January 2019

Web Service Composition Optimization

Hussain Aljafer
Wayne State University, aljafer.hussain@gmail.com

WEB SERVICE COMPOSITION OPTIMIZATION

by

HUSSAIN ALJAFER

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2019

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

_____

_____

_____

_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: BACKGROUND AND INTRODUCTION

1.1 Introduction

In recent years, a number of organizations and users have started exchanging workstation-based systems with web services or what is sometimes known as "Cloud Computing". Cloud computing is centered on the services notion which are defined as "independently developed and deployed artifacts to which clients subscribe on per need basis and pay the service providers based on their usage" [3]. Web services can be defined as a set of related functionalities to be accessed using an XML-based standard. One of the main reasons for changing from workstation-based applications to Web services-based applications is the cost of deployment and maintenance. Web services reduce this cost to the user dramatically since the user will not be involved in any of these, instead, the service provider is the responsible party. The users then do not need to own and operate their server and maintain it which is quite costly. This has helped many of the web apps to grow and be comparable to some large companies. According to Forbes.com, Pinterest had a huge growth in terms of the generated traffic, and it can be compared to Facebook. The former uses Amazon's Web services while the latter maintain their own servers.

One of the main concerns of the users and organizations is the security and privacy of their data. The data can be hosted on some services such as Apple's iCloud or transmitted using services. Either of these cases, we need to prevent unauthorized data access at all time, hence the need of encryption schemes which will help achieving this. another issue is due to the huge number of services provided by service providers, the users look service composition optimization to reduce the cost and/or enhance the application's performance. The service provider on the other hand looks to make the highest possible revenue while providing competitive prices to attract users. This

brings the need of a composition optimization model that serves both the user and the service provider.

1.2 Service Oriented Architecture

The service oriented architecture is usually defined as a "Paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [41] [27]. SOA's boundaries are explicit so the communication between the services would be across different zones geographically, trust wise ownership wise and according to operating environment. The SOA has two main components which are the service provider and the user (also called customer). The services are provided by the service providers by having the service information published in the UDDI [52] [45] which is a public directory where the users query these directories to find the best services for their applications.

1.3 QoS and Composition Optimization

Quality of Service (QoS) parameters refer to the features of the service such as response time, availability, cost...etc. These play a major role when it comes to determining the failure and success of the application to be composed [4]. This brings the need of a Service Level Agreement (SLA) that is set between the service provider and the service user and provides the expected QoS details. In Web service composition we aim at finding the set of services that satisfy the QoS constraints of set by the user or the application to be composed. Just as Real-world negotiations do not require the parties to reach a negotiated agreement; similarly, the automated negotiation has the same options. An entity can choose "no deal" if it cannot negotiate a satisfactory agreement. Furthermore, there are distinct negotiation strategies for "open" and "closed" marketplaces. A closed marketplace is based upon a predefined set of users, who "enroll" in the marketplace and agree to a certain set of rules. An

open marketplace has no such agreement; entities are welcome to enter and exit at any time and are not required to agree to any rules. This adds to the complexity and uncertainty of information. Hence entities need to take into account these uncertain information patterns and deal accordingly.

CHAPTER 2: LITERATURE REVIEW

2.1 Existing Web Service Composition Optimization Schemes

Different Schemes were proposed to tackle the Web service composition optimization problem using different techniques. The following will summarize some of the most popular ones and what technique they have used,

2.2 Genetic Algorithms

Due to the presence of multiple objectives in some problems, the solution to this kind of problem consists of a set of optimal solutions known as Pareto-optimal solution instead of a single optimal solution. Since we cannot judge weather one set of Pareto-optimal solutions is better than the other, this arises the need of finding as many sets as possible [20]. Multi-objective evolutionary algorithms were proposed as a solution of this problem. These algorithms use a population of random solutions to produce another population of solutions that converges toward the true Pareto-optimal solutions region. Genetic algorithms evolved from that concept.

NSGA (Non-dominated Sorting Genetic Algorithm) was one of the first evolutionary algorithms. This algorithm uses the principles of evolutionary biology to solve this problem. It applies techniques such as inheritance and mutation [58]. The optimization process begins with a population containing randomly generated solutions and has a fitness value for these solutions. The individuals having higher fitness values are saved into another population where the winners form a mating pool. the algorithm is then applied to the newly generated population and this step is repeated until the termination criteria is met. NSGA-II was implemented to solve multi-objective optimization problems as an improved version of NSGA. The difference in NSGA-II is that: after the random population is created and crossover/mutation phase, the individuals are ordered into non-dominated sets. These sets are then reduced to find

the next non-dominated sets. This is repeated until enough individuals are sorted into the non-dominated sets where the sorting here is using Quick sort algorithm [49]. The following are some of the schemes using GA as a Base for the solution.

2.2.1 Multi-objective Service Composition with Time- and Input-Dependent QoS

Since dependency on execution time or input data is mostly ignored in current QoS models, this paper introduces a QoS model that covers the dependencies and discuss how this can be used to consider multiple workflows at once. The paper also proposes a multi-objective optimization approach to offer solutions varying on the time and price to give the user the ability to make decisions. Fictional and non-functional requirements are both considered to achieve the user satisfaction. In QoS there is no single selection that dominates all other selections, so the users need to define their preferences for the solution they want and then service composition is applied to choose the services on behalf of the user. The proposed approach consists of three steps: 1. Modeling the problem using Hierarchical Workflow Graph.

2. Use this Graph to determine time and input dependencies.

3. Compute the set of feasible solutions using the multi-optimization algorithm.

The approach is based on GA.[77]

2.2.2 Single-Objective versus Multi-Objective Genetic Algorithms for Workflow Composition based on Service Level Agreements

This paper analyzes both the single objective and the multi-objective genetic algorithms in the context of Web service composition measuring the success rate along with the execution time SLA (Service Level Agreement) is defined as a contract be-

tween the service provider and the consumer with respect to QoS parameters. The paper first revisits the GA. The paper has the functions for both the single objective and the multi-objective problems based on four factors: Reliability, Availability, Execution time and Execution price. In the single-objective GA the algorithm is used to find the approximate solutions to the difficult-to-solve problems by applying the evolutionary biology principles to the problems. The optimization takes a randomly generated population (solutions) and evaluates the fitness of each individual and then saves the fittest ones. In the multi-objective GA Approach, the most fit individuals from both archive and child population are determined by a ranking mechanism. This is referred to as NSGA-II which is differs from the GA in: all individuals are ordered into non-dominated sets then the reduced set is used to find the next non-dominated solutions which is executed iteratively until enough individuals are sorted into non-dominated sets. Detailed comparison with the experiments and results are presented in graphs in the paper proving that NSGA-II has a slightly higher success rates for smaller population sizes.[49]

2.2.3 QoS decomposition for service composition using genetic algorithm

This paper presents a top-down approach that uses genetic algorithm to decompose the global constraints into local constraints then select the best web service for each task using a simple linear search. The composite service structure is defined as a workflow that is conceptually specified based on tasks and control structures. The paper then represents some definitions that are used in the proposed approach such as global constraints and local constraints. The quality model in this paper depends on a set of attributes that are categorized into 2 main types: positive and negative. The objective is to maximize the positive ones and minimize the negatives. The

approach handles the QoS composition through 2 sub problems: quality constraint decomposition, Local selection. The paper then explains each of these concepts and how these ones are done. After that some information about genetic algorithm is explained. Local selection is the explained along with some formulas followed by the experiment and performance evaluation.[53]

2.2.4 Solving Multi-Objective and fuzzy multi-attributive integrated technique for the QoS-Aware Web Service Selection

The paper proposes a multiple criteria decision-making methodology for global web service -selection based on QoS criteria and this integrates the multi-objective optimization with a fuzzy multi-attributive group decision-making. For the optimization, the scheme uses a genetic algorithm basis. The scheme consists of two major stages:

1) Multi-objective optimization stage to find the Pareto-optimal design alternatives.

2) Ranking these PODAs according to the predefined attributes.

In the first stage, a multi-objective optimization technique called FRONTIER is used which is basically a genetic algorithm that generates individuals randomly and these individuals represent the service nodes that can be assessed on several QoS criteria. This finds the trade-off to best approach the solution. Next the solutions can be combines as parents that generates new (child) solution that is better than the parents and as this continues, the generations come closer to the optimal solution. Next stage is the decision-making stage and it consists of three phases:

- Rating phase: where the decision matrix is established for each PODA so the data need some conversion (the formula for conversion is in the paper).

- Attribute based aggregation: which creates another matrix according to a six-stage algorithm to find the aggregation result of the fuzzy opinion based on relative agreement in the opinions.

- Selection Phase: which has two sub-phases: Defuzzification which generates another decision matrix that contains only crisp data and Ranking sub-phase that determines the ranking order of the PODAs which has six steps detailed in the page to get the result as a ranked order.[84]

2.2.5 Web Service Dynamic Composition Based on Decomposition of Global QoS Constraints

Since most QoS existing solutions are based on global optimization, they suffer from poor performance, so they are becoming inappropriate for applications that have dynamic and real time requirements. This paper proposes an approach that decomposes global constraints into local ones. The paper includes 9 properties in the attributes of QoS composition and these include cost, response time, reliability, availability, rate of successful execution security reputation throughput and load. The user requirements are the global constraints, and these are represented as upper and lower bound for the aggregated values of the QoS attributes. For a solution to be feasible, all values have to satisfy the global QoS constraints. The paper uses Simple Additive Weighting approach for the utility function so before the computation of the utility value of composite web service, each value is transformed into a value between 0 and 1 by a comparison to minimum and maximum possible values in the service class. Formulas for each attribute and for the composition function are available in the paper. The main idea of the decomposition is decomposing each global QoS constraint into a set of local constraints and these local constraints serve as conservative lower and upper bounds that are used by the local service selection

algorithm to get the best services or each abstract service. A new algorithm called Culture Genetic Algorithm is developed to handle the growth of the number of quality level combinations for CWS. An evaluation function is also developed to evaluate the fitness of different quality level combinations. Detailed algorithms are included in the paper. For Future work, the research will continue on applying the method on solving manufacturing problems like partner selection, supply chain planning? etc. [47]

2.2.6 Selecting Web Services for optimal composition

This paper proposes service quality variables as non-functional criteria to get the optimal web service composition for a certain goal and also proposes a scheme for multi-objective optimization to find the set of Pareto Solutions so the user can then choose one of these solutions depending on the trade-off the user chooses. The paper also proposes an environment to allow this composition to be automatic. In the quality model, the scheme considers 4 attributes: cost, time, availability and reputation. The problem model also models three constraints: 1- only one service belongs to a composition for each task.

2- User budget.

3- Xij must have binary values (Xij specifies the service belongs to a composition matrix).

Genetic algorithm NSGA-II is used as a resolution method where it is not necessary to create a mathematical expression to determine the weights associated to the criteria which will be needed with aggregation approaches. The experiments results gave sets of chromosomes (population) that are represented by a set of binary values to determine whether the service belongs to the composition or not. These have shown that the approach is feasible for real life applications. Pros: including reputation in the attributes for checking security, Giving the user ability to choose one solution

among the set of optimal ones.

Cons: not including scalability which is an important factor. [16]

2.2.7 A QoS-based service composition optimization method

This paper proposes an approach for service composition by first using Pareto dominant elite screening collection of services to remove redundancy in the candidate services then genetic algorithm is used to optimize the set. Pareto dominance is a classic heuristic algorithm that is used for solving combinational optimization problems of web services. Genetic algorithms are also classical heuristic algorithm that are used in the optimization problems and this paper combines both of these technologies to get the optimized solution to the problem of Web service composition. The description of these algorithms and their combination is presented in the paper. [79]

Pros: explained the algorithms and used some solid technologies (that are the algorithms) to resolve the problem.

Cons: just combined the technologies and did not add a new thing.

2.2.8 Trust-oriented QoS-aware Composite Service Selection Based on genetic Algorithms

This paper proposes a scheme for service selection in web service composition as well as a trust evaluation method for the service composition plan based on the subjective probability theory and based on this, a trust-oriented genetic algorithm is proposed to find the near-optimal service composition plan with the QoS constraints. The paper considers four attributes in the QoS composition attributes: Time, Price, Availability and Reliability and the calculation methods for each one of these in the composition cases (sequential invocation and loop) are presented in the paper along with the fitness function of the genetic algorithm. There is a difference in

this approach in the genetic algorithm which is that the paper assumes that a large volume of services with their ratings are stores in a third party trust management authority. The paper only focuses on numerical attributes of the QoS. Experimental results show the effectiveness and efficiency in finding the near optimal solution.[26]

Pros: providing equation for all different cases handled in the approach.
Cons: not handling the conditional case in the service composition.

2.3 Bees Algorithm

Bees Algorithm is a population-based search algorithm that performs search in the neighborhood combined with random search to find the best solution for multi-objective optimization problems.

2.3.1 Optimal web service selection and composition using Multi-objective bees algorithm

The paper proposes a multi objective bees algorithm to select the optimal service in the service composition resolving the service composition optimization problem. The paper considers 4 attributes for the quality: cost, response time, reliability and availability formulated as the QOS vector. The paper provides computation models for computing the QOS for the 4 control models provided by BBEL4WS which are: Sequential, Conditional, Loop and Parallel. A fitness evaluation built as a matrix records the quality information for the services. The selection is based on Bees algorithm which is a population-based search algorithm performing search in the neighborhood combined with random search.[42]

2.4 Particle Swarm

In Particle Swarm Based Algorithms, the solution is encoded as particles in PSO where there is a position and a velocity for each particle. These particles adjust

their selves according to pbest (best position experienced by a particle) and gbest (best position experienced by the swarm).

2.4.1 Efficient Multi-Objective Services Selection Algorithm Based on Particle Swarm Optimization

This paper proposes a service selection model based on the particle swarm optimization model. The paper starts by defining the terms task, service and vector that will be used in the paper. The multi-objective optimization problem falls into NP complete class problems and PSO has been used in solving many NP complete problems much more efficiently than genetic algorithms. These are the comparison rules between two solutions I copy pasted from the original paper: 1) If two solutions meet constraints, the dominant one is preferable;

2) If one solution meets constraints and the other doesn't, the one meeting constraints is preferable;

3) If two solutions have different deviation from constraints, the smaller deviation one is preferable;

4) If two solutions have the same deviation from constraints, the dominant one is preferable;

5) If two solutions have the same deviation from constraints and are non-inferior, both are preferable

The solution is then encoded as particles in PSO where there is a position and a velocity for each particle. These particles adjust their selves according to pbest (best position experienced by a particle) and gbest (best position experienced by the swarm). Initialization, update and gbest, pbest selection strategies are described in the paper. The paper also proposes an algorithm EMOSS (efficient multi-objective

service selection algorithm based PSO) with QoS global guarantee. Detailed algorithm along with analysis and comparison are in the paper. The proposed algorithm does not require heuristic knowledge related to the application background and can get high quality solutions adapting to the user's requirements and work even under QoS dynamic change without the increase in the running time as the problem size increases.[39]

2.4.2 A Hybrid Multiobjective Discrete Particle Swarm Optimization Algorithm for a SLA-Aware Service Composition Problem

This paper proposes an algorithm for hybrid multi-objective discrete particle swarm optimization and a model to optimize SLA-aware service composition. The paper proposes a local searching strategy based on constraint domination that is introduced to the algorithm to accelerate the process of obtaining a feasible solution. The particle updating strategy is made by a crossover operator in genetic algorithm based on the exchange of candidate service. The paper proposes a strategy to mutate the particles for the purpose of increasing the diversity of the particle swarm. Service throughput, cost and also latency are the QoS attributes considered by the paper for parallel and sequence composition structures, the aggregation functions for each case are presented in the paper. The paper has details on the design of the proposed algorithm.[80]

Pros: adding performance to resolve the issue.

Cons: considering only three QoS attributes and only two composition structures.

2.4.3 An Effective Dynamic Web Service Selection Strategy with Global Optimal QoS Based on Particle Swarm Optimization Algorithm

This paper proposes an approach to solve the web service selection in the web service composition based on the theory of particle swarm optimization algorithm. The strategy is called PSO-GODESS (Global Optimization of Dynamic Web Service Selection based on PSO). The goal is to implement web service selection with QoS global optimization. First the algorithm transforms the original selection problem into a multi-objective service composition optimization problem subject to global QoS constraints. This is further transformed to a single-objective problem by using the method of ideal point. Lastly, the intelligent optimization theory is applied to produce a set of optimal services composition process with the QoS constraints. Details of the steps and algorithm details are included in the paper. Test results show the feasibility of the proposed approach and the efficiency tests prove that it is faster than multi-objective genetic algorithms.[40]

2.5 Domain Specific Language DSL

2.5.1 An End-to-End Approach for QoS-Aware Service Composition

This paper proposes a composition approach based on DSL (Domain Specific Language) to specify the functional requirements of QoS and also the expected constraint hierarchies by leveraging hard and soft constraints. To optimize the composition in a semi-automatic way, as composition runtime resolves the user's constraints and find the solution. Using VERSCO as a runtime environment, the core services can be access directly via SOAP or using the client library that provides the simple

API. Leveraging VERSCO core services helps integrating Composition Service on the level of the infrastructure. The user interacts with the system by specifying a composition using VCL. The paper then explains VCL. A VCL composition consists of a set of features where each feature has a set of candidates to implement the feature. Querying and matching all candidates that implement certain feature is called feature resolution and it is used to reduce the number of candidates to speed up the overall process. QoS aggregation is needed to determine a formula for the composition. The first approach is Constraint Optimization Problem. In order for the solution to be found, all constraints have to be fulfilled including feature constraints and global ones. The other approach is integer programming approach where we define a new objective function for calculating an overall utility value for features while considering the user's constraints. Details about constraints in both approaches are in the paper. The test results show that the performance is promising and as a future work, VERSCo's infrastructure needs to be leveraged to do efficient runtime re-composition and reduce the continuous querying.[69]

2.6 Novel Algorithms

These solutions come up with their own algorithms for Web service selection and do not depend on one of the well-known algorithms such as GA, Bees or PSA.

2.6.1 An Improved Ant Colony Optimization Algorithm for QoS-Aware Dynamic Web Service Composition

This paper proposes an algorithm for QoS aware dynamic web service composition by transforming the problem of global optimality into a multi-objective, multi-choice Quality of Service optimization for Web service composition. This is reduced into the QoS-aware dynamic web service composition. The infrastructure's topology can be presented as a multi-stage graph where the edges connecting the

nodes are the values of QoS attributes between these nodes. The approach uses an improved OAC algorithm to solve the multi-objective selection. [72]

2.6.2 TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition

The paper tries to solve the issue of web service selection and composition problem not only by considering the functional requirements but also by the transactional properties and QoS characteristics. The selection algorithm expresses the user's preferences as weights over QoS criteria and as levels of risk to define the transactional requirements semantically. The paper defines three categories of web services:

- Pivot WS: these services leave the effect forever once they are executed and cannot be undone.
- Compensable WS: These ones have other services that can undo what these do.
- Retriable WS: those ones guarantee a successful termination after finite invocation number.

The paper considers five factors affecting the QoS: price, execution duration, reputation, rate of success and availability. There are also other definitions for the services according to the transactional properties: Atomic WS: like Pivot WS Transactional WS which has a transactional behavioral property in a vector. For the users to express their criteria, the paper defines the risk:

- Risk 0: where the system guarantees compensating the obtained result in the case of the successful execution.
- Risk 1: the system does not guarantee compensating the result by the user.

In order to obtain TCWS (Transactional Composition of the Web Services), the paper assigns a WS for each activity in the workflow. This depends on the workflow being Sequential, Parallel .... etc. detailed work in the paper. [23]

2.6.3 A Novel Web Service Composition Recommendation Approach Based on Reliable QoS

The paper proposes a novel approach for web service composition. The QoS information is classified into 2 categories: local and Global. For data definition, there are N global QoS dimensions and M local dimensions so there are I QoS where I = M+N. the rest of the paper provides mathematical formulas to calculate the credibility which is the greater distance = the worse the credibility. The approach finally computes the local QoS and the global QoS separately then the paper provides a real-life case study to validate the feasibility of the proposed approach. [36]

2.6.4 RMORM: A framework of Multi-objective Optimization Resource Management in Clouds

This paper proposes a design on multi-objective serial optimization with priorities for the purpose of finding the resource deployment in the clouds. The approach intends to help the service providers in estimating and satisfying the user requirements such as cost, performance, availability and reliability. The main constraints for the users to measure their applications in the cloud are: security, data, SLA and performance which is mainly affected by the number of virtual machines and the distance between them. The proposed scheme is composed of 5 components:

- Performance model: that accepts the user's SLA and satisfies performance requirements.

- Availability model: for accepting the data form the performance model and obtaining the topological structure of the virtual machines.
- Placement model: gets the real locations of the virtual machines.
- Consolidation model: for the providers to save cost and improve resource utilization.
- Monitor: to monitor the rest of the models and also does the database access part.

The rest of the paper is detailed information about these models followed by experiments and evaluation which shows that the algorithms used in the scheme need to be improved. [19]

2.6.5 Metaheuristic Optimization of Large-Scale QoS-Aware Service Compositions

This paper proposes an approach for optimizing web service composition in large-scale service-oriented systems using constraint hierarchies. A composition model contains composite service consisting of a set of abstract services that are non-executable and describe the core functionality in terms of operations, I/O and conditions. In the QoS model, the selection can be improved by considering the candidate service and a vector is used to denote the available services. The paper focuses on two categories: operational attributes and business-related attributes. The model also includes a flexible model for the QoS constraints which have two categories: Global and local constraints. The objective function is defined and a metaheuristic (which is an iterative generation process that guides a subordinate heuristic via combining different concepts to explore and exploit the search space) is used and a neighbor generation function is used to generate new and possibly better solutions. The next

step is Candidate evaluation and solution evaluation which are described in detail in the paper. The QoS values are normalized based on the aggregated values to different scales where the closer to 0 is better and closer to 1 is worse. Next step is computing the penalty values for global constraints violation. The scores then must be combined to a single value. The next step is solution modification which contains some steps like improved mutation, improved local and global penalty and some other steps that are detailed in the paper. The evaluation shows that the scheme performs better than the existing ones and the next step is to extend it for nominal QoS attributes.[70]

2.6.6 An Optimal QoS-Based Web Service Selection Scheme

This paper proposes a service selection scheme for helping the service requesters in selecting the services in web service composition. The scheme considers two contexts: single QoS-based service discovery and QoS based optimization of the composition. The attributes considered in the selections are response time, reliability, availability and cost. Since UDDI provides a way of retrieving qualified services for functional and text-based matchmaking, this paper focuses on numeric based QoS matchmaking. The paper discusses also the single QoS Based discovery which evaluates task individually and picks the one with highest performance while for the QoS based optimization, the overall workflow structure for a given composition with highest performance is selected. These are discussed in detail in the paper and each case of the composition is also discussed along with calculation formulas. Mathematical programming techniques are used for scheme implementation and solving objective function. The algorithm of the scheme has two main phases: pruning phase which filters out the candidates in each task then comes the branch-and-bound phase which selects the maximum of the bounds of all branches. The experiments show that this scheme's performance is better than the enumerative method in terms of effi-

ciency and effectiveness. The next enhancement is focusing on semantics-enhanced service discovery and composition to provide higher precision in candidate service discovery.[35]

2.6.7 QoS-Aware and Multi-granularity service composition

This paper considers how to produce a new service composition plan while preserving original observable behaviors seen by the user. This new plan aims at finding better quality services. The first step here is defining a model that captures the observable behavior to the user and then define that two compositions are equivalent if they have the same observable model. To model the behavioral signature graph, a directed acyclic graph is used to model the processes which model services and the reason for choosing DAG is to be able to apply some of the graph algorithms to check the behavioral compliance. A complete definition of the graph is in the paper. The algorithms and experiment are also in the paper. The future enhancement on the proposed mechanism is to do some formal analysis of the proposed algorithms.[24]

Pros: clear and understandable mechanism.

Cons: does not show the gain or loss in the new composition plan i.e. there is nothing that shows the performance increase or if it is even there in the new plan.

2.6.8 A fuzzy multi-objective model for provider selection in data communication services with different QoS levels

This paper proposes a scheme to solve the multi-criteria decision-making problem where the user must weigh up the relative importance of the factors considering nonlinear objective membership function, multi-class services, price, penalty definition in different tasks and different QoS levels. The proposed solution reflects both subjective judgment and objective information in real life circumstances and incorporates the concepts of stochastic theory, fuzzy sets and scenario analysis to do the

provider selection to handle the vagueness and ambiguity that is available in the information as well as essential fuzziness in human judgment and preference. Objective function and penalty functions handling different scenarios are available in the paper. Also functions to calculate cost, reliability, delay, demand and cases for provider selection are presented in the paper. The functions are also defined in the case of fuzzy environment with defined symbols to represent fuzziness. One of the major problems is the uncertainty of the data, therefore an algorithm combining features of fuzzy data and stochastic data was developed to resolve this issue. The proposed scheme gives less computational complexity and makes the application more understandable[64].

Pros: handling the missing or uncertain data.

Cons: paper has lots of function without a clarification on how to combine all these things.

2.6.9 A QoS preference-based algorithm for service composition in service oriented networks

This paper proposes an order relation model that can be used to calculate and represent user's preference information in an effective manner. The ordered QoS attributes in the vector represent the user's QoS requirements. To manage the QoS values, a QoS attributes matrix is introduced. This paper considers four attributes: response time (as delay), reliability, reputation and cost. The paper defines a QoS preference function that reflects the user's preference information. In the order relation vector, the users sort the QoS attributes according to preference level. Since the QoS attributes have different units, these need to be normalized prior to calculation on the matrix to make it easier to get the result. Defuzzication, normalization and vector calculation method are explained in the paper along with formulas used in these stages. To summarize the steps: the user provides QoS preference order, this is

converted to order relation vector, the QoS value matrix is defizzified and normalized, the QoS weight vector is obtained from the matrix, fine degree function is used to calculate the candidate service compositions and finally the matrix is stored to get the optimal solution[83].

Pros: finds and stores optimal solution.

Cons: lots of calculations: matrix, defuzzication, normalization then function.

2.6.10 An Efficient and Reliable Approach for Quality-of-Service-Aware Service Composition

The paper addresses the problem that the current existing solutions of QoS composition do not suit real time decision making that are required to get the optimal solution within a reasonable amount of time and also the reliability issues. A novel heuristic algorithm is proposed to serve efficient and reliable selection of trustworthy services in a service composition. The algorithm consists of three steps. At the first step, a trust-based selection method is performed to filter the untrusted services. At the second step, the search space is reduced for the composition. Finally, a global optimization approach tries to get the near optimal solution. The system architecture contains four components:

- Service community: groups services related to specific area.
- Composition Manager: considers functional user requests to create an execution plan.
- Trust Manager: evaluates trustworthiness of the matched services.
- Selection Manager: selects services along the optimal path from the graph generated by the composition manager.

QoS is divided into three categories: Generic such as response time, availability, cost and reliability. Domain specific that depends on the nature of service like in video service where resolution and color depth are considered, and User perceived that combines both. Utility function along with algorithm details and test results are described in the paper [44].

Pros: including domain specific QoS attributes.

Cons: could not think of a negative part.

2.6.11 Toward Better Quality of Service Composition Based on a Global Social Service Network

The paper proposes an approach to move from isolated service islands to global social service network by developing a model for the network that supports service sociability. A construction of a GSSN is first proposed based on quality of social links and then an algorithm for mapping GSSN into service cluster is proposed to reduce the search space. The paper also proposes a quality-driven approach to enable exploitation of the service cluster by providing workflow as a service. The idea of sociability is for a service to see with whom it has worked before and with whom it is willing to work in the future. Considering isolated services leads to several difficulties such as poor scalability, expanded search time and lack of sociability. The GSSN uses social links to connect cross-domain distributed services and this is similar to RDF to connect distributed files into single global space. The social link can be illustrated by rules that include composition patterns such as sequential, parallel and conditional. The quality of social links is determined through some evaluation criteria described in the paper. The evaluation of the scheme used three metrics: response time, number of services required for fulfilling the composition goal and success rate. The evaluation shows that the approach improved the response time and success

rate, but the limitation is with the static generation of the GSSN for each group of users instead of adapting to user preferences. Another limitation is that the approach cannot incorporate the user feedback [13].

2.6.12 A Multi-Criteria QoS-aware Trust Service Composition Algorithm in Cloud Computing Environments

the paper proposes a global trust service composition approach based on random QoS and trust evaluation. The approach considers the multi-criteria assessment of service quality. The first step is removing the uncertain outliers and estimate the ideal value of the collected objective QoS data, and this is done using statistical testing (hypothesis test). Then subjective QoS evaluations of the providers and users are aggregated according to direct trust and recommended trust. The final step is composing the services through global QoS optimization. Formulas along with detailed techniques for the steps are included in the paper. The test results show that the approach improves accuracy and is suitable for dynamically changing cloud environments.[48]

Pros: accuracy improvement and it is suitable for dynamic clouds.
Cons: does not include the cost of the service and it has lots of calculations.

2.7 Linear Programming

2.7.1 Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes

This paper considers a broker offering composite service with multiple classes to several users creating flow requests. the scheme proposed here optimizes the end-to-end aggregated QoS of the incoming requests and is based on linear programming. The approach guarantees that the QoS constraints will be met for each request. The

paper then describes the architecture of the broker which acts as an intermediary between the service providers and the users and this broker is maintained and operated by a third party. The paper assumes that composite service structure is defined using BPEL definition. For the QoS composition model the approach considers the response time, execution cost and availability and have equations to calculate matrices for these parameters. For the optimization problem, the goal of the engine here is to determine the variables that maximize the objective function. The problem is a linear programming problem that can be solved by standard linear programming techniques. The approach can be easily modified to take other QoS attributes. For future enhancement, the work will address supporting more general types of statistical guarantees, dynamic re-binding and service sharing among multiple brokers [9].

2.8 Analysis works

These papers represent some analysis works for the QoS optimization problem. They do not necessarily propose a solution in their work.

2.8.1 QoS Analysis and Service Selection for Composite Services

In this paper, a scheme for calculating the Quality of Service is proposed. The scheme is intended to handle the composite services having complex structure. The paper defines the services as simple graph where the nodes are the services and the edges between these nodes are transitions. The paper then defines the patterns of service composition such as: parallel, conditional, sequential and loop. In the loop construct, the paper assumes only the structured loop which has a single-entry point and a single exit. The paper visits only three composition patterns: loop, parallel and sequential. The proposed algorithm takes a graph of services as input and outputs the QoS along with the execution probability for every path. The approach calculates the

quality of service for the composition pattern and the places the calculation results into a vertex. This vertex is then used in the graph to take the place of the calculation results. Detailed calculation methods along with the experiment are included in the paper [81].

2.8.2 QoS Analysis for Web Service Compositions with Complex Structures

This paper proposes a semantic approach for calculating QoS for complex structured composite web services while considering probability and conditions for each execution path. The patterns discussed in the paper are: loop, loop, sequential and conditional. The paper gives the good s ordering example as a real-life application for the composite services to describe the patterns. The composite services are represented as a graph where the vertices represent services. Formulas and description of each pattern are included in the paper. The next section contains the QoS calculation methods for each pattern. The conditional pattern is divided into two types: structured and unstructured where in the unstructured pattern, two or more paths can share the one task before joining. The graph is then passed to a function that transforms it into a rooted tree for analysis and the time complexity for this is O(—V—+—A—) where V is the number of vertices and A is the number of arcs. The paper then does some performance evaluation of the proposed approach by constructing a service selection to calculate QoS and then compare this with the service selection based on the other calculation methods. This comparison show that the proposed approach is much more effective than the previous ones due to some performance issues and due to the fact that some approaches have open issues that are covered by this approach. As a future work, the next step is to study QoS cal-

culation method for composite service with component QoS modeled as general QoS probability distribution [82].

2.8.3 Scaling the Performance and Cost for Elastic Cloud Web Services

The paper inspects the tradeoff between the consumer's costs for the resources and the gain in performance since the pricing of the resources is basically controlled by the CSPs and most of the studies consider the benefits on their side and not the user side. So, the privileges and the expenses of the customer are not fully covered. The experiment was carried out in a way to simulate the real-life example of renting multiple resources and the testing environment followed the pricing of the CSPs. The VMs analyzed were Windows Azure, Google Compute and Amazon EC2 since these are the mostly present on the market. The experiment basically uses 2 web services hosted on the VM: one demands memory and the other one demands memory and CPU utilization. The goals are to test the scaling of the performance if the resources are scaled for the same server load and the second is to find if there is a region in the server load that maximizes the performance. There is a section on the paper with the details on how the experiment was carried out. The results show that for the first web services that needs only memory, performs better for environments with VM instances with 2 or 4 CPUs by 3.6, 4.8 times than when it is hosted on a one core VM. For the second one the response time have no such increase but around 1.7 only so the performance depends on the type of the web service used. So as the result of the comparison, the more the customers pay the more performance they get. This shows that the pay-per-use model is convenient to the user. The paper concludes that the cost-performance relationship really depends on the characteristics of the web service [75].

2.8.4 A Comprehensive Infrastructure of Constraint Optimizer in Dynamic Web Service Composition

The paper presents the principles leading to constraint-based selection and preferring the best services suiting the customer and supplier demands. This will also support semantic web and its services. The paper proposes a multi-phase approach for the selection of the optimal web services that has been used. The paper introduces the Constraint optimizer that consists of Constraint Analyzer, Cost Estimation and Constraint Representation. Figure 2A in the paper shows the data flow chart which is the infrastructure of the proposed approach. OWL-S web service registries are where the web service descriptions are stored, and these reduce the discovery time and enhance efficiency and accuracy of web services operations. Web Service Search Engine is an interface that provides the information about semantic publishing and discovery of the web services with OWL-s registries. The Constraint optimizer's major function is choosing the optimal web service and satisfying the customer's requirements. The selection is done dynamically and from the set of services provided by the web service search engine. The paper also contains descriptions and tasks of the other components of the optimizer [12]. Pros: seems like a clear approach and good way to handle various parts of the service composition.

Cons: no scheme testing results or detailed implementation of the scheme is provided.

2.9 Discussion

The Quality of Service in web service composition is affected by several parameters such as Availability of the service, Service reliability, Service Response time and service cost per transaction or cost per usage. These parameters are the main or we can call generic parameters. Other factors that may affect the decision according

to the nature of the task to be accomplished by the Web service composition. These factors fall into one of the three categories of QoS composition [44] which are:

1. Generic: such as response time, cost, availability and reliability as we just mentioned.

2. Domain Specific: which depends on the nature of the service, for example in the case of video broadcasting services where there are some extra parameters such as resolution and color depth.

3. User Defined: which is a combination of both categories.

There are some cases where the user needs to take into account some of the domain specific characteristics e.g. in the case of video streaming service where the video depth and resolution matters. To provide a good solution to this multi-objective optimization problem we need to consider all these parameters along with any other parameters associated with affecting the quality of the solution.

To find the best possible solution, we need first to set some requirements for the solution so we can compare the available solutions and see which is the best one.

R1- The solution should be a real time solution: in other words, we need a high-performance algorithm that gives us the desired solution with the highest performance in terms of CPU usage and number of computations which at the end will result in shorter execution time. Genetic Algorithms are widely used in these problems and the population generation is one of the things that consumes CPU usage and therefore time. In order for the solution to be a real time solution, we want it to at least outperform GAs in terms of the time or generations needed to get the solution or converge to the optimal solution.

R2- Should give Pareto Optimal Solution set: Pareto optimal solutions set is one of the essential parts needed for any multi-objective optimization problem. It shows the tradeoff between two characteristics or properties since sometimes maximizing one property will result in minimizing others in such problems [32]. In order for the solution to express the tradeoff between the properties in the optimization problem we require the solution to be expressed as Pareto Optimal Solution Set.

R3- Considering User Defined QoS factors: Some solutions cover only partial aspects of the QoS optimization properties such as response time and cost. In fact, there are some applications that need more properties, so the approach needs to be flexible enough to handle such applications and add those functional properties into consideration. Moreover, as we discussed above, there are some other factors affecting the optimization which are domain specific requirements. These play an important role in the optimization in some applications and have a big impact on the solution. We refer to these here as non-functional properties. The solution should be able to handle both functional and non-functional QoS attributes.

R4- Considering invocation patterns: Different applications have different workflows and therefore different Web service invocation patterns. Types of the invocation patterns are sequential, parallel, conditional ...etc. The more patterns handled by the approach means the more application types the approach is suitable for. We require the solution to handle the invocation types to be capable of handling different applications.

R5- Dynamic checking and switching for better provider: As the number of Web services grow, we will have new services performing the same tasks required by

some applications that already bind to some services to perform those tasks. These new services might be better than the current ones and also (according to the user preferences) might not. Our solution should not only figure out the best service for a new application but also needs to check the feasibility to switch from one service provider to another given the information of the service we have and any SLA initiation/break fees.

We have surveyed many of the major approaches used in solving the problem and have compared twenty-five different approaches against our requirements. These approaches were based on a variety of algorithms from GA based to novel algorithms. The table below shows the approaches we surveyed and what requirements they implement:

As we can clearly see from the comparison table (Table 2.1), there are many existing solutions that implement R1 and therefore these are real time solutions, but they lack some other requirements. We also can see that none of the existing solutions implement R5 which is dynamically checking and switching to the better service provider which is an important property in optimizing the solution.

Approach	*R1*	*R2*	*R3*	*R4*	*R5*
Wanger et al.	✓	✓	✓	×	×
Ludwig et al.	✓	✓	×	×	×
Mardukhi et al.	✓	×	✓	✓	×
Zhuang et al.	×	✓	×	×	×
Liu et al.	✓	×	×	×	×
Claro et al.	✓	✓	×	×	×
Xiaolong et al.	✓	✓	×	×	×
Gao et al.	✓	✓	×	×	×
Kousalya et al.	✓	×	×	✓	×
Jiuxin et al.	✓	✓	✓	×	×
Yin et al.	✓	✓	×	×	×
Kang et al.	✓	✓	×	✓	×
Rosenberg et al.	×	×	✓	✓	×
Shanshan et al.	✓	✓	✓	×	×
El Haddad et al.	×	×	✓	✓	×
Hwang et al.	✓	×	×	×	×
Dai et al.	×	×	×	✓	×
Rosenberg et al.	✓	×	×	✓	×
Z.Liu et al.	✓	×	×	✓	×
Feng et al.	×	×	×	×	×
Pan et al.	✓	×	×	×	×
Zhou et al.	×	×	×	×	×
Li et al.	✓	×	✓	✓	×
Chen et al.	✓	×	×	×	×
Cardellini et al.	×	×	✓	✓	×

Table 2.1: Current Approaches and Requirements Implementation

CHAPTER 3: WEB SERVICE COST OPTIMIZATION

3.1 Introduction

Due to the rapid increase and spread of Web enabled computing many service providers compete to offer better services to the users. The provided quality of the offered service differs form one service user to another according to the relative importance of each feature provided by the service. To judge the service quality i.e. to assess which service may be better for a certain user in a certain task, several parameters are put to consideration. Finding the cheapest rate does not mean this service is the best service since there are other parameters judging this. For example in video streaming services, the video resolution is a primary feature of the service. In this case, having a service with a very low cost and a lower resolution than what the application needs is not an effective solution. In some cases, the user already have a service that performs the desired tasks but want some improvement to be added to the application. The improvement could be in terms of performance or operational cost. The problem here is to find another service such that it is feasible to switch from the current service provider to the new one.

Minimizing the cost to the user does not mean finding the cheapest service since most of the time the cheapest service is not the most efficient. There is usually a relation between cost and performance and in most cases the cheaper the cost leads to lower performance measures. those performance measures include Availability, Reliability, Response Time, Cost and in some cases some other domain specific measures depending on the type of application. Therefore our main problem becomes " Given a relative importance for each measure, find the best candidate service". This includes the case whether the user has no current service for the task or even if the user has a service but looking for further enhancement. In other words, is it feasible

and cost effective for the user to switch from one service provider to another service provider and if it is then which one is the best solution. Our proposed scheme uses the P-OCEA algorithm proposed in [10] to find the solution to optimize the cost in Web service composition according to a given objective function. The P-OCEA algorithm is faster in finding the solution than Genetic algorithms. The Algorithm uses the P-Optimality Criteria combined with Genetic Algorithm's operations to find the solution. P-Optimality plays the role of candidate selection for solution comparison according to the provided objective function instead of calculating a fitness value for each candidate which will enhance the performance of finding a solution to the problem.

The rest of the chapter is divided as follows:Section II contains a brief discussion of Genetic Algorithms, Section III describes P-Optimality, Section IV Describes P-OCEA. Section V discusses the derivation of our objective function derivation and how it can be customized. We discuss some related works in Section VI, Section VII shows the experiments we conducted along with experimental results and comparison with some recent approaches and finally insights and discussion in Section VIII.

3.2 Genetic Algorithms

In some cases, an apparently attractive solution may not be optimal. For example, trying to find a lower price for the service this results in lower performance hence there is a tradeoff between cost and performance. Trying to solve such problems results in having a set of solutions referred to as "Pareto-optimal solution" and its curve shows us the tradeoff between the parameters included in the problem. To find the best possible solution "the optimal solution" we need to generate as many Pareto-optimal sets as possible[20]. Evolutionary Algorithms were proposed to solve such problems. The idea behind these algorithms is to start with a set of random

solutions referred to as initial population. The algorithms use the initial population to create another new generation containing the best solutions. This population generation continues and each time the population converges more to towards the true optimal solution region. This concept is the base of Genetic Algorithms. NSGA (Non-dominated Sorting Genetic Algorithm) was one of the first evolutionary algorithms. This algorithm uses the principles of evolutionary biology to solve the above-mentioned problem. It applies techniques such as inheritance and mutation [58]. NSGA-II was implemented to solve multi-objective optimization problems as an improved version of NSGA which was mostly used for single objective optimization problems. The difference in NSGA-II is that after the random population is created and crossover/mutation phase, the individuals are ordered into non-dominated sets. These sets are then reduced to find the next non-dominated sets [49].

3.3 P-Optimality

In this section, we provide a brief definition of P-Optimality which is central to our algorithm for multi-objective optimization problem. It is defined as follows: for a value x that belong to a set of values S to be optimal over the other values in S implies that x has a higher probability to be the winner of any comparison between x and any other randomly selected member of S according to a predefined objective function. That is:

$$forx \in S \rightarrow P(x) \geq P(x*), \forall x* \in S \tag{3.1}$$

where P(x) is the probability that x has the winning criteria according to a predefined objective function f(x) and x* is any randomly selected member of the set S. So x should maximize the function:

$$\sum_{i=1}^{k} P_i(x) \tag{3.2}$$

$$\sum_{i=1}^{k}(1-(P_i(x))) \tag{3.3}$$

which is the 1-norm function of x, where k is the number of objectives. Then for x to be better than y the 1-norm function of x has to be less than the 1-norm function of y:

$$\sum_{i=1}^{k}(1-(P_i(x)))<\sum_{i=1}^{k}(1-(P_i(y))) \tag{3.4}$$

In other words, x is optimal over a set S if and only if:

$$P-function(x) \geq P-function(x*) \forall x* \in S \tag{3.5}$$

where :

$$P-function(x) = (\sum_{i=1}^{k}(1-(P_i(x)))^r)^{1/r} \tag{3.6}$$

Where, S: finite set of feasible solutions, x*: randomly selected solution, r: real number larger than zero and P(x): the probability of x winning the comparison with other randomly selected members of S according to the predefined objective function.

The Optimality Criteria is not the same as selection method as might think, but it is what allows us to compare two feasible solutions to know which solution is better. The basic characteristics of the P-optimality criteria are as follows: (1) P-Optimality allows us to assess the importance of each parameter in the problem we are trying to solve in order to get a compromise solution. (2) The criteria are defined over the set C which is the finite set of feasible solutions. (3) $P_i(x)$ is the probability that x will win the comparison against randomly chosen individuals (solutions) according to the objective function. The definition of P-Optimality is based on $P_i(x)$. (4) Probability $P_i(x)$ and the "inverse cripple" concept are relative to the considered

solutions. (5) The definition is based on equation 3. 6. (6) The optimality criteria generate interpretations and gives us what is optimal.

3.4 P-OCEA: A Multi-Objective Optimization Algorithm Based on P-Optimality and Genetic Algorithms

The algorithm which we used to get the solution for the problem according to the objective function we derived is called P-OCEA [10] which is an evolutionary multi-objective optimization algorithm based on P-Optimality and Genetic algorithms [34]. The implementation of the P-OCEA algorithm considers a posteriori approach which first begins by filling the sets of solutions and then applying the preference criteria to these sets [28]. The algorithm starts by generating random populations to begin with. Each one of these m populations contains n values having upper and lower bounds. The P-OCEA algorithm performs three major operations to each one of these m populations to create a new population. These three operations are as follows, Selection: The selection method here is called tournament selection [56] where some individuals are chosen randomly from the population and the best solution of these ones is the one that will be selected. Variation: The algorithm was implemented with flip mutation and a two point-crossover which results in creating an offspring population having size N. Elite Preservation: The operator here is the same operator of the elitist NSGA-II [20] which combines the offspring population with the old one then chooses the best solutions out of this combination. More details about each one of these operations can be found in [10].

3.5 Objective Function Derivation

To Derive a good and suitable objective function for a certain problem, we need first to consider all parameters associated with it [8]. Since the main goal of our objective function is to optimize the service cost from the user's perspective, we need

to include all parameters associated with the cost of the current service used by the user and parameters associated with the next potential service to check the feasibility of changing the service provider. These parameters are: Number of service usages per day, number of days remaining in the SLA, contract/agreement breaking fee, new contract/agreement initiation fee, current service price per usage and candidate service price per usage. So, for the current service and next service respectively we have:

$$C1 = rd * c1 * u \tag{3.7}$$

$$C2 = (rd * c2 * u) + BFee + NFee \tag{3.8}$$

where: rd= Number of remaining days, c1= cost for current service per usage, c2= cost for candidate service per usage, u= average daily service usage, BFee= contract/agreement breaking fee and NFee= New contract/agreement initiation fee. Now we calculate:

$$\Delta C = c2 - c1 \tag{3.9}$$

Finally, we need to include other service parameters in the objective function. Parameters include service availability, which is defined as " the probability at steady state that requests submitted are successfully processed" [54]. Another factor that we consider is "reliability" which refers to the probability of successful service execution [37]. Response time which refers to the time it takes for the service to process the request and send back the result to the user. Web service architecture follows "pay-per-use" pricing model where the service user needs to pay the usage cost every time the service is used.

Let vectors A1= $\{A1_1, ...A1_n\}$ and A2= $\{A2_1, ...A2_n\}$ hold the attributes of the candidate service and the current service respectively, U= $\{U_1, ..., U_n\}$ holds the user preference for each attribute and S= $\{S_1, .., S_n\}$ hold the signs indicating maximizing or minimizing certain attribute. A general function in this case is:

$$\sum_{i=1}^{n} A_i S_i \tag{3.10}$$

In our case we have two or more services to compare in each iteration and we also want to consider user preferences for the importance of each parameter. Since we are taking Δ C we need Δ for the other parameters to be included in the objective function. Therefore we define ΔA to include the difference between the attributes of A1 and A2 for each attribute instead of the attribute itself. Then our objective function for a candidate Web service x in this case becomes:

$$f(x) = \sum_{i=1}^{n} U(x)_i \Delta A(x)_i S(x)_i \tag{3.11}$$

According to this, the services that maximizes the objective function should be the best available solution. As we mentioned above in equation 3.6, P-Optimality depends on the probability that a candidate solution will be the winner when compared to any randomly chosen solution according to some objective function which, in our case, is equation 3.11. Combining our objective function with equation 3.6 yields:

$$P - function(x) = (\sum_{i=1}^{k}(1 - (P_i(\sum_{i=1}^{n} U(x)_i \Delta A(x)_i S(x)_i)))^r)^{1/r} \tag{3.12}$$

There are some cases where the user needs to take into account some of the domain specific characteristics e.g. in the case of video streaming service where the

video depth and resolution matters. The objective function we discussed earlier is designed to calculate the generic characteristics of QoS. Our proposed function has the flexibility to be customized in order to be suitable for such problems. To customize our function for the above case, we need to set user preferences for each additional parameter. Each one of these user preferences is multiplied by the difference of the corresponding parameters. Multiplication by - sign is needed in the case of minimizing the parameter where in maximizing it is left as is. In this case, append the signs for the new parameters to the vector S2, create a new vector U2 for user preferences, create vector $\Delta \mathrm{V} = \{V_1, V_2, ..., V_n\}$ and append difference of the user defined attributes into ΔV. Now we have to modify our objective function to consider these non-functional QoS attributes. The new customized function then becomes:

$$f(x) = \sum_{i=1}^{n} U(x)_i \Delta A(x)_i S(x)_i + \sum_{j=1}^{n} U2(x)_j \Delta V(x)_i S2(x)_j \tag{3.13}$$

Plugging this function back into the P-function of x we get:

$$\begin{aligned} P - function(x) = (\sum_{i=1}^{k} (1 - (P_i (\sum_{i=1}^{n} U(x)_i \Delta A(x)_i S(x)_i \\ + \sum_{j=1}^{n} U2(x)_j \Delta V(x)_i S2(x)_j))))^r)^{1/r} \end{aligned} \tag{3.14}$$

3.6 Experiments and Results

We have implemented the solution in two different ways. First, we compared our proposed scheme with a scheme using Genetic Algorithms approach with the "Roulette Wheel Selection". This will reflect the performance of the current technology in processing the solution to the problem using our objective function. In the roulette wheel selection, the algorithm gives each solution some probability of choos-

ing it and then the solutions are compared according to the fitness function which is calculated according to our proposed objective function. The fittest solution (the one with higher fitness value) is then returned to go to the next population. The other way we implemented the solution is using P-OCEA algorithm with "Tournament Selection" [10]

$$\begin{aligned} Fitness(x_A) > Fitness(x_B) \Leftrightarrow \\ p - function(x_A) < p - function(x_B) \end{aligned} \tag{3.15}$$

In our Implementation we have a predefined Web service from the set of the 100 services we used for testing. The need of this service is to denote the current service from the current service provider. We ran both solutions on the same initial population of 100 services. The solution has 200 iterations which makes 200 population generations, 0.7 crossover rate and 0.02 mutation rate. We had our elitism set to 5 which means in each generation we will take the best 5 individuals to the next generation. We ran the approaches more than 100 times to see the trend of each one in finding the solution and to get an idea of the outlier values that we get due to random number generation. In Table 3.1 we show the min, max and average of a sample run after the data was cleaned from outliers in both approaches.

Methodology	Min	Max	Average
GA	82	129	106
Wcost	79	126	104

Table 3.1: Number of Generations Needed

Table 3.2 shows that the average convergence to the optimal solution in the proposed approach is faster than the average convergence using Genetic Algorithms approach

in the 200 generations which means, in average, using the same number of generations produces a better solution and to get the same solution as the Genetic Algorithms it needs less generations which means that the proposed approach outperforms the GA approach in terms of performance. P-OCEA algorithm outperforms GA algorithm in terms of runtime. In GA, in each call to the selection procedure, the fitness calculation function will be called and that what consumes more time. In our proposed approach we have used Tournament selection where the individuals are compared based on the p-value which leads to faster execution time since no need to calculate the fitness function in each selection call. In our experiment we have compared the runtime of both approaches im milliseconds. We took a sample of runs and removed the outlier values in both approaches and placed them in a comparison table which can be seen below.

Methodology	Min	Max	Average
GA	126	153	144.25
Wcost	121	144	136

Table 3.2: Runtime Comparison

From the previous table we can clearly see the difference in the time our approach needs compared to the GA approach. Our approach's maximum time was around the average time needed by GA. In the following figure we show the comparison result of the generations needed for convergence to the optimal solution for our proposed approach compared to three more approaches along with the Genetic Algorithms approach. These three approaches are: SBA [21] which was proposed in 2007 by Nitto et al. which is an approach that is based on Genetic Algorithms , NBA [62] that was proposed in 2008 by Niu and Wang which is also based on Genetic Algorithms and SWC [43] that was proposed in 2009 by Lecue et al for semantic

composition of Web services using GA based Algorithm. The following Chart Shows the convergence to the optimal solution for each approach. The horizontal access shows the number of generations needed for convergence.

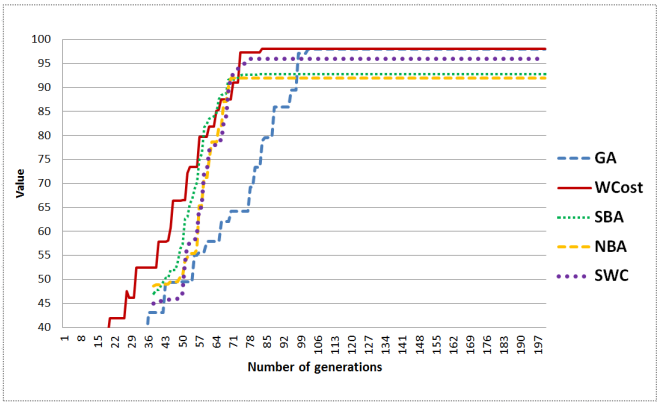

Figure 3.1: Sample Run Comparison

Therefore, by looking at the experimental results we can clearly see the enhancement in the number of generations our approach needs to find the optimal solution. It clearly shows that our approach needs less generations to converge to the optimal solution. Moreover, the value we get for the solution is higher than the values for the other approaches. That shows the efficiency of our approach in terms of solution's accuracy and time to get the solution.

3.7 Insights and Discussion

In this chapter of the dissertation we proposed Wcost, a cost-based web service composition optimization approach based on P-OCEA algorithm. The algorithm is based on Genetic algorithms but with different selection methodology where we used tournament selection based on the p-value of each service to prevent the computational overhead of calculating the fitness of each individual every time we perform the selection operation. This leads us to simple comparison to get the better individual

instead of calculation before the comparison and the theory here is simply the smaller the p-value means the higher the fitness and therefore the better the individual. The p-value of each individual is relational to the probability that this individual would win the comparison with another randomly chosen individual according to our objective function. In our objective function we have considered all the parameters that associated with the cost that will have an effect on it such as the number of service usages per day and the remaining time in the SLA. We implemented the approach and tested it to prove its efficiency and accuracy. We also implemented the problem's solution in a normal Genetic Algorithm approach to see the performance difference between the two approaches. We have proved that Wcost is faster in terms of runtime than Genetic Algorithms based approach since from the sample runs, we have shown that the average runtime of Wcost is less than the average runtime of the Genetic Algorithms based Approach. Moreover, in terms of number of generations needed to get the solution we have proved it needs less generations than what is needed by the GA approach.

To see how our proposed Approach performs compared to the other approaches, we have compared it to three other approaches in terms of the number of generations needed to converge to the optimal solution. Comparison shows that our approach outperforms the other approaches since it needs less generations to converge to the optimal solution. Moreover, it gives us higher value for the convergence. As a future extension, we think of extending Wcost to cover the optimization on the service provider's side such that the service provider can provide a pricing model such that this price will be the best price for the user according to the other QoS parameters.

CHAPTER 4: MULTI-USER WEB SERVICE OPTIMIZATION

4.1 Introduction

In recent years, a number of organizations and even single users have started using Web based services instead of workstation-based systems primarily due to the maintenance and the overall cost of deployment. This translates to huge savings as the customers do not have to own servers to run their applications, and hence can deploy large applications fairly easily. A very good example of is Pinterest which according to Forbes.com is growing steadily since its launch and is estimated to have about seventy million active users with a net worth of about five billion US dollars. The primary source of this site‘s revenue to this site is the traffic it generates, which is comparable to Facebook. The latter maintains its own servers while Pinterest depends on Amazon‘s Web services. Web services are offered by various service providers with various performance parameters and pricing schemes to satisfy the users' needs. Users look for services that perform the desired tasks, have the desired performance parameters and at a good price from the user's perspective. Due to the increasing number of services, it is not feasible to search them manually; resulting in a body of research. Existing technology in this regard usually take the user's preferences and find the best services that can satisfy the given requirements. Some requirements come from single users while others involve multiple users. In this chapter of the dissertation, we target the latter. To illustrate the problem, we consider the following scenario: an organization O is trying to find some services (likely part of a Web service composition) to perform a certain task. This organization has multiple departments that can influence the service choice through their inputs. Each department also has a different vision of the solution, and therefore has different preferences.

The problem statement then is to find the ‘best‘ services that suit the needs of the organization, and at the same time will be optimal when combining the preferences of the different departments influencing the decision. Many solutions were proposed to tackle the composition optimization problem considering different scenarios. Most of the proposed schemes are based on Genetic Algorithms to find a set of optimal solutions known as Pareto optimal solution set [20]. Within these algorithms the schemes use some selection mechanism that will select the best set of services to be the output of the algorithm. To the best of our knowledge, these solutions do not take the discussed scenario, or similar case, into account. Moreover, some solutions for the multi-user decision making produce a very large matrix that needs to be processed which results in computational overhead. The selection mechanism is one of the factors that affect the performance of the overall scheme. For this reason, we use a selection method based on P-Optimality [10] that depends on the probability of the candidate being optimal according to an objective function. We have proved that this procedure outperforms roulette wheel selection method in terms of accuracy and number of generations needed to get the optimal solution [2].

Our main contribution in this dissertation is proposing a PROMETHEE [68] based approach to tackle the multi-user Web service composition optimization. The scheme utilizes the P-Optimality theory for faster and more accurate results. The rest of the chapter is organized as follows: Section 3 presents some of the related work, the proposed approach is discussed in section 4 followed by the invocation patterns and the objective function discussion in section 5. Section 6 contains the details of the conducted experiments and finally a conclusion and some future works in section 7.

4.2 Proposed Approach

Our proposed solution considers the scenario we mentioned earlier in the introduction. Supposing that the organization is seeking service composition to stream safety awareness videos to educate its employees. Each department (denoted by D) in O needs to participate in the awareness. The organization O in this case is seeking a single service composition that will satisfy the needs of all departments to save money rather than having multiple compositions. Then the problem statement here is to find the set of services that will be the best composition choice for the combined constraints set by all departments such as price limitation and minimum resolution. We need to take into account here the difference in preferences set by each department since not all departments have the same needs from the Web service. Moreover, some of the service parameters which are considered as critical to one department might be ignored by another one. This is a source of conflict in the decision which might prevent finding a single solution that satisfies the needs of all parties included in the decision making. This conflict needs to be resolved first prior to proceeding with finding the solution. Our approach consists of two main components: Conflict resolution that will deal with conflicts between the decision makers that might prevent finding a group solution and Group decision making component which will process the decisions of all decision makers to propose a unified decision.

1. ***Conflict resolution:***In some cases, contradicting requirements may prevent finding a group solution. To avoid this problem, we need to resolve these contradictions prior to proceeding with finding the solution. For example: one department may require the video resolution of 720p at most for faster download while another department may require a 1080p minimum resolution HD purposes. Another source of conflict might not be as obvious as the previous

one but trying to maintain both will result in two disjoint solution sets. For example: one department requires the cost per invocation for the service to be at most $x while another department requires a minimum availability y where there is no service provider offering this resolution at this price or lower. We resolve this problem by assigning weights to each department in each attribute. Note that there are some different preferences that will not affect the solution which is the case when the needs of one department fall into the needs of another one. An example of this is when one department needs the resolution to be at least 720p where another department needs the resolution to be at least 1080p. The first department preference in this case will be satisfied when we satisfy the requirement of the second one, so we do not need to consider such contradictions. Let $w_{D_i}^{A_j}$ denote the weight of the department's D_i in affecting the attribute A_i. This allows us to construct the Weights matrix containing the weight of the decision on each attribute by each department as follows:

$$w_M = \begin{bmatrix} w_{D_1}^{A_1} & w_{D_1}^{A_2} & .... & w_{D_1}^{A_j} \\ w_{D_2}^{A_1} & w_{D_2}^{A_2} & .... & w_{D_2}^{A_j} \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ w_{D_i}^{A_1} & w_{D_i}^{A_2} & .... & w_{D_i}^{A_j} \end{bmatrix}$$

Then we perform a pairwise comparison based on the Analytic Hierarchy Process (AHP) [25] to decide which preference is dominant as per the algorithm shown in Algorithm 1. A threshold here is set prior to submitting the matrix to the algorithm. This threshold will be used in the comparison to determine how much of a difference we need to have for a group to dominate the others

i.e. if the difference in the comparison is more than or equal to the predefined threshold t then we consider it as a dominant group. By default, t is set to one so if the difference in the pairwise comparison is more than or equal to one, we consider that as a dominant decision. In case of multiple departments, we add the weights that fall in the same solution set and then proceed. In case there is no dominant decision, we go with the most feasible decision that will maximize the objective function and will result in better performance at lower cost. As a result of applying this procedure we might discard some of the candidate services resulting in a smaller pool of services to deal with. Algorithm 1 demonstrates our AHP based conflict resolution algorithm.

Algorithm 1: Conflict resolution

Input: User decision weights matrix M,
threshold t={1,2...10}
Output: Dominant decision.
1: for each row in M:
2: if (no contradiction): move to next row
3: if (contradiction): divide users into groups $G_1...G_n$ decisions falling in same area.
4: sum user weights within group.
5: perform Pairwise comparison and compare to threshold.
6: Return dominant group.

The preferences of the dominant group, which is the output of the conflict resolution algorithm, will be used to discard the services that do not satisfy the preferences and then generate the selection pool that contains the remaining

services. The original population then will be replaced by the new generated population.

2. ***Group Decision:*** After resolving the contradictions, we generate one set of solutions that contains the optimal solution from each user's perspective. So, let $R = NumberOfDepartments$ and $M = NumberOfCandidates$. Each department will set some weight to each QoS attribute such that

$$\sum_{i=1}^{n} w_i^r = 1 \tag{4.1}$$

To analyze the combined preference of each QoS attribute, our approach uses an evaluation matrix of size $R \times M$ to get a group decision. The analysis of the matrix entries depends on some predefined preference function (objective function). Let

$$\begin{gathered} P_j(a,b) = G_j[f_j(a) - f_j(b)] \\ \text{where } P_j(a,b) \text{ is between 0 and 1} \end{gathered} \tag{4.2}$$

be the preference function associated with the criterion f_j where G_j is a non-decreasing function of the deviation between $f_j(a)$ and $f_j(b)$. Thus, when we want to maximize the criterion f_j we will have the following:

$$\begin{aligned} G_i[f_i(a) - f_i(b)] = 0 &\Rightarrow \text{no preference} \\ G_i[f_i(a) - f_i(b)] \sim 0 &\Rightarrow \text{weak preference} \\ G_i[f_i(a) - f_i(b)] \sim 1 &\Rightarrow \text{strong preference} \\ G_i[f_i(a) - f_i(b)] = 1 &\Rightarrow \text{strict preference} \end{aligned} \tag{4.3}$$

Now we can calculate the weakness and strength of each candidate compared to other candidates from each department's perspective as follows:

$$\begin{aligned} \Phi^{+r}(a) &= \sum_{x \in A} \sum_{j=1}^{k} P_j(a,x) w_j^r \\ \Phi^{-r}(a) &= \sum_{x \in A} \sum_{j=1}^{k} P_j(x,a) w_j^r \\ \Phi^{r}(a) &= \Phi^{+r}(a) - \Phi^{-r}(a) \end{aligned} \tag{4.4}$$

This applies $\forall a \in A$ where A is the set of alternatives or candidates, $\Phi^{+r}(a)$ is the power of a compared to other candidates, $\Phi^{-r}(a)$ is the weakness of a compared to the rest of candidates and $\Phi^{r}(a)$ is called "the net flow alternative" of the decision maker where a higher the value of $\Phi^{r}(a)$ is better.

From these net flow values, we can get the evaluation matrix to get the best candidate that suits all departments. At the end of each individual evaluation stage we get the value $\Phi^{r}(a)$. Hence, if we have R decision makers then we have:

$$\Phi^{r}(a_i), \text{ i= 1,...n and r= 1,...R} \tag{4.5}$$

Then we have:

$$\begin{aligned} \phi^{r}(a_i) &= \sum_{j=1}^{k} \phi_j^r(a_i) w_j \\ &\text{given that} \\ \phi_j^r(a_i) &= \sum_{x \in A} (P_j(a,x) - P_j(x,a)) \end{aligned} \tag{4.6}$$

where $\phi_j^r(a_i)$ is the net flow of a single criterion considered by a decision maker. This net flow corresponds to the relative importance of a criterion to a decision

maker. The higher the net flow value means the better the candidate to suit the decision maker's needs. The evaluation matrix then will be

$$M = \begin{bmatrix} \phi_1^r(a_1) & \phi_1^r(a_2) & .... & \phi_1^r(a_M) \\ \phi_2^r(a_1) & \phi_2^r(a_2) & .... & \phi_2^r(a_M) \\ ..... & & & \\ ..... & & & \\ \phi_R^r(a_1) & \phi_R^r(a_2) & .... & \phi_R^r(a_M) \end{bmatrix}$$

We can get a global solution then by computing the weighted sum of the individual net flow values where the global net flow for the whole decision makers group on a certain candidate will be:

$$\Phi^a(a_i) = \sum_{r=1}^{R} \Phi^r(a_i) w_r \tag{4.7}$$

and the candidate with the highest value will then be the best candidate for the group decision.

Note that in the world, this will be a huge matrix due to the number of candidate services, and the matrix computation will be complex. Thus, we need to reduce the matrix size to decrease this computational overhead. We cannot decrease the number of departments, but what we can do is reduce the number of candidates by getting the top Y candidates for each department's preferences and generate a new population of candidates that will be used to generate the evaluation matrix. We employ a Genetic algorithm to extract the top Y candidates and integrate the P-Optimality theory in our selection mechanism (instead of roulette wheel selection) for finding the optimal solution [2].

The algorithm divides the problem into user blocks such that each user block will output a set of Y candidates that will be evaluated. P-Optimality states that:

$$forx \in S \rightarrow P(x) \geq P(x*), \forall x* \in S \tag{4.8}$$

where $P(x)$ is the probability of x winning the comparison according to a predefined function $f(x)$ and $x*$ is a randomly selected value belonging to the set S. In this case, x should maximize the 1-norm of x which is the following function:

$$\sum_{i=1}^{k}(1-(P_i(x))) \tag{4.9}$$

Hence for x_1 to be a better candidate than x_2 the following condition needs to hold:

$$\sum_{i=1}^{k}(1-(P_i(x)))<\sum_{i=1}^{k}(1-(P_i(y))) \tag{4.10}$$

Therefore, for x to be optimal over the set S it has to satisfy the following:

$$P-function(x) \geq P-function(x*) \forall x* \in S \tag{4.11}$$

where :

$$P-function(x) = (\sum_{i=1}^{k}(1-(P_i(x)))^r)^{1/r} \tag{4.12}$$

where, S: finite set of feasible solutions, x*: randomly selected solution, r: positive real number and P(x): the probability of x winning the comparison with x* according to a predefined objective function. This will reduce the matrix size from $N \times M$ to $N \times Y$ where Y can be increased if no common candidate

is present among all user block outputs. The resulting matrix is smaller than the original matrix since $Y \ll M$. The new matrix in this case will be:

$$M = \begin{bmatrix} \phi_1^r(a_1) & \phi_1^r(a_2) & .... & \phi_1^r(a_Y) \\ \phi_2^r(a_1) & \phi_2^r(a_2) & .... & \phi_2^r(a_Y) \\ ..... & & & \\ ..... & & & \\ \phi_R^r(a_1) & \phi_R^r(a_2) & .... & \phi_R^r(a_Y) \end{bmatrix}$$

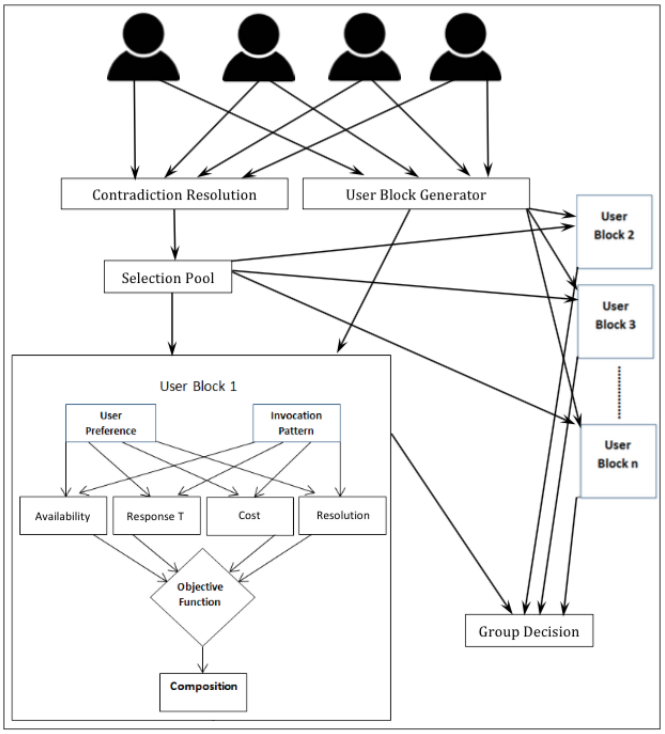

Figure 4.1: Proposed Algorithm Flow

Figure 4.1 illustrates the flow of our proposed approach. The users submit their requirements to the contradiction resolution algorithm and the user generator

shown in Algorithm 1. The contradiction resolution algorithm then resolves the contradiction and discards the candidates that will not satisfy the requirements. This then results in a new population of candidate services that will be the selection pool which will be used in the next steps. The user block generator on the other hand will use the preferences of each user to create a corresponding user block. Each user block will generate an objective function depending on the user preferences and the composition pattern used for the composition. The Objective function will then be used by the our proposed MWC algorithm to get the top candidates for the corresponding user. When we get the top candidates for all users from the user blocks, we then generate the evaluation matrix that will be processed to get the best candidate for the group of the users.

4.3 Invocation Patterns and Objective Function

Due to the variety of tasks to be accomplished using Web service composition and the nature of the procedures required to get the task done we need to use different composition patters for different tasks. Invocation patterns include Sequential invocation, Parallel invocation, Loop invocation and conditional invocation [53]. These patterns affect the calculation of the QoS parameters and hence affect the objective function that will be used to find the solution, so they have to be included. These composition patterns can be seen in the following figure:

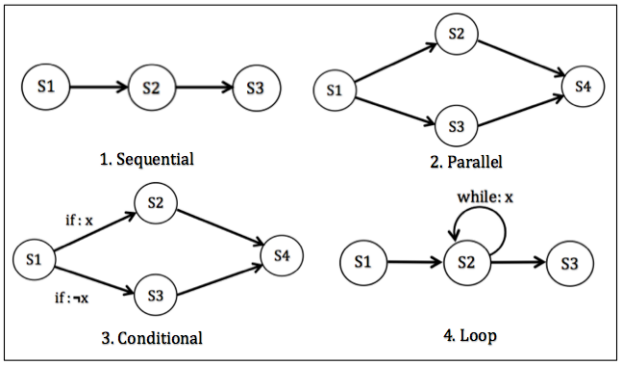

Figure 4.2: Web Service Composition Patterns

- Sequential Invocation: In this pattern the Web services are invoked one after the other in a certain sequence. The response time of the composition in this case will be the sum of the response times of all services in the composition as well as the cost since all services in the sequence will be invoked to get the solution. The availability on the other hand will be the product of availability of the services to be composed as well as the reliability which will be the product of the reliability of the composed services[47].

- Parallel Invocation: In this pattern we have two or more services invoked at the same time and their output is the input of the next service in the composition.

  As can be seen from Figure 4.2, S4 will not be invoked until both S2 and S3 are invoked successfully which means the response time of the parallel invocation will be dependent on the highest response time i.e. the slowest service will delay the whole composition even if the services that works in parallel with it is fast since S4 needs both results from S2 and S3 as its input. The availability of the composition however will be affected by the minimum availability of the services in the composition as well as the reliability while the cost will be the sum of costs for all services to be composed since all services will be invoked.

- Conditional Invocation: In Conditional Web service invocation the service will be invoked if a certain condition holds, otherwise the service will not be invoked. As we can see in Figure 4.2, S2 will be invoked only if condition x holds, otherwise S3 will be the invoked service. In this case, the QoS attributes will be subject to the probability that the services S will be invoked which means the probability that condition x holds. Other services such as S1 and S4 will be invoked for sure so the probability of invocation will be 1 while in this case S2 and S3 will depend on the probability of condition x holding or not.

- Loop Invocation: In Loop invocation pattern the service will be invoked several times as long as certain condition/s hold. Since we will keep invoking the service for k times then the response time will be multiplied by k as well as the cost of the composition. The availability however will be multiplied by itself k times as well as the reliability of the composition [53]. According to the above figure, we have a probability p for service invocation such that $p(S2) = 1 - p(S3)$ and $p(S1) = p(S4) = 1$ since they will be invoked for sure.

The following table presents the aggregate functions for the QoS attributes needed for the objective function for the discussed composition patterns. These are used to compose the objective function for each composition depending on the pattern and are preceded with a + sign for maximizing the attribute or a − sign for attribute minimization.

4.4 Experiments

In order to test the accuracy and performance of our proposed approach we have set our scenario as follows: we have set the number of departments to five departments. Each department has its own preferences in terms of Availability, Reliability, Response Time and Resolution. We have also considered the sequential Web

Pattern/Attribute	Availability	Response time	Cost/invocation	Resolution
Sequential	$\prod_{i=1}^{n} Av(S_i)$	$\sum_{i=1}^{n} RT(S_i)$	$\sum_{i=1}^{n} C(S_i)$	$R(S_i)$
Parallel	$Min_{Av}(S_1, S_2, ..., S_n)$	$Max_{RT}(S_1, S_2, ..., S_n)$	$\sum_{i=1}^{n} C(S_i)$	$Min_R(S_1, S_2, ..., S_n)$
Conditional	$p(S_i) \prod_{i=1}^{n} Av(S_i)$	$p(S_i) \sum_{i=1}^{n} RT(S_i)$	$p(S_i) \sum_{i=1}^{n} C(S_i)$	$p(S_i) Min_R(S_1, S_2, ..., S_n)$
Loop	$(Av(S))^k$	$k \times RT(S_i)$	$k \times C(S_i)$	$R(S_i)$

Table 4.1: Aggregate Functions

service composition pattern for easiness of the scenario and due to the fact that other patterns can be reduced to the sequential composition form [4]. Thus, we have the following objective function as per Table 4.1:

$$Max(\prod_{i=1}^{n} Av(S_i) - \sum_{i=1}^{n} RT(S_i) - \sum_{i=1}^{n} C(S_i) + Min(R(S_i))) \tag{4.13}$$

We then performed our experiment starting with the contradiction resolution algorithm to get rid of any contradiction that prevents finding a group decision such as when two departments have totally contradicting preferences. We used a set of one hundred services to find the optimal solution for each user. We ran the conflict resolution algorithm several times with different numbers of contradictions to verify the runtime of the algorithm. We ran our experiments on a Mac computer with 2.5 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 memory. Figure 4.3 shows the runtime for the conflict resolution algorithm:

As can be seen from the graph in Figure 4.3, we have almost a linear runtime complexity. Moreover, the runtime is not increasing when we increase the number of conditions as the algorithm will generate a new selection pool of candidates then it is actually affected by the number of the candidates in the new pool. The results show that increasing the number of conflicts will result in a smaller or equal population therefore smaller or equal runtime.

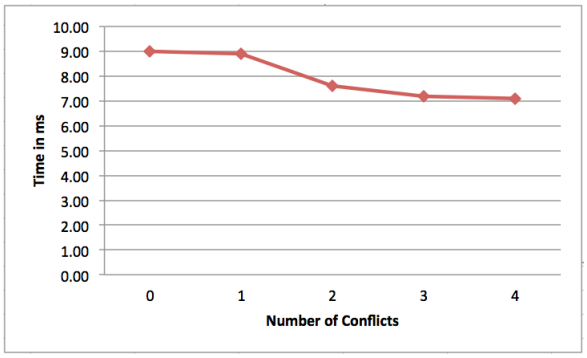

Figure 4.3: Conflict Resolution Average Runtimes

Our setting finds the five best services for each user to reduce the matrix size from $D \times N$ to $D \times 5$ where D is the number of departments and N is the number of services. We have used our algorithm that is based on P-Optimality which as we have shown is more accurate and faster that other algorithms in finding the optimal solution. Upon acquiring the top candidates from each user block, we then used our PROMETHEE based procedure to find the group decision. First, we need to set the preference Matrix of the departments which is as follows:

$$P_M = \begin{bmatrix} P_{D_1}^{a_1} & P_{D_1}^{a_2} & .. & P_{D_1}^{a_n} \\ P_{D_2}^{a_1} & P_{D_2}^{a_2} & .. & P_{D_2}^{a_n} \\ ..... & & & \\ ..... & & & \\ P_{D_m}^{a_1} & P_{D_m}^{a_2} & .. & P_{D_m}^{a_n} \end{bmatrix}$$

where $P_{D_i}^{a_j}$ corresponds to the weight of attribute a_j from the perspective of the department D_i. Accordingly, we set our preference matrix as follows:

$$P_M = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 2 & 0 & 1 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 3 \\ 2 & 1 & 3 & 2 \end{bmatrix}$$

The numbers here are set by the decision makers. The decision makers here do not have to worry about an upper bound or a scale for the numbers as they will normalized to reflect the relative importance of each of the preferences. The normalization is performer row by row on the matrix as each row contains the preferences of a certain decision maker hence the scale of one decision maker will not affect others. After normalizing the above matrix, we get N_{DM} as follows:

$$N_{P_M} = \begin{bmatrix} 0.2 & 0.2 & 0.4 & 0.2 \\ 0.4 & 0 & 0.2 & 0.4 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.166 & 0.333 & 0 & 0.5 \\ 0.25 & 0.125 & 0.375 & 0.25 \end{bmatrix}$$

Accordingly, the decision making matrix D_M becomes:

$$D_M = \begin{bmatrix} S58 & S91 & S10 & S02 & S95 \\ S10 & S91 & S02 & S58 & S95 \\ S58 & S10 & S91 & S02 & S95 \\ S58 & S91 & S10 & S95 & S02 \\ S10 & S58 & S91 & S95 & S02 \end{bmatrix}$$

The above matrix (obtained from the generated user blocks) shows the top five common services from the participating decision makers perspectives. We have used MWC [2] for the extraction of these candidates which is based on P-Optimality for the service selection mechanism providing higher accuracy with lower number of generations compared to using roulette wheel selection as per Genetic Algorithms. Figure 4.4 that compares our approach to other approaches that tackle the same problem [2] The figure shows how our approach reaches the optimal solution area before GA based approaches. Moreover, it shows more accuracy than the other approaches.

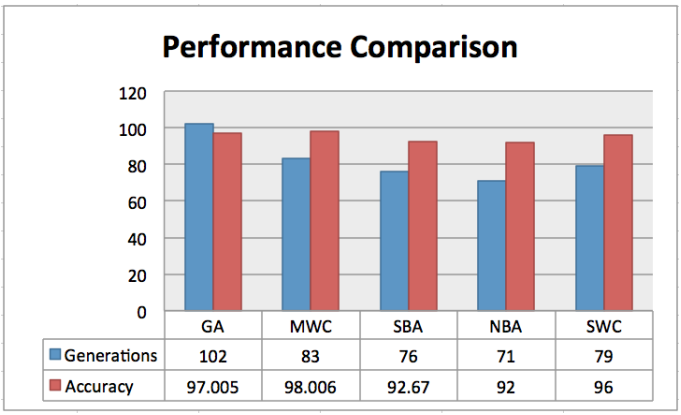

Figure 4.4: MWC Vs other schemes performance comparison

We performed a manual calculation as a test to get the group decision and to compare it with the solution we get from running the proposed approach. Each row in the above matrix corresponds to one of the departments as the first row corresponds to department D1, second row represents D2 and so on. From the above matrix we can see that S58 is the top candidate for department D1. We can also see that it is the best candidate according to departments D3 and D4 although it is the second-best option for department D5 and the fourth best option for department D2. Similarly, the matrix shows that S10 is the best candidate according to the preferences

of department D2 and department D5, where it is the second option for department D3 and the third option for departments D1 and D4. Following the same inference logic, we can clearly see that the Top candidate for the group decision is S58 followed by S10 then S02, S91 and S95 respectively. We then performed the computation on the matrix D_M to get the evaluation matrix M that gives us the best service for all departments as follows:

$$M = \begin{bmatrix} 0.40 & -0.10 & -0.30 & -0.40 & 0.40 \\ -0.10 & 0.10 & 0.00 & -0.20 & 0.20 \\ 0.35 & 0.30 & -0.20 & -0.60 & 0.15 \\ 0.3125 & 0.00 & -0.375 & -0.25 & 0.3125 \\ 0.4167 & 0.500 & -0.5833 & -0.1667 & -0.1667 \end{bmatrix}$$

After global evaluation we get the following NetFlow values for the candidate Web services:

Candidate Service	NetFlow
S58	0.3698
S10	0.1750
S02	-0.3646
S91	0.1740
S95	-0.3542

Based on PROMETHEE Group decision[68], the candidate with the highest NetFlow value should be the best group solution. This implies that service S58 is the best candidate for the composition considering the case scenario we set prior to the experiment followed by S10 then S02, S91 and S95 respectively which matches the manual calculation we performed. This shows the accuracy of the proposed approach.

4.5 Insights and Discussion

We proposed a novel Web service composition approach to tackle multi-user Web service composition optimization problem. Our proposed approach uses the concepts of P-Optimality for accuracy and efficiency and is based on AHP [25] to remove any constraints contradiction among users and the group decision extraction mechanism is based on PROMETHEE decision support [68]. We performed some experiments that show the feasibility of our proposed approach and to show how it can get a group decision that will be the optimal solution for the problem. Our experiments show the accuracy and the performance of our proposed approach. As a future work we are considering the case when we cannot find a unified solution and how we can solve the problem providing the lowest number of compositions needed.

CHAPTER 5: PROFIT MAXIMIZATION IN LONG-TERM E-SERVICE AGREEMENTS

5.1 Introduction

Web services are becoming the technology of choice for conducting business [57] due to their ease in integration and cost of operation and maintenance. Many companies provide the services to the users with different performance parameters, usually at different prices. This results in a competition among the service providers to attract more users and therefore make more profit. To do so, the service provider needs to set the best price possible that will maximize the revenue while attracting users. This cannot be done by simply offering the services at the lowest market rate. Doing so will attract the users but will not guarantee the desired maximum profit to the service provider since this rate might actually be less than or equal to the operation and maintenance costs.

One problem that a service provider might encounter in trying to distribute the resources is that some users may not be willing to pay the price, that is set for the service, therefore we need to adjust the price to attract these users. To address this problem, we use dynamic pricing where prices are adjusted based on the value the customers attribute to a product [67]. Some research works refer to this as flexible pricing or customized pricing [60] and some call it price discrimination as different users will get different rates according to how much those users are willing to pay [76]. Another Approach that helps attracting more users is to bundle the services at a rate that is lower than the total price of these services. This will not only attract users, but it will help the service provider in selling more resources, specifically the ones that have the least demand. This is important to the service provider since these resources are already running so the service provider is already paying for the

running and maintenance costs. The problem here is what resources to bundle and at what rate? To solve this problem, we use Instance Based Risk Assessment and a Kernel Regression Model to assess the risk of the bundle not having a good demand.

The service provider may also face a favorable scenario where it gets an offer from one or more users requesting to use certain resources that are currently being utilized by other users For example in the case of storage services, if a new user requests an amount of storage space, the available amount that is not being used and can be offered by the service provider is less than the one requested by the user. The new users could bring more profit to the service provider as they can offer to pay more for the resources. This case may require breaking an agreement with one or more existing users which will have some consequences that the service provider needs to take into account in such case such as the service level agreement breaking fee which the provider will need pay to the user. other things apply to this problem as we will discuss later in the chapter.

In this dissertation we propose a scheme that handles the above-mentioned case. Specifically: proposing a price that maximizes the profit for some resource (in our case Web service resource usage) depending on the market demand for such a resource. Distributing all the available resources on the market by utilizing the concepts of economic equilibrium that are based on Nash equilibrium [46]. Proposing a service bundling scheme based on instance-based risk assessment and kernel regression, and finally proposing an automated negotiation mechanism to decide if it is feasible to exchange one or more users with a new user that is requesting to utilize some of the resources. The rest of the chapter is divided as follows: Section 3 presents some of the related works, Section 4 presents our proposed approach and in Section 5 we discuss

the experiments we performed to test the approach's feasibility. Finally, we conclude in Section 6.

5.2 Related Works

The literature has a number of works negotiating pricing of information goods. For instance, Wei-Lun Chang and Soe-Tsyr Yuan [11] proposed an approach for pricing the information goods in which the users or customers are an active part in the decision since the prices are adopted to meet the changing needs of the users. This collaborative pricing model is based on the concepts of bundling and user discrimination according to how much a certain user is willing to pay. Y Narahari et. al. [60] discussed the dynamic pricing models for electronic business surveying the different models that have been used in dynamic pricing such as inventory-based models, Data driven models and other models. B. Edelman et. al [22] investigated the generalized second price that is used by the search engines to sell online advertising encountered daily by the internet users. A decentralized information pricing model was presented by A. Polanski [66] where he found out that the price is intimately related to the existence of cycles in the network where the price is zero if the cycle covers the trading pair and is proportional to the direct and indirect utility generated by the goods otherwise. Similarly, C. Maina [51] examined whether the price contributes to the emerging information. He also divided and identified the existing gaps in the economics of information research.

Halliday [31] values information goods based on the availability of the information goods and the number of consumers that are willing to pay for them (supply and demand). Rowley [71] describes the price as the dominant force in resource allocation that determines supply, demand and income among the consumers and the providers. Sharma et al. [73] proposed a financial model to provide a high QoS to the consumers

employing the financial option theory and treating the cloud resources as assets to determine their realistic values. There was a shortcoming to their approach which is not taking maintenance costs into consideration. Wang et al. [78] proposed an algorithmic solution to optimize the net profit where they developed two algorithms for the net profit optimization. Macias [50] proposed a model utilizing genetic algorithms for pricing cloud resources which they were able to prove its efficiency in acquiring the highest revenue. [76] Discussed pricing policies for information goods where he proved that bundling products increases the profit via reducing the heterogeneity of the customers. Bakos et. al. [6] investigated the effect of bundling on the competition where they concluded that large bundles may provide significant advantages in competition. Altinkemer and Jaisingh [5] have devised a mathematical model for the bundle and price determination.

Estimating the demand on certain goods is one of the crucial steps prior to offering those goods to the market. Many approaches to estimate the demand have been proposed and in different fields since each field of marketing has different factors that affect the demand for certain goods. D. Besanko et.al. [7] proposed a demand estimation framework for competitive pricing in 1998. Their approach estimates the value created by a brand which looks for the willingness of the consumers to pay certain prices for certain brands. A Bayesian approach for the workload and demand estimation was then proposed in 2004 by Pena et. al [65] which estimates the workload on the Web farms. These Web farms include Web services and estimating the workload on a service will help determine the number of users that are willing to pay a certain price to use a service. Aksoy et. al. [1] proposed another Bayesian approach for the demand estimation in 2015 and that approach was proved to give better estimates than previous ones.

Multi-attribute optimization problems solving is an evolving area of interest in many fields such as computer science and others. Evolutionary algorithms are widely used to help solve such problems [17]. These algorithms can help the user find the best service composition for certain tasks which involves having a Service Level Agreement between the user and the service provider. This brings the need to negotiate the SLA terms and hence the need of some automated negotiation approaches which is a complex problem in terms of providing the feasible solution and the time it needs to get such one [33]. Many efforts were employed to enhance the negotiation via Genetic Algorithms such as Matos et.al [55] who presented an approach in which strategies and tactics correspond to the genetic material in a Genetic Algorithm. Others also proposed many approaches, but they lack a crucial requirement and that is the support for dynamic selection of the decision-making models [33]. Since Negotiation will include selection then we also have concern for the selection method to be used and the effectiveness of it. Many selection methods are available such as tournament selection, ranking selection and roulette wheel selection. EC Jara [38] proposed an approach based on P-optimality that uses tournament selection and gave very good results. We have Used tournament selection in a previous work and compared it with roulette wheel selection and showed increased accuracy and performance [2]. Chhetri et al. [14] presented a multi-agent framework that uses local QoS are coordinated to satisfy the global QoS constraints for the composition. These local constraints have to be inferred from the global ones using the workflow topology. Comuzzi [18] proposed a model that utilizes a broker that carries out the one-to-one negotiations on behalf of the consumers and providers. An approach using multi-agent system paradigm along with Web service technology to perform online bargaining was presented by Ncho in [61]. Oliver utilized Genetic algorithms to calculate the fittest strategy based on the negotiation outcome runs in [63].

5.3 Proposed Scheme

The proposed approach is divided into four main parts. In the first two sections we use simple methodologies to set the price which will be a base for the main contribution which is bundling services. In the first part, we calculate and provide the pricing model that will guarantee the maximum revenue, based on the market demand for certain performance characteristics provided by the Web services. The second part utilizes the concept of Nash Equilibrium for resource distribution among the services to implement a dynamic pricing model which will guarantee the distribution of all available resources, and hence a higher profit. The third part of the approach is a bundling scheme where for each bundle we calculate the feasibility of bundling some resources and offering them to the users by utilizing an instance-based risk assessment and kernel regression. The last part helps in deciding if and when to break an SLA with some users if there are new requests and negotiates the price and other SLA related fees (such as initiation and/or breaking fees) with the potential users.

Figure 5.1 shows the flow of the proposed scheme which starts by setting an initial price for the service. The next step is to offer the service to the users at the initial price. We then check if all resources are in use or not. If not, we go to the dynamic pricing and bundling phase and offer the services again, at the new price. This process repeats till all or maximum resources are in use. Then the approach can view other user offers and negotiate the price with them if it is feasible to exchange the new user with one or more existing users.

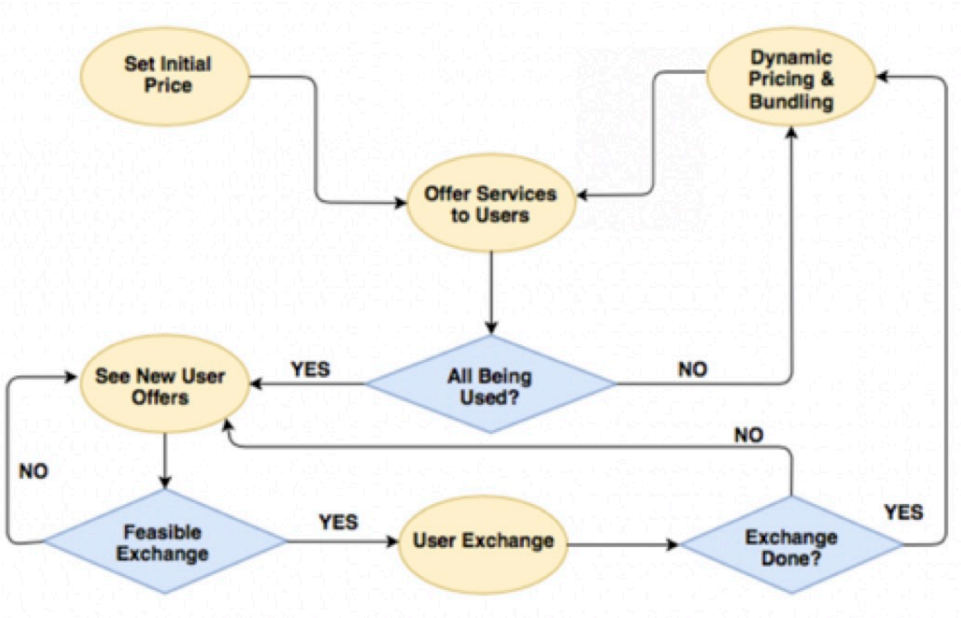

Figure 5.1: Proposed Scheme Flow

5.3.1 Pricing Model - Setting Initial Price

In elementary economics, revenue is defined as the price of the unit times the quantity sold, minus the original costs. In the case of Web services, this maps to the price per service usage, the number of usages and the operational costs that the service provider has to pay in order to operate the services (servers, production...etc.). A higher price may correspond to lower demand as users will look to optimize their costs too. On the other hand, higher proposed prices correspond to higher profits for the service provider. This is a standard demand-supply problem, where revenue as R, price $= P$, usages $= U$ and operational costs, then:

$$R = (P * U) - C \tag{5.1}$$

Moreover, increasing the price by a variable ax will lower the demand by bx where a and b are integers and x is the increment coefficient which is a real number larger

than zero. Equation 5.1 thus becomes:

$$R = ((P + ax)(U - bx)) - C \tag{5.2}$$

We want to maximize the revenue so to get the highest value of the variable x that will maximize the value of R we need to first derive equation 5.2 to find its curve and then look for the value x in which the slope is zero so it is at the top of the curve or R'. Expanding equation 5.2 will give us:

$$R = PU - bPx + aUx - abx^2 - C \tag{5.3}$$

The derivative of equation 5.3 with respect to x is then:

$$R' = 2abx - bP + aU \tag{5.4}$$

In order to get the value of the variable x that will maximize R', we need to look for slope zero in the equation which will correspond to the peak of the curve as shown in Figure 5.2. Solving for x by setting equation 5.4 to zero will give us the price increase that guarantees the maximum revenue. We call this the Highest Possible Price (HPP).

Demand Estimation

The demand is one of the main factors to set the initial price as it will provide the values of a and b. We rely on server log data to help provide information of the demand on a certain service. We then apply the basic Bayesian theorem to estimate the probability of a certain demand given certain log file history. Bayes' theorem is

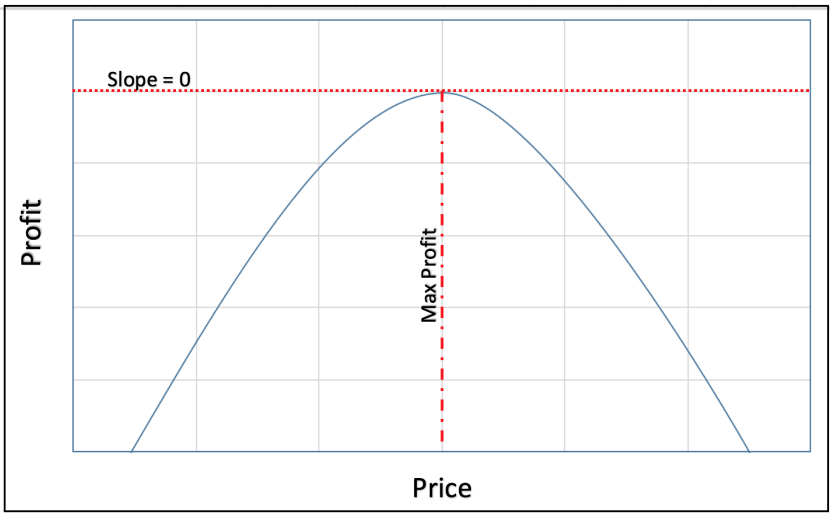

Figure 5.2: Revenue vs price increase

expressed as follows

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.5}$$

That is the probability of an event A happening given another event B happened is equal to the probability of event B happening given that event A happened times the probability of event A all over the probability of event B. In our case this translates to the following [1]:

$$p(D|x) = P(D_i|x_1, x_2, x_3, ...x_n) = \frac{P(x_n|D_i, x_1, ..., x_{n-1})P(D_i|x_1, ..., x_{n-1})}{\int_{m=1}^{I} P(x_n|D_m, x_1, ...x_{n-1})P(D_m|x_1, ..., x_{n-1})dD} \tag{5.6}$$

where $p(D|x)$ refers to the probability of having a demand D given that the price changes by x. Estimating this demand will help getting the values of a and b in equation 5.2 as the demand will translate to the number of usages such that $(D|x)$ equals to the demand (number of usages) where $a = 1$ then we solve for b in equation 5.2 to get its value.

5.3.2 Dynamic Pricing for Ultimate Resource Distribution

After setting the HPP, the service provider offers the resources to the consumers. It is not necessary that all resources are booked at the respective HPPs, so the next step may alter it a little bit, so we can use all the resources and generate more profit. The idea here is that not all users are willing to pay the assigned HPP for the provided services. This implies that we should check how much the users are willing to pay and alter the prices accordingly. The approach is to divide and distribute the available resources for the users based on the concept of Nash Equilibrium to get the best resource utilization. Consider we have n users and m resources to be distributed among these users. Initially we grant each user some amount $g_i \geq 0$ and an increasing utility function f. Nash theory states that there exists some unique allocation for the resources y on each user i that will maximize the user satisfaction in terms of spending and budget constraints $py \leq pg_i$. In this case, given the price that we generate from the first step as a vector p and the user i then the resource allocation for i will be:

$$\begin{aligned} S_i(p) &= (S_{i1}(p), S_{i2}(p), ...S_{in}(p) \\ &= max f_i(y) \text{ , such that: } py \leq pg_i \end{aligned} \tag{5.7}$$

Then for each resource or service j let us define the total demand function of j as:

$$D_j = \sum_{i=0}^{n} S_{ij}(p) \tag{5.8}$$

Then we extract a pricing vector such that all resources will be utilized (market equilibrium). This vector will satisfy: p is greater than or equal to zero, $D(p) - \sum g_i \leq 0$ and $p(D(p) - \sum g_i) = 0$ [46]. The Price curve then will be as shown in Figure 5.3.

Figure 5.3 shows the Dynamic price change which guarantees the maximum profit

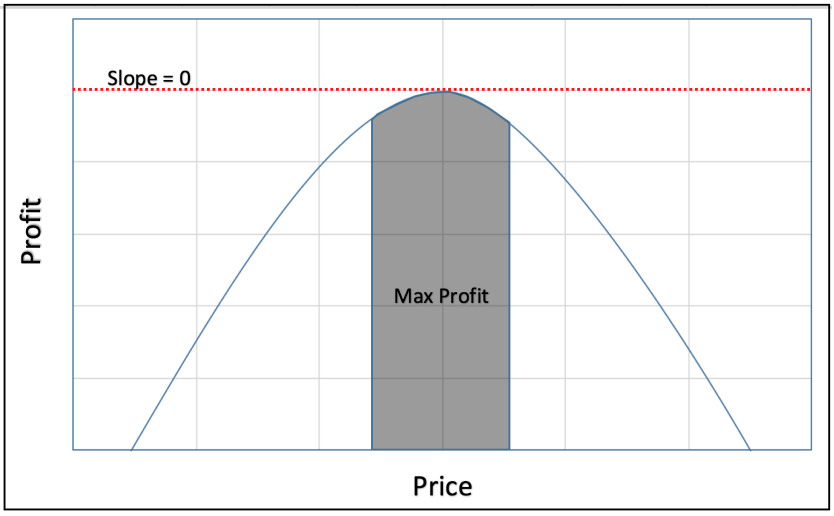

Figure 5.3: Revenue vs price increase

and the distribution of all available resources. The shaded area denotes the range we can go in price change such that we will be able to distribute the available resources. This area is directly affected by the number of users that are willing to pay to use the resources provided by the service provider.

A negotiation process might be needed to be performed between the service provider and the customers to determine the price for the service. We use an automated negotiation based on WebNeg [33] which is a Genetic Algorithm- based scheme that has been proven to outperform other similar schemes. We use Tournament selection [29] which pairs the candidates and extracts the best candidate out of them according to the probability that this candidate is the optimal solution. We employ the concepts of P-Optimality [38] which translates to the following: "for a value x that belong to a set of values S to be optimal over the other values in S implies that x has a higher probability to be the winner of any comparison between x and any other randomly selected member of S according to a predefined objective function"

[2]. This is represented as

$$forx \in S \rightarrow P(x) \geq P(x*), \forall x* \in S \tag{5.9}$$

This, as we have proved in our previous work [2] minimizes the calculation as we will not need to calculate the fitness value for each candidate when we perform the selection mechanism. Instead, we rely on the p-function [38] as follows, x is optimal over a set S if and only if:

$$P - function(x) \geq P - function(x*) \forall x* \in S \tag{5.10}$$

where :

$$P - function(x) = (\sum_{i=1}^{k} (1 - (P_i(x)))^r)^{1/r}) \tag{5.11}$$

where, S: finite set of feasible solutions, x*: randomly selected solution, r: real number larger than zero and P(x): the probability of x winning the comparison with other randomly selected members of S according to the predefined objective function [2].

5.3.3 Service Bundling based on Instance Based Risk Assessment and Kernel Regression

Some goods tend to be more popular and people buy them more often where some others are not. In the case of the service provider, the services that are not sold have a remaining cost since the service provider has to keep them operational, and this cost is totally paid by the service provider, whether it is being used by a user or not. To maximize the profit of the service provider, these services need to be used by users; but what if the users are not interested in such a service? The solution here is to "sell as a package", in our approach we perform service bundling based

on the assessment of the risk of creating a bundle that will have low or no demand. The assessment is based on examining similar bundles in the marketplace in the case of Pure Bundling which refers to selling the items as bundles and not offering the services individually (instance-based method). We define S_j as the selling rate in which j ranges from 1 to n so we will be able to predict the demand D_i of a bundle B_i via the use of the weighted average of the selling rates of the previous bundles as follows:

$$D_i = \sum_{j=1}^{n} w_{ij} S_j \tag{5.12}$$

w_{ij} here represents the weight of bundle B_j for predicting the demand of a bundle B_i in a way that is similar to voting schemes where each vote has a different weight assigned to it. In order to predict the demand of the new bundle we can use the weighted average but the problem here is that this average cannot be taken directly since the sell rates are not quantified so we need to quantify these values as the weighted variance of the votes as follows:

$$v_i = \sum_{j=1}^{n} w_{ij} (S_j - D_i)^2 \tag{5.13}$$

Higher variation of the votes indicates a higher risk of a low demand for the bundle B_i. The distance between two instances is primarily tied to the probability of default (in our case the default refers to the bundle selling with a good demand) of these instances. We define the distance between the bundles B_i and B_j as:

$$d_{ij} = |b_i - b_j| \tag{5.14}$$

Where b_i and b_j are the probabilities of default for bundles B_i and B_j. The more distance between the instances implies less weight to the vote.

Determining the weights is a very critical step and we use Kernel Regression for this step because although the naive weighting schemes are easy to implement, they are not optimized to provide the most accurate prediction [30]. Kernel Regression is a statistical method that aims at finding the non-linear relation among a pair of random variables [59]. It states that, supposing that each observation is evaluated based on two dimensions namely x being the predictive variable and y being the response variable. Observing n instances of x_i and y_i helps us in the prediction of an outcome given its predictive observation X a s follows:

$$Y = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{ß})y_i}{\sum_{i=1}^{n} K(\frac{x-x_i}{ß})} \tag{5.15}$$

Where $K()$ is the kernel function such that :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{5.16}$$

this allows the assignment of more weight as the observation gets closer to x and less weight as it gets further away from x. Also, ß here refers to the bandwidth efficiency. The bandwidth is a smoothing factor parameter for the estimator's bias-variance tradeoff which we calculate by adopting the mixed bandwidth selection strategy by Silverman[74] that searches for the optimal bandwidth. As can be seen from Figure 5.4, the first bandwidth is overfitting the data generating too much variance in the prediction while the third one is too smooth resulting in more bias for the estimator. The second one is balancing the bias and variance tradeoff of the estimator.

The bandwidth is calculated as follows:

$$ß = \left(\frac{4}{3n}\right)^{\frac{1}{5}} \sqrt{v} \tag{5.17}$$

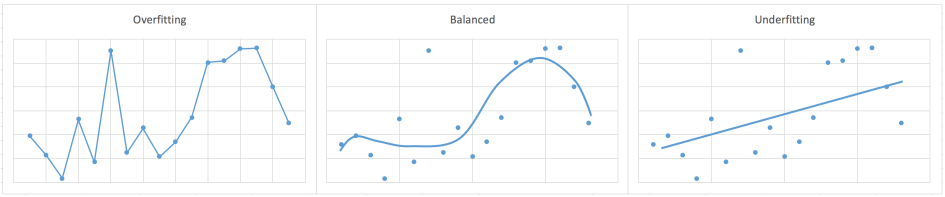

Figure 5.4: Bandwidth Selection

Where v is the variation, we get in equation 5.13. This bandwidth needs to be optimized based on some training data in order to fit the kernel regression model. To do so, a cross validation method was proposed by[15] namely the "leave-one-out least-square cross-validation" in which the bandwidth ß is chosen to minimize the cross-validation error as follows:

$$CV(ß) = \frac{1}{n}\sum_{i=1}^{n}\left(f_{ß}(x_{-i}) - y_i\right)^2 \tag{5.18}$$

Where $f_{ß}(x_{-i})$ is what is called the leave-one-out estimation of y_i using the kernel regression which refers to D_i (equation 5.12) but the instances for the distance are not equal i.e. the summation is: $\sum_{i=1,j\neq i}^{n}$. in our case, this maps to the following:

$$CV(ß) = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{d_{ij}}{ß})^2}S_i}{\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{d_{ij}}{ß})^2}} - S_i\right)^2 \tag{5.19}$$

To Estimate the demand of a bundle we will substitute the values of x and y with b and S. Substitution and expansion of equation 5.15 will give us:

$$D_i = \frac{\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{d_{ij}}{ß})^2}S_i}{\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{d_{ij}}{ß})^2}} \tag{5.20}$$

Where D_i is the demand for a bundle B_i and d_{ij} is the distance from equation 5.14. Now we can calculate the weights as follows:

$$w_{ij} = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d_{ij}}{\mathbb{B}})^2}}{\sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d_{ij}}{\mathbb{B}})^2}} \tag{5.21}$$

This implies that the smaller the distance between the instances, the higher the assigned weights to their vote. Now to create the bundles we make the bundle such that it maximizes the demand D so our objective then becomes finding bi that maximizes D_i.

Bundle Size and Demand Estimation

Although bundling helps increase the demand on the low demanded resources, the overall demand may affect the demand of the highly demanded resources negatively. Thus, adding many resources to the bundle while implementing Pure Bundling principle may result in lower revenue. Changing the bundle size has a direct effect on the demand of the bundle. Assuming that we bundle the low-demanded services, this implies that increasing the number of services in the bundle decreases the demand of the bundle. To determine the ultimate bundle size, we need to find the number of services $i \leq m \leq n$ that maximizes the revenue equation:

$$MAX((P_B + \sum_{i}^{n} axP_i)(D_B - \sum_{i}^{n} bx(\varphi_i(\Delta(D_B D_i))))) \tag{5.22}$$

Where P_B is the bundle price, P_i is the price of each individual service to be added to the bundle, D_B is the initial bundle demand, φ_i is the coefficient of the change in demand associated with adding service i, D_i is the demand for service i and a, b and x are real numbers. This will show the increase/decrease of the demand resulting from adding more services to the bundle. To get the values of φ_i, a, b and x we need

to perform instance-based comparison with similar bundles using the same procedure for creating the bundles proposed in section 4.3.

5.3.4 User Exchange

In some cases, the resources provided by the service provider are already booked and a new request comes in but there is no capacity to handle that request. In this case the options are either to deny the request or break the SLA with one or more users to be able to process the new request. In the second case, the service provider needs to consider many factors such as how much is the difference in the revenue and how much gain will be the result of this action. Another thing to be considered is the reputation since it will be affected by such an action. The type and reputation of the new potential user also plays a role.

To be able to decide on such cases we need to consider all these factors along with how much more profit we will be making. First thing we need to do is to calculate how much profit is to be gained by keeping the current users and this depends on the amount of resources they use and the price they pay for the usage. We also need to calculate how much profit the new contract will guarantee. Let us define a ranking mechanism that will give the users ranks according to their reputation and the size of the organization as a threshold T. When we get a new request from a user Ui with a threshold T_{Ui} then $\forall Uj \in S$ where S is the set of current users using the resources.

- If: $T_{Ui} < T_{Uj}\ \forall U \in S$, deny the request.
- else, calculate the total gain as follows:

Let SLA_{R_i} denote the number of days remaining on the SLA with the users to be compared, S_{usage} denotes the service usage per day ρ_i denoting the net profit from every usage, B_{fee_i} is SLA breaking fees and I_{fee} corresponds to the Initiation fees

of the SLA. We can define the total profit gained from that user i over the contract period as:

$$\sum(\rho_i) = SLA_{R_i} \times P_i \times S_{usage_i} \tag{5.23}$$

Then to calculate the net gain from discarding user to grant the request of a new user will be:

$$\frac{\sum(\rho_i) + I_{fee_i} - \sum_{j=1}^{n}(\sum(\rho_j)) - \sum_{j=1}^{n}(B_{fee_j})}{\sum_{j=1}^{n}(\sum(\rho_j))} \times 100 \tag{5.24}$$

This will give the percentage increase in profit which then can be compared to some preset percentage to decide either grant the request or negotiate higher price for the service with the new potential user.

Another aspect affecting the user exchange is the reputation. Exchanging many users will affect the reputation of the service provider which in turn will impact the demand on the offered services. Since the reputation has a direct relationship with the demand, it needs to be considered. The service provider gains reputation by several means such as the quality of offered resources, the quality of customers e.g. if the customers are large organizations then this will result in higher reputation for the service provider, and the loyalty to customers which implies how often does the service provider breaks an SLA.

Taking these factors into consideration, we can derive the equation for reputation $\Re$ calculation as follows:

$$\Re = \sum(\Re_{QoS}, \Re_u, \eth) \tag{5.25}$$

where $\Re_{QoS}$ is the reputation gained by the quality of provided services, $\Re_u$ is the reputation gained by the type and reputation of users using the services and $\eth$ is the negative impact factor of breaking an SLA. We can rewrite equation 5.25 as:

$$\Re = \sum(\frac{QoS_{avg}}{QoS_{avg_c}}, \Re_u, \lambda^n) \tag{5.26}$$

where QoS_{avg} is average Quality of Service provided by the service provider, QoS_{avg_c} is the average Quality of Service provided by other competitors in the market, λ is the one-time SLA break factor and n is a real number. Since the impact of breaking an SLA is not linear i.e. the change in reputation will not change by a fixed amount for every time the service provider breaks an SLA, we calculate the impact as an exponent. The reputation will have a direct relationship to the average demand on the services provided by the service provider as follows:

$$D'_{avg} = \frac{D_{avg}\Re}{D_{avg}} \tag{5.27}$$

where D'_{avg} is the change in the average demand. Now we can negotiate the price and SLA agreement with the new user accordingly using the Same negotiation process used for Dynamic Pricing but this time we negotiate multiple attributes instead of just negotiating the price which is supported by the negotiation algorithm we use.

5.4 Experiments

5.4.1 Setting Highest Possible Price

In order to calculate the revenue R we need to multiply the price of the service by the number of usages and subtract the operational costs from the total and that will give us the net profit as in equation 5.1. As equation 5.2 implies, when we change the price for the service the demand will change accordingly. Increasing the price will

result in lower demand and vice versa. In order to calculate the effect of the price change we introduced the values a as the price change, b as the demand change and x as the number of times this change is applied. This means whenever we add x times a to the price then the demand will change by x times b, where a and b are real numbers and x is an integer.

Expanding equation 5.2 will result in a polynomial as shown in equation 5.3 which will have a curve shaped graph that represents the profit or the revenue R. In order to finds the highest revenue, we need to find the peak of the curve of equation 5.3 which is when the slope of the curve is equal to zero. The way to find the slope is to derive the equation as we did in equation 5.4 and then solve for x by setting the equation to zero and that will give us the amount of change that we can apply to both a and b in order to find the highest revenue R.

5.4.2 Dynamic Pricing

In order to test the efficiency of the Dynamic Pricing we have used the WS Dream dataset. First, we set the Highest Possible Price (HPP) and compare the total revenue with the original service revenue that is calculated using the initial service price that is in the dataset and the estimated demand. We have used the service throughput as our measure for the service performance to compare similar services in order to estimate the demand change according to the price. We then calculate the revenue for the HPP which as can be seen from the figure is more than or equal to the original revenue since the HPP will provide the price that generates the highest revenue. Next, we use Dynamic pricing based on a certain threshold TH such that the service provider is not willing to sell for lower than TH as the price. We then construct the price vector along with the corresponding demand vector and calculate

the total revenue which will be more than the Dynamic pricing revenue or equal to it depending on the threshold. The lower the threshold the more revenue will be accomplished.

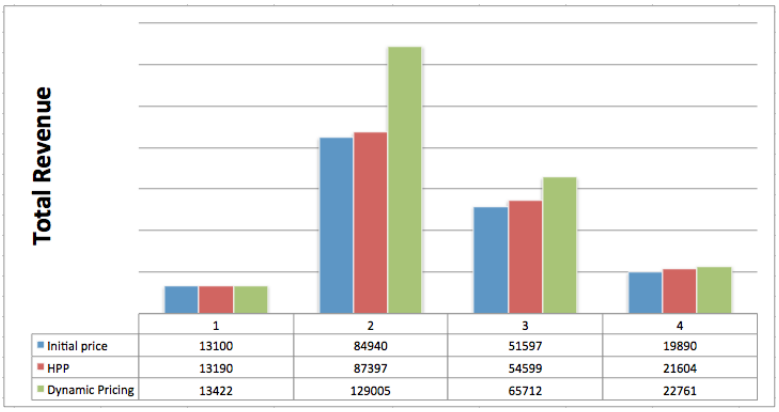

Figure 5.5: Initial Price vs HPP vs Dynamic Pricing

We can see from Figure 5.5 that the Dynamic pricing clearly generates more revenue as it will maximize the resource distribution and in the optimal case (depending on the threshold) will distribute all the available resources which will generate the maximum revenue.

5.4.3 Resource Bundling

To test the bundle creation, we have created several bundles to increase the selling rates of some services that have low selling rates. We have used the service throughput as a parameter to calculate the distance between instances. We have assumed that these services have already been priced using our proposed pricing model. Table 5.1 shows the revenue increase and the selling rate increase of the services after they are bundled.

The service ID refers to the ID of the service in the dataset, SR refers to the selling rate, $Rev.$ stands for revenue, B denotes the bundled services and E in the cases of the revenue and the selling rate refers to the estimated value from the

S_1ID	S_2ID	S_1SR	$S_1Rev.$	S_2SR	$S_2Rev.$	$Rev._C$	$B\ SR_E$	$B\ Rev._E$
1819	1012	94%	69.56	19%	3.23	72.79	84%	76.44
1526	1486	83%	29.05	51%	54.06	83.11	83%	117.03
3024	794	77%	107.03	45%	11.25	118.55	73%	119.72
4068	4756	94%	29.14	74%	46.62	75.76	68%	114.24
688	304	81%	72.09	34%	28.9	100.99	81%	140.94
5426	3374	72%	64.06	38%	12.92	77	71%	87.33
3686	3580	80%	80	24%	13.44	93.44	74%	115.44
162	1943	82%	52.29	14%	9.1	61.39	72%	92.16
4702	3942	75%	14.25	28%	5.88	20.13	58%	22.4
3094	4062	71%	50.41	26%	12.48	62.89	66%	78.54
4887	1568	79%	34.76	36%	22.32	57.08	6%	63.6
168	4522	94%	141	26%	28.86	169.86	88%	229.68
5187	3450	85%	56.1	23%	6.21	62.31	84%	78.12
1640	3956	86%	63.64	45%	25.2	88.84	77%	100.1
5289	4749	89%	77.34	57%	11.97	89.4	84%	90.72
3734	917	84%	43.68	14%	2.66	46.34	79%	56.09
4252	4081	80%	94.4	41%	18.45	112.85	77%	125.51
3440	2835	89%	19.58	31%	7.44	27.02	81%	37.26
684	3533	82%	97.54	21%	9.3	88.84	71%	90.17
4904	2251	85%	99.45	36%	20.88	120.33	74%	129.5

Table 5.1: Bundle Creation Simulation

similar bundles. As can be seen from Table 5.1, although in some cases the selling rate of one service decreases, the overall revenue of selling as a bundle will increase the total revenue as the revenue from the other service will compensate for the loss in the revenue. For example, let us take the first bundle (row 1 of Table 5.1), the first service ($S\ ID$: 1819) tends to sell 94% of the time and the second one ($S\ ID$: 1012) tends to sell only 19% of the time. Considering the case of 100 times then we will get the total revenue of selling these two services as: $S_1Rev. + S_2Rev$, that is 69.56 + 3.23 = $ 72.79. Bundling these two services together and comparing to similar bundles we estimate the selling rate of this bundle to be 84% which will result in $76.44 in estimated revenue which is higher than the combined revenue of selling both services. Similarly, in the second row of Table 5.1, the combined revenue of

S_1ID	S_2ID	S_1SR	$S_1Rev.$	S_2SR	$S_2Rev.$	$Rev._C$	$B\ SR_E$	$B\ Rev._E$
302	3773	94%	128.78	32%	18.24	147.02	70%	135.8
4914	4081	97%	64.99	41%	18.45	83.44	69%	77.28

Table 5.2: Bad Bundles

selling 100 services of each will be \$83.11 at 83% selling rate for the first service and 51% selling rate for the second service while the estimated revenue of the bundle is \$117.03 at 83% estimated selling rate for 100 bundles. Moreover, we can then use our proposed pricing model based on the other similar bundles to achieve the highest possible revenue.

Similar to creating bundles that are close to the high-selling bundles, creating bundles at random or bundles that are similar to low-selling bundles may result in a revenue loss as can be seen from Table 5.2. The services in these bundles have similar throughput to services in bundles we have in the dataset, therefore have similar estimated selling rates. We can see that in these particular cases, the services tend to generate more revenue if not combined in a bundle due to the face that the bundle has a lower selling rate and the generated revenue from the one service (the one that gained more selling rate) is not sufficient to cover for the loss in selling rate and revenue of the other service.

Bundle Size

To test the ultimate bundle size, we have generated different bundles using twelve services. The results indicate that enlarging the bundle might result in a bundle with high risk of not selling which will result in lower revenue.

Figure 5.6 shows the impact of increasing the number of services per bundle on the total revenue. As the figure shows, selling twelve bundles of single services (offering the services individually without any bundling) has lower revenue than selling

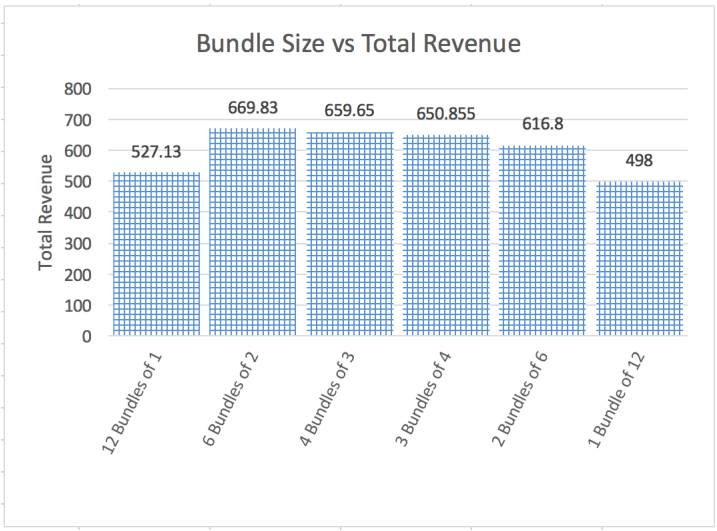

Figure 5.6: Bundle Size vs Revenue

six bundles of two services. In our particular case, the ultimate number of services per bundle came out to be two services per bundle. This number may be different based on the selected services. The main observation here is that the revenue tends to increase to a certain level and then decreases when the number of services per bundle exceeds the optimal number. It might even result in lower revenue than selling single services as we can see from Figure 5.5.

Generating bundles with optimal number of services per bundle will result in higher revenue than selling individual services as the bundling will increase the selling rate of the low demanded services without a severe effect on the highly demanded services as can be seen from Figure 5.7.

in Figure 5.7, the lower part of each column corresponds to the revenue generated by single services while the top part corresponds to the revenue generated by bundling services with the number of bundles below each column. The figure shows the revenue increase when selling services as bundles of optimal number of services

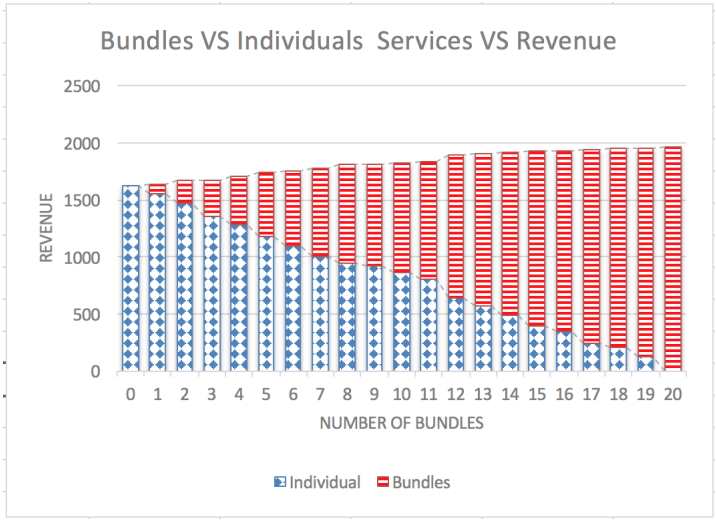

Figure 5.7: Bundle Revenue vs Individual

per bundle. Moreover, it shows that increasing the number highly demanded bundles and decreasing the number of individually sold services can generate more revenue.

5.4.4 Negotiation

As we mentioned earlier, we have done some improvement in the negotiation process through using P-optimality in the selection process via tournament selection instead of roulette wheel selection. We have implemented both methods and ran both of them on the same dataset of one hundred services as an initial population with more than one hundred generations. After running them for several times we were able to see the trend of finding the solution in each approach and we were able to clean our run data from outliers due to random number generation. Averaging the performed runs clearly shows how our approach improves the accuracy of the produces solution as can be seen from Figure 5.8. The figure shows the accuracy which refers to how close is the solution to the optimum solution are and that is in terms of percentage vs number of iterations needed to get to the solution. As can be seen from Figure 5.8, Although the runtime and number of iterations needed for WebNeg is slightly lower

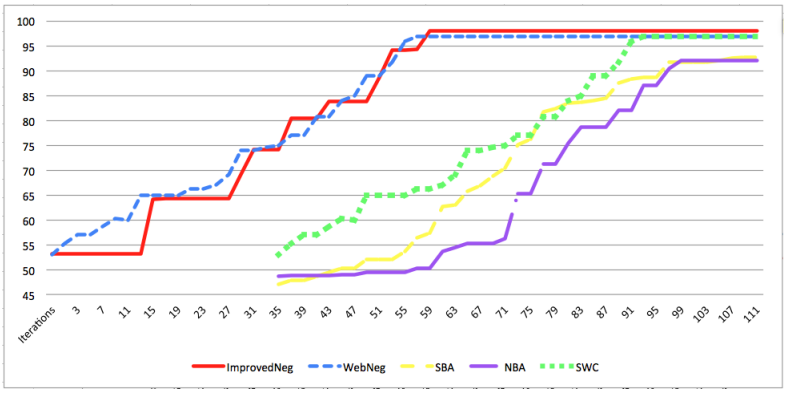

Figure 5.8: WebNeg VS Other Schemes

than those of ImprovedNeg, the ImprovedNeg performs better than WebNeg in terms of solution accuracy since it converges more to the optimal solution, that is about 98 percent for ImprovedNeg versus about 96 percent for WebNeg. Since the runtime of ImprovedNeg is not significantly higher than that of WebNeg and we were able to achieve a more optimal solution then trading-off the runtime for accuracy we can conclude that it is an improvement over WebNeg. Moreover, in our previous work we have found out that our approach outperforms using normal genetic algorithm with roulette wheel selection in terms of runtime, number of generations and solution accuracy [2]. It also outperforms SBA, NBA and SWC [21] [62] [43] in terms of both runtime and accuracy.

5.5 Insights and Discussion

Web services are becoming widely used nowadays and hence becoming a major interest for investment in the information market. Service providers are competing to offer the best services at competitive rates but at the same time make the most profit they can. Since raising the prices may result in lowering the number of customers then we needed to find a way to set the highest possible price that guarantees the highest profit. In this dissertation we proposed an approach that utilizes the concepts of Nash

Equilibrium and automated negotiation to tackle this problem. Our approach helps setting the HPP for the information goods by estimating the demand based on the server logs and posterior probability. The approach then applies Nash equilibrium concepts to alter the price to guarantee maximum resource distribution. We have also proposed a service bundling approach based on Instance Based Risk Assessment and Kernel Regression to increase the revenue by increasing the selling rates of the least desired services. Moreover, our approach negotiates the incoming users and the possibility of user exchange using automated negotiation.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Web Service cost optimization is one of the most important aspects in Web service composition as it matters to bot the user and the service provider. Cost saving may encourage the users to seek different service providers that will offer similar performance characteristics. This requires comparing the current solution with other possible candidates in order to find the best suitable one. Additional parameters need to be considered here such as SLA initiation fees ans SLA break fees (if applicable). We have proposed a scheme that will provide the best available solution for the user according to their preferences. Our proposed approach is based on P-Optimality in which we have proved that it enhances the performance of getting the solution. Moreover, our approach calculates the feasibility of switching from one service provider to the another considering the parameters associated with it.

We then have expanded our approach to account for multiple users seeking one service composition. An example for that is when an organization has multiple departments and seeking one service composition that will suit the most requirements. In our approach we first handle the conflicts that will result in two or more mutually exclusive solutions. We used the concepts of PROMETHEE group decision support to help get a group decision from the candidate matrix which we used our proposed approach in WCost to reduce its size. We simulated our approach and shown that it is able to provide the most suitable group decision for the Web service composition.

As for the service provider, we have proposed an approach that aims at maximizing the profit by employing a pricing model that sets initial prices for the services and then dynamically adjusts these prices based on the client's willingness to pay. This as we showed in our experiments generates higher profit as we will attract more users and will achieve higher resource distribution. Moreover, we tackled the problem of the resources that have low demand by bundling them with the highly-demanded

services to achieve more sales for these services. The bundling is based on risk assessment to assess the risk of a bundle not selling and the risk of the low demand service affecting the bundle demand in a way that generates less overall profit. We utilized kernel regression in our risk assessment for bundle creation. As adding more services to the bundle affects the bundle demand, we proposed a way to estimate the optimal bundle size that will generate the highest profit. Finally, our approach checks for the feasibility of discarding one or more users to grant a new upcoming user taking multiple factors into account, namely: overall profit, the reputation of the upcoming user and how is that switch going to affect the reputation of the service provider since the reputation of the service provider will affect the demand on the offered services. We ran experiments on the different aspects of our proposed scheme and the results show how our proposed scheme maximizes the profit of the service provider.

For future work, we plan on further improving the profit gain by utilizing the concepts of green cloud computing, specifically load balancing. The idea here is when the load is balanced over the servers, these servers will consume less power which translates to lower energy costs to the service provider and therefore higher profit.

REFERENCES

1. H Kıvanc Aksoy and Asli Guner. A bayesian approach to demand estimation. Procedia Economics and Finance, 26:777–784, 2015.
2. Hussain Aljafer, Khayyam Hashmi, Zaki Malik, and Abdelkarim Erradi. Web service cost optimization. In Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems, pages 116–120. ACM, 2015.
3. Hussain Aljafer, Zaki Malik, Mohammed Alodib, and Abdelmounaam Rezgui. A brief overview and an experimental evaluation of data confidentiality measures on the cloud. journal of innovation in digital ecosystems, 1(1):1–11, 2014.
4. Mohammad Alrifai, Dimitrios Skoutas, and Thomas Risse. Selecting skyline services for qos-based web service composition. In Proceedings of the 19th inter- national conference on World wide web, pages 11–20. ACM, 2010.
5. Kemal Altinkemer and Jeevan Jaisingh. Pricing bundled information goods. In Advanced Issues of E-Commerce and Web-Based Information Systems, 2002.(WECWIS 2002). Proceedings. Fourth IEEE International Workshop on, pages 89–96. IEEE, 2002.
6. Yannis Bakos and Erik Brynjolfsson. Bundling and competition on the internet. Marketing science, 19(1):63–82, 2000.
7. David Besanko, Sachin Gupta, and Dipak Jain. Logit demand estimation under competitive pricing behavior: An equilibrium framework. Management Science, 44(11-part-1):1533–1547, 1998.

8. Robert Bradley, Anthony Brabazon, and Michael O'Neill. Objective function de- sign in a grammatical evolutionary trading system. In Evolutionary Computation (CEC), 2010 IEEE Congress on, pages 1–8. IEEE, 2010.
9. Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, and F Lo Presti. Flow-based service selection forweb service composition supporting multiple qos classes. In Web Services, 2007. ICWS 2007. IEEE International Conference on, pages 743–750. IEEE, 2007.
10. Emiliano Carreno Jara. Multi-objective optimization by using evolutionary al- gorithms: the p-optimality criteria. 2014.
11. Wei-Lun Chang and Soe-Tsyr Yuan. Collaborative pricing model for bundling information goods. Journal of Information Science, 34(5):635–650, 2008.
12. N CHANNA+, KT PATHAN, and NH ARIJO. A comprehensive infrastructure of constraint optimizer in dynamic web service composition.
13. Wuhui Chen and Incheon Paik. Toward better quality of service composition based on global social service network.
14. Mohan Baruwal Chhetri, Jian Lin, SukKeong Goh, Jun Yan, Jian Ying Zhang, and Ryszard Kowalczyk. A coordinated architecture for the agent-based service level agreement negotiation of web service composition. In Software Engineering Conference, 2006. Australian, pages 10–pp. IEEE, 2006.
15. R Malcom Clark. A calibration curve for radiocarbon dates. Antiquity, 49(193):251, 1975.

16. Daniela B Claro, Patrick Albers, and Jin-Kao Hao. Selecting web services for op- timal composition. In ICWS International Workshop on Semantic and Dynamic Web Processes, Orlando-USA, 2005.
17. Carlos Coello Coello, Gary B Lamont, and David A Van Veldhuizen. Evolution- ary algorithms for solving multi-objective problems. Springer Science & Business Media, 2007.
18. Marco Comuzzi and Barbara Pernici. An architecture for flexible web service qos negotiation. In EDOC Enterprise Computing Conference, 2005 Ninth IEEE International, pages 70–79. IEEE, 2005.
19. Wenyun Dai, Haopeng Chen, Wenting Wang, and Xi Chen. Rmorm: A frame- work of multi-objective optimization resource management in clouds. In Services (SERVICES), 203 IEEE Ninth World Congress on, pages 488–494. IEEE, 2013.
20. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, 6(2):182–197, 2002.
21. Elisabetta Di Nitto, Massimiliano Di Penta, Alessio Gambi, Gianluca Ripa, and Maria Luisa Villani. Negotiation of service level agreements: An architecture and a search-based approach. Springer, 2007.
22. Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertis- ing and the generalized second-price auction: Selling billions of dollars worth of keywords. The American economic review, 97(1):242–259, 2007.

23. Joyce El Hadad, Maude Manouvrier, and Marta Rukoz. Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. Services Computing, IEEE Transactions on, 3(1):73–85, 2010.

24. Zaiwen Feng, Rong Peng, Raymond K Wong, Keqing He, Jian Wang, Songlin Hu, and Bing Li. Qos-aware and multi-granularity service composition. Information Systems Frontiers, 15(4):553–567, 2013.

25. Ja´nos Fu¨lo¨p. Introduction to decision making methods. In BDEI-3 Workshop, Washington. Citeseer, 2005.

26. Hao Gao, Jun Yan, and Yi Mu. Trust-oriented qos-aware composite service selection based on genetic algorithms. Concurrency and Computation: Practice and Experience, 26(2):500–515, 2014.

27. Debanjan Ghosh, Raj Sharman, H Raghav Rao, and Shambhu Upadhyaya. Self- healing systems—survey and synthesis. Decision Support Systems, 42(4):2164– 2185, 2007.

28. Bart Goethals and Jan Van den Bussche. A priori versus a posteriori filtering of association rules. 1999.

29. David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms, 1:69–93, 1991.

30. Yanhong Guo, Wenjun Zhou, Chunyu Luo, Chuanren Liu, and Hui Xiong. Instance-based credit risk assessment for investment decisions in p2p lending. European Journal of Operational Research, 249(2):417–426, 2016.

31. Sharon Halliday, Karin Badenhorst, and Rossouw Von Solms. A business ap- proach to effective information technology risk analysis and management. Infor- mation Management & Computer Security, 4(1):19–31, 1996.

32. Khayyam Hashmi, Amal Alhosban, Zaki Malik, and Brahim Medjahed. Webneg: A genetic algorithm based approach for service negotiation. In Web Services (ICWS), 2011 IEEE International Conference on, pages 105–112. IEEE, 2011.

33. Khayyam Hashmi, Zaki Malik, Erfan Najmi, Amal Alhosban, and Brahim Med- jahed. A web service negotiation management and qos dependency modeling framework. ACM Transactions on Management Information Systems (TMIS), 7(2):5, 2016.

34. John H Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michi- gan Press, 1975.

35. Angus FM Huang, Ci-Wei Lan, and Stephen JH Yang. An optimal qos-based web service selection scheme. Information Sciences, 179(19):3309–3322, 2009.

36. San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, and Cheng-Hung Chen. Dy- namic web service selection for reliable web service composition. Services Com- puting, IEEE Transactions on, 1(2):104–116, 2008.

37. San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, and Cheng-Hung Chen. Dy- namic web service selection for reliable web service composition. Services Com- puting, IEEE Transactions on, 1(2):104–116, 2008.

38. Emiliano Carreno Jara. Multi-objective optimization by using evolutionary algo- rithms: The-optimality criteria. IEEE transactions on evolutionary computation, 18(2):167–179, 2014.

39. Cao Jiuxin, Sun Xuesheng, Zheng Xiao, Liu Bo, and Mao Bo. Efficient multi- objective services selection algorithm based on particle swarm optimization. In Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific, pages 603– 608. IEEE, 2010.

40. Guosheng Kang, Jianxun Liu, Mingdong Tang, and Yu Xu. An effective dy- namic web service selection strategy with global optimal qos based on particle swarm optimization algorithm. In Parallel and Distributed Processing Sympo- sium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, pages 2280–2285. IEEE, 2012.
41. Michael J Katchabaw, Hanan L Lutfiyya, Andrew D Marshall, and Michael A Bauer. Policy-driven fault management in distributed systems. In Software Reliability Engineering, 1996. Proceedings., Seventh International Symposium on, pages 236–245. IEEE, 1996.
42. Tai-hoon Kim, D Palanikumar, and G Kousalya. Optimal web service selection and composition using multi-objective bees algorithm. INFORMATION-AN IN- TERNATIONAL INTERDISCIPLINARY JOURNAL, 14(10):3289–3295, 2011.
43. Freddy L´ecu´e, Usman Wajid, and Nikolay Mehandjiev. Negotiating robustness in semantic web service composition. In Web Services, 2009. ECOWS'09. Seventh IEEE European Conference on, pages 75–84. IEEE, 2009.
44. Jun Li, Xiao-Lin Zheng, Song-Tao Chen, William-Wei Song, and De-ren Chen. An efficient and reliable approach for quality-of-service-aware service composi- tion. Information Sciences, 269:238–254, 2014.
45. Cui Lin, Shiyong Lu, Zhaoqiang Lai, Artem Chebotko, Xubo Fei, Jing Hua, and Farshad Fotouhi. Service-oriented architecture for view: A visual scientific workflow management system. In Services Computing, 2008. SCC'08. IEEE International Conference on, volume 1, pages 335–342. IEEE, 2008.
46. Richard J Lipton and Evangelos Markakis. Nash equilibria via polynomial equa- tions. In Latin American Symposium on Theoretical Informatics, pages 413–422. Springer, 2004.

47. Zhi-Zhong Liu, Xiao Xue, Ji-quan Shen, and Wen-Rui Li. Web service dynamic composition based on decomposition of global qos constraints. The International Journal of Advanced Manufacturing Technology, 69(9-12):2247–2260, 2013.

48. Weina Lu, Xiaohui Hu, Shangguang Wang, and Xiaotao Li. A multi-criteria qos-aware trust service composition algorithm in cloud computing environments. International Journal of Grid & Distributed Computing, 7(1), 2014.

49. Simone A Ludwig. Single-objective versus multi-objective genetic algorithms for workflow composition based on service level agreements. In Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on, pages 1–8. IEEE, 2011.

50. Mario Mac´ıas and Jordi Guitart. A genetic model for pricing in cloud computing markets. In Proceedings of the 2011 ACM Symposium on Applied Computing, pages 113–118. ACM, 2011.

51. Charles Maina. Valuing information in an information age: The price model and the emerging information divide among individuals, societies, and nations. In Proceedings of the Annual Conference of CAIS/Actes du congr`es annuel de l'ACSI, 2013.

52. Zaki Malik and Athman Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. The VLDB Journal—The International Journal on Very Large Data Bases, 18(4):885–911, 2009.

53. Farhad Mardukhi, Naser NematBakhsh, Kamran Zamanifar, and Asghar Barati. Qos decomposition for service composition using genetic algorithm. Applied Soft Computing, 13(7):3409–3421, 2013.

54. Magnos Martinello, Mohamed Kaaniche, and Karama Kanoun. Web service availability—impact of error recovery and traffic model. Reliability Engineering & System Safety, 89(1):6–16, 2005.

55. Noyda Matos, Carles Sierra, and Nicholas R Jennings. Determining successful negotiation strategies: An evolutionary approach. In Multi Agent Systems, 1998. Proceedings. International Conference on, pages 182–189. IEEE, 1998.

56. Brad L Miller and David E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 9(3):193–212, 1995.

57. Sajib Mistry, Athman Bouguettaya, Hai Dong, and A Kai Qin. Predicting dy- namic requests behavior in long-term iaas service composition. In Web Services (ICWS), 2015 IEEE International Conference on, pages 49–56. IEEE, 2015.

58. Melanie Mitchell. An introduction to genetic algorithms (complex adaptive sys- tems). A Bradford Book, third printing edition, 55:02142–1493, 1998.

59. EA Nadaraya. On non-parametric estimates of density functions and regression curves. Theory of Probability & Its Applications, 10(1):186–190, 1965.

60. Y Narahari, CVL Raju, K Ravikumar, and Sourabh Shah. Dynamic pricing models for electronic business. Sadhana, 30(2-3):231–256, 2005.

61. Ambroise Ncho and Esma Aimeur. Building a multi-agent system for automatic negotiation in web service applications. In Proceedings of the Third Interna- tional Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3, pages 1466–1467. IEEE Computer Society, 2004.

62. Xiaotai Niu and Su Wang. Genetic algorithm for automatic negotiation based on agent. In 2008 7th World Congress on Intelligent Control and Automation, pages 3834–3838, 2008.

63. James Robert Oliver. On artificial agents for negotiation in electronic commerce. In System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on,, volume 4, pages 337–346. IEEE, 1996.

64. Wei Pan, Lean Yu, Shouyang Wang, and Xianjia Wang. A fuzzy multi-objective model for provider selection in data communication services with different qos levels. International Journal of Production Economics, 147:689–696, 2014.

65. Jos´e M Pena, V´ıctor Robles, Oscar Marb´an, and Mar´ıa S P´erez. Bayesian meth- ods to estimate future load in web farms. In International Atlantic Web Intelli- gence Conference, pages 217–226. Springer, 2004.

66. Arnold Polanski. A decentralized model of information pricing in networks. Journal of Economic Theory, 136(1):497–512, 2007.

67. Werner Reinartz. Customizing prices in online markets. Symphonya. Emerging Issues in Management, (1):55–65, 2002.

68. Abbas Roozbahani, Banafsheh Zahraie, and Massoud Tabesh. Promethee with precedence order in the criteria (ppoc) as a new group decision making aid: an application in urban water supply management. Water resources management, 26(12):3581–3599, 2012.

69. Florian Rosenberg, Predrag Celikovic, Anton Michlmayr, Philipp Leitner, and Schahram Dustdar. An end-to-end approach for qos-aware service composition. In Enterprise

Distributed Object Computing Conference, 2009. EDOC'09. IEEE International, pages 151–160. IEEE, 2009.

70. Florian Rosenberg, MB Muller, Philipp Leitner, Anton Michlmayr, Athman Bouguettaya, and Schahram Dustdar. Metaheuristic optimization of large-scale qos-aware service compositions. In Services Computing (SCC), 2010 IEEE In- ternational Conference on, pages 97–104. IEEE, 2010.
71. Jennifer Rowley. Principles of price and pricing policy for the information mar- ketplace. Library Review, 46(3):179–189, 1997.
72. Zhao Shanshan, Wang Lei, Ma Lin, and Wen Zepeng. An improved ant colony optimization algorithm for qos-aware dynamic web service composition. In In- dustrial Control and Electronics Engineering (ICICEE), 2012 International Con- ference on, pages 1998–2001. IEEE, 2012.
73. Bhanu Sharma, Ruppa K Thulasiram, Parimala Thulasiraman, Saurabh K Garg, and Rajkumar Buyya. Pricing cloud compute commodities: A novel financial economic model. In Proceedings of the 2012 12th IEEE/ACM International Sym- posium on Cluster, Cloud and Grid Computing (ccgrid 2012), pages 451–457. IEEE Computer Society, 2012.
74. Bernard W Silverman. Density estimation for statistics and data analysis, vol- ume 26. CRC press, 1986.
75. Monika Simjanoska, Marjan Gusev, Sasko Ristov, and Goran Velkoski. Scal- ing the performance and cost for elastic cloud web services. CIT. Journal of Computing and Information Technology, 21(2):85–95, 2013.
76. Hal R Varian. Pricing information goods, 1995.

77. Florian Wagner, Adrian Klein, Benjamin Klopper, Fuyuki Ishikawa, and Shinichi Honiden. Multi-objective service composition with time-and input-dependent qos. In Web Services (ICWS), 2012 IEEE 19th International Conference on, pages 234–241. IEEE, 2012.

78. Wei Wang, Peng Zhang, Tian Lan, and Vaneet Aggarwal. Datacenter net profit optimization with individual job deadlines. In Proc. Conference on Inform. Sci- ences and Systems, 2012.

79. Wang Xiaolong, Zou Peng, He Jun, Wang Peng, and Chen Liang. A qos-based service composition optimization method. In 1st International Workshop on Cloud Computing and Information Security. Atlantis Press, 2013.

80. Hao Yin, Changsheng Zhang, Bin Zhang, Ying Guo, and Tingting Liu. A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem. Mathematical Problems in Engineering, 2014, 2014.

81. Huiyuan Zheng, Jian Yang, and Weiliang Zhao. Qos analysis and service selection for composite services. 2010.

82. Huiyuan Zheng, Weiliang Zhao, Jian Yang, and Athman Bouguettaya. Qos anal- ysis for web service compositions with complex structures. Services Computing, IEEE Transactions on, 6(3):373–386, 2013.

83. Wei Zhou, Junhao Wen, Min Gao, and Junwei Liu. A qos preference-based al- gorithm for service composition in service-oriented network. Optik-International Journal for Light and Electron Optics, 124(20):4439–4444, 2013.

84. Liu Zhuang, Guo HeQing, Li Dong, Han Tao, and Zhang Juan Juan. Solving multi-objective and fuzzy multi-attributive integrated technique for qos-aware web service

selection. In Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on, pages 735–739. IEEE, 2007.

ABSTRACT

WEB SERVICE COMPOSITION OPTIMIZATION

by

HUSSAIN ALJAFER

August 2019

Advisor: Dr. Zaki Malik

Major: Computer Science

Degree: Doctor of Philosophy

In recent years, users and organizations started switching from workstation-based applications to Web services also known as cloud services. Web services offer many advantages such as cost of use and maintenance. Web services follow the pay-per-use pricing model, so the users pay for their usage only. Due to the huge number of services, a composition optimization mechanism is needed to help the users find the best service/set of services for their application/s. On the other hand, service providers look for generating the highest profit possible from the offered services. In this dissertation, we address the problem from both the user's perspective and the service provider's perspective. We propose a mechanism to find the best service for a composition based on the user's preferences. We also extend our work to account for multiple users. Last but not least, we propose a scheme for the service provider to help generate higher revenue by proposing a pricing model that includes dynamic pricing. We also tackle the issue of the low-demanded services by utilizing resource bundling to help sell these services.

AUTOBIOGRAPHICAL STATEMENT

I have always been passionate about computers and technology since I was young. I still remember owning my first PC when I was in 5th grade. Since then, my dream became to be able to dive more and write computer programs and be one of the innovative people who bring such new technologies to the world. My passion pushed me to complete my B.S. degree in information and computer science at King Fahd University of Petroleum and minerals.

Upon completing my B.S. degree, I worked in couple companies but still I felt that my dream wasn't fulfilled. This led me to pursue my M.S. degree in computer science at Wayne State University in which I was planning to focus on bioinformatics as my concentration. I still remember when I took the Web service's course which opened up my eyes to a new world that became my new passion. This steered my direction to Web services and Cloud computing, and I completed my master's thesis in cloud data confidentiality.

I didn't feel enough satisfaction in terms of achieving my dream, which is to be one of the future builders and be a beneficial member to the society, so I decided to complete a Ph.D. in computer science and focus my career goal on teaching and performing more research. This in my opinion will fulfill my dream in two ways. First, I will still engage in research activities and deliver new useful solutions to problems, and second, I will deliver my gained knowledge to the new generation to help them in building a better future.