the constraint violation of the generated solutions.
- The rising number of classes involved in the prediction process did not affect the prediction accuracy of the classifier. The structure of the business process plays an important role in the prediction accuracy results. The results make the proposed approach suitable for any compositions regardless of the number of involved execution paths.

The second part has been evaluating the performance of the optimization mechanism. The new simulation prototype has been developed for this aim. The evaluation was conducted by comparing the performance of the CP1 & CCP1 algorithms, which represented the proposed optimization mechanism, with the performance of the CP2 & CCP2 and the CP3 & CCP3 algorithms, which represented the relevant optimization techniques.

The evaluation goal was to evaluate three aspects of the performance of the optimization techniques: the resulting QoS ratio, the constraints violated number, and the computation time. In summary, the evaluation has been revealed the following points:

- The CP1 & CCP1 algorithms achieved the highest total QoS ratio and average of the total QoS ratio 205.318, 10.266 respectively with a small standard deviation. They produced the lowest percentage of constraint violated number 17.850. Moreover, the algorithms consumed small computational time by achieving the lowest total average computation time 2186.200 µs.
- The CP2 & CCP2 and CP3 & CCP3 algorithms showed close resulting QoS performances by achieving total QoS ratios 200.196, 198.897 and the average total QoS ratio 10.010, 9.945 respectively. Compared with CP2 & CCP2, CP3 & CCP3 algorithms performed better by achieving a percentage of the constraints violated number 31.550, compared to 34.350 achieved by CP2 & CCP2. The results of CP3 & CCP3 show that the algorithms have reasonable

average computation time, i.e., 6720.350 µs compared to CP2 & CCP2 which have the highest average of the computation time 31078.700 µs.

7.2 Contributions of the Research

The major contribution of this research work is proposing a smart approach for QoS-aware service composition. The approach is designed to efficiently solve the multiple paths composition problem. Rather than computing the optimization for all execution paths, the proposed approach computes the optimization for any composition instance based only on its corresponding execution path.

The proposed approach enhances the performance of the optimization process by generating high overall QoS ratio solutions, and significantly reducing the constraints violated number, while consuming small computation time. These excellent results make the approach efficient for the real-time application scenarios.

The proposed approach can be used by clients (organizations) to build their business processes. There are many advantages that organizations can gain when relying on the proposed approach:

- Allow organizations to increase the QoS performance of their business processes.
- Allow organizations to efficiently select the outsourced web services that guarantee satisfying their QoS requirements as much as possible.
- Allow organizations to quickly respond to their business process requester.

The research work makes several important contributions which are:

1. **A New Optimization Mechanism**

This research work has proposed a new optimization mechanism which computes the optimization considering the execution path that will potentially be executed by a composition instance. The mechanism is a combination between runtime path prediction method and heuristic algorithms. The runtime path prediction method predicts, at runtime, and just before the actual composition execution, the execution path that will potentially be executed. Then the heuristic algorithms compute the optimization considering only the execution path that is predicted by the runtime path prediction method. Thus, the proposed optimization mechanism generates, within small computation time, a set of web services that delivers the best possible overall QoS ratio and meets the clients' requirements.

2. **A Runtime Path Prediction Method**

A runtime path prediction method is proposed for the purpose of predicting at runtime the execution path that will potentially be executed based on the information provided by a composition requester. This method is composed of four phases. The first phase is adopted from Cardoso (2005, 2008) and Cardoso and Lenic (2006) and aimed at extending the composition logs to store information that indicates the input (output) value parameters passed (received) to/ or from web services and their types. A class path is an extra field needs to be added to the log to store path information. The second phase aimed at using the instances data contained in the logs as a training dataset for machine learning algorithms. The third phase aimed at building classifiers using the training dataset. The fourth phase aimed at performing runtime path prediction based on the information provided by a composition requester.

To the best of our knowledge, this work is one of the first that has employed machine learning algorithms (include NB, SMO, and J48) in the area of QoS-aware web service composition in order to learn and then predict at runtime the path that will potentially be executed, and has used the predicted path to be optimized.

3. **Heuristic Optimization Algorithms**

The QoS-aware service composition problem was mapped to MMKP, due to the similarity between both problems, which allows to select heuristic algorithms, namely CP and CCP algorithms to be applied to solve the QoS-aware service composition problem. The first algorithm is a constructive approach called constructive procedure (CP) that is applied to generate a feasible solution while the second one is a complementary approach called a complementary procedure (CCP) that is used to improve the quality of the solution generated from CP. A machine learning algorithm is combined with a CP algorithm in order to optimize only the predicted path. CP and CPP algorithms have been selected because of their ability to generate quality solutions within small computation efforts. This makes this approach efficient in real-time scenarios. Moreover, the algorithms can be easily applied to solve the selection problem.

4. **QoS Characteristics for a Web Service Composition**

This research work has suggested eight QoS characteristics that can be considered as selection criteria when composing web services. These criteria were suggested with respect to web service composition features and were identified after investigating and analyzing the related works in the area of web service and SOA. These characteristics can be used to evaluate the QoS of composite services. Furthermore, clients can use the characteristics to specify their QoS requirements. To assist clients when assigning weights, prioritizing the selected QoS criteria was suggested.

7.3 **Research Limitation**

The following are the constraints and limitations in this approach to making further study necessary in order to improve its performance.

- The QoS characteristics considered in this approach are a set of fixed characteristics that can be applied to all domains. QoS characteristics should differ depending on the domain. For example, it is important for the E-Learning domain to consider QoS characteristics like accuracy and reputation while E-Publishing service should consider the security (Sathya, Swarnamugi, Dhavachelvan, & Sureshkumar, 2010).

- The proposed approach is not capable to deal with structures such as loop or parallel structures. It is designed to suit the multiple paths in business processes, i.e., business processes that defined using sequential and conditional structure. Knowing that loop structure, for example, may be reduced to sequential as in Zeng et al. (2004).

- The evaluation results of the proposed optimization mechanism have shown that the mechanism is capable to generate the best possible overall QoS ratio solutions while consuming small computation time. However, the mechanism is not capable to generate exact optimal solutions. This is because the optimization is based on heuristic algorithms.

7.4 **Future Works**

The work reported in this research has opened up several areas for further research work. Based on this work, possible areas are:

Designing QoS model: It is unlikely that a fixed set of QoS characteristics can be considered for all domains. Instead, the dynamic QoS model is required to be designed in a way that QoS characteristics should differ depending on the services' domains.

Other optimization strategy: The proposed approach is based on data mining techniques and heuristic algorithms. Beside this approach, another research opportunity would be investigating a new approach that can be used for stochastic analysis techniques and algorithms that guarantee to find optimal solutions.

Developing other heuristic algorithms: Along with the presented heuristic algorithms, new heuristic and powerful algorithms are still needed to be investigated. The goal is to investigate algorithms that are able to reduce the computation efforts, and at the same time generate better quality solutions.

Developing simulation software: Simulation software is required to evaluate the performance of the optimization algorithms. The lack of general simulation software that can be used in the area of QoS in service composition makes the researcher implement their own simulation.

REFERENCES

Acuna, E., & Rodriguez, C. (2004). The treatment of missing values and its effect on classifier accuracy. In D. Banks, F.R. McMorris, P. Arabie, & W. Gaul (Eds.), *Classification, Clustering, and Data Mining Applications* (pp. 639-647). Springer Berlin Heidelberg.

Agrawal, R., Gunopulos, D., & Leymann, F. (1998). *Mining process models from workflow logs*. Springer Berlin Heidelberg.

Alrifai, M., & Risse, T. (2009). Combining global optimization with local selection for efficient QoS-aware service composition. *Proceedings of the 18th international conference on World wide web*, 881-890.

Alrifai, M., Risse, T., Dolog, P., & Nejdl, W. (2009). A scalable approach for qos-based web service selection. In G. Feuerlicht, & W. Lamersdorf (Eds.), *Service-Oriented Computing – ICSOC 2008 Workshops* (pp. 190-199). Springer Berlin Heidelberg.

Alrifai, M., Skoutas, D., & Risse, T. (2010). Selecting skyline services for qos-based web service composition. *Proceedings of the 19th international conference on World wide web*, 11-20.

Arab Bank. (2012). *Sustainability Report 2012*. Arab Bank. Retrieved 25, May, 2013 from www.arabbank.com/uploads/File/SustainabilityReport2012_English.pdf?CSRT=8982756408031428478

Ardagna, D., & Mirandola, R. (2010). Per-flow optimal service selection for Web services based processes. *Journal of Systems and Software*, *83*(8), 1512-1523.

Ardagna, D., & Pernici, B. (2006). Global and local qos guarantee in web service selection. In C.J. Bussler, & A. Haller (Eds.), *Business Process Management Workshops* (pp. 32-46). Springer Berlin Heidelberg.

Ardagna, D., & Pernici, B. (2006). Adaptive service composition in flexible processes. *Software Engineering, IEEE Transactions on, 33*(6), 369-384.

Austin, D., Daniel, A., Ferris, C., & Garg, S. (2004). *Web services architecture requirements*. W3C Working Group Note. Retrieved 5, April, 2012, from http://www.w3.org/TR/wsa-reqs/

Bahadori, S., Kafi, S., Far, K. Z., & Khayyambashi, M. R. (2009). Optimal web service composition using hybrid GA-TABU search. *Journal of Theoretical and Applied Information Technology*, *9*(1), 10-15.

Balas, E., & Zemel, E. (1980). An algorithm for large zero-one knapsack problems. *Operations Research*, *28*(5), 1130-1154.

Baryannis, G., Carro, M., Danylevych, O., Dustdar, S., Karastoyanova, D., Kritikos, K., ... & Wetzstein, B. (2008). *Overview of the state of the art in composition and coordination of services*. S-CUBE Software Services and Systems Network Consortium. Retrieved 11, July, 2012 from http://www.s-cube-network.eu/results/deliverables/wp-jra-2.2/PO-JRA-2.2.1-Overview-of-the-state-of-the-art-in-composition-and-coordination-of%20services.pdf.

Bayt. (2013). *Average Monthly Salary in Jordan*. Bayt.com. Retrieved 3, April, 2013 from http://www.bayt.com/en/jordan/jobs/

Behkamal, B., Kahani, M., & Akbari, M. K. (2009). Customizing ISO 9126 quality model for evaluation of B2B applications. *Information and Software Technology, 51*(3), 599-609.

Benatallah, B., Dumas, M., Sheng, Q. Z., & Ngu, A. H. H. (2002). Declarative composition and peer-to-peer provisioning of dynamic web services. *Proceedings of the 18th International Conference on Data Engineering*, 296-308.

Berbner, R., Heckmann, O., & Steinmetz, R. (2005). An Architecture for a qos driven composition of web service based workflows. *Networking and Electronic Commerce Research Conference (NAEC 2005).*

Berbner, R., Spahn, M., Repp, N., Heckmann, O., & Steinmetz, R. (2006). Heuristics for qos-aware web service composition. *Proceedings of the IEEE International Conference on Web Services*, 72-82.

Bilchev, G., & Parmee, I. (1995).The ant colony metaphor for searching continuous design spaces. In T.C. Fogarty (Ed.), *Selected Papers from AISB Workshop on Evolutionary Computing* (pp. 25-39). Springer Berlin Heidelberg.

Blomberg, L. C., & Ruiz, D. D. A. (2013). Evaluating the influence of missing data on classification algorithms in data mining applications. *SBSI 2013: Simpósio Brasileiro de Sistemas de Informação*.

Bolcer, G. A., & Kaiser, G. (1999). SWAP: Leveraging the web to manage workflow. *Internet Computing, IEEE*, *3*(1), 85-88.

Booth, D., & Liu, C. K. (2006). *Web services description language (WSDL) version 2.0 Part 0: Primer*. W3C Working Draft. Retrieved 16, June, 2010, from http://www.w3.org/TR/wsdl20-primer/

Botella, P., Burgués, X., Carvallo, J. P., Franch, X., & Quer, C. (2002). Using quality models for assessing COTS selection. *Proceedings of WER 2002*, 263-277.

Burstein, M., Bussler, C., Zaremba, M., Finn, T., Huhns, M. N., Paolucci, M., ... & Williams, S. (2005). A semantic web services architecture. *IEEE Internet Computing*, *9*(5), 72.

Canfora, G., Penta, M. D., Esposito, R., & Villani, M. L. (2005). An approach for qos-aware service composition based on genetic algorithms. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, 1069-1075.

Cardellini, V., Casalicchio, E., Grassi, V., & Mirandola, R. (2006). A Framework for optimal service selection in broker-based architectures with multiple qos classes. *Proceedings of the IEEE Services Computing Workshops*, 105-112.

Cardellini, V., Di Valerio, V., Grassi, V., Iannucci, S., & Presti, F. L. (2011). A performance comparison of QoS-driven service selection approaches. In *Towards a Service-Based Internet* (pp. 167-178). Springer Berlin Heidelberg.

Cardoso, J. (2005). Path mining in web processes using profiles. *Encyclopedia of data warehousing and mining*, 896-901.

Cardoso, J. (2008). Applying data mining algorithms to calculate the quality of service of workflow processes. In P. Chountas, L. Petrounias, & J. Kacprzyk (Eds.), *Intelligent Techniques and Tools for Novel System Architectures*. Springer Berlin Heidelberg.

Cardoso, J., & Lenic, M. (2006).Web process and workflow path mining using the multi-method approach. *International Journal of Business Intelligence and Data Mining, 1*(3), 304-328.

Cardoso, J., Miller, J., Sheth, A., & Arnold, J. (2004).Modeling quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web Journal, 1*(3), 281–308.

Cardoso, J., & Sheth, A. (2003). Semantic e-workflow composition. *Journal of Intelligent Information Systems (JIIS), 21*(3), 191-225.

Casati, F., Ilnicki, S., Jin, L. J., Krishnamoorthy, V., & Shan, M. C. (2000). E-flow: A platform for developing and managing composite e-services. *Proceedings Academia/Industry Working Conference on Research Challenges*, 341-348.

Choen, D. (2012). *Bank loans and how to qualify for one*. Loan Article. Retrieved 14, July, 2013 , from http://www.bills.com/bank-loans/

Choi, S., Her, J., & Kim, S. (2006). Modeling qos attributes and metrics for evaluating services in SOA considering consumers' perspective as the first class requirement. *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference*, 398-405.

Clement, U., Hately, A., Riegen, C. v., & Rogers, T. (2004). *UDDI version 3.0.2*. DDI Spec Technical Committee Draft. Retrieved 16, June, 2011, from http://www.uddi.org/pubs/uddi_v3.htm

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 263-296.

Degwekar, S., Su, S. Y. W., & Lam, H. (2004). Constraint specification and processing in web services publication and discovery. *Proceedings of the IEEE International Conference on Web Services*, 210-217.

Dimitrios, G., Hans, S., Andrzej, C., & Donald, B. (1999). Managing process and service fusion in virtual enterprises. *Information Systems - Special issue on information systems support for electronic commerce, 24*(6), 429-456.

Dong, S., & Dong, W. (2009). A qos driven web service composition method based on ESGA (Elitist Selection Genetic Algorithm) with an improved initial population selection strategy. *International Journal of Distributed Sensor Networks, 5*(1), 54-54.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26*(1), 29-41.

Dromey, R. G. (1995). A model for software product quality. *IEEE Transactions on Software Engineering, 21*(2), 146-162.

Du, Y., Wang, X., Ai, L., & Li, X. (2012). Dynamic Selection of Services under Temporal Constraints in Cloud Computing. In *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*, 252-259.

Dustdar, S., & Papazoglou, M. P. (2008). Services and service composition - an introduction. *it - Information Technology, 50*, 086 - 092.

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. *International Journal on Web and Grid Services, 1*(1), 1-30.

Fitzpatrick, R. (1996). *Software quality: Definitions and strategic issues*. Reports. Retrieved 19, May, 2010, from http://www.comp.dit.ie/rfitzpatrick/papers/quality01.pdf

Freund, Y., & Schapire, R. E. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence, 14*(5), 661-680.

Ganesarajah, D., & Lupu, E. (2002). Workflow-based composition of web-services: A business model or a programming paradigm?. *Proceedings of the Sixth International Enterprise Distributed Object Computing Conference (EDOC'02)*, 273-284.

Gao, Z.-p., Chen, J., Qiu, X.-s., & Meng, L.-m. (2009). QoE/QoS driven simulated annealing-based genetic algorithm for web services selection. *The Journal of China Universities of Posts and Telecommunications, 16*(1), 102-106.

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., & Shan, M. C. (2004). Business process intelligence. *Computers in Industry*, *53*(3), 321-343.

Guoping, Z., Huijuan, Z., & Zhibin, W. (2009). A qos-based web services selection method for dynamic web service composition. *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, 832-835.

Gutiérrez-Peña, E. (2004). Bayesian classification methods. *Psychology Science*, *46(1)*, 52-64.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18.

Hifi, M., Michrafy, M., & Sbihi, A. (2004). Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, 55(12), 1323–1332.

Hilari, M. O. (2009). *Quality of service (QoS) in SOA systems. A Systematic Review.* (Master's thesis, UniversitatPolitècnica de Catalunya, 2009). Retrieved 7, May, 2013 from http://upcommons.upc.edu/pfc/handle/2099.1/7714

Huang, A. F. M., Lan, C.-W., & Yang, S. J. H. (2009). An optimal qos-based Web service selection scheme. *Information Sciences: An International Journal, 169*(19), 3309-3322.

Hull, R., Benedikt, M., Christophides, V., & Su, J. (2003). E-services: A look behind the curtain. *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 1-14.

ISO-9000:2005. (2005). *Quality management systems fundamentals and vocabulary*. International Organization for Standardization. Retrieved 24, August, 2010, from http://www.iso.org/iso/watermarksample.pdf

ISO/IEC 25012:2008. (2008), *Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model*. Retrieved 24, June, 2014, from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35736

ISO/IEC. (1998*). Information technology – quality of service: Framework (ISO/IEC 13236)*. International Organization for Standardization. Retrieved 15, April, 2010, from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26993

Ivanovic, D., Carro, M., & Hermenegildo, M. (2010). Towards data-aware qos-driven adaptation for service orchestrations. In *Web Services (ICWS), 2010 IEEE International Conference on*, 107-114).

Jacobsson, M., Lidén, P., Stjernschantz, E., Boström, H., & Norinder, U. (2003). Improving structure-based virtual screening by multivariate analysis of scoring data. *Journal of medicinal chemistry*, *46*(26), 5681-5689.

Jaeger, M., Muhl, G., & Golze, S. (2005). QoS-aware composition of web services: An evaluation of selection algorithms. In R. Meersman, & Z. Tari (Eds.), *On the Move to Meaningful Internet Systems 2005* (Vol. 3660, pp. 646-661). Springer Berlin Heidelberg.

Jaeger, M. C. (2007). *Optimising quality-of-service for the composition of electronic services* (Doctoral dissertation, Berlin University of Technology). Retrieved from http://opus4.kobv.de/opus4-tuberlin/frontdoor/index/index/docId/1413

Jaeger, M. C., Rojec-Goldmann, G., & Muhl, G. (2004). QoS aggregation for web service composition using workflow patterns. *Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International*, 149-159.

Jafarpour, N., & Khayyambashi, M. R. (2010). QoS-aware selection of web service composition based on Harmony Search algorithm. *Proceedings of the 12th international conference on Advanced communication technology*, 1345-1350.

Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., ... & Yiu, A. (2007). Web services business process execution language version 2.0. *OASIS standard*, *11*, 11.

Jiang, H., Yang, X., Yin, K., Zhang, S., & Cristoforo, J. A. (2011). Multi-path qos-aware web service composition using variable length chromosome genetic algorithm. *Information Technology Journal*, *10*(1), 113-119.

Jin, C., Wu, M., Jiang, T., & Ying, J. (2008). Combine automatic and manual process on web service selection and composition to support qos. *12th International Conference on Computer Supported Cooperative Work in Design, 2008 (CSCWD 2008),* 459-464.

Kim, E., Kang, G., Lee, Y., & McRae, M. (2005). *OASIS web services quality model TC*. Advanced Open Standards for the Information Society (OASIS). Retrieved July 13, 2010, from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm

Kim, E., Lee, Y., Kim, Y., Park, H., Yun, J., & Kang, G. (2011). *OASIS web services Quality factors version 1.0*. Advanced Open Standards for the Information Society (OASIS). Retrieved July 13, 2012, from http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cs01/WS-Quality-Factors-v1.0-cs01.html

Khan, S., Li, K. F., Manning, E. G., & Akbar, M. M. (2002). Solving the knapsack problem for adaptive multimedia systems. *Studia Informatica Universalis.*, *2*(1), 156-168.

Ko, J. M., Kim, C. O., & Kwon, I.-H. (2008). Quality-of-service oriented web service composition algorithm and planning architecture. *Journal of Systems and Software, 81*(11), 2069-2090.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI*, 1136-1145.

Kotsiantis, S. B. (2006). Supervised machine learning: A review of classification techniques. *Informatica, 31*(3), 249–268.

Lécué, F. (2009). Optimizing qos-aware semantic web service composition. In A. Bernstein, D. R. Karger, T. Heath, & K. Thirunarayan (Eds.), *Proceedings of the 8th International Semantic Web Conference* (pp. 375-391). Springer Berlin Heidelberg

Lee, J. (2003). Matching algorithms for composing business process solutions with web services. In K. Bauknecht, A. Tjoa, & G. Quirchmayr (Eds.), *E-Commerce and Web Technologies* (Vol. 2638, pp. 393-402). Springer Berlin Heidelberg.

Lee, K., Jeon, J., Lee, W., Jeong, S.-H., & Park, S.-W. (2003). *Qos for web services: requirements and possible approaches*. W3C Working Group Note. Retrieved 9, August, 2010, from http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

Leitner, P., Hummer, W., & Dustdar, S. (2013). Cost-based optimization of service compositions. *Services Computing, IEEE Transactions on*, *6*(2), 239-251.

Li, S., & Chen, M. (2010). An adaptive-GA based QoS driven service selection for Web services composition. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. IEEE.

Li, W., & Yan-xiang, H. (2010). Web service composition based on qos with chaos particle swarm optimization. *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010*, 1-4.

Liao, J., Liu, Y., Zhu, X., Wang, J., & Qi, Q. (2013). Accurate QoS-based service selection algorithm for service composition. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, 344-347.

Liu, D., Shao, Z., & Yu, C. (2009a). Optimizing service selection by user's qos expectation. *Proceedings of the International Multi Conference of Engineers and Computer Scientists*.

Liu, D., Shao, Z., Yu, C., & Fan, G. (2009b). A heuristic qos-aware service selection approach to web service composition. *Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science*.

Liu, P., Lei, L., & Wu, N. (2005). A quantitative study of the effect of missing data in classifiers. *The Fifth International Conference on Computer and Information Technology, 2005 (CIT 2005)*, 28-33.

Liu, Y., Ngu, A. H., & Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 66-73.

Liu, Y.,Wu, W., & Liu, S. (2012). A novel qos-aware service composition approach based on path decomposition. *2012 IEEE Asia-Pacific Services Computing Conference*, 76-82.

Lou, Y.-s., Tao, Z.-h., Wang, Z.-j., & Yue, L.-l. (2009). Research on a global optimization method for qos of web service composite. *Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering*, 352-355.

Luo, Y. S., Qi, Y., Hou, D., Shen, L. F., Chen, Y., & Zhong, X. (2011). A novel heuristic algorithm for qos-aware end-to-end service composition. *Computer Communications*, *34*(9), 1137-1144.

Mani, A., & Nagarajan, A. (2002). *Understanding quality of service for web services*. IBM. Retrieved 9 August, 20011, from https://www.ibm.com/developerworks/java/library/ws-quality/

Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, *45*(3), 414-424.

Martello, S., & Toth, P. (1988).A new algorithm for the 0-1 knapsack problem. *Management Science*, *34*(5), 633-644.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., ... & Sycara, K. (2005). Bringing semantics to web services: The OWL-S approach. In J. Cardoso, & A. Sheth (Eds.), *Semantic Web Services and Web Process Composition* (pp. 26-42). Springer Berlin Heidelberg.

Martello, S. & Toth, P.(1986). Algorithms for knapsack problems. In S. Martello, G. Laporte, M. Minoux, & C. Ribeiro (Eds.), *Surveys in Combinatorial Optimization*, (Vol. 31, pp. 213–258). Amsterdam: Annals of Discrete Mathematics.

Masseglia, F., Poncelet, P., & Teisseire, M. (2008). *Successes and new directions in data mining*. IGI Global.

Menasce, D. A. (2002). QoS issues in web services. *Internet Computing, IEEE, 6*(6), 62-65.

Menasce, D. A. (2004).Composing web services: A qos view. *Internet Computing, IEEE, 8*(6), 88-90.

Michlmayr, A., Rosenberg, F., Platzer, C., Treiber, M., & Dustdar, S. (2006). Towards recovering the broken SOA triangle: A software engineering perspective. *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting*, 22-28.

Ming, C., & Zhen-wu, W. (2006). An Approach for web services composition based on qos and discrete particle swarm optimization. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006 (SNPD 2006*), 37-41.

Missaoui, A., & Barkaoui, K. (2010). A neuro-fuzzy model for qos based selection of web service. *Journal of Software Engineering and Applications, 3(6)*, 588-592.

Mitra, N., & Lafon, E. Y. (2006). *SOAP version 1.2 Part 0: Primer (second edition)*. W3C Recommendation. Retrieved 12, June, 2010, from http://www.w3.org/TR/2006/REC-soap12-part0-20060426/

Monmarch, N., Venturini, G., & Slimane, M. (2000). On how pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems, 16*(9), 936-946.

Moser, M., Jokanovic, D. P., & Shiratori, N. (1996). An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, *80*(3), 582-589.

Mostofa Akbar, M., Sohel Rahman, M., Kaykobad, M., Manning, E. G., & Shoja, G. C. (2006). Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & operations research*, *33*(5), 1259-1263.

Neelavathi, S., & Vivekanandan, K. (2011). An innovative quality of service (QOS) based service selection for service orchrestration in SOA. *International Journal of Scientific & Engineering Research, 2(4).*

Newcomer, E., & Lomow, G. (2004). *Understanding SOA with web services (independent technology guides)*. Addison-Wesley Professional.

Norinder, U., Lidén, P., & Boström, H. (2006). Discrimination between modes of toxic action of phenols using rule based methods. *Molecular diversity*, *10*(2), 206-212.

O'Brien, L., Bass, L., & Merson, P. (2005). *Quality attributes and service-oriented architectures*. Software Engineering Institute. Retrieved 12, July, 2012, from http://www.sei.cmu.edu/library/abstracts/reports/05tn014.cfm

Parasuraman, A., Zeithaml, V. A., & Berry, L. L. (1988). SERVQUAL: A multi-item scale measuring consumer perceptions of service quality. *Journal of Retailing, 64*, 12-36.

Parejo, J. A., Fernandez, P., & Cortes, A. R. (2008). Qos-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, *2*(1), 55-66.

Patel, C., Supekar, K., & Lee, Y. (2003). A qos oriented framework for adaptive management of web service based workflows. In V. Marík, W. Retschitzegger, & O. Štepánková (Eds.), *Database and Expert Systems Applications* (Vol. 2636, pp. 826-835). Springer Berlin Heidelberg.

Patel, C., Supekar, K., & Lee, Y. (2004). Provisioning resilient, adaptive web services-based workflow: A semantic modeling approach. *IEEE International Conference onWeb Services, 2004,* 480-487.

Pei, S., Shi, X., & Hu, D. (2014). Research on the Particle-Ant Colony Algorithm in Web Services Composition Problem. *Journal of Applied Sciences*, *14*(8).

Pisinger, D. (1996). A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, *45*(5), 658-666.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods* (pp. 185-208). MIT Press.

Prekopcsák, Z., Henk, T., & Gáspár-Papanek, C. (2010). Cross-validation: The illusion of reliable performance estimation. *RCOMM 2010: RapidMiner Community Meeting and Conference*.

Qiqing, F., Xiaoming, P., Qinghua, L., & Yahui, H. (2009). A global qos optimizing web services selection algorithm based on moaco for dynamic web service composition. *International Forum on Information Technology and Applications, 2009 (IFITA'09)*, 37-42.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco: Morgan Kaufmann Publishers Inc.

Ran, S. (2003). A model for web services discovery with QoS. *ACM SIGecom Exchanges, 4*(1), 1-10.

Rajeswari, M., Sambasivam, G., Balaji, N., Saleem Basha, M. S., Vengattaraman, T., & Dhavachelvan, P. (2014). Appraisal and analysis on various web service composition approaches based on QoS factors. *Journal of King Saud University-Computer and Information Sciences*, *26*(1), 143-152.

Razzazi, M. R., & Ghasemi, T. (2009). An exact algorithm for the multiple-choice multidimensional knapsack based on the core. In H. Sarbazi, B. Parhami, S.G

Miremadi, & S. Hessabi (Eds.), *Advances in Computer Science and Engineering* (pp. 265-282). Springer Berlin Heidelberg.

Rosenberg, F., Leitner, P., Michlmayr, A., Celikovic, P., & Dustdar, S. (2009). Towards composition as a service - a quality of service driven approach. *Proceedings of the 2009 IEEE International Conference on Data Engineering*.

Rozinat, A., & van der Aalst, W. M. (2006). Decision mining in ProM. In S. Dustdar, J. L. Amit, & P. Sheth (Eds.), *Business Process Management* (pp. 420-425). Springer Berlin Heidelberg.

Sasikaladevi, N., & Arockiam, L. (2014). LASA-HEU: Heuristic Approach for Service Selection in Composite Web Services. In *Computing and Communication Technologies (WCCCT), 2014 World Congress on*, 256-259.

Sathya, M., Swarnamugi, M., Dhavachelvan, P., & Sureshkumar, G. (2010). Evaluation of qos based web-service selection techniques for service composition. *International Journal of Software Engineering*, *1*(5), 73-90.

Sbihi, A. (2006). A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. *Journal of Combinatorial Optimization*, *13*(4), 336-351.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., & Van der Aalst, W. (2008). Process flexibility: A survey of contemporary approaches. In W. Van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, & C. Szyperski (Eds.), *Advances in Enterprise Engineering I* (Vol. 10, pp. 16-30). Springer Berlin Heidelberg.

Schuller, D., Eckert, J., Miede, A., Schulte, S., & Steinmetz, R. (2010). QoS-aware service composition for complex workflows. *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, 333-338.

Schuller, D., Polyvyanyy, A., García-Bañuelos, L., & Schulte, S. (2011). Optimization of complex qos-aware service compositions. In G. Kappel, Z. Maamar, & H.R. Motahari-Nezhad (Eds.), *Service-Oriented Computing* (pp. 452-466). Springer Berlin Heidelberg.

Shen, J., & Yuan, S. (2009). QoS-aware peer services selection using ant colony optimization. In W. Van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, & C. Szyperski (Eds.), *Business Information Systems Workshops* (Vol. 36, pp. 362-364). Springer Berlin Heidelberg.

Singh, A. K. (2012). Global optimization and integer programming networks. *International Journal of Information*, *2*(8).

Sumathi, S., & Esakkirajan. S. (2007). *Fundamentals of relational database management systems.* Springer Berlin Heidelberg.

Tang, M., & Ai, L. (2010). A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In M. Tang (Ed.), *Proceeding of the 2010 World Congress on Computational Intelligence*. Barcelona.

Tao, F., LaiLi, Y., Xu, L., & Zhang, L. (2013). FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. *Industrial Informatics, IEEE Transactions on*, *9*(4), 2023-2033.

Toma, I., & Foxvog, D. (2006). *Non-functional properties in web services*. Web Service Modeling Ontology (WSMO) Working Draft. Retrieved 13, June, 2010, from http://www.wsmo.org/TR/d28/d28.4/v0.1/

Todorovski, L., & Džeroski, S. (2000). Combining multiple models with meta decision trees. In D. Zighed, J. Komorowski, & J. Zytkow (Eds.), *Principles of Data Mining and Knowledge Discovery* (Vol. 1910, pp. 69-84). Springer Berlin Heidelberg.

Tong, H., Cao, J., & Zhang, S. (2006). A distributed genetic algorithm for optimizing the quality of grid workflow. In K. Chang, W. Wang, L. Chen, C. Ellis, C.-H. Hsu, A. Tsoi, & H. Wang (Eds.), *Advances in Web and Network Technologies, and Information Management* (Vol. 4536, pp. 408-419). Springer Berlin Heidelberg.

Toyoda, Y. (1965). A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, *21*(12), 1416-1426.

Ukor, R., & Carpenter, A. (2009). Flexible service selection optimization using meta-metrics. *Proceedings of the 2009 Congress on Services – I*, 593-598.

Ukor, R., & Carpenter, A. (2008.). On modeled flexibility and service selection optimisation. *9th Workshop on Business Process Modeling, Development and Support*.

van der Aalst, W. M., Adriansyah, A., de Medeiros, A. K., Arcieri, F., Baier, T., Blickle, T., ... & Pontieri, L. (2012). Process mining manifesto. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Business process management workshops* (pp. 169-194). Springer Berlin Heidelberg.

van der Aalst, W. M., Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases, 14*(1), 5-51.

van der Aalst, W., M. (2009). Process-aware information systems: Lessons to be learned from process mining. *Transactions on Petri Nets and Other Models of Concurrency II*, 1-26.

van der Aalst, W., M., Dongen, B. F. v., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. J. M. M. (2003). Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering, 46*(2), 236-266.

van der Aalst, W. M., Weijters, A. J., & Maruster, L. (2002). *Workflow mining: Which processes can be rediscovered*. BETA Working Paper Series, WP 64, Eindhoven University of Technology, Eindhoven.

van der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and parallel databases*, *14*(1), 5-51.

van der Werf, J. M., van Dongen, B. F., Hurkens, C. A., & Serebrenik, A. (2008). Process discovery using integer linear programming. In K.M. van Hee, & R. Valk (Eds.), *Applications and Theory of Petri Nets* (pp. 368-386). Springer Berlin Heidelberg.

Wan, C., C., U., Chen, L., Huang, R., Luo, J., & Shi, Z. (2008). On solving qos-aware service selection problem with service composition. *Seventh International Conference on Grid and Cooperative Computing, 2008 (GCC '08),* 467-474.

Wang, J., & Hou, Y. (2008). Optimal web service selection based on multi-objective genetic algorithm. *Proceedings of the 2008 International Symposium on Computational Intelligence and Design*, 553-556.

Wang, L., & He, Y.-x.(2010). A web service composition aalgorithm based on global qos optimizing with MOCACO. In C.-H.Hsu, L. Yang, J. Park, & S.-S.Yeo (Eds.), *Algorithms and Architectures for Parallel Processing* (Vol. 6082, pp. 218-224). Springer Berlin Heidelberg.

Wang, R., Chi, C.-H., & Deng, J. (2009). A fast heuristic algorithm for the composite web service selection. In Q. Li, L. Feng, J. Pei, S. Wang, X. Zhou, & Q.-M. Zhu (Eds.), *Advances in Data and Web Management* (Vol. 5446, pp. 506-518). Springer Berlin Heidelberg.

Wang, S., Zhu, X., & Yang, F. (2014). Efficient QoS management for QoS–aware web service composition. *International Journal of Web and Grid Services*, *10*(1), 1-23.

Weinhardt, C., Anandasivam, A., Blau, B., & Stosser, J. (2009). Business models in the service World. *IT Professional*, 11, 28-33.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wu, M., Xiong, X., Ying, J., Jin, C., & Yu, C. (2011). QoS-driven global optimization approach for large-scale web services composition. *Journal of Computers*, *6*(7), 1452-1460.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ... & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1-37.

Xia, Y.-m., Chen, J.-l., & Meng, X.-w. (2008). On the dynamic ant colony algorithm optimization based on multi-pheromones. *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008).*

Yang, L., Dai, Y., Zhang, B., & Gao, Y. (2006). A dynamic web service composite platform based on qos of services. In H. Shen, J. Li, M. Li, J. Ni, & W. Wang (Eds.), *Advanced Web and Network Technologies, and Applications* (Vol. 3842, pp. 609-616). Springer Berlin Heidelberg.

Yang, Z., Shang, C., Liu, Q., & Zhao, C. (2010). A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. *Journal of Computational Information Systems, 6*(8), 2616- 2622.

Yoon, K. P., & Hwang, C. L. (1995). *Multiple attribute decision making*. Thousand Oaks, CA: Sage Publication.

Yu, D., Li, C., & Yin, Y. (2014). Optimizing Web Service Composition for Data-intensive Applications. *International Journal of Database Theory & Application*, *7*(2).

Yu, T., & Lin, K.-J.(2005). A broker-based framework for qos-aware web service composition. *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, 22-29.

Yu, T., Zhang, Y., & Lin, K. J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, *1*(1), 6.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., & Sheng, Q. Z. (2003). Quality driven web services composition. *Proceedings of the 12th international conference on World Wide Web*, 411-421.

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering, 30*(5), 311-326.

Zhang, C., Su, S., & Chen, J. (2006). DiGA: Population diversity handling genetic algorithm for qos-aware web services selection. *Computer Communications, 30*(5), 1082-1090.

Zhang, C., Su, S., & Chen, J. (2006a). A novel genetic algorithm for qos-aware web services selection. In J. Lee, J. Shim, S.-g. Lee, C. Bussler, & S. Shim (Eds.), *Data Engineering Issues in E-Commerce and Services* (Vol. 4055, pp. 224-235). Springer Berlin Heidelberg.

Zhang, C., Su, S., & Chen, J. (2006b). Efficient population diversity handling genetic algorithm for qos-aware web services selection. In V. Alexandrov, G.

van Albada, P. Sloot, & J. Dongarra (Eds.), *Computational Science – ICCS 2006* (Vol. 3994, pp. 104-111). Springer Berlin Heidelberg.

Zhang, L.-J., Li, B., Chao, T., & Chang, H. (2003). On demand web services-based business process composition. *International Conference on Systems, Man and Cybernetics,* 2003, 4056-4064.

Zhang, W., Chang, C. K., Feng, T., & Jiang, H.-y. (2010). QoS-based dynamic web service composition with ant colony optimization. *34th Annual Computer Software and Applications Conference (COMPSAC), 2010*, 493-502.

Zhang, W., Yang, Y., Tang, S., & Fang, L. (2006). QoS-driven service selection optimization model and algorithms for composite web services. *Proceedings of the 31st Annual International Computer Software and Applications Conference*, 425-431.

Zheng, H., Zhao, W., Yang, J., & Bouguettaya, A. (2013). Qos analysis for web service compositions with complex structures. *Services Computing, IEEE Transactions on*, *6*(3), 373-386.

Zibanezhad, B., Zamanifar, K., Nematbakhsh, N., & Mardukhi, F. (2009). An approach for web services composition based on qos and gravitational search algorithm. *Proceedings of the 6th international conference on Innovations in information technology*, 340-344.