3-12-2020

An Approach of QoS Evaluation for Web Services Design With Optimized Avoidance of SLA Violations

Peng Cheng
University of Windsor

An Approach of QoS Evaluation for Web Services Design with Optimized Avoidance of SLA Violations

By

Peng Cheng

A Dissertation
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2020

An Approach of QoS Evaluation for Web Services Design with Optimized Avoidance of SLA Violations

by

Peng Cheng

APPROVED BY:

H. Lutfiyya, External Examiner
University of Western Ontario

H. Wu
Department of Electrical and Computer Engineering

A. Ngom
School of Computer Science

D. Wu
School of Computer Science

X. Yuan, Advisor
School of Computer Science

February 14, 2020

Declaration of Co-Authorship/Previous Publication

I thereby certify that this dissertation incorporates material that is the result of my research conducted under the supervision of my advisor Dr Xiaobu Yuan. Parts of Chapter 3 and 4 of the dissertation was co-authored with Dr Xiaobu Yuan. In all cases, the key ideas, primary contributions, justifications of the methodologies, the design of the experiments, data analysis and interpretations, were performed by myself with the guidance of Dr Xiaobu Yuan.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

This thesis includes two original papers that have been previously published / submitted for publication in peer reviewed journals, as follows:

1. Xiaobu Yuan, Peng Cheng. 2019. An Approach of Sensitivity and Metamodel-Based Analyses for SLA Violation Prediction. 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE). Copyright 2019, IEEE. Published.

2. Xiaobu Yuan, Peng Cheng. 2019. A Strategy of QoS Optimization for Web Services. 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE). Copyright 2019, IEEE. Published.

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above

material describes work completed during my registration as a graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Quality of service (QoS) is an official agreement that governs the contractual commitments between service providers and consumers in respect to various non-functional requirements, such as performance, dependability, and security. While more Web services are available for the construction of software systems based upon service-oriented architecture (SOA), QoS has become a decisive factor for service consumers to choose from service providers who provide similar services. QoS is usually documented on a service-level agreement (SLA) to ensure the functionality and quality of services and to define monetary penalties in case of any violation of the written agreement. Consequently, service providers have a strong interest in keeping their commitments to avoid and reduce the situations that may cause SLA violations.

However, there is a noticeable shortage of tools that can be used by service providers to either quantitively evaluate QoS of their services for the predication of SLA violations or actively adjust their design for the avoidance of SLA violations with optimized service reconfigurations. Developed in this dissertation research is an innovative framework that tackles the problem of SLA violations in three separated yet connected phases. For a given SOA system under examination, the framework employs sensitivity analysis in the first phase to identify factors that are influential to system performance, and the impact of influential factors on QoS is then quantitatively measured with a metamodel-based analysis in the second phase. The results of analyses are then used in the third phase to search both globally and locally for optimal solutions via a controlled number of experiments. In addition to technical details, this dissertation includes experiment results to demonstrate that this new approach can help service providers not only predicting SLA violations but also avoiding the unnecessary increase of the operational cost during service optimization.

Dedication

This dissertation is dedicated to my wife, Juan Fan, and to my parents, Maidi Su and Xiaoxin Cheng, for their endless love, support, and encouragement.

Acknowledgements

It gives me immense pleasure to thank a few very important people who have helped me a lot during the whole process of my graduate studies.

First and foremost, I would like to present my gratitude to my supervisor Dr. Xiaobu Yuan for his valuable assistance and support during the past fours years. This work could not have been achieved without his continuous encouragement, valuable advices, suggestions and cooperation.

I would also like to thank my committee members, Dr. Huapeng Wu, Dr. Alioune Ngom and Dr. Dan Wu for their encouragement, insightful comments, and advice.

Finally, I want to thank my wife and parents for their love and support throughout my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The dramatically increased software complexity has made productivity and time-to-market two of the major concerns of software industry. Traditional approaches of software development failed to cope with sophisticated computer applications. In comparison, Service-Oriented Architecture (SOA) has emerged as a new technology that helps organizations to deal with the interoperability of software systems in heterogeneous environments. An SOA system consists of a collection of services that interact with each other, and its inter-service infrastructure focuses on Web services when services communicate via the Internet. SOA promises added benefits of easier components reuse, increased reliability, and reduced development and deployment costs [46].

More recently, Web services have become the technology of choice for realizing Service-Oriented Architecture and its associated set of strategic goals [1]. Web services are considered as self-contained, self-descriptive, modular applications that can be published, located, and invoked across the Web [2]. The communication between Web services is done via standards-based technologies that give users the opportunity

to access different Web services, independent of their hardware, operating system, or even programming environment. This supports organizations with a technology to create services that can be easily discovered and consumed by external users [1]. Software can be redelivered and paid for as streams of services as opposed to packaged products, thus making it possible to achieve automatic and dynamic interoperability between systems while accomplishing business tasks [6]. Businesses can be released from the burden of complex, low, and expensive software integration and focus instead on the value of their mission and critical tasks. Consequently, the Internet becomes a global common platform where organizations and individuals communicate with each other to carry out various commercial activities and to provide value-added services [6]. The rest of this section briefly presents the main concepts, principles and research problems relevant to Service-Oriented Architecture. In Chapter 2, the background and literature review of Service-Oriented Architecture will be discussed in more details.

As shown in Fig. 1.1, Web services play three major roles in an SOA system. With service registry acting as a centralized directory, service providers publish the information of their Web services for service consumers to select according to their needs of software development. The selected Web services are then bound at run time and invoked during the operation of software systems. Usually, Web services are third-party software made available by service providers, which means that these distributed applications based on Web services rely on these providers to ensure that their services comply with the agreed Quality of Service(QoS) [3]. Meanwhile, with tens of thousands of Web services available on the Internet, many of which are of equiva-

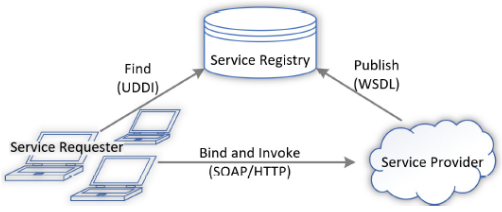

Figure 1.1: Three Web Service Roles

lent functions, QoS values are important criteria for service consumers to choose the web services that meet their requirements [5]. Many investigations have been carried out focusing on different kinds of QoS-related issues, such as Web service selection, Web service composition, SOA failure prediction, SOA performance prediction, fault tolerance, resiliency quantification, service searching, resource consistency, resource allocation, workload balance, as well as dynamically resource management [10]. As a result, QoS is currently a leading research topic in the field of SOA [3].

QoS is widely employed for describing non-functional characteristics of Web services. With the increasing number of Web services, QoS has become an important differentiating point of different functionally equivalent Web services. A Web service's QoS includes a number of properties, such as response time, throughput, failure probability, availability, price, popularity, and so on [4]. As a service-level agreement (SLA), QoS is documented to guarantee that services fulfill their official commitments in terms of both functionality and quality. While service consumers focus on suitability when selecting from a pool of similar candidates, service providers compete with each other to offer not only the anticipated functionality but also attractive quality. If service provider fails to ensure SLA for service consumers, this will have negative impact on customer satisfaction and even lead to a disaster. For instance, when Amazon Elastic Cloud crashed in 2011 in terms of availability, it faced an outage and many big customers such as Quora and Reddit were down for more than a day. Such crashes affects service providers and service consumers. As SLA also defines monetary penalties in case of any violation of the written agreement, service providers have a strong interest in keeping their commitments to avoid and reduce the situations that may cause SLA violations.

Nevertheless, avoidance of SLA violations has been proven to be a challenging task as QoS is influenced by a variety of factors [29]. Because of variations in workload, computing resources, and even network conditions, it is common for Web services to exhibit fluctuations in performance, leading to the possibility of violating SLA. Any service provider who does not take action to prevent SLA violation will have to face

monetary penalties and lost revenue due to damaged relationships with clients [38]. Therefore, service providers are in desperate need of such tools that can not only help them predict if their service design provokes SLA violations but also guide them to optimize service design and prevent SLA violations from happening. When SLA violations are predicted, service providers can always optimize QoS of Web services by configuring different contextual factors, such as CPU, storage, distributed computing, or data cache though at the risk of increased operational cost due to over provision [38].

The extraordinary potential of SOA in software development has stimulated continuous research in the academic community and explorations in the software industry, as to be discussed in the next chapter of the literature survey. To improve the design of Web services, encouraging progress has been achieved for the evaluation of QoS, prediction of SLA violations, and QoS optimization. For example, most of the approaches use the technique of collaborative filtering to evaluate SOA systems with historical datasets obtained from other systems running a similar application and to predict the possibility of SLA violations based upon the calculation of QoS values for existing services [12, 13, 15, 16, 17, 74]. For QoS optimization, some approaches work for groups of services by maximizing aggregated utility values [89, 90, 91, 92] and others for individual services by allocating more resources [87, 88]. However, current methods still lack objective evaluation of software operating in different application domains, and their separation of related processes has resulted in limited success in practice.

In this dissertation, the avoidance of SLA violation is divided into two subproblems: QoS evaluation and optimization. For QoS evaluation, the dissertation research first applies the principle of design of experiments to analyze how sensitive a system is influenced by uncontrollable factors of a Web service and to find the influential factors that dominate the service's performance. The identification of influential factors then helps to develop an algorithm that constructs a surrogate model for the Web service and to use it to predict situations in which violations of SLA may take place. QoS

optimization is necessary only when SLA violations are supposed to happen, and it is carried out in the dissertation research by a guided search to globally identify sub-regions with local optima and to locally locate the best optimum with constraints to minimize operational cost. In such a way, the three normally separated processes of sensitivity analysis, SLA violations predication, and QoS optimization are integrated into a unified framework to help service providers to analyze, predict, and prevent SLA violations.

1.2 Research Contributions

This dissertation makes the following contributions in QoS evaluation for Web services design with optimized avoidance of SLA violations:

1. Multi-factor sensitivity analysis:

   (a) Created a general multi-factor model of sensitivity analysis that decomposes the total sensitivity measure into several sensitivity measures for each of individual factors and for all their joint effects in different combinations.

   (b) Based upon this general model, developed a new algorithm of sensitivity analysis for the identification of influential factors. As the algorithm calculates the influence of different input parameters on the model's output and identifies the most influential parameters, it can be used to improve model quality and to reduce parametrization and computational costs. In addition, this algorithm requires a relatively small number of experiments with high accuracy and robustness.

2. Metamodel-based performance analysis:

   (a) The identification of influential factors helps to overcome the problem of over-parameterization and allows the development of a novel algorithm to create a fitted metamodel for quantitative analysis of QoS.

(b) This algorithm of performance analysis uses the metamodel for quantitative analysis of service performance and for the prediction of SLA violations without domain limits.

3. QoS optimization:

(a) Created a formal model with influential controllable factors for the problem of QoS optimization.

(b) Developed a search algorithm to fit a global surrogate model with multivariate adaptive regression splines and to identify sub-regions with potential local optima.

(c) Extended the response surface methodology to allow the search algorithm finding the global optima among the local optimum in the identified subregions. Furthermore, the algorithm takes minimizing operational costs into consideration to prevent over-provisioning resources.

1.3 Overview of the Unified Framework

Sensitivity analysis, SLA violation prediction, and design optimization are three important yet normally separated techniques for QoS evaluation and optimization [67]. As illustrated in Fig. 1.2, this dissertation research proposes an innovative framework to tackle the main issues of SLA violation by joining these three techniques into a unified framework to analyze, predict, and prevent SLA violations. Based upon a newly invented multi-factor model, the process of sensitivity analysis in this framework initiates the construction of a variance table with a chosen experimental strategy, e.g. Taguchi's orthogonal arrays [68], by calculating the F-distribution values to determine the impact of factors, both individually and jointly, that are beyond the control of a service on a Web service's performance. Among all the uncontrollable factors, those whose individual or joint F-distribution values exceed the threshold value of a cumulative F-distribution are influential factors of the service.

With the number of uncontrollable factors being reduced to only those influential to the service's performance, the second process treats the Web service under examination as a black box, use these identified influential uncontrollable factors as input variables and fits it with a surrogate model, or metamodel. The development of a metamodel starts with a candidate in a specific type and form, such as a quadratic model in polynomial regression, from a pool of choices. The process of SLA violation prediction then checks the adequacy of this chosen candidate with new experiments and verifies that the observable performance merits of the metamodel are within the 90% prediction interval at a residual error rate less than 10%. Unless a rejection results in the choice of another candidate, the metamodel is ready to be used for quantitative analysis of service quality and for the prediction of situations when SLA violations could take place. Meanwhile, the metamodel can help service provider better understand the performance of a service and identify performance bottlenecks caused by influential factors, which can help service provider choose the effective controllable factors to optimize performance in the next phase.

The last process of QoS optimization becomes inevitable to avoid the predicted SLA violations by adjusting the values of controllable factors by service providers, such as CPU, storage, and cache. This process includes five steps as below:

1. Identify the most influential controllable factor;
2. Conduct experiments and collect observation data;
3. Fit the collected dataset with the model of multivariate adaptive regression splines (MARS);
4. Search globally and create a list of subregions in which potential local optima exist;
5. Identify global optimum by searching local optima.

This five-step procedure repeats until either SLA prevention is achieved or the

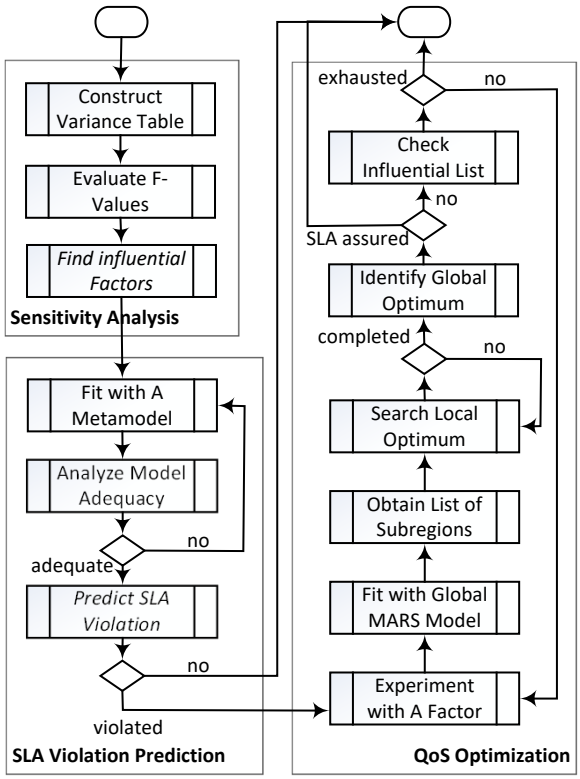

Figure 1.2: A Unified Framework of QoS Optimization

list of controllable factors is exhausted. In the latter case, a redesign of the Web service by the provider becomes a sensible recommendation.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 introduces the background of SOA and the latest literature reviews related to SLA violation, SLA violation prediction, QoS optimization and the design and analysis of experiments. After presenting the first and second phases of the proposed framework on SLA violation prediction in Chapter 3, Chapter 4 presents the third phase of the proposed framework on QoS optimization for Web services. Finally, conclusions with some suggestions for future works are given in Chapter 5.

Chapter 2

Background and Literature Review

2.1 Service-Oriented Architecture (SOA)

The dramatically increased complexity of software systems in the past decades has made traditional approaches of software development inadequate for the software industry to handle sophisticated computer applications. In the demand of new technologies, SOA has has emerged as an efficient way to help organizations dealing with the interoperability of software systems in heterogeneous environments. SOA offers the much-needed benefits of easier components reuse, increased productivity, improved reliability, shorter time-to-market, and reduced deployment costs. It is one of the most successful architectural styles in which applications make use of reusable services via internet. In the next decade, the SOA principles will be at the core of a new era of business engagements that transact at Internet scale across locations, devices, people, processes and information [1].

An SOA system consists of a collection of services that interact with each other, and its inter-service infrastructure focuses on Web services when services communicate via the Internet. It is based on some key principles listed as follows [69]:

1. Loose coupling: Services maintain a relationship that minimizes dependencies and only maintain an awareness of each other. This principle advocates the creation of a specific type of relationship within and outside of service boundaries, with a constant emphasis on reducing loosening dependencies between the service contract, its implementation, and its service consumers.

2. Service contract: Services adhere to a communicative agreement as defined collectively by one or more service description documents. Services express their purpose and capabilities via a service contract.

3. Service abstraction: Beyond what is described in the service contract, services hide logic from the outside world. Abstraction ties into many aspects of service-orientation. On a fundamental level, this principle emphasizes the need to hide as much of the underlying details of a service as possible.

4. Service reusability: Services provide specific functionality with the intention of promoting reuse. Reuse is strongly advocated within service-orientation; so much so, that it becomes a core part of typical service analysis and design processes, and also forms the basis for key service models. The advent of mature, non-proprietary service technology has provided the opportunity to maximize the reuse potential of multi-purpose logic on an unprecedented level.

5. Service composability: Collections of services can be coordinated and assembled to form composite services. As the sophistication of service-oriented solutions continues to grow, so does the complexity of underlying service composition configurations.

6. Service autonomy: Services have control over the logic they encapsulate. For services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources.

7. Service statelessness: The management of excessive state information can compromise the availability of a service and undermine its scalability potential.

Services are therefore ideally designed to remain stateful only when required.

8. Service discoverability: Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

Each service can be provided by different providers and used by one or multiple consumers. A service consumer can be an application or a service that reuses other services. According to the SOA systems that service consumers are developing, they choose the most suitable services from different candidates with similar functionality and use them to compose their application. A service provider, on the other hand, can be an individual or an organization that develops and maintains reusable services. These services are available for service consumers to reuse. Competition is extremely fierce between different service providers as there are always others with the same features. If a service is not capable of satisfying service consumers in terms of quality and functionality, service consumers may abandon this service and choose another service provider.

Web services offer a potential solution for developing distributed business processes and applications that can be accessed via the Internet. The use of Web services in SOA systems have have many benefits for the development of new applications [76]. While QoS encompasses various non-functional issues, such as performance, dependability, and security, SLA as an official document of QoS is in place to assure service consumers that they get the services paid for by obligating service providers to fulfill contractual promises. As more competitive Web services become available for service consumers to choose, QoS has become a decisive factor to distinguish the reputation of service providers [77]. In the next section, QoS and SLA will be discussed in detail.

2.2 Service Level Agreement (SLA)

As QoS tells the ability of a service to meet certain requirements for different aspects of the service, such as performance, availability, reliability, and cost, it plays an important role in the selection and composition of services for the development of SOA systems. For qualitative and quantitative QoS evaluation, service providers and customers negotiate the QoS attributes they mutually agree on, including rejection rate, mean time between failures, response time, throughput, financial cost, or energy consumption. As part of the contract between service consumers and providers that defines the level of service, all SLAs need to formally describe both the different Service Level Objectives (SLO) in terms of values for QoS metrics and penalties if the objectives have not been accomplished [70]. For instance, the following specifications could be in an SLA for a Web service that generates a list of top ten webpages linking to different e-commerce websites with the lowest prices for certain products.

1. The average response time for the operation of this Web service should be less than 50 milliseconds.

2. This service must be available from 6:00 AM to 8:00 PM, Monday through Sunday.

3. The service provider should seek 100% availability during working hours for this Web service.

4. A penalty may be applied upon the service provider if the fault lies on the side of the service provider. The penalty can also be waived if both parties reach an agreement to do so. If applied, the penalty will be: Penalty = Amount * Times. In this equation, the penalty is the total compensation service consumer will receive from the service provider for SLA violation in one year; Amount is the penalty for each SLA violation; Times is the total number of SLA violations taking place in one year.

SLA helps service consumers and providers understand the trade-offs inherent among cost, schedule, and quality with explicitly documented contracts, but the fulfillment of contracts has proven to be difficult. A report published by Forrester Consulting indicates that service providers fail to meet the expected outcomes for service consumers 26% of the time, directly resulting in 57% of the service providers noticing a decrease in revenue and negative impact on customer satisfaction due to their failure in meeting SLA requirements [71]. As a result, the avoidance of SLA violation for SOA has become an import topic of research that attracts interest from both the academic environment and the software industry.

By introducing a utility model for contract-based service provisioning, a method was developed in [34] to help service providers establish an optimal service provision contract under uncertainty. Based on the idea of risk assessment, this work focuses on SLA criteria formation and evaluation, instead of the quantitative measurement. Similarly, the authors of [84] proposed an optimal SLA formation architecture, enabling service providers to consider consumers' reliability in their commitment to SLA. The authors classify existing consumers into three categories based on their reliability or trustworthiness value and use that knowledge to ascertain whether to accept a consumer request for resource allocation and then to determine the extent of the allocation. Their approach enables service providers to monitor the behavior of service consumers in the post-interaction time phase and to minimize service violations by using the information of consumer behavior to form viable SLAs in the pre-interaction time phase. From a risk management perspective, another paper proposed a process of SLA violation detection and suggested service providers to consider abatement of SLA violations with SLA monitoring, violation prediction, and recommendations for risk reduction [35].

Meanwhile, some SLA management-based methods have been proposed to prevent the violation of SLA. By introducing a mechanism for contract renegotiation and modification, a method was developed in [36] to revise the current format of SLA specifications and allows the involved parties of an SLA to renegotiate at run-time. It

Table 2.1: Risk Assessment-based Methods

Publication	**Contributions**
Mitigating provider uncertainty in service provision contracts by C Smith and A Van Moorsel(2010) [34]	- Introduce a utility model for contract-based service provisioning. - Help service providers establish an optimal service provision contract under uncertainty.
Provider-based optimized personalized viable SLA (OPV-SLA) framework to prevent SLA violation by W Hussain, FK Hussain, OK Hussain and E Chang(2016) [84]	- Introduce a novel optimal SLA formation architecture. - Help the service provider monitor the behavior of service consumers in the post-interaction time phase.
Risk-based framework for SLA violation abatement from the cloud service provider's perspective by W Hussain, FK Hussain and O Hussain(2018) [35]	- Introduce a risk management-based Framework for SLA violation abatement. - Help the service provider to form an SLA which comprises SLA monitoring, violation prediction and decision recommendation.

avoids both the suspension of service provisioning and the brutal termination of the contract. After examining how Service Level Agreement management(SLAM) deals with the dynamic nature of cloud and inspecting issues concerning SLAM, the authors of [37] suggested SLA renegotiation as an efficient way of managing SLA within a cloud-based system. To guarantee SLA for cloud services, a recent paper proposed a cloud model [70] that used a new language to formally describe cloud-oriented SLAs and to define different Service Level Objectives with utility functions. It developed several online controlling algorithms to monitor and ensure the SLOs for the objective of achieving a systematic and transparent integration of SLAs in the cloud.

Table 2.2: SLA Management-based Methods

Publication	**Contributions**
Dynamic SLAs management in service oriented environments by G Di Modica and O Tomarchio, L Vita(2009) [36]	- Enhance the current WS-Agreement specification. - Allow the involved parties of an SLA to renegotiate at run-time. - Avoid both the suspension of service provisioning and the brutal termination of the contract.
Renegotiation in service level agreeent management for a cloud-based system by AFM Hani, IV Paputungan and MF Hassan(2015) [37]	- Propose the approach managing Service Level Agreement within a cloud-based system. - Use SLA renegotiation as an efficient way of managing SLA within a cloud-based system.
SLA guarantees for cloud services by D Serrano, S Bouchenak and Y Kouki(2016) [70]	- Formally describe cloud-oriented SLAs. - Proposed a model to guarantee SLA for cloud services. - Develop several online controlling algorithms to monitor and ensure the SLOs.

Listed in Table 2.1 and Table 2.2 are two groups of publications surveyed in this section, with highlights of their contributions. While both risk assessment-based and SLA management-based methods can help service providers reduce or minimize SLA violations, existing approaches in these directions lack the capability of optimizing QoS according to SLA requirements when SLA violations are predicted or take place. As discussed in Section 1.1 on background, service providers who do not take action to optimize the performance when SLA violation occurred will not only face monetary penalties but also results in lost revenue due to their damaged relationship with service

consumers [38]. Meanwhile, QoS is largely impacted by a variety of factors in the real-world service invocation. The typical number of controllable and uncontrollable factors that service providers need to deal with ranges from one to eight factors. Therefore, service providers need effective tools to help them analyze how different factors affect the QoS, to predict the situations in which these factor's influence cause SLA violations, and to optimize performance when an SLA violation is predicted or happens.

2.3 SLA Violation Prediction

In recent years, SLA violation prediction has become a popular topic in the study of cloud services. This subject can be viewed either from the perspective of the service consumer or from that of the service provider.

For SLA violation prediction from the perspective of the service consumer, most of the existing works focus on using or extending approaches based on collaborative filtering(CF), which encompasses techniques for matching people with similar interests and making recommendations on this basis. Generally, collaborative filtering algorithms can be divided into two main categories: memory-based and model-based algorithms. Memory-based methods employ similarity computation with past usage experiences to find similar users and services for making the performance prediction and, include user-based approaches, item-based approaches, and their fusion [14]. Memory-based CF algorithms are easy to implement and highly effective, but they suffer from a fundamental problem: the inability to scale-up. To address the scalability issue, model-based methods were proposed to utilize the models(such as clustering models, latent factor models, aspect models, and so on) to recognize complex patterns by seeking users for the recommendation within smaller and highly similar clusters, rather than within the entire database [29]. However, model-based methods are often time-consuming to build and update. Listed in Table 2.3 are publications surveyed

Table 2.3: Collaborative Filtering (CF)-based Methods

Publication	**Contributions**
Qos-aware Web service recommendation by collaborative fltering by Z Zheng, H Ma, MR Lyu and I King(2011) [12]	- Present a user-collaborative mechanism to predict QoS values of Web services.
Collaborative Web service qos prediction via neighborhood integrated matrix factorization by Z Zheng, H Ma, MR Lyu and I King(2013) [13]	- Solve the issue of the expensive and time-consuming invocations of Web services.
TAP: a personalized trust-aware QoS prediction approach for Web service recommendation by K. Su, B. Xiao, B. Liu, H. Zhang and Z. Zhang(2017) [15]	- Present a trust-aware approach for reliable personalized QoS prediction. - Address the data credibility problem that most existing CF-based approaches ignore.
New QoS-Aware Web Service Recommendation System based on Contextual Feature Recognition at Server-Side by S Li, J Wen, F Luo, M Gao and J Zeng(2017) [16]	- Propose an improved prediction approach by considering the contextual feature similarities of different services.
Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization by J Zhu, P He, Z Zheng and MR Lyu(2017) [17]	- Propose an adaptive matrix factorization approach for runtime QoS prediction with high accuracy, efficiency, and robustness.

in the direction of CF-based methods, with highlights of their contributions. After presenting a user-collaborative mechanism for past QoS information that are collected from different users, a collaborative filtering approach is proposed in [12] to predict QoS value of Web services and make Web service recommendation by taking advantages of past usage experiences of service users. To avoid the issue of the expensive and time-consuming invocation of Web services in [12], a neighborhood-integrated approach is designed in [13] for QoS prediction of personalized Web services. Based on the fact that the quality of Web service is affected by its context feature, the au-

thors of [16] proposes a new QoS-aware Web service recommendation system, which considers the contextual feature similarities of different services. To address the data credibility problem that most existing approaches ignore, the authors of [15] propose a trust-aware approach for reliable personalized QoS prediction. Meanwhile, to support timely and accurate adaptation decisions, the authors of [17] proposed an adaptive matrix factorization approach to perform online QoS prediction for candidate services. The approaches based on collaborative filtering operate on a common assumption that multiple applications are likely to support similar functionalities when the evaluation is conducted similarly for a large part of their services, and, therefore, their remaining services can be evaluated similarly [24]. However, when the new systems under development are for different application domains, their design involves significant structural or contextual changes and thus destroys the applicability of collaborative-filtering-based approaches. As a result, there is a demand for a more general approach that is capable of predicting the quality of services without the constraints on application domains.

For SLA violation prediction from the perspective of the service provider, several approaches have been proposed in recent years. Listed in Table 2.4 are publications surveyed in this direction, with highlights of their contributions. To guarantee the business requirements of service providers and maximize their profit, the authors of [83] proposed an intelligent and profile-based model of SLA violation prediction. This prediction model intelligently predicts the consumer's likely resource usage by considering the consumer's reputation from its previous transaction history and determines the level of required resources based on their reliability. While this method can help service providers make decisions about whether to form SLA and recommends how to mitigate those violations to avoid any penalties, it lacks the capability of predicting SLA violation in post-interaction time phase when the SLA contract has already been made between the service provider and the service consumer. To achieve service level agreements, a prediction method based on Bayes model was designed in [72] to predict the mean load over a long-term time interval, as well as the mean load in

consecutive future time intervals by identifying novel predictive features of host load that capture the expectation, predictability, trends and patterns of host load. This prediction model of the workload can help service provider estimate the possibility of whether SLA violation will occur. Meanwhile, the authors of [73] proposed to use the SVM model to predict possible SLA violation before any issue emerges so then remedial action can be taken. While the approaches in [72] and [73] can help the service provider know beforehand whether SLA violation will take place, it lacks the capability of helping service provider evaluate QoS quantitatively. However, service providers need an effective approach to measure and analyze the impact of influential factors on system performance. As a result, this dissertation will develop a novel approach to help service providers evaluate QoS by using techniques of the design and analysis of experiments. As part of the proposed unified framework, this approach uses sensitivity analysis to identify influential input factors of services, produces a fitted model to approximate the relationship between factors and QoS, and uses this fitted model to analyze and predict QoS values. In Chapter 3, this approach will be discussed in more details.

2.4 QoS Optimization

The subject of QoS optimization can be either composite or individual services. For composite services, several approaches that are listed in Table 2.5 suggested to calculate aggregated QoS values of all possible service combinations and choose the one that maximizes the aggregated utility value while satisfying global constraints [89, 90, 91, 92]. While most of these approaches in this direction are proved to be able to optimize QoS of the composite services effectively and efficiently, they become inapplicable for the individual service.

For individual services, resource provision has been utilized in [87] to analyze workloads, to categorize them on the basis of common patterns, and to plan for

Table 2.4: Methods of SLA Violation Prediction

Publication	**Contributions**
Profile-based viable service level agreement(SLA) violation prediction model in the cloud by W Hussain, FK Hussain and O Hussain(2015) [83]	- Introduce an efficient system that predicts SLA violation before it occurs and recommends how to mitigate those violations to avoid any penalties. - Propose a profile-based model of SLA violation prediction from the provider's perspective. - Help service providers in making decisions about whether to form SLA and avoiding SLA violations.
Host load prediction in a Google compute cloud with a Bayesian model by Di Sheng, Derrick Kondo, and Walfredo Cirne(2012) [72]	- Use a Bayesian model to predict the mean workload. - Help service providers judge whether there is a possibility that SLA will be violated.
A machine learning model for detection and prediction of cloud quality of service violation by TS Wong, GY Chan and FF Chua(2018) [73]	- Use SVM model to help the cloud service provider know beforehand whether SLA violation will take place.

workloads before actual scheduling. The authors of [87] later enhanced their work with automated processing in the [88]. Although these methods are able to optimize QoS by allocating more resources, no attention has been given to resource over-provision, which is a serious issue for Web services as it wastes resources and causes the increase of operational cost [38].

In this dissertation, optimization techniques are used to optimize QoS. Good optimization algorithms should possess the properties[97] listed below and should be taken into consideration when designing approaches for QoS optimization.

Table 2.5: QoS Optimization Methods Based on Service Composition

Publication	**Contributions**
Optimal service selection and composition or service-oriented manufacturing network by S Huang, S Zeng, Y Fan and GQ Huang(2011) [89]	- Propose a performance evaluation model to analyse the local and global performance. - Realise the optimal service selection and composition.
Two-stage approach for reliable dynamic Web service composition by ZZ Liu, DH Chu, ZP Jia, JQ Shen and L Wang(2016) [90]	- Proposes a two-stage method for reliable dynamic Web service composition. - Improves the reliability of dynamic Web service composition.
Support timely and accurate adaptation A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing by F Chen, R Dou, M Li and H Wu(2016) [91]	- Optimize the QoS-aware service composition based on multi-objective optimization.
A Multi-Objective Optimization of Cloud Based SLA-Violation Prediction and Adaptation by V Gaur, P Dhyani and OP Rishi(2016) [92]	- Use a GA-Tabu based approach for optimal cloud service composition selection.

1. Robustness: They should perform well on a wide variety of problems in their class, for all reasonable values of the starting point.

2. Efficiency: They should not require excessive computer time or storage.

3. Accuracy: They should be able to identify a solution with precision, without being overly sensitive to errors in the data or to the arithmetic rounding errors that occur when the algorithm is implemented on a computer.

Various types of optimization techniques are used to solve research problems in different fields. These approaches can be broadly classified into two strategies: direct

Table 2.6: QoS Optimization Methods Based on Resource Provisioning

Publication	**Contributions**
Q-aware: Quality of service based cloud resource provisioning by S Singh and I Chana(2015) [87]	- Cloud workloads have been analyzed and clustered through workload patterns. - Identify QoS metrics of each workload. - Analyze the effect of number of workloads and resources on execution time and cost. - Proposed technique for the minimization of cost and time simultaneously while adhering to workload deadline.
STAR: SLA-aware autonomic management of cloud resources by S Singh, I Chana and R Buyya(2017) [88]	- Develop a resource management technique that automatically manages QoS requirements of cloud users. - Help the cloud providers in achieving the SLAs and avoiding SLA violations.

and metamodel-based optimization.

For direct optimization, it is performed by evaluating real objective function. For the Service-Oriented system, the real objective function between influential factors and the performance is usually unknown, which means that the real performance model is regarded as a black box. Each evaluation of the real objective function typically involves the following steps: perform one experiment by configuring values of contextual factors, execute the designed Web services and collect the experimental data. As a result, the evaluation of real objective function for the Service-Oriented system is computationally expensive and involves cost. Meanwhile, when the formula of the real objective function is unknown, the heuristic optimization methods, such as genetic algorithms [98] and simulated annealing [99], are usually used to search the

optimum. These methods have fewer restrictions on the mathematical characteristics of the function, and do not require any gradient information. However, many objective function evaluations are usually needed for heuristic optimization methods, which means that a huge amount of time will be spent on performing these experiments. As a result, the strategy of direct optimization is inapplicable when optimizing the performance of the Service-Oriented system.

For Metamodel-based optimization, it makes use of some surrogate models to provide an efficient representation of the costly true model. When a surrogate model is used for this purpose, it is called a metamodel. This process is called metamodel-based design optimization(MBDO) [94, 95, 96] when metamodels are utilized for the evaluations during the optimization process. Metamodel-based design optimization usually includes three steps as below:

1. Perform a small or modest number of experiments and collect the experimental data;
2. Use these experimental data to fit a set of mathematical surrogate models that are used to approximate the real unknown function;
3. Identify the optimum by searching on the most appropriate surrogate model.

Since these surrogate models are mathematical functions with a specific formula, evaluation on these functions are efficient. As a result, the strategy of metamodel-based optimization will be used to optimize the performance of the Service-Oriented system in this dissertation. Because metamodel-based design optimization originates from techniques of the design and analysis of experiments, the relevant literature review will be discussed in the next subsection : design and analysis of experiments.

2.5 The Design and Analysis of Experiments

2.5.1 Motivation: Why the Design and Analysis of Experiments

In essence, QoS defined in the SLA is the quality attribute requirement for Web services. Whether a system will be able to exhibit its desired quality attributes is substantially determined by its architecture. In other words, quality attribute requirements are satisfied by the various structures designed into the architecture, and the behaviors and interactions of the elements that populate those structures [78]. As a result, a well-designed architecture can help service providers ensure SLA when designing Web services.

When performing architectural design, the architecture should be evaluated for its ability to deliver the system's important quality attributes. This should occur early in the life cycle, when it returns the most benefit, and repeated as appropriate, to ensure that changes to the architecture (or the environment for which it is intended) have not rendered the design obsolete [78]. Meanwhile, there are various architectural tactics that can be used to achieve the requirement of quality attributes, such as provisioning more resources, caching and maintaining multiple copies of computation [79][80]. Service designers need to compare alternative architectural tactics and, hence, make well-informed design decisions when making trade-offs between cost, quality, and performance[33]. For example, when doing performance-driven architectural design for Web services, service designers may make different design decisions based on different environments in which Web services execute. Service designers usually need to ask some questions about the contextual information of Web services.

1. Will SLA be violated in the future?

2. Which factors have the significant influence on the performance?

3. How is the performance impacted by uncontrollable factors?

4. What are the performance bottlenecks that are caused by variations of different uncontrollable factors?

5. Which controllable factors can be used to improve performance effectively?

6. Which architectural tactics related to controllable factors can be used to improve the performance?

To answer these questions, service designers need to gain insight into the performance model and then can make the right design decisions based on this knowledge. Otherwise, a long time will be spent and wasted on the implementation of architectural tactics that will only give us a small percent performance boost. However, most of the answers to these questions are usually unknown during the architectural design stage. As a result, this kind of decision-making process for designing architecture is empirical and need to make use of techniques of the design and analysis of experiments(DAAE) to investigate the performance model.

2.5.2 The Basic Concept Of the Design and Analysis of Experiments

In general, the design and analysis of experiments(DAOE) is to use statistical techniques to analyze the results from designed experiments in an iterative manner. Specifically, we may want to determine which input variables are responsible for the observed changes in the response, develop a model relating the response to the important input variables and to use this model for system analysis, system improvement or other decision-making [8]. Because the results and conclusions that can be drawn from the experiment depend to a large extent on how the data were collected, the experiment is needed to be well-designed via the design of experiments. Design of experiments(DOE) is a test or series of tests in which changes are made on purpose

to the input variables of a process or system so that the causes for changes in the output response can be observed and identified. After data are collected from the well-designed experiment, some statistical techniques will be used to analyze these results to gain insight into the unknown system, to decide how to optimize the system or to make some other decision [8].

The motivation of design and analysis can be traced back to Fisher's work on data analysis for agriculture experiments[75]. Since then, over thousands of activities and publications on it have been conducted and reported to characterize a system or product, to optimizing a product or process, and to design a product. Its applications cover a wide range of disciplines, including biology, agriculture, medicine and pharmacy, electronic and mechanics, and even social and human management[19, 104, 26, 27, 28].

The most commonly used terms in the design and analysis of experiments include controllable input factors, uncontrollable input factors, responses, hypothesis testing, and interaction.

1. Controllable input factors are those input parameters that can be controlled or configured, such as CPU, storage, cache, etc.

2. Uncontrollable input factors are those parameters that can not be controlled,

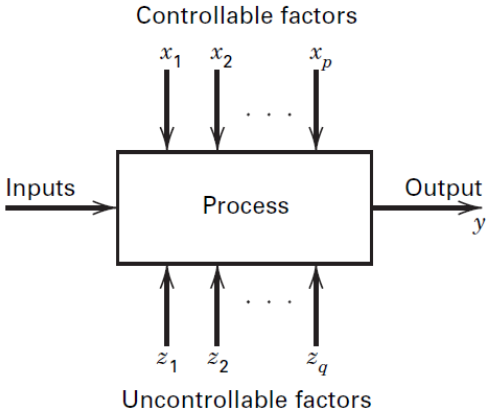

Figure 2.1: General model of a process or system

such as workload, network environment or the requirements of services consumer.

3. Responses or output measures are the elements of the process outcome.

4. Hypothesis testing helps determine the significant factors using statistical methods.

5. An interaction is a situation in which the simultaneous influence of two variables on a third is not additive when an experiment has three or more variables.

In general, experiments are used to study the performance of processes and systems. The process or system can be represented by the model shown in Fig. 2.1. The process can be visualized as a combination of operations, machines, methods, people, and other resources that transforms some input into an output that has one or more observable response variables. Some of the process variables [x1,x2,...xn] are controllable, whereas other variables [z1,z2,...zn] are uncontrollable. The overall objectives of the experiment include[8]:

1. Understanding: two main objectives of the design and analysis of experiments is to learn which factors have the most influence on the response(s) of interest and to understand the behavior of a system. Specifically, in order to avoid wasting valuable resources, to reduce the total number of experiments and to get a surrogate model with better quality, the service designer may be interested in understanding whether some factors, such as workload, CPU, memory, cache, etc, influence the performance of a service and get a list of important factors that have significant impact on the performance. They can apply techniques of DOAE to do sensitivity analysis to screen factors. Meanwhile, the service designer is interested in understanding the performance model that reflects the cause-effects relationship between influential factors and the response time. After constructing one surrogate model approximating the true performance model by using data from the well-designed experiments, it can be explored to help

service designers better understand the performance of Web services and predict SLA violation.

2. Optimization: after the system has been better understood, the next objective is usually optimization. That is to find the settings or levels of the important factors that result in desirable values of the response. An optimization experiment is usually a follow-up after the screening experiment, surrogate model fitting and exploration on this model.

3. Confirmation: in a confirmation experiment, the experimenter is usually trying to verify that the system operates or behaves in a manner that is consistent with some theory or experience.

2.5.3 Performance Analysis via the Design and Analysis of Experiments

To investigate the unknown system, several sensitivity analysis methods relying upon DOE were proposed to perform the factor screen. Sensitivity in this context is a measure of the contribution of an independent variable to the total variance of the dependent data. They can help us find how different model input parameters influence the model output, what are the most influential parameters and how to evaluate such effects quantitatively [39]. Meanwhile, when working with new systems or technologies, valuable resources will not be wasted by focusing on important factors and ignore unimportant ones [8].

By deriving measures of global sensitivity from a set of local derivatives or elementary effects, and sampling on a grid throughout the parameter space, a method of sensitivity analysis was proposed in the [40] to identify the main effects. In this paper, the proposed experimental plans are composed of individually randomized one-factor-at-a-time designs and data analysis is based on the resulting random sample of observed elementary effects. While the method in the [40] can identify the main

effects, it becomes inapplicable in terms of interaction effects. Similiarly, the authors of [41] later use metamodeling approaches to estimate sensitivity indices. Instead of evaluating real model or performing the real experiment, this method first perform several experiments, then fit a surrogate model and finally calculate the sensitivity indices by using the data coming from the evaluation on the fitted surrogate model. Although this method can reduce the number of the experiment, its accuracy depends on the surrogate model. If the fitted model is not accurate enough, the sensitivity indices will be not capable of providing useful information to make decision. To overcome the issues of lack of efficiency and accuracy, the authors of [60] and [21] present the investigation on performance analysis with sensitivity analysis and a generalization for the statistical method to handle two or three factors. The approach in these two papers can be used to analyze main and interaction effects, and require a relatively small number of experiments. However, the effects models and algorithms can only be used to analyze two or three factors. As a result, the research in this dissertaion will enhance and extend the models and algorithms in [60] and [21] to improve its robustness.

In order to gain a deeper insight into the behavior of a system, many papers in a variety of fields use metamodeling techniques to further explore the unknown system and predict the system behavior. Meanwhile, Complex computational models often suffer from over-parameterization that can reduce model quality and increase computational costs [42]. As a result, before using metamodeling techniques to further explore the behavior of a system, sensitivity analysis can be used to exclude unimportant parameters to potentially improve model quality and to reduce parametrization. In recent years, many papers [19, 104, 26, 27, 28] covering a wide range of fields, including intelligent manufacturing, civil engineering, powder technology, electronic and mechanics, use techniques combing metamodel-based analysis with sensitivity analysis to evaluate the performance model. As a result, techniques of the design and analysis of experiments have already been proven to be an effective way to model the cause-effects relationship between influential factors and the performance, to explore

unknown performance system, and to optimize the performance. In this dissertation, sensitivity analysis combined with the metamodel technique will be employed to evaluate the performance model of Web services. In Chapter 3, this approach will be discussed in more details.

2.5.4 Performance Optimization via the Design and Analysis of Experiments

For optimization, many metamodel-based design optimization (MBDO) methods that rely upon DOE were proposed. Response surface methodology is the most popular metamodel-based optimization method used in recent years [100]. It has been studied and used for optimization in a variety of different fields [101, 102, 103, 104, 105] due to its efficiency and accuracy for finding the optimum. However, RSM uses the linear and quadratic model to approximate the real response model and can only guarantee to find a local optimum. When the real response model of the system is complex and has multiple local minimum or maximum optima, the global optimum can not be obtained in some cases. As a result, the global metamodel-based optimization approaches [106, 107, 108, 109, 110] are used to overcome this issue of being trapped by the local optimum and have a better chance of pinpointing the global optimum. These methods fit one or more global metamodels following by searching global optimum on those global models. When the global metamodel is used to approximate the true unknown model, it need to accurately capture the real model of the system to ensure the searching efficiency and accuracy. However, it is difficult to build a highly accurate global model because the error occurring in the global surrogate model will usually become larger than that in the local surrogate model when the global region becomes large. As a result, the global and local metamodel-based optimization approaches should be combined to take advantage of their merits. In this dissertation, QoS optimization is carried out by a guided search to globally identify subregions with local optima and to locally locate the best optimum with constraints

to minimize operational cost. In Chapter 4, this approach will be discussed in more details.

Chapter 3

Sensitivity and Metamodel-Based Analyses for Services Design

3.1 Introduction

In support of software development based upon Service-Oriented Architecture, service providers use quality of service (QoS) to guide their design of services. By utilizing historical information of QoS collected from service consumers operating similar software systems, a group of approaches [12, 13, 15, 16, 17, 74] has recently been developed to analyze and predict QoS values for new systems with impressive accuracy as long as these systems fall into the same category of application domain, e.g., for buying, watching, or listening online [61]. Based upon collaborative filtering, those approaches rely on a common assumption that multiple software systems are likely to support similar functionality. After evaluating QoS for a large part of their services, the results are valid for the remaining services [62]. However, when the new systems under development are for different application domains, their design involves significant structural or contextual changes and thus destroys the applicability of collaborative-filtering-based approaches.

In the demand for a more general approach that is capable of predicting quality of services without the constraints on application domains, this chapter proposes a framework to use sensitivity analysis for the identification of influential factors with dominating impacts on the quality of services and use metamodel-based analysis to select a fitted surrogate model for the prediction of potential violation of SLA. In the remaining of this chapter, Section 3.2 provides details of sensitivity analysis and metamodel-based analysis after introducing a unified framework to integrate the two processes together. Afterwards, Section 3.3 goes through the main steps with experiment results on a real Web service and uses it as an example to illustrate the use of this framework for service design. Finally, this chapter concludes with discussions and future work in Section 3.4.

3.2 Metamodeling for Prediction of SLA Violation

Every service in an SOA system has a number of parameters as input variables, whose values can be either discrete or dividable into discrete segments. Assuming the true performance model of a service can be expressed as a function in (3.1),

$$Y = f(A^{(1)}, A^{(2)}, \cdots, A^{(m)}) \tag{3.1}$$

where Y is the performance of service under the influence of m input factors $A^{(v)}$, $1 \leq v \leq m$. As the true functional relationship between performance and factors is unknown, the prediction of SLA violation needs an approximation model $g(\cdot)$, or a metamodel, to imitate the behavior of $f(\cdot)$.

$$\hat{Y} = g(A^{(1)}, A^{(2)}, \cdots, A^{(m)}) \tag{3.2}$$

In practice, the metamodel will be different from the real model in most cases, i.e.,

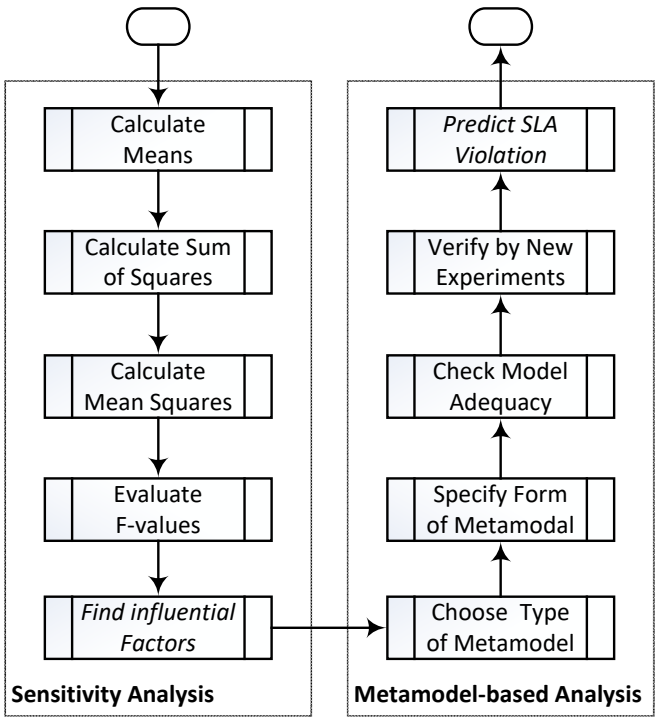

Figure 3.1: A Unified Framework of Sensitivity and Metamodel-Based Analyses

$Y = \hat{Y} + \varepsilon$, but $g(\cdot)$ is an adequately fitted model if the difference ε is less than a predetermined threshold. Presented in this section is a unified framework (Fig. 3.1) that integrates the processes of sensitivity analysis for the identification of influential factors and metamodel-based analysis for the prediction of SLA violation.

3.2.1 Multi-Factor Sensitivity Analysis

Sensitivity analysis is the study of how the uncertainty in the output of a system or mathematical model can be apportioned to different sources of uncertainty in its inputs [18]. As illustrated in (3.3) for a service with m input factors $A^{(v)}$, $1 \leq v \leq m$, the observation of an experiment with each factor taking a specific value indexed at i_v, $1 \leq i_v \leq a_v$, is the system's output $Y_{i_1 \cdots m l}$ collected with a performance metric, where $1 \leq l \leq n$ for n experiments.

$$Y_{i_1 \cdots m l} = \mu + Y'_{i_1 \cdots m} + Y''_{i_1 \cdots m} + \varepsilon_{i_1 \cdots m l} \tag{3.3}$$

In (3.3), μ is the population mean obtained from the average of all n observations per experiment arrangement, and $\varepsilon_{i_{1\cdots m}l}$ is a random error. In addition, the components of $Y'_{i_{1\cdots m}}$ and $Y''_{i_{1\cdots m}}$ counter in the influence of all factors individually and jointly as formulated in (3.4) and (3.5) respectively.

$$Y'_{i_{1\cdots m}} = \sum_{v=1}^{m} A_{i_v}^{(v)} \tag{3.4}$$

$$Y''_{i_{1\cdots m}} = \sum_{w=1}^{m-1} \sum_{u=1}^{m-w} \left(\prod_{v=u}^{w} A_{i_v}^{(v)} \cdot \sum_{v=u+w+1}^{m} A_{i_v}^{(v)}\right) \tag{3.5}$$

The preparation for sensitivity analysis needs a total number of $n \times \prod_{v=1}^{m} a_v$ experiments, from each of which an observed response $y_{i_{1\cdots m}l}$ is collected. The overall process of analysis goes through five steps to calculate individual/joint means, sum of squares, mean squares, and F distribution values for the identification of influential factors. In the first step, the overall mean $\hat{y_{i_{1\cdots m}}}$ is obtained from (3.8), and all the other individual and joint means are obtained by simply taking out corresponding items from the nested sums in the dividend and from the product in the divisor of (3.8). For example, (3.6) tells how to calculate individual mean $\hat{y_{i_2}}$ for factor $A^{(2)}$ and (3.7) to calculate the joint mean $\hat{y_{i_{23}}}$ between factors $A^{(2)}$ and $A^{(3)}$.

$$\hat{y_{i_2}} = \frac{\sum_{i_1=1}^{a_1} \sum_{i_3=1}^{a_3} \cdots \sum_{i_m=1}^{a_m} \sum_{l=1}^{n} y_{i_{1\cdots m}l}}{a_1 a_3 \ldots a_m n} \tag{3.6}$$

$$\hat{y_{i_{23}}} = \frac{\sum_{i_1=1}^{a_1} \sum_{i_4=1}^{a_4} \cdots \sum_{i_m=1}^{a_m} \sum_{l=1}^{n} y_{i_{1\cdots m}l}}{a_1 a_4 \ldots a_m n} \tag{3.7}$$

$$\hat{y_{i_{1\cdots m}}} = \frac{\sum_{i_1=1}^{a_1} \sum_{i_2=1}^{a_2} \cdots \sum_{i_m=1}^{a_m} \sum_{l=1}^{n} y_{i_{1\cdots m}l}}{n \prod_{i=1}^{m} a_i} \tag{3.8}$$

Let $\hat{y_I}^2$ denote the square of overall mean $\hat{y_{i_{1\cdots m}}}$ and $\sum^{a_I}$symbolize the nested sums $\sum_{i_1=1}^{a_1} \sum_{i_2=1}^{a_2} \ldots \sum_{i_m=1}^{a_m}$. The second step of calculation produces the sum of squares SS_{i_v} for each individual factor $A^{(v)}$ with (3.9), the sum of squares $SS_{i_{1\cdots m}}$ for all m factors with (3.10), and the error of sums of squares with (3.13), where SS_J is specified in (3.11).

$$SS_{i_v} = \frac{a_v}{n\prod_{i=1}^{m} a_i}\sum_{i_v=1}^{a_v} \hat{y_{i_v}} - \hat{y_I}^2 \tag{3.9}$$

$$SS_{i_{1\cdots m}} = \frac{1}{n}\sum^{a_I} y^2_{i_1\ldots i_m} - \hat{y_I}^2 - \sum_{i=1}^{m} SS_{i_i} - SS_J \tag{3.10}$$

$$SS_J = \sum_{w=1}^{m-1}\sum_{u=1}^{m-w} (SS_{\prod_{v=u}^{w} i_u \sum_{v=u+w}^{m} i_v}) \tag{3.11}$$

$$SS_T = \sum^{a_I} y^2_{i_1\ldots i_m} - \hat{y_I}^2 - SS_J \tag{3.12}$$

$$SS_E = SS_T - \sum_{i=1}^{m} SS_{i_i} - SS_J \tag{3.13}$$

In addition, after converting $1/n$ to $\frac{\prod_{i=1}^{m} a_i}{n\prod_{i=1}^{m} a_i}$, the sum of squares for any other combinations of factors are obtained by keeping $\hat{y_I}^2$ and those items related to the factors in the numerator $\prod_{i=1}^{m} a_i$ and all sum components in (3.10). For example, (3.14) illustrates the calculation of $SS_{i_{23}}$ for two factors $A^{(2)}$ and $A^{(3)}$.

$$SS_{i_{23}} = \frac{a_2 a_3}{n\prod_{i=1}^{m} a_i}\sum_{i_2=1}^{a_2}\sum_{i_3=1}^{a_3} \hat{y_{i_{23}}} - \hat{y_I}^2 - SS_{i_2} - SS_{i_3} \tag{3.14}$$

The third step of sensitivity analysis uses the results from the previous step to produce mean squares with (3.15) for the whole list of factors or any individual or joint factors by using the corresponding sum of squares as the dividend and keeping only related items in the product of the divisor. As illustrated in (3.16) for $F^{A^{(1\cdots m)}}$, the calculation of F distributions in the fourth step is a simple division of a mean square with MS_E, which is the mean square error given by (3.17).

$$MS_{A^{(1\cdots m)}} = \frac{SS_{i_1\cdots i_m}}{\prod_{i=1}^{m}(a_i - 1)} \tag{3.15}$$

$$F^{A^{(1\cdots m)}} = \frac{MS_{A^{(1\cdots m)}}}{MS_E} \tag{3.16}$$

$$MS_E = \frac{SS_E}{(n-1)\prod_{i=1}^{m} a_i} \tag{3.17}$$

Finally, the process of sensitivity analysis compares each F distribution value with the cumulative F distribution value F_{α,df_1,df_2}, where α is a confidence level, df_1 is the degree of freedom associated with the numerator of the mean square, and df_2 is the degree of freedom associated with the denominator the mean square [52]. A significant impact on the performance of a service can be identified if the F distribution value of an individual factor or a joint group of several factors exceeds F_{α,df_1,df_2}. By the end, this process produces a list of influential factors, each of which has significant impact on service performance individually or jointly with other factors.

3.2.2 Metamodel-based Performance Analysis

In the process of sensitivity analysis, the performance model $f(\cdot)$ of a service is treated as a blackbox as influential factors are identified by analyzing only the outputs of selective inputs. As the prediction of SLA violation is expected to work in all use cases of the service, the proposed framework fits the approximation model $g(\cdot)$ with a surrogate model and predicts service performance by evaluating this fitted model. While the elimination of insignificant factors in the previous process of sensitivity analysis greatly reduces the computational costs caused by over-parameterization of metamodeling techniques, the process of metamodel-based analysis starts with the identified influential factors and produces a surrogate model for the prediction of SLA violation. As illustrated in Fig. 3.1, the overall process goes through the following steps.

Choose the type of metamodel

This step requires experiments to be conducted with the influential factors using a chosen experimental strategy, which could be the Taguchi's orthogonal arrays or any of the others as suggested in [7][8][115]. By inspecting the relationship between input factors and service performance after plotting experiment results onto a graph,

this step uses the distribution of marks in the graph to select a type of metamodel among the choices of polynomial regression, multivariate adaptive regression splines, support vector regression, or regression tree.

Specify the form of the metamodel

As a metamodal is not complete without specifying a form, the next step determines a selection of form from different candidates for the selected metamodel type. While linear model, two-factor interaction (2FI) model, and quadratic model are commonly used in polynomial regression, polynomial kernel and radial basis function kernel should be used for support vector regression.

Check the adequacy of metamodel

The objective of this step is to check the fitting quality of chosen metamodel in the selected form with experiments. If the candidate model lacks the required quality, the process of metamodel-based analysis repeats the work in the previous two steps to choose another metamodel. Otherwise, this step determines the most appropriate form of the chosen metamodel by evaluating its quality using a polynomial model defined by its coefficient of determination R^2, adjusted coefficient of determination R^2_{adj}, and predicted coefficient of determination R^2_{pdt} as formulated below,

$$R^2 = 1 - SS_E/SS_T \tag{3.18}$$

$$R^2_{adj} = 1 - R^2 \times DOF_{SS_T}/DOF_{SS_E} \tag{3.19}$$

$$R^2_{pdt} = 1 - SS_{PE}/SS_T \tag{3.20}$$

where SS_{PE} is prediction error sum of squares, and SS_E and SS_T are the error sum of squares and total sum of squares defined in (3.13) and (3.12) respectively, with their

corresponding degree of freedom given by DOF_{SS_E} and DOF_{SS_T}. The fitted model is adequate only if the values of the three coefficients are within the range between 0.9 and the maximum of 1.0. Among them, R^2 and R^2_{adj} are used to determine how well the model fits the experiment observations, and R^2_{pdt} tells how well the model predicts responses for new observations. The larger value R^2_{pdt} has, the greater predictive capability the fitted model exhibits.

Verify the metamodel with new experiments

After the analysis finishes validating the metamodel, it continues to assess the model's accuracy when being used to predict service performance in different use cases. This assessment leads to a new series of experiments. Let a new observation be $x_0 = [1, x_{01}, x_{02}, \cdots, x_{0n}]$. A point estimate $\hat{y}_0$ for the future observation y_0 at the point $(x_{01}, x_{02}, \cdots, x_{0n})$ is calculated with the fitted model. The following formula gives the $(1-\alpha)\times 100\%$ prediction interval for y_0,

$$\hat{y}_0 \pm t_{\alpha/2,n-p} \times \sqrt{MS_E + se(\hat{y}_0)^2} \tag{3.21}$$

where $\sqrt{MS_E + se(\hat{y}_0)^2}$ is the standard error of prediction, and $t_{\alpha/2,n-p}$ is the t-multiplier with the same degree of freedom as the mean square error MS_E.

Predict SLA violation

Finally, the residual error between predicted and validated response time are calculated to check how well the fitted model predicts the response time in the future. Unless a rejection results the whole process going back to step one for model selection, the fitted model is valid and prediction will be accurate when the new observed response time lies within the 90% prediction interval and the residual errors are less

that 10%. At this time, the metamodel is ready to be used for quantitative analysis of the influence of different factors on service quality and for the predication of situations when SLA could be violated.

3.3 Experiments

3.3.1 Experiments Setup

To examine the effectiveness of the proposed approach, experiments have been conducted on a Web service that belongs to an e-commerce system. A typical e-commerce system is complicated and consists of a variety of modules, such as search engine, order processing, payment, discount tools, etc. The Web service on which experiments will be performed is one of the components of the e-commerce system and is capable of generating a list of top ten webpages linking to different e-commerce websites with the lowest prices for certain products. It can be used to help users compare one item's price sold in this e-commerce website with that of other e-commerce websites and make a reasonable price policy. This Web service uses the ASP.NET Web API and consists of a product controller, a thread manager, a cache manager, product services, and product DAL. The experiments were implemented by employing .NET Framework 4.5, Visual Studio 2017 Professional and SQL Server 2012 on Windows 10 Professional with Inter(R) Core(TM) i5-4210U CPU and 8G RAM.

3.3.2 Design and Perform Experiemnts

Among the candidate factors that may influence the quality of this Web service in terms of response time, three are considered potentially significant, i.e., the number of sources , the number of products and the number of users to search, which are labeled A, B and C respectively in Table 3.1. As the five, five and three different levels of

Table 3.1: Data Collection

		A				
B	**C**	50	100	150	200	250
10,000	1	35.702	67.2039	87.605	119.807	142.508
	2	35.621	68.9726	87.8588	119.689	145.796
	3	35.8181	67.9856	85.1732	117.995	143.657
20,000	1	52.103	95.7055	135.408	189.911	242.014
	2	54.2969	100.978	132.662	192.722	240.799
	3	53.4769	96.8925	131.261	189.251	241.956
30,000	1	60.4034	114.507	172.21	232.113	297.317
	2	59.154	112.976	174.699	235.977	295.266
	3	58.9963	114.677	175.118	231.928	297.665
40,000	1	82.2047	138.208	220.113	294.717	353.92
	2	83.6584	133.922	221.019	292.562	352.827
	3	82.9885	139.655	217.923	293.379	356.284
50,000	1	85.6049	157.809	259.715	340.119	402.923
	2	87.2549	160.143	255.222	340.876	401.116
	3	84.2962	159.657	256.799	345.358	405.328

factors A, B and C showing in the table generate 75 different combinations, a total of 150 experiments are conducted to collect two measurements for each combination. Provided in each cell of Table 3.1 is the average of observations in milliseconds.

Table 3.2: Variance Table

Source	Sum of Square	DOF	Mean Square	F-Value
Factor A	973897.464	4	243474.366	1592.790
Factor B	463802.886	4	115950.721	758.540
Factor C	480.904	2	240.452	1.573
Joint AB	111626.179	16	6976.636	45.640
Joint AC	2044.983	8	255.622	1.672
Joint BC	2222.770	8	277.846	1.817
Joint ABC	7155.858	32	223.620	1.462
Error	11464.517	75	152.860	
Total	1572695.561	149		

3.3.3 Perform Screen Design

The five-step process of sensitivity analysis produces a variance table in Table 3.2. For a confidence level of 95%, factor A has a significant impact on the response time as its F-value exceeds 2.50, the value of $F_{0.05,4,75}$, and so does factor B. However, factor C does not influence the response time significantly because its F-value is less than 3.13, the value of $F_{0.05,2,75}$. In addition, the response time of the Web service is largely affected by the joint effect of A and B due to the fact that the interaction F-value exceeds the value of $F_{0.05,16,75}$. The result of sensitivity analysis shows that the Web service is sensitive to the change of both factors A and B, and is insensitive to factor C. As a result, while factor C is removed from the factor list, factors A and B will be used as input factors to fit the surrogate model, with the experiments shown in Table 3.3 for the next process of metamodel-based analysis.

Table 3.3: Data Collection

Factor	**Factor A**				
B	50	100	150	200	250
10,000	35.702	67.2039	87.605	119.807	142.508
20,000	52.103	95.7055	135.408	189.911	242.014
30,000	60.4034	114.507	172.21	232.113	297.317
40,000	82.2047	138.208	220.113	294.717	353.92
50,000	85.6049	157.809	259.715	340.119	402.923

3.3.4 Specify One Appropriate Type of Metamodel

After performing sensitivity analysis to identify that both factor A and B have a significant influence on the response time, the next step is to perform experiments based on these two influential factors, to specify an appropriate type of metamodel and to fit its surrogate model. The experiment data in Table 3.3 will be used to fit the surrogate model. It can be found that the performance behavior in this case is low-order nonlinearity by an initial analysis of these experimental data. As a result, the polynomial regression model is selected as the metamodel type.

3.3.5 Specify a Series of Candidate Forms for the Metamodel and Construct These Surrogate Models

The six models listed in the second column of Table 3.4 become the candidate forms of polynomial regression model. For each of the six candidate forms, a fitting with the method of least squares [7] produces their R^2, R^2_{adj}, and R^2_{pdt} values (Table 3.4). As quartic model yields the largest value of R^2_{pdt}, it is selected to predict the performance of the Web service, and the final approximating function of the polynomial regression model is formulated in (3.22) for response time (rt) in reference to

Table 3.4: R^2 ,R^2_{adj} and R^2_{pdt} of fitted models

No	**Fitted Response Model**	R^2	R^2_{adj}	R^2_{pdt}
1	linear	0.9253	0.9185	0.8904
2	2FI(Factor Interaction)	0.9935	0.9925	0.9888
3	quadratic	0.9949	0.9935	0.9893
4	cubic	0.9969	0.9950	0.9884
5*	quartic	0.9993	0.9984	0.9929
6	fifth	0.9999	0.9994	0.9920

the values of factors A and B, where a_A stands for the number of sources and a_B for the number of products.

$$\begin{aligned} rt \;=\; & -106.35207 + 1.27249 \times a_A + 0.017516 \times a_B \\ & -2.79588 \times 10^{-5} \times a_A \times a_B \\ & -8.39782 \times 10^{-3} \times a_A^2 \\ & -9.58326 \times 10^{-7} \times a_B^2 \\ & +6.03267 \times 10^{-7} \times a_A^2 \times a_B \\ & -7.03235 \times 10^{-10} \times a_A \times a_B^2 \\ & +1.32674 \times 10^{-5} \times a_A^3 \\ & +2.41739 \times 10^{-11} \times a_B^3 \\ & -3.50836 \times 10^{-12} \times a_A^2 \times a_B^2 \\ & -9.37387 \times 10^{-10} \times a_A^3 \times a_B \\ & +1.64843 \times 10^{-14} \times a_A \times a_B^3 \\ & +6.13368 \times 10^{-9} \times a_A^4 \\ & -2.15762 \times 10^{-16} \times a_B^4 \end{aligned} \tag{3.22}$$

3.3.6 Check Model Adequacy

After fitting and identifying the polynomial regression model, several assumptions have to be checked to determine whether this fitted model is adequate.

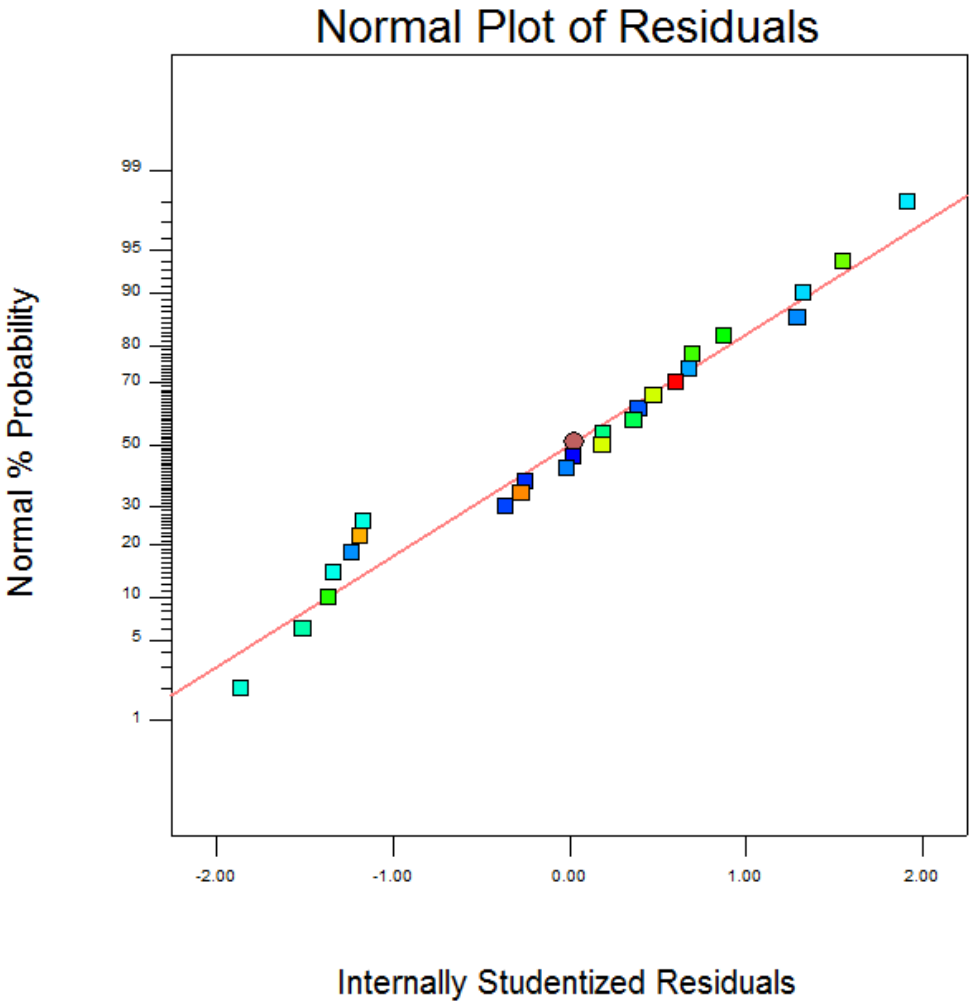

Figure 3.2: Normal Plot Of Residuals.

To check whether residuals are normally distributed, a probability plot shown as Fig. 3.2 is used. By analyzing the normal plot of residuals, it is obvious that these points form a line. Consequently, a conclusion can be drawn that residuals come from a normal distribution.

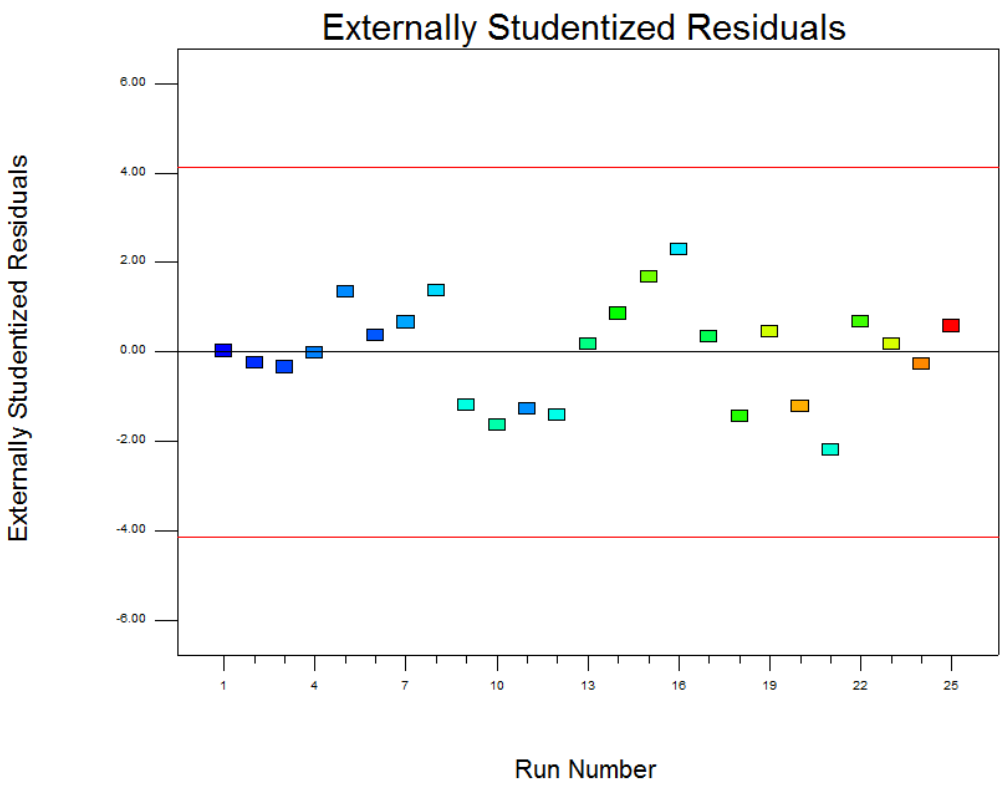

Figure 3.3: Externally Studentized Residuals.

To check whether the residual variance is constant, a plot with externally studentized residuals shown as Fig. 3.3 is used. It shows that the average of the residuals remains approximately 0 and the variation of the residuals appears to be roughly constant. As a result, residual variance is constant.

To check whether residuals are independent across the runs, a plot with residuals vs. Run shown as Fig. 3.4 is utilized. It shows that there is no strong linear or simple nonlinear trend in this plot. Therefore, residuals are independent across the runs.

In addition to residual normality, variance, autocorrelation, how accurate the fitted model is should be checked as well. The plot of predicted VS. atual shown as Fig. 3.5 is used, and it shows that this fitted model is accurate. Consequently, there is a strong correlation between the predicted response time of the fitted model and

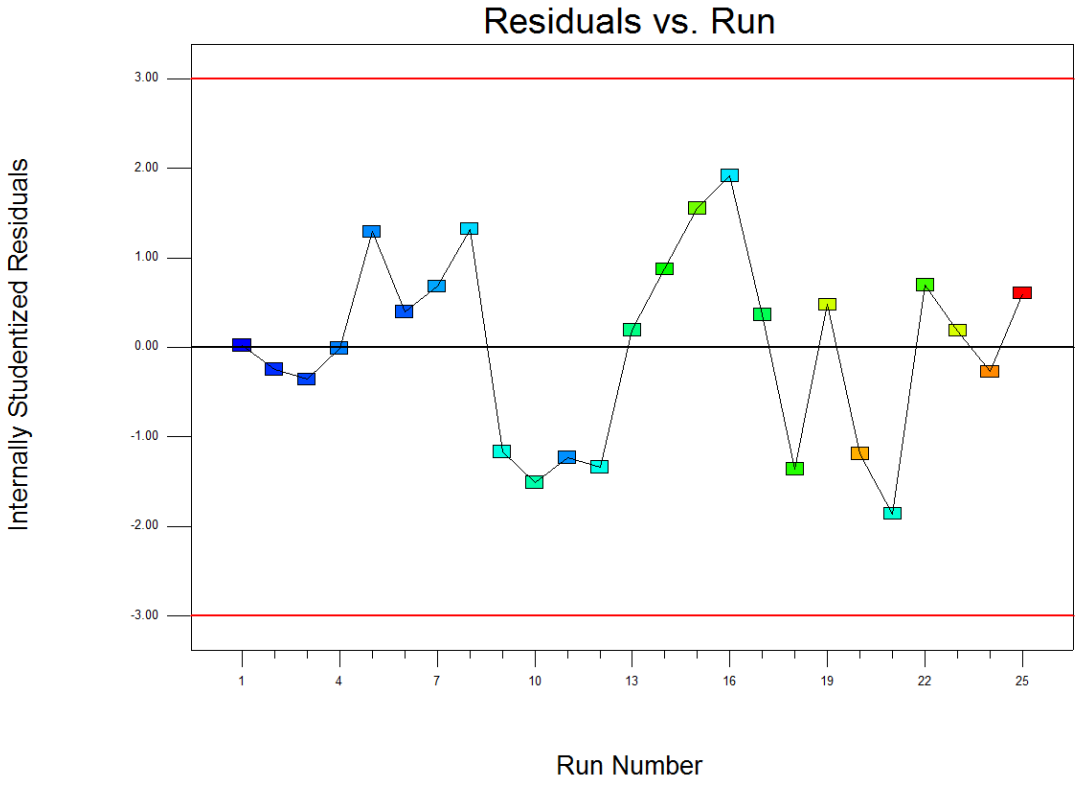

Figure 3.4: Residuals VS. Run.

corresponding actual observed results.

3.3.7 Verify the Fitted Model

Once the model adequacy is validated, model confirmation should be performed to assess the accuracy of the prediction of the fitted model in the future. As a result, another round of ten different experiments produces new results as shown in Table 3.5, which indicates that the fitted model provides an accurate approximating function to the true function for predicting response time within 90% (90%-) prediction interval and the residual error relative to predicted values are less than 10% (90%+) in the future.

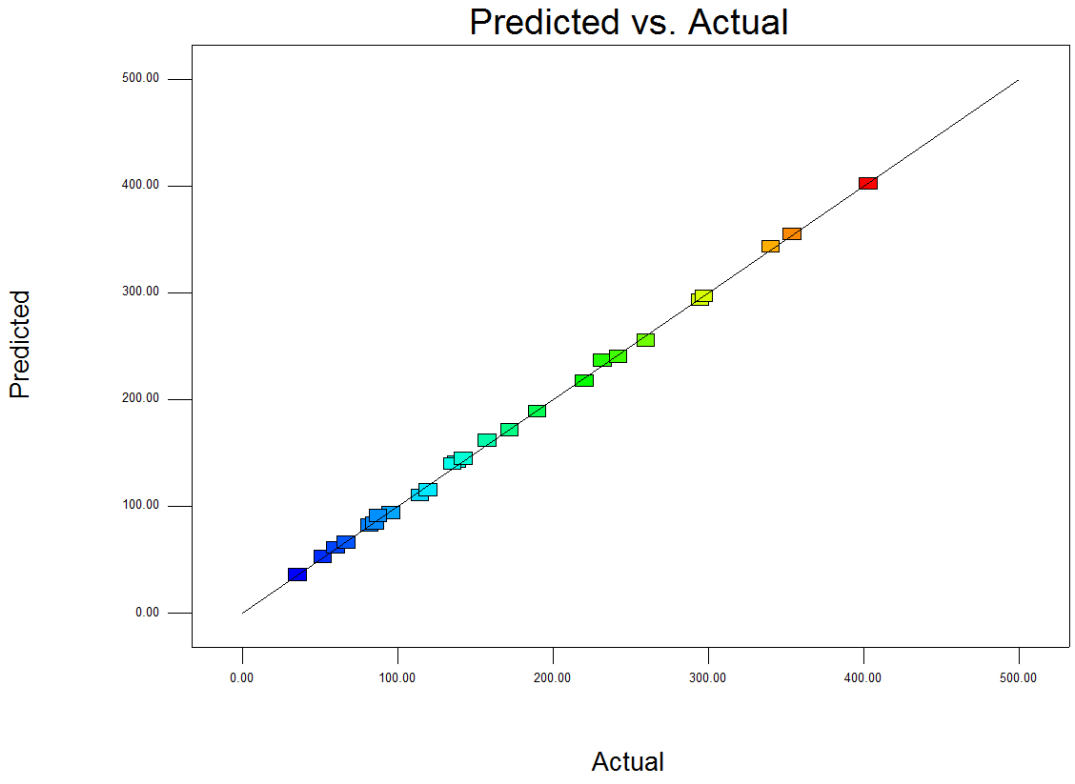

Figure 3.5: Predicted VS. Actual.

3.3.8 Analyse the Effects of Factors and their Interaction on Response Time

All the results of analyses indicate with consistent evidence that the selected model can be used for predicting the response time of the Web service under examination when the numbers of sources and products change due to structural or contextual variation of different SOA systems. As a result, this model can be used to analyze the effect of the number of sources, number of products and their interaction on response time and to predict SLA violation.

Graphically, a three-dimensional response surface plots as shown in Fig. 3.6 can be used to represent this fitted surrogate model and to help analyze and understand

Table 3.5: Results of Verifying Experiments

A	B	Actual	Predict	90%-	90%+	Error
75	20,000	70.904	72.410	62.883	81.938	1.506
75	50,000	123.807	119.225	109.375	129.075	-4.582
125	10,000	70.804	79.035	69.766	88.3054	8.231
125	30,000	142.308	140.003	131.254	148.752	-2.305
125	40,000	175.210	178.518	169.772	187.265	3.308
175	10,000	97.805	102.767	93.4969	112.036	4.962
175	20,000	162.009	163.903	155.156	172.65	1.894
175	40,000	248.414	256.237	247.49	264.984	7.823
225	30,000	269.315	267.621	259.688	275.554	-1.694
225	50,000	355.82	377.488	368.573	386.402	21.668

both the main and the interaction effects of these two factors. In terms of the number of sources, the change of the number of sources from 50 to 250 results in a significant increase in response time. This reflects the great effect of the number of sources on the response time. In terms of the number of products, the increase of the number of products from 10000 to 50000 results in a significant increase in response time as well. This shows the significant impact of the number of products on the response time. The interaction of these two factors have significant effects on the response time both in Fig. 3.6 and Fig. 3.7. Specifically, an increase in the number of sources results in the rise of response time and the rate of increase when the number of products increases from 10000 to 50000.

A further contour plot of the three-dimensional fitted surface model by the response time with respect to the two factors produces a map in Fig. 3.8. This figure shows that when both the number of sources and the number of products increase together, response time of the service has a tendency of going up from the blue region

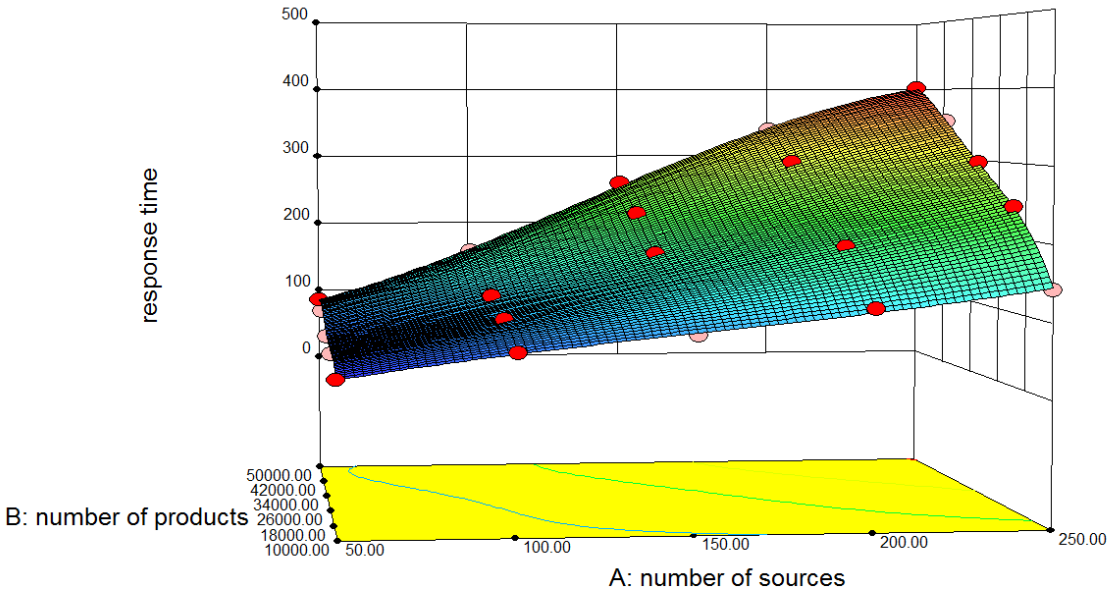

Figure 3.6: Three-dimensional Response Surface Plot of the fitted model.

for 100 milliseconds or less to the red region for higher than 300 milliseconds. The significant increase of response time caused by the two factors is a clear indication of their impact on service performance.

Suppose that an SOA system requires in its SLA that the response time must be less than 50.0 milliseconds. An analysis with the fitted model predicts that the response time cannot meet the requirement when the number of sources increases from 50 to 250 or the number of products goes up from 10000 to 50000. To avoid the potential violation of SLA, the service provider may decide to either improve the service design or reconfigure the service. As the increase of response time is mainly caused by the time reading data from disk to memory when the number of sources and products starts to go up, the disk IO can be identified as the performance bottleneck. Accordingly, the service designer can make a decision to adapt caching strategy as an effective way to boost the performance among some candidate architectural tactics, rather than to choose provisioning more resources and maintaining multiple copies of

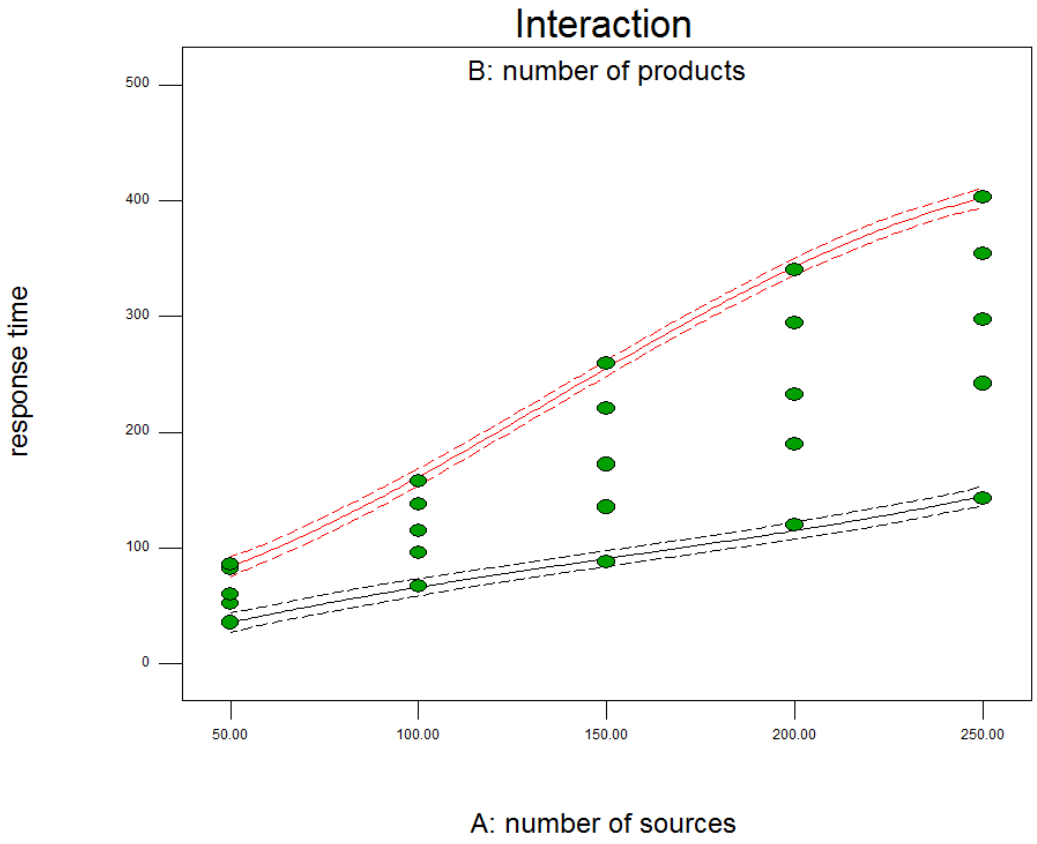

Figure 3.7: Interaction between the Number of Sources and the Number of Products with Respect to Response Time.

computation. This insight gained from performance analysis can help service designer avoid wasting time on the implementation of architectural tactics that may deliver small improvements with a lot of work.

3.3.9 Prediction Accuracy Comparison

The root mean squared error(RMSE) and mean absolute error (MAE) are used to measure and evaluate the accuracy of prediction. RMSE is defined as:

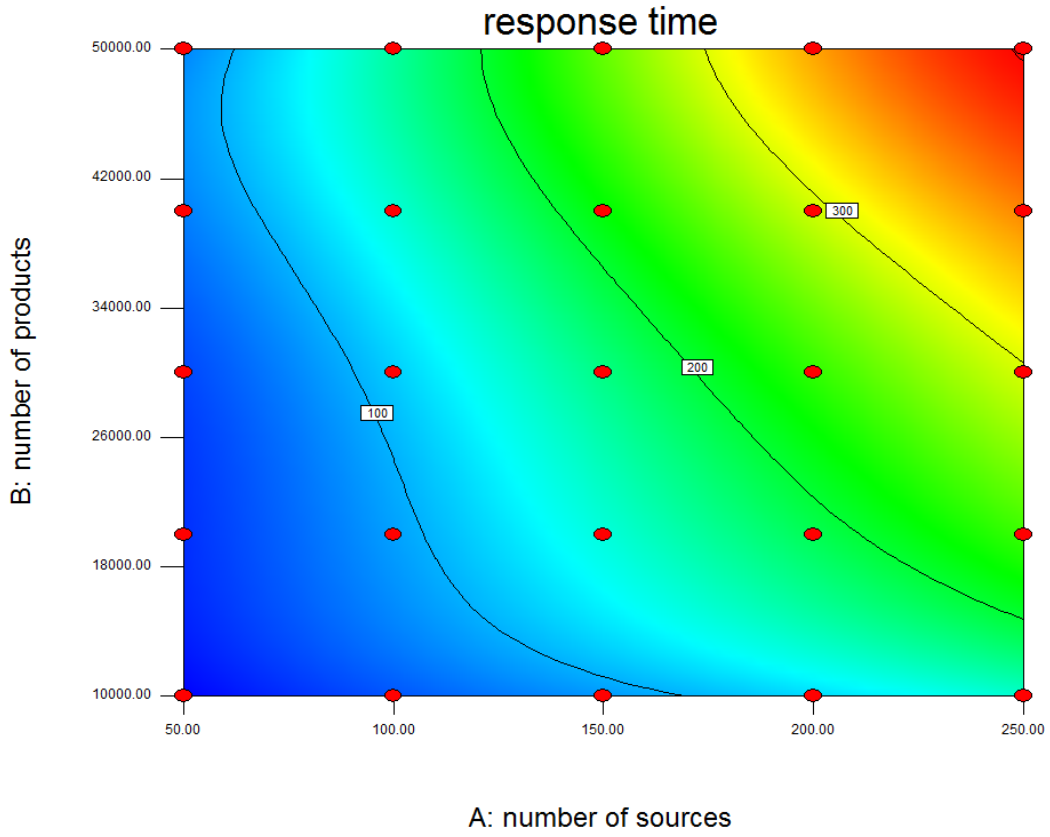

Figure 3.8: Contour plot between the number of sources and the number of products with respect to response time.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{3.23}$$

MAE is defined as:

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N} \tag{3.24}$$

where y_i denotes the ith observerd QoS value, $\hat{y}_i$ is the corresponding predicted QoS value, and N is the number of predicted values.

A new series of experiments were conducted and 20 observations were collected to assess the prediction accuracy among 25%, 50% and 100% experimental regions.

Table 3.6: Comparison Based on Different Number of Factors

Model	**RMSE (25%)**	**MAE (25%)**	**RMSE (50%)**	**MAE (50%)**	**RMSE (100%)**	**MAE (100%)**
Model 1	8.3517	7.8511	18.5245	12.2719	14.0389	9.2505
Model 2	44.3053	35.2979	71.5077	56.4806	57.2106	43.5665
Model 3	7.8136	6.7709	16.6585	11.7055	12.7683	8.7619

Then, the prediction accuracy were first compared among the following three models:

1. Mode 1 is fitted by using three candidate factors without utilizing sensitivity analysis to remove the insignificant factors from the model.

2. Model 2 is fitted by selecting two factors randomly from three candidate factors and performing sensitivity analysis on these two factors.

3. Model 3 is fitted by using significant factors after performing sensitivity analysis on three candidate factors.

Results in Table 3.6 demonstrate that model 3 is able to consistently achieve higher accuracy among 25%, 50% and 100% experimental regions, and therefore outperforms the other models in term of prediction accuracy. The comparison between model 3 and model 1 shows that it is necessary to use sensitivity analysis to identify significant factors and ignore insignificant ones before doing metamodel-based performance analysis. It helps to solve the issue of overfitting the model and improve the prediction accuracy. Meanwhile, the comparison between model 3 and model 2 shows that it is necessary to improve the robustness of the sensitivity analysis method. In its original form [21], sensitivity analysis works with no more than two factors. As a result, two factors have to be randomly selected among three candidate factors, which happened in experiment to be the number of users and the number of sources. It causes the issue of underfitting the model because the number of products is a significant factor but not be used to fit the model. As a result, it is necessary to

Table 3.7: Comparison Based on Different Models

Model	**RMSE (25%)**	**MAE (25%)**	**RMSE (50%)**	**MAE (50%)**	**RMSE (100%)**	**MAE (100%)**
Linear Model	15.4784	12.7151	23.3908	16.7449	22.5625	17.2117
2FI Model	10.0049	9.0506	20.3665	14.3188	15.5506	10.7687
Quadratic Model	8.4832	8.0019	18.6009	12.4312	14.0881	9.2929
Cubic Model	8.4006	6.7427	17.0605	12.2050	13.1324	8.9688
Quartic Model	7.8136	6.7709	16.6585	11.7055	12.7683	8.7619

propose an approach of sensitivity analysis with high robustness as the one developed in this dissertation.

The value of R^2_{pdt} determines how well a model will predict the response time for new observations. According to the largest R^2_{pdt} value of the quartic model, it is selected as the most appropriate form of the metamodel to predict response time and expected to have a better prediction accuracy than the other four models. To verify this, another round of comparison of the prediction accuracy were made between a series of models, including linear model, factor interaction model, quadratic model, cubic model, and quartic model. All these models are fitted by using significant factors after performing sensitivity analysis on three candidate factors. The smallest RMSE and MAE values of the quartic model, shown in Table. 3.7, suggest that it outperforms the other models in terms of prediction accuracy. As a result, it is necessary to use the R^2_{pdt} to select the most appropriate one from a series of candidate forms for the metamodel.

3.4 Summary

This chapter presents a framework that uses sensitivity analysis to identify influential input factors of services, produces a fitted model to approximate the relationship between factors and service quality, and uses this fitted model to analyze and predict QoS values. The process of QoS evaluation can be used by service consumers to select the most appropriate services when constructing an SOA system. Meanwhile, service providers can use the evaluation results to find out if and when SLA might be violated due to the fluctuation in different factors, to better understand the performance of a service with its fitted model, and to identify performance bottlenecks caused by influential factors. Future research will investigate automation of the proposed framework by using data mining and machine learning.

Chapter 4

A Strategy of QoS Optimization for Web Services

4.1 Introduction

The subject of QoS optimization can be either composite or individual services. For composite services, several approaches suggested to calculate aggregated QoS values of all possible service combinations and choose the one that maximizes the aggregated utility value while satisfying global constraints [89, 90, 91, 92]. Most of these methods are able to effectively and efficiently optimize QoS of the composite services. However, they fails to work for individual services. For individual services, resource provision has been utilized in [87] to analyze workloads, to categorize them on the basis of common patterns, and to plan for workloads before actual scheduling. The authors of [87] later enhanced their work with automated processing in [88]. Although these methods are able to optimize QoS by allocating more resources, no attention has been given to resource over-provision, which is a serious issue for Web services as it wastes resources and causes the increase of operational cost [38].

Presented in this chapter is a new approach of QoS optimization for quality

improvement of Web service. After briefly discussing the relationship between this chapter and the work presented in Chapter 3 in a unified framework for both prediction and prevention of SLA violation, Section 4.3 provides details of a novel strategy that helps service providers to achieve design optimum by searching globally to identify subregions with local optima and locally to locate the best optimum with constraints to minimize the operational cost. As an assessment of the proposed approach, Section 4.4 first uses multiple benchmark optimization problems to examine its accuracy, efficiency and robustness in comparison with existing methods, and then uses a Web service to check its efficiency in practice. Finally, this chapter concludes with discussions in Section 4.5.

4.2 Definition of the Optimization Problem and Related Conceptual Models

In this section, the optimization problem and conceptual models relevant to QoS of Web services are formally defined. These concepts will be used by the proposed optimization method in the following section.

4.2.1 Optimization Context Model

Optimization context model specifies the optimization objectives, the environmental context and the environment type. It can be modeled as a 3-tuple(Objectives, Factors, Environment), in which:

1. Objectives are QoS that need to be optimized, such as response time, security, etc.

2. Factors are those that may have a potential influence on the Objective. Each factor has its low and upper bound.

3. Environment represents the type of environment where Web services runs, such as cloud computing, traditional server, etc.

4.2.2 QoS Model

QoS model specifies the quality characteristic of objectives and can be modeled as a 3-tuple(Name, Type, Desirability), in which:

1. Name is the QoS name.
2. Typ is the data type of factors, such as numeric, string or boolean.
3. Desirability is the sense of desirability of QoS value that represents the direction in which the result is expected to go. It has three options: smaller is better, bigger is better or nominal is best.

4.2.3 Factor Model

Influential factors can be categorized as either uncontrollable or controllable ones [111]. Controllable factors refer to the ones that service providers can control or configure their level(value) during the execution of Web services, such as CPU, storage, cache, etc. Uncontrollable factors are those that can not be controlled by service providers, such as workload, network environment or the requirements of services consumer. For controllable factors, they can be further subdivided into two groups. One group includes factors that can cause an increase or decrease in the operational cost when changing their values. Another group contains factors that have no impact on the operational cost while their levels are modified. The objective of the proposed approach is to achieve design optimum without causing an unnecessary increase in the operational cost when reconfiguring Web services and optimizing its QoS. A factor can be modeled as a 4-tuple(Name, Type, Category, Cost), in which:

1. Name is the factor name.

2. Type is the data type of factor, such as numeric or string.

3. Category represents whether the factor is controllable or not.

4. Cost is an object that contains two properties: weight and influential type. Weight refers to the value of operational cost and its value ranges from 0 to 10. Influential type is a enumerable type that shows how this factor will influence operational cost. Its value can be either negative or positive.

4.2.4 SLA Model

An SLA specifies an agreement between service provider and service consumer and can be modeled as a 4-tuple(Name, QoS, RS, Penalty), in which:

1. Name is the SLA name.

2. QoS is the QoS model object that is used in the SLA.

3. RS is the average response time of Web services that the service provider promises to achieve.

4. Penalty is compensation that the service consumer will receive once the service provider fails to meet its promises.

4.2.5 Service Model

A service model specifies information relevant to Web Services and can be defined as a 3-tuple(Name, Factors, SLA), in which:

1. Name is the name of Web Services.

2. Factors are a collection of factors that have a significant influence on QoS.

3. SLA is the SLA model for this Web service.

4.2.6 Global Search Model

A global search model specifies information relevant to global search algorithm and can be defined as a 5-tuple(Factors, QoS, MARS, Knots, Results), in which:

1. Factors is a set of input variables of the global model, such as CPU, cache level, etc.

2. QoS is the output response(variable) of the global model, such as response time.

3. MARS is the fitted global model that reflects the relationship between one controllable factor and the response.

4. Knots is a collection of points that divide the region of the input variable into several subregions.

5. Results is a set of result objects that contain a candidate subregion in which a local optimum is located and a starting search point that is near the real local optimum. The results will be used by the local search algorithm.

4.2.7 Local Search Model

A local search model specifies information related to local search algorithm. For each result in the result list obtained from the global search, a corresponding local search model will be created. It can be modeled as a 6-tuple(Factor, QoS, Region, Model, Path, Result), in which:

1. Factor is the input variable of the local model, such as CPU, cache level, etc.

2. QoS is the output response(variable) of the local model, such as response time.

3. Region is one of subregions that are obtained from the global search algorithm.

4. Model is the fitted local surrogate model that reflects the relationship between one controllable factor and the response. It is either a first or second order polynomial model.

5. Path refers to a searching path of the steepest ascent or descent computed by using the fitted first order model.

6. Result is the local optimum of one subregion.

4.2.8 Optimization Problem

Mathematically speaking, QoS optimization for Web services is the minimization of a function subject to constraints on the influential controllable factor and can be written as follows:

$$Minimize : f(X) \tag{4.1}$$

$$Subjectto : c_i(X) = 0, i \in E; c_i(X) > 0, i \in I \tag{4.2}$$

where:

1. X refers to a set of influential controllable factors.

2. f(X) refers to the real relation between the controllable factor and QoS and its underlying formula is usually unknown. As a result, it is regard as a black box and the surrogate model will be used to approximate it.

3. E is a set of equality constraints.

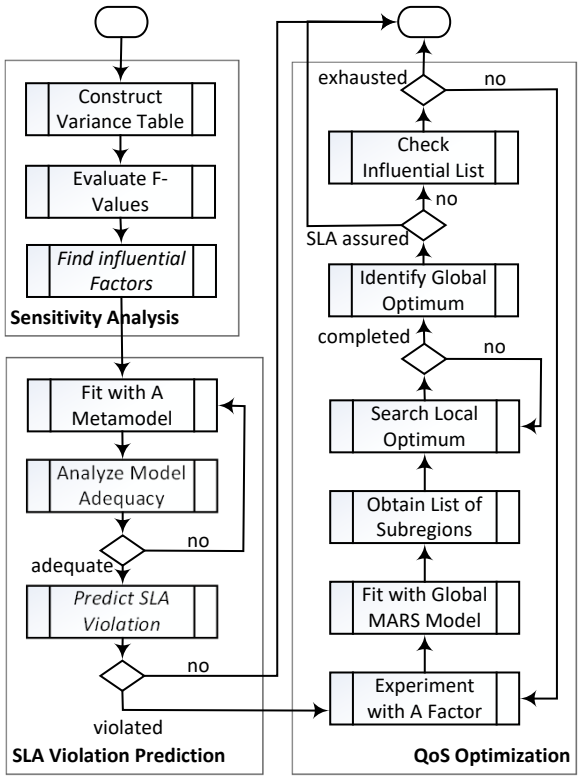

Figure 4.1: A Unified Framework of QoS Optimization

4. I is a set of inequality constraints.

4.3 SLA Violation Prevention with Optimization

This section discusses prediction and prevention of SLA violation with a unified framework, and provides details of QoS optimization with joint global search and local search algorithms.

4.3.1 A Unified Framework of QoS Optimization

Among a diverse range of approaches working on QoS optimization, sensitivity analysis, SLA violation prediction, and design optimization are three important yet normally separated techniques [67]. Presented in Fig. 4.1 is an innovative framework

Table 4.1: M-Factor Variance Table

source	**mean square** (MS_{source})	**f distribution** (F^{source})
$A^{(1\cdots m)}$	$SS_{i_{1\cdots m}} / \prod_{i=1}^{m}(a_i - 1)$	$MS_{A^{(1\cdots m)}}/MS_E$
error	$SS_E/[(n-1)\prod_{i=1}^{m} a_i]$	
total	$SS_T/[(n-1)\prod_{i=1}^{m} a_i]$	

that tackles the main issue of SLA violation by joining these three techniques into a unified process to analyze, predict, and prevent SLA violation. In the proposed framework, sensitivity analysis emphasizes on external observations of a service's performance under the influence of different factors $A^{(v)}$, $1 \leq v \leq m$, that are beyond the control of the service provider. By developing a multi-factor model with a chosen experimental strategy, e.g. Taguchi's orthogonal arrays [68], a variance table can be constructed from observations in the format of Table 4.1 for each of the individual factors and for all their joint effect in different combinations.

For all the m uncontrollable factors, only those whose individual or joint f-distribution values exceed the threshold value of a cumulative f distribution are considered to be influential factors of the service. After fitting an surrogate model, or metamodel, to external observations of the service with only the influential factors and verifying the model's adequacy, accurate approximations of the service's performance become accessible to predict the situations when SLA violation might happen and to prevent violation by means of QoS optimization. With details of sensitivity analysis and violation prediction being presented in Chapter 3, this chapter focuses on a new strategy of QoS optimization with controllable factors. This new strategy consists of a global search algorithm, which search globally to create a list of subregions with potential local optima, and a local search algorithm, which searches local optima to identify global optimum.

4.3.2 Global Search Algorithm

The aim of the proposed global search algorithm is to find one or more subregions in which potential local optima exist, and to obtain their corresponding starting search points that should be near the true local optimum in each subregion. These subregions will be used by the local search algorithm. Each subregion and its corresponding starting search point should have the following characteristics:

1. Each subregion must contain only one local optimum.
2. Each starting search point should be near the local optimum in order to increase the efficiency of the upcoming local search.

The global search algorithm will first fit a global surrogate model and then identify one or more subregions and their starting search points. The new algorithm chooses multivariate adaptive regression splines(MARS) for the following reasons:

1. It is capable of modelling both simple and complex performance model.
2. It can reflect the shape of real model of system accurately and help the global search algorithm to find subregions and their starting search points.

Specifically, the global search algorithm uses MARS to partition a given dataset into separate regions and fit each of them with a basis function [113]. With the basis function $H_i(X)$ taking the form of (4.3), $1 \leq i \leq n$, the overall model $f(\hat{X})$ of MARS mainly consists of these n basis functions as formulated in (4.4). In these two equations, X is the factor vector; K_i is a parameter introduced to limit the order of interactions; s_{ki} assumes only two values that can be either 1 or -1 and indicates the left/right sense of the truncation; $[exp]_+$ produces zero if the value of exp is negative or its actual value otherwise; t_{k_i}, known as the knot of a spline, has a value unique to $x_{v(k,i)}$; B_0 is an intercept parameter; and B_i is the coefficient of $H_i(X)$.

$$H_i(X) = \prod_{k=1}^{K_i} [s_{ki}(x_{v(k,i)} - t_{k_i})]_+ \tag{4.3}$$

$$\hat{f(X)} = B_0 + \sum_{i=1}^{n} B_i H_i(X) \tag{4.4}$$

The approximation of a dataset with MARS starts with only B_0 being included in the model. After that, basis functions are created by iterating all factors and all possible knots, and those that give the maximum reduction in residual error will be added to the candidate list; This search-and-add activity continues recursively until reaching a model with pre-determined maximum complexity. Afterwards, a backward pruning procedure removes those basis functions that contribute the least to the approximation. Finally, its generalized cross validation error as introduced in [112] tells the adequacy of the fitted MARS model in terms of both its residual error and complexity.

In the proposed searching algorithm, the fitted global model is the relationship between one controllable factor and the performance. As a result, the characteristics of MARS model with one variable will be studied. According to [114], a MARS model in this case has the following characteristics:

1. An approximating spline function is obtained by dividing the range of x values into K + 1 range regions separated by K knots;

2. The approximation takes the form of a separate qth degree polynomial model in each region;

3. The flexibility of the regression spline approach can be enhanced by incorporating automatic strategy for knot selection as a part of the data fitting process.

Based on these characteristics, the global search algorithm, shown in Algorithm 1, is designed to find candidate subregions and their corresponding starting search points for the local search phase. With a list of knots created by fitting the dataset

Algorithm 1: The Global Search Algorithm

```
1  MARSModel ← fit the MARS model with D;
2  knots ← Get Knot list from MARSModel;
3  add the starting and ending point of D to knots and sort knots;
4  R ← create an empty subregion list;
5  IsCreateNewResult ← true;
6  foreach knot of knots do
7      if IsCreateNewResult is true then
8          subRegion ← create a new subregion;
9          subRegion.StartingPoint ← knot;
10         IsCreateNewResult ← false;
11     else
12         spline1, spline2 ← get spline models of 2 subregions which are
             divided by this knot;
13         if spline1 is monotonically increasing and spline2 is monotonically
             decreasing then
14             subRegion.EndPoint ← knot;
15             IsCreateNewResult ← true;
16             add subRegion to R;
17     end
18 end
19 foreach subRegion of R do
20     subRegion.StartingSearchPoint ← Get x_min
21 end
22 MinStartingSearchPoint ← Get the minimum starting Search Point from R;
23 foreach subRegion of R do
24     diff ← abs(subRegion.StartingSearchPoint - MinStartingSearchPoint);
25     if diff > Threshold, then
26         Remove this subRegion from R;
27 end
```

with a MARS model, the proposed global search algorithm completes its initialization by sorting the knot list after adding starting and ending points, creating an empty list of subregions, and setting flag `IsCreateNewResult` to true in line 3, 4, and 5 respectively. While the first loop (lines 6–18) adds subregions to `R`, the second loop (lines 19–21) sets the starting points for all subregions in the list. Finally, the third loop (lines 23–27) removes some subregions in which the starting search points are far away from the global optimum to increase the searching efficiency for the following local search algorithm.

4.3.3 Local Search Algorithm

The local search algorithm will use the subregions and their starting search points identified by the global search algorithm, and iterate on each of these subregions. For each iteration, this local search algorithm will first start with its starting search point, fit one or more first order polynomial models, and use these models and the steepest descent or ascent method to get closer to new experimental regions that are closer to the real local optimum. When approaching to the real optimum on this subregion, one or more second order polynomial models will be used to optimize the response function. When there is no significant iteration-to-iteration improvement, the search on this subregion will come to an end and the point found will be regarded as a candidate point. After all subregions are searched, the best one will be used as the final solution among all candidate points.

In this approach, first or second order polynomial models are chosen as the local surrogate models and are given in Equations 4.5 and 4.6,

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \varepsilon \tag{4.5}$$

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i<j} \sum_{i=1}^{n} \beta_{ij} x_i x_j + \sum_{i=1}^{n} \beta_{ii} x_i^2 + \varepsilon \tag{4.6}$$

where $\hat{y}$ is approximating function; x_i and x_j is corresponding input factors; β_0, β_i, β_{ii} and β_{ij} is unknown coefficients that need to be estimated; ε is Bias or lack of f.

When fitting the local surrogate models, design of experiments will be used in order to produce a good fit of the model to the data in a minimum number of experimental runs and control errors due to both variance and bias [7]. An experiment can be defined as a test or series of runs in which purposeful changes are made to the input variables of a process or system [8]. There are many types of experimental design, including a full factorial design(FFD), a central composite design(CCD), box-behnken design, the d-optimality, the space filling design and taguchi's orthogonal arrays [7] [8] [115]. Because the proposed local search algorithm is developed by extending the response surface methodology, full factorial design will be used to fit the first order polynomial model and central composite design will be utilized to build the second order polynomial models [7].

After experiments based on full factorial design have been conducted and the first order polynomial model has been fitted, the steepest descent or ascent method will be used to find the searching path. The coordinates along the path of steepest descent or ascent depend on the signs and magnitudes of the regression coefficients in the fitted first order polynomial model [7]. Consider the following equation:

$$L = \hat{y} + \lambda(\sum_{i=1}^{n} x_i^2 - r^2) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \lambda(\sum_{i=1}^{n} x_i^2 - r^2) \tag{4.7}$$

where $\hat{y}$ is approximating function; x_i is corresponding input factors; β_0 and β_i is coefficients; r is a fixed distance r from the center of the design; λ is a constant of proportionality.

The derivative with respect to x_i is

$$\frac{\partial L}{\partial x_i} = b_i - 2\lambda X_i = 0 \tag{4.8}$$

As a result, the coordinate of x_i of the path of steepest descent or ascent is

$$x_i = \frac{b_i}{2\lambda} \tag{4.9}$$

When moving to the point that is closer enough to the optimum along the steepest descent or ascent path, experiments based on central composite design will be performed and the second order model will be fitted. After that, canonical and possible ridge analysis will be used to determine the location and the nature of the stationary point of the second order polynomial model [7]. The fitted second order polynomial model can be written in Equation 4.6.

$$\hat{y} = \beta_0 + X'b + X'\hat{B}X \tag{4.10}$$

where β_0, b, and $\hat{B}$ are the estimates of the intercept, linear, and second order coefficients, respectively.

The stationary point x_s of the second-order polynomial is determined by

$$x_s = -\frac{1}{2}\hat{B}^{-1} \tag{4.11}$$

If all eigenvalues of B are positive(negative), the quadratic surface has a minimum(maximum) at the stationary point x_s. If the eigenvalues have mixed signs, the stationary point x_s is a saddle point and the search is in the direction of the negative gradient of the fitted quadratic, or may be a ridge direction which gives the best predicted value of the quadratic on a hypersphere of fixed radius [7].

Based on the above analysis, searching for optimum design will continue locally within each of the subregions that have been created by the global search algorithm. As illustrated in Algorithm 2, the algorithm repeats its search within each iteration

Algorithm 2: The Local Search Algorithm

```
1  L ← create an empty list which is used to contain local optimum for each
     subregion;
2  foreach subRegion of R do
3      fit a first-order model around the starting search point of this subregion;
4      if the controllable factor have negative influence on the operational cost
         and the performance on the starting search point meet the requirement
         of the SLA then
5          a path of steepest descent will be computed;
6      else if the controllable factor have positive influence on the operational
         cost and the performance on the starting search point meet the
         requirement of the SLA then
7          a path of steepest ascent will be computed;
8      else
9          a path of steepest descent will be computed;
10     end
11     perform a series of line searches along the computed path until a certain
         criteria is meet;
12     finalPoint ← the last point along the searching path;
13     if the controllable factor influences on the operational cost and the
         performance on the final point meet the requirement of the SLA then
14         localOptium ← finalPoint;
15     else
16         secondOrderModel ← fit a second-order model around the final point;
17         localOptium ← locate the stationary point of the fitted second-order
             polynomial model;
18     end
19     add localOptium to L;
20 end
21 get final optimum solution in L;
```

(lines 2–18) by first fitting a first-order polynomial model and then following a path in the steepest descent or ascent direction to approach the regional optimum. When the flow reaches line 14, it is the case to find the local point by avoiding SLA violation while keeping the operational cost as low as possible at the same time. When the flow reaches line 16, it is the case to find the local optimum by using a second-order polynomial model for the optimization of response function. At the end of each iteration, a regional optimum is added to the candidate list for global optimum. When the algorithm finishes searching through all the subregions, the final optimum solution will be identified by getting the minimum value from the candidate list or the value that helps the service avoiding SLA violation while keeping the operational cost as low as possible at the same time.

4.4 Experiments

The effectiveness of one optimization algorithm against others cannot be simply measured by the problems that it solves if the set of problems are too specialized and without diverse properties [125]. As a result, this section uses experiments on both benchmark functions and a Web service to assess the efficiency, accuracy and robustness of the proposed optimization method.

4.4.1 Experiments with Benchmark Functions

Shown in Figure. 4.2 is the benchmark function of a well-known optimization problem [116], whose optimum is $f_{min}(x^*) = -0.8690$ at $x^* = 0.5485$. For its unusual shape and multiple local optima as defined in (4.12), where $0.5 \leq x \leq 2.5$, this function is commonly used to test the accuracy and efficiency of optimization

algorithms.

$$f(x) = \frac{sin(10\pi x)}{2x} + (x-1)^4 \tag{4.12}$$

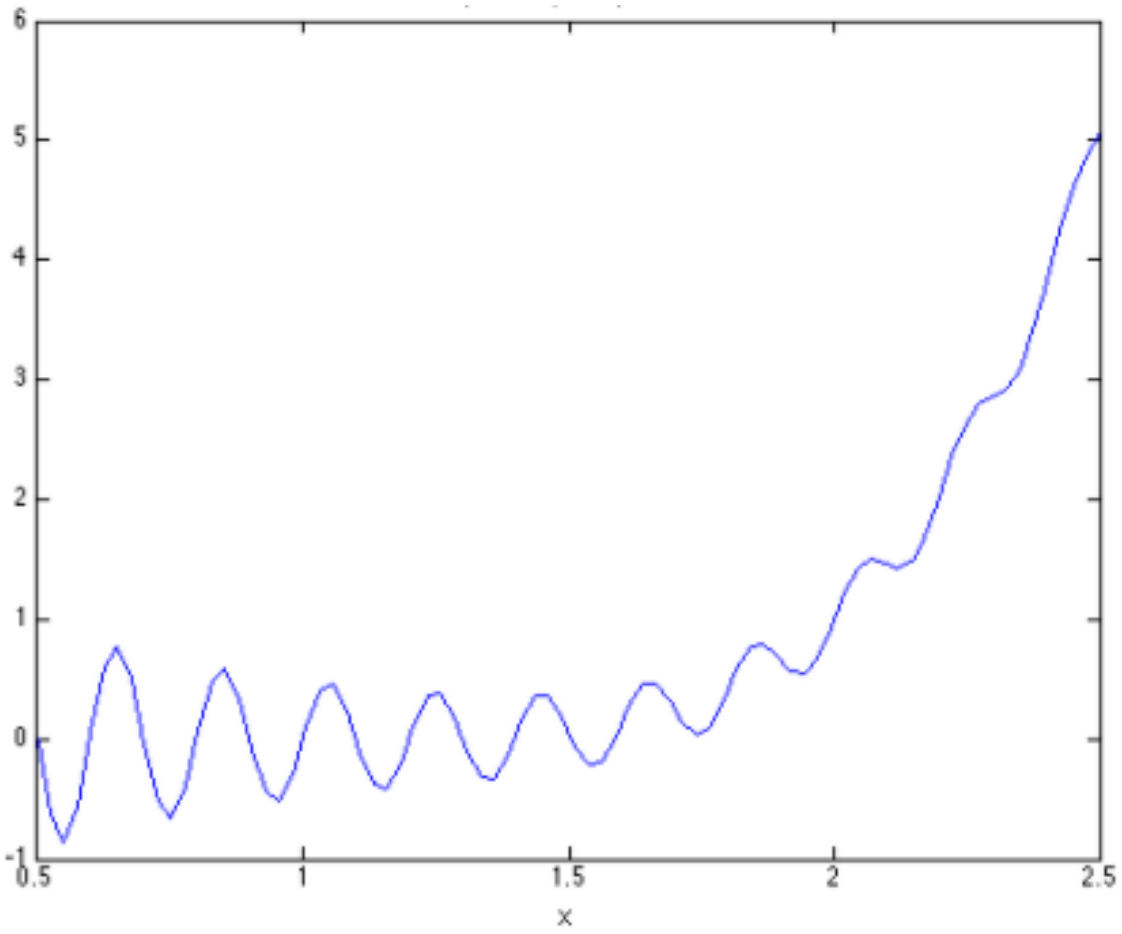

Figure 4.2: The GRAMACY-LEE (2012) function.

Apply the Proposed Global and Local Search Algorithm

Full factorial design is used in the experiment to divide the design space of x into a series of uniformly spaces and the experimental data is collected to fit the global MARS model. As shown in Table. 4.2, a sequence of 100 values of the benchmark function $f(x)$ is collected with x increasing by an increment of 0.02.

Table 4.2: Result of Experiment for Global Search Algorithm

No	x	f(x)	No	x	f(x)	No	x	f(x)	No	x	f(x)
1	0.50	0.0704	2	0.52	-0.5056	3	0.54	-0.8333	4	0.56	-0.8141
5	0.58	-0.4820	6	0.60	0.0176	7	0.62	0.4884	8	0.64	0.7573
9	0.66	0.7362	10	0.68	0.4490	11	0.70	0.0160	12	0.72	-0.3955
13	0.74	-0.6355	14	0.76	-0.6247	15	0.78	-0.3808	16	0.80	-0.0063
17	0.82	0.3529	18	0.84	0.5642	19	0.86	0.5557	20	0.88	0.3405
21	0.90	0.0080	22	0.92	-0.3129	23	0.94	-0.5033	24	0.96	-0.4977
25	0.98	-0.3062	26	1.00	-0.0079	27	1.02	0.2816	28	1.04	0.4547
29	1.06	0.4510	30	1.08	0.2785	31	1.10	0.0080	32	1.12	-0.2557
33	1.14	-0.4142	34	1.16	-0.4116	35	1.18	-0.2544	36	1.20	-0.0063
37	1.22	0.2367	38	1.24	0.3842	39	1.26	0.3843	40	1.28	0.2421
41	1.30	0.0160	42	1.32	-0.2056	43	1.34	-0.3389	44	1.36	-0.3352
45	1.38	-0.1985	46	1.40	0.0176	47	1.42	0.2315	48	1.44	0.3651
49	1.46	0.3728	50	1.48	0.2580	51	1.50	0.0704	52	1.52	-0.1137
53	1.54	-0.2212	54	1.56	-0.2088	55	1.58	-0.0792	56	1.60	0.1216
57	1.62	0.3226	58	1.64	0.4551	59	1.66	0.4785	60	1.68	0.3951
61	1.70	0.2480	62	1.72	0.1043	63	1.74	0.0291	64	1.76	0.0610
65	1.78	0.1986	66	1.80	0.4016	67	1.82	0.6070	68	1.84	0.7537
69	1.86	0.8050	70	1.88	0.7623	71	1.90	0.6640	72	1.92	0.5698
73	1.94	0.5382	74	1.96	0.6043	75	1.98	0.7675	76	2.00	0.9920
77	2.02	1.2214	78	2.04	1.4003	79	2.06	1.4956	80	2.08	1.5081
81	2.10	1.4720	82	2.12	1.4414	83	2.14	1.4693	84	2.16	1.5881
85	2.18	1.7976	86	2.20	2.0656	87	2.22	2.3411	88	2.24	2.5739
89	2.26	2.7332	90	2.28	2.8196	91	2.30	2.8640	92	2.32	2.9158
93	2.34	3.0235	94	2.36	3.2172	95	2.38	3.4969	96	2.40	3.8336
97	2.42	4.1807	98	2.44	4.4920	99	2.46	4.7393	100	2.48	4.9227

Table 4.3: The Fitted Mars Model for the Benchmark Function

no.	basis functions	coefficient	no.	basis functions	coefficient
1	Intercept	883.19	2	h(1.94-x)	-1025.41
3	h(x-2.1)*h(x-1.94)	-556.012	4	h(2.1-x)*h(x-1.94)	532.607
5	h(x-0.64)*h(1.94-0)	771.606	6	h(x-0.64)	-944.482
7	h(x-0.84)*h(x-0.64)	86.2546	8	h(0.84-x)*h(x-0.64)	-217.846
9	h(x-1.14)*h(x-0.94)	502.194	10	h(1.14-x)*h(x-0.94)	-380.269
11	h(1.24-x)*h(x-0.64)	-34.3123	12	h(x-1.54)*h(x-1.34)	93.1461
13	h(1.54-x)*h(x-1.34)	-42.5101	14	h(1.64-x)*h(x-0.64)	-21.9965
15	h(x-1.74)*h(1.94-x)	-225.029	16	h(1.74-x)*h(1.94-x)	297.525
17	h(x-1.44)*h(x-0.94)	-302.771	18	h(1.44-x)*h(x-0.94)	265.226
19	h(0.74-x)	259.016	20	h(x-2.34)*h(x-0.74)	151.82
21	h(2.34-x)*h(x-0.74)	-141.199	22	h(2.14-x)*h(x-0.94)	17.828
23	h(x-0.62)*h(0.94-x)	-114.808			

The MARS model is fitted by using this dataset with the pyearth module in Python as specified in (4.13) after setting maximum terms to 1000, the maximum degree to 2, and penalty to 0.01. This fitted global surrogate model contains 23 basis functions shown in Table. 4.3.

$$\hat{f(x)} = \sum_{i=1}^{23} C_i B_i \tag{4.13}$$

In (4.13), B_i is a basis function and C_i is the coefficient of the basis function in the i_{th} row of Table 4.3. By applying the proposed global search algorithm, the whole dataset is divided into subregions as listed in Table 4.4 with their starting points to search for local optima. After setting the searching threshold to 0.2, only the first two subregions remain for further processing in the phase of local search while the

Table 4.4: Subregions and Their Starting Search Points

no.	**subregions**	**starting points**	**f(x_s)**
1*	[0.5 , 0.64]	0.54	-0.8333
2*	[0.64 , 0.84]	0.74	-0.6355
3	[0.84 , 1.14]	0.94	-0.5033
4	[1.14 , 1.24]	1.14	-0.4142
5	[1.24 , 1.44]	1.34	-0.3389
6	[1.44 , 1.64]	1.54	-0.2212
7	[1.64 , 2.10]	1.74	0.0291
8	[2.10 , 2.50]	2.12	1.4414

others are abandoned.

Around the starting point of the first subregion, five $\hat{f(x)}$ values are collected (Table. 4.5). Fitting data from no. 1 to 3 produces a fitted model in the form of $\hat{f(x)} = 3.47043 - 7.91500 * x$. When searching along the path in the positive direction for two more increments of x, no further improvement of performance is observed as the starting point is already near the local optimum. As a result, the surrounding values around the last searching point from no. 1 to no. 5 are used to fit the second-order model, which produces $\hat{f(x)} = 129.16100 - 473.61437 * x + 431.27143 * x^2$. A canonical and ridge analysis on this model delivers the local optimum in first subregion, i.e. $f_{min}(x^*) = -0.8680$ at $x^* = 0.55$.

Similarly, the local optimum in the second subregion can be found with the proposed local search algorithm as $f_{min}(x^*) = -0.6627$ at $x^* = 0.75$. As $f_{min}(0.55)$ is better than $f_{min}(0.75)$, the global optimum is achieved at 0.55 and the difference from the real x value is 0.01. The process of optimization terminates after performing 106 experiments.

Table 4.5: Result of Experiments on Local Search Algorithm

no.	**x**	$f(\hat{x})$	**no.**	**x**	$f(\hat{x})$
1	0.53	-0.70971	2	0.54	-0.8333
3	0.55	-0.868	4	0.56	-0.8141
5	0.57	-0.6801	6	0.73	-0.544
7	0.74	-0.6355	8	0.75	-0.6627
9	0.76	-0.6247	10	0.77	-0.5271

Comparison of the Accuracy of Approximation and Prediction among Different Fitted Global Models

In this subsection, the error sum of squares (SSE) is used to measure the capability of how well the fitted global surrogate models approximate the true model and is defined as:

$$SSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.14}$$

Meanwhile, the root mean squared error(RMSE) and mean absolute error (MAE) are utilized to measure and evaluate the accuracy of prediction. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}} \tag{4.15}$$

MAE is defined as:

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{4.16}$$

where y_i denotes the ith observerd QoS value, $\hat{y}_i$ is the corresponding predicted QoS value, and N is the number of predicted values.

The fitted MARS model will be compared with the following three commonly used modelling techniques in terms of SSE, RSME and MAE:

1. Polynomial regression(PR) [7]:

   (a) Polynomial regression is the most commonly used modeling technique for its simplicity [117].

   (b) The sklearn.linearmodel module in Python will be used to fit the polynomial regression model by setting the degree of the polynomial features to 10.

2. Support vector regression(SVR) [118]:

   (a) Support vector regression is a modeling technique. It extends support vector machine to the domain of regression problems and gains in popularity in the machine learning community for its capability of producing prediction models with excellent generalization performance.

   (b) The sklearn.svm module in Python will be used to fit the SVR model by setting the kernel function to the radial basis function, C (that is a penalty parameter of the error term) to 1e3 and gamma (that is the kernel coefficient for radial basis function to) to 0.1.

3. Regression tree(RT) [119]:

   (a) Regression tree is a variant of decision tree that predicts values of continuous variables instead of labels of classification [120] and is frequently used to model non-linear relationships between input factors and output response.

   (b) The sklearn.tree module in Python will be used to fit the RT model by setting the minimum samples' split to 30, the minimum samples' leaf to 10 and the random state to 0.

In comparison with these three modeling techniques regarding the accuracy of the global fitting, Table 4.6 shows that the MARS model has the smallest SSE value,

Table 4.6: Comparison of Approximation and Prediction Accuracy of Fitted Global Models)

Model	SSE	RMSE	MAE
Multivariate Adaptive Regression Splines	0.62255	0.06208	0.04611
Support Vector Regression	10.24481	0.29062	0.24912
Polynomial Regression	9.02551	0.27201	0.23638
Regression Tree	24.19414	0.45257	0.35261

which indicates that MARS model has the least error among all the fitted models and performs the best in terms of approximation accuracy. In addition, the smallest RM_{SE} and MAE values of MARS suggest that it outperforms the other models in term of prediction accuracy.

By plotting the real model and all the four fitted models onto the graphs, Fig. 4.3 and Fig. 4.4 show that the MARS model fits the true model with higher accuracy than others and exhibits a similar shape as the true model across the global region.

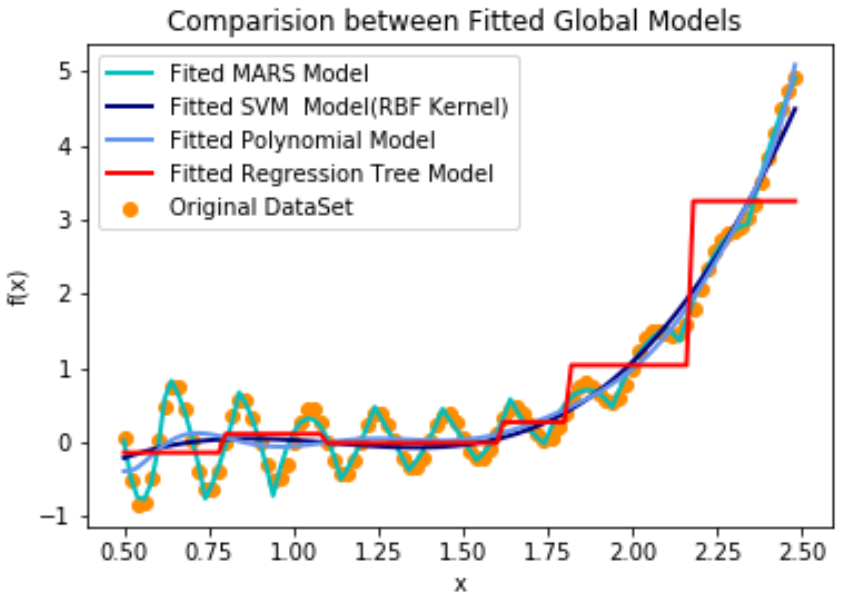

Figure 4.3: The Comparison between All Fitted Global Models

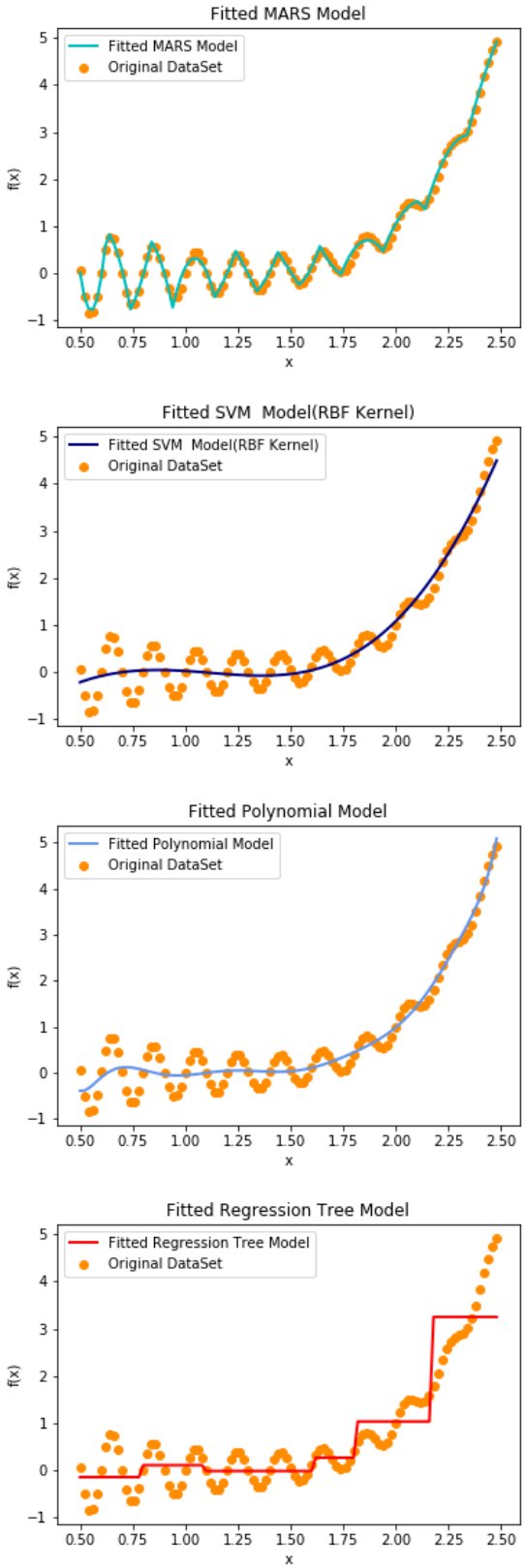

Figure 4.4: The Comparison Between the True Model and the Fitted Model

Table 4.7: Optimization Results by Using Response Surface Methodology

Starting Search Point	**Number of Experiments**	**Optimum Design Point**	**Optimum Obtained**	**Error**
0.623	8	0.552	-0.8609	0.0080
0.791	5	0.751	-0.6618	0.2072
0.852	11	0.952	-0.5246	0.3444
1.227	9	1.147	-0.4326	0.43631
1.452	11	1.552	-0.2290	0.6399

Comparison with Several Well-known Optimization Methods

In this subsection, the proposed optimization method will be compared with response surface methodology(RSM) and the global metamodel-based optimization methodology. These comparisons will cover two aspects:

1. The optimization accuracy indicates the error between the identified and real global optimum. It tests how close the found global optimum comes to the real one;

2. The number of experiments tests the search efficiency of the optimization method. For benchmark optimization problem, performing one experiment refers to one benchmark function evaluation.

The first comparison was made with the response surface methodology [100], which is the most popular metamodel-based optimization method. The five rows in Table. 4.7 are the optimization results starting with five different searching points by using the response surface methodology. Table. 4.7 shows that the number of experiments needed by RSM is smaller than that of the proposed method. However, the optimization results are different from each other because RSM can only guarantee

Table 4.8: Optimization Results by Using Global Metamodel-based Optimization Methodology

Model	**Number of Experiments**	**Optimum Design Point**	**Optimum Obtained**	**Error**
MARS	106	0.550	-0.868	0.010
SVR	101	1.373	-0.251	0.617
PR	101	0.963	-0.477	0.391
RT	101	1.396	-0.023	0.845

to find a local optimum and has the issue of being trapped by the local optimum. As a result, the proposed method is more competitive than RSM in terms of finding the global optimum when there are multiple local optima in the performance model.

The second comparison was made with the global metamodel-based optimization methodology. For the global metamodel-based optimization methodology, three commonly used modeling techniques are used, including polynomial regression, support vector regression and regression tree. These three global models are fitted by using the result of 100 experiments in Table. 4.2. Then, genetic algorithm(GA) is used to find the global optimum on these three surrogate models. While the first row in Table. 4.8 shows the result of the proposed method, the other rows show results of PR, SVR and RT, respectively. The smallest error of the proposed method suggests that it outperforms the global metamodel-based optimization methodology that uses PR, SVR and RT, in terms of optimization accuracy.

The Robustness Evaluation of the Proposed Optimization Method

In previous subsections, the proposed method has been evaluated on the benchmark function with high degrees of non-linearity in terms of accuracy and efficiency.

Table 4.9: Robustness Comparison of Approximation Models

$f_1(x) = -(16x^2 - 24x + 5)e^{-x}$				$f_2(x) = -x * sin(x)$			
no.	**point**	**opti.**	**error**	**no.**	**point**	**opti.**	**error**
23	2.885	-3.850	0.00030	37	7.958	-7.915	0.00163
20	2.888	-3.850	0.00043	34	7.960	-7.915	0.00124
20	2.846	-3.849	0.00048	34	7.991	-7.916	0.00066
20	3.127	-3.788	0.06200	34	6.658	-2.438	5.47811
$f_3(x) = sin(x) + sin(10x/3)$				$f_4(x) = -(1.4 - 3x)sin(18x)$			
no.	**point**	**opti.**	**error**	**no.**	**point**	**opti.**	**error**
52	5.193	-1.885	0.01356	62	0.966	-1.489	0.00001
49	4.756	-1.144	0.75458	60	0.114	-0.928	0.56046
49	5.153	-1.899	0.00041	60	0.999	-1.204	0.28485
49	5.511	-1.160	0.73885	60	0.171	-0.054	1.43445

Table 4.10: Details of the Four Additional Functions

	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
range of x	[1.9 , 3.9]	[2.7 , 7.5]	[2.7 , 7.5]	[0 , 1.2]
x^*	2.868034	7.9787	5.145735	0.96609
$f_{min}(x^*)$	-3.85054	-7.916727	-1.899599	-1.48907

This subsection is going to evaluate its robustness and verify its ability to consistently achieve similar accuracies on different problems. This evaluation results in more experiments on four additional benchmark functions in different degrees of nonlinearity. Going through the same procedure as for the first benchmark function in (4.12), more results are obtained and listed in Table 4.9. While the formula of each function is given in the table, results for the proposed method, polynomial regression, support vector regression, and regression tree are listed in the first, second, third, and

fourth row under each of the functions. The four benchmark functions have one, two, three, and four local optima respectively, with more details provided in Table 4.10. Underneath each function, results are listed in Table 4.9 in the order of the number of experiments, point of optimal design, the value of optimal design, and error between the real and fitted model.

Results in Table 4.9 demonstrate that the proposed method is able to consistently achieve high accuracy when applied to solve different problems, and therefore is robust. In comparison, metamodel-based optimization based upon polynomial regression and support vector regression may produce competitive results only when the benchmark functions have one, two, or three local optima. Once the number of local optima exceeds three, all the tested methods, except the proposed method, fail to achieve satisfying accuracy.

4.4.2 Experiments with A Web Service

Presented in this subsection is the second group of experiments that examines the proposed method about its use in the optimization of a Web service for users to identify the top ten webpages linking to different e-commerce Websites with the lowest prices for certain products.

Experiments Setup

For the implementation of Web services and collection of experimental data, the experiments use .NET Framework 4.5, Visual Studio 2017 Professional and SQL Server 2012 on Windows 10 Professional with Inter(R) Core(TM) i5-4210U CPU and 8G RAM. In addition, this group of experiments uses the python program language and the pyearth module for data analysis, the fitting of surrogate model, and performance optimization.

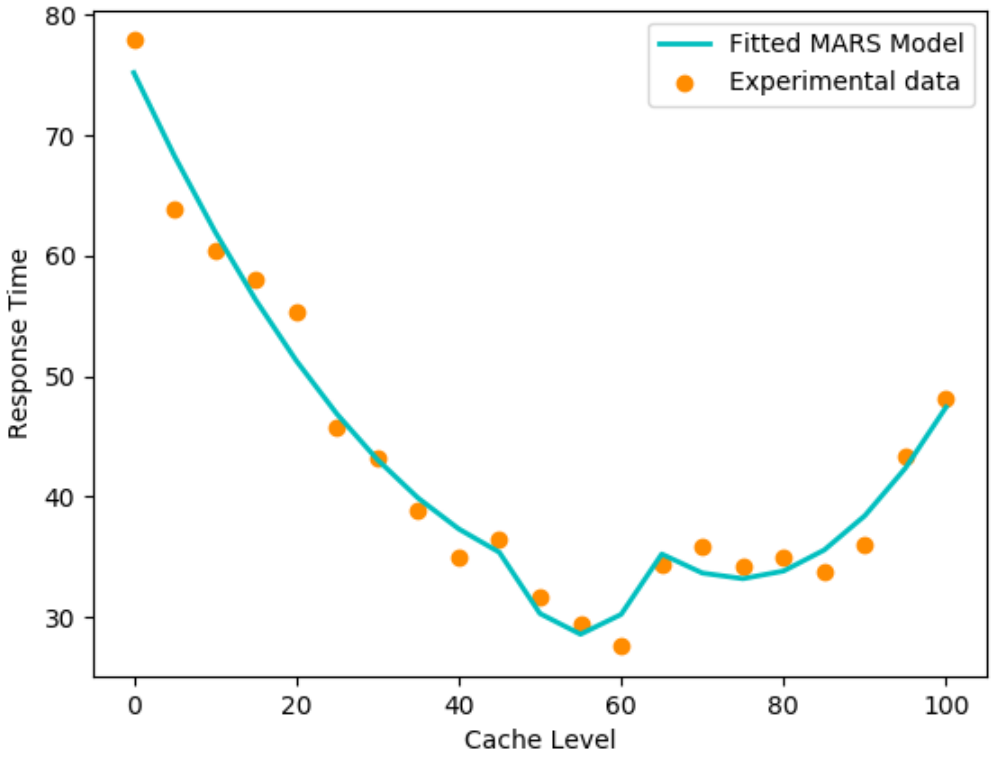

Figure 4.5: Fitted MARS Model for Web Service

Optimization Planning and Analysis

Among different factors that influence the performance of this service, the number of sources and number of products to search are two factors beyond the control of the service provider as they vary from time to time and from application to application. In comparison, the cache level is a controllable factor as the service provider may use it to improve performance by loading product data into memory for fast processing.

The number of sources and the number of products have been identified in Chapter 3 as influential factors that cause SLA violation. For example, it is a violation of SLA if the required response time is less than 50 milliseconds because the Web service takes 77.8801 milliseconds when searching 40 sources for 10000 products. As

Table 4.11: Response Time of Web Service

level	time	level	time	level	time
0	77.8801	5	63.9250	10	60.4861
15	57.9892	20	55.3350	25	45.7550
30	43.1849	35	38.8399	40	35.0098
45	36.4552	50	31.7500	55	29.5000
60	27.5900	65	34.3057	70	35.9400
75	34.2906	80	35.0053	85	33.8550
90	36.0740	95	43.3450	100	48.1450

Table 4.12: Fitted Mars Model for the Web Service

no.	basis functions	coefficient
1	Intercept	33.839
2	h(Cache-45)	-3.49659
3	Cache*h(Cache-45)	0.0548757
4	h(Cache-65)*h(Cache-45)	-0.0327714
5	h(Cache-65)	-0.937721
6	h(65-Cache)	0.636234
7	Cache*h(65-Cache)	-0.012408

a result, the SLA is violated and the performance of this Web service needs to be optimized.

Optimization Process

To optimize the Web service using the proposed method, a series of experiments are first performed based upon full factorial designs. By keeping the two factors

Table 4.13: Subregions and Starting Search Points

no.	**subregions**	**starting points**	**f(x_s)**
1*	[0 , 65]	60	27.5900
2	[65 , 100]	85	33.8550

constant at 10000 and 40 while changing the cache level from 0 to 100 in an increment of 5, a total of 21 observations of response time are collected (Table. 4.11) and Fig. 4.5 shows the plotting graph. Using the same setting of experiments on the benchmark functions, the MARS model uses seven basis functions to fit the dataset (Table. 4.12), which allows the identification of two subregions (Table. 4.13), with the second one being abandoned after setting the threshold of the global search to 5.

Even though the global optimum is the local optimum in the subregion [0, 65], a forward search along the path of the fitted MARS model (Fig. 4.5) from the starting point will miss the optimum as it has already surpassed the optimum point. In fact, the higher the cache level is, the more the memory will be occupied by cache data, causing more operational costs. To allow the Web service to use less memory by lowering the cache level, the search for the optimum in practice goes backward from 60 in a decrement of 5. Due to the required 50 milliseconds, the search stops at 25 as it uses the least cache level to yield a response time of 45.7550. This decision of QoS optimization not only improves service performance but also avoids causing the cache data to occupy too much unnecessary memory.

4.5 Summary

As part of an overall framework developed to predict and prevent SLA violation, this chapter proposes a novel strategy that uses combined global and local searching to help service providers achieve design optimum without over-provision of resources. Experimental results demonstrate that the proposed method outperforms

other commonly used methods in search efficiency while retaining competitive with those methods in optimization accuracy.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

This Ph.D. dissertation introduces a framework to integrate sensitivity analysis, SLA violation prediction, and SLA violation prevention into a unified process for quality improvement of Web services. This framework identifies influential input factors of services, produces a fitted model to approximate the relationship between factors and service quality, and uses this fitted model to analyze and predict QoS values. The process of QoS evaluation can be used by service consumers to select the most appropriate services when constructing an SOA system. Meanwhile, service providers can use the evaluation results to find out if and when SLA could be violated due to the fluctuation in different factors, to better understand the performance of any service with its fitted model, and to identify performance bottlenecks caused by influential factors.

When SLA violation is predicted by analysis or detected at run time, the optimization strategy presented in this proposed framework can be used to prevent SLA violation with a combined search of global optimum through the identification of local optima in subregions created by a fitted MARS model. A decision of opti-

mization based upon the type of controllable factors further allows Web services to be optimized with minimized operational cost. In addition to technical details, this dissertation also includes results from experiments on multiple benchmark optimization problems to demonstrate the accuracy and robustness and on a Web service to validate the efficiency of the proposed method.

5.2 Future Works

In the proposed framework for SLA prediction, a manual process is employed, which is time consuming. In some cases, SLA violation prevention has to be performed in real-time to detect violations quickly. Therefore, an automated technique is required. In addition, only one controllable factor(cache level) is used for optimization, which will not lead to optimization for complex system. Therefore, the optimization method will be extended to be capable of preventing the SLA violation by using multi-factors optimization.

Future research will investigate the automation of the proposed approach using machine learning techniques that can easily adapt to any complex system and overcome the limitation of these approaches. Meanwhile, directions of future research will also include experiments on more Web services or SOA applications and development of a working system for use in practice. The plan is:

1. A real-world dataset(Google Cloud Compute Trace) will be used to perform experiments. This dataset will first be analyzed to understand its characteristics, features, and class distribution. This can help discover effective models to ascertain the best response time prediction with such features.

2. After finishing the data analysis, different regression models will be fitted to predict the violation. From the evaluation results, the best-fitted model will be selected for prediction. Vector autoregressive moving average model with

exogenous variables will be used to predict SLA violations

3. Once the SLA violation is detected through the selected model, SLA violation will be avoided by considering a multi-factor technique for optimizing a service.

Bibliography

[1] Mahboobeh Moghaddam, Joseph G. Davis *"Service Selection in Web Service Composition: A Comparative Review of Existing Approaches,"*. Web Services Foundations, Pages 321-346, Springer, New York, 2014.

[2] Jinghai Rao, Xiaomeng Su *"A Survey of Automated Web Service Composition Methods,"*. Semantic Web Services and Web Process Composition, Volume: 3387, Pages 43-54, Springer, Berlin, Heidelberg, 2004.

[3] M Oriol, J Marco, X Franch *"Quality models for web services: A systematic mapping,"*. Information and Software Technology, Volume: 56, Issue: 10, Pages 1167-1182, 2014.

[4] Z Zheng, Y Zhang, MR Lyu *"Investigating QoS of Real-World Web Services,"*. IEEE Transactions on Services Computing, Volume: 7, Issue: 1, Pages 32-39, 2014.

[5] SY Hwang, CC Hsu, CH Lee *"Service Selection for Web Services with Probabilistic QoS,"*. IEEE Transactions on Services Computing, Volume: 8, Issue: 3, Pages 467-480, 2015.

[6] H Wang, JZ Huang, Y Qu, J Xie *"Web services: problems and future directions,"*. Journal of Web Semantics, Volume: 1, Issue: 3, Pages 309-320, 2004.

[7] Myers Raymond H., D.C. Montgomery and C. Anderson-Cook *"Response Surface Methodology : process and product optimization using designed experiment,"*. 4th ed. Wiley, 2016.

[8] Douglas C. Montgomery, *"Design and Analysis of Experiments,"*. 8th ed. Wiley, 2012.

[9] IN Vuchkov and NL Boyadjieva, *"Quality improvement with design of experiments : A response surface approach,"*. Springer Science and Business Media. 2001.

[10] Yilei Zhang, Michael R. Lyu, *"Online QoS Prediction,"*. QoS Prediction in Cloud and Service Computing, Pages 55-80, Springer, 2017.

[11] S Deng, H Wu, W Tan, Z Xiang, *"Mobile service selection for composition: an energy consumption perspective,"*. IEEE Transactions on Automation Science and Engineering, Volume: PP, Issue: 99, Pages 1478-1490, 2015.

[12] Z Zheng, H Ma, MR Lyu, I King, *"Qos-aware web service recommendation by collaborative filtering,"*. IEEE Transactions on Services Computing, Volume: 4, Issue: 2, Pages 140-152, 2011.

[13] Z Zheng, H Ma, MR Lyu, I King, *"Collaborative web service qos prediction via neighborhood integrated matrix factorization,"*. IEEE Transactions on Services Computing, Volume: 6, Issue: 3, Pages 289-299, 2013.

[14] Y Zhang, Z Zheng, MR Lyu, *"An online performance prediction framework for service-oriented systems,"*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Volume: 44, Issue: 9, Pages 1169-1181, 2014.

[15] K. Su, B. Xiao, B. Liu, H. Zhang, Z. Zhang. *"TAP: a personalized trust-aware QoS prediction approach for web service recommendation,. Knowledge-Based Systems*, Pages 55-65, 2017.

[16] S Li, J Wen, F Luo, M Gao, J Zeng, *"A New QoS-Aware Web Service Recommendation System based on Contextual Feature Recognition at Server-Side,"*. IEEE Transactions on Network and Service Management, Volume: 14, Issue: 2, Pages 332-342, 2017.

[17] J Zhu, P He, Z Zheng, MR Lyu, *"Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization,"*. IEEE Transactions on Parallel and Distributed Systems, Volume: 28, Issue: 10, Pages 2911-2924, 2017.

[18] A Saltelli, M Ratto, T Andres, F Campolongo, J Cariboni, D Gatelli ,M Saisana, S Tarantola, *"Global sensitivity analysis: the primer,"*. John Wiley Sons, 2008.

[19] KM Shirvan, M Mamourian, S Mirzakhanlari, R Ellahi , *"Numerical investigation of heat exchanger effectiveness in a double pipe heat exchanger filled with nanofluid: a sensitivity analysis by response surface methodology,"*. Powder Technology Volume 313, Pages 99-111, 2017.

[20] A Cicek, T Kivak, E Ekici, *"Optimization of drilling parameters using Taguchi technique and response surface methodology (RSM) in drilling of AISI 304 steel with cryogenically treated HSS drills,"*. Journal of Intelligent Manufacturing, Volume: 26, Issue 2, Pages 295-305, 2015.

[21] X. Yuan and C. Ji, *"Performance analysis of service-oriented architectures with multi-factor sensitivity analysis,"*. 2007 IEEE International Conference on Electro/Information Technology, 2007.

[22] A Iosup, N Yigitbasi, D Epema, *"On the Performance Variability of Production Cloud Services,"*. Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on.

[23] D.A. Menasce, *"QoS Issues in Web Services,"*. IEEE Internet Computing, Volume: 6, Pages 72-75, 2002.

[24] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, *"Eigentaste: a constant time collaborative filtering algorithm,"*. Information Retrieval, Volume: 4, Pages 133-151, 2001.

[25] Bertrand Iooss and Paul Lemaître, *"A Review on Global Sensitivity Analysis Methods,"*. Dellino G., Meloni C. (eds) Uncertainty Management in Simulation-Optimization of Complex Systems. Operations Research/Computer Science Interfaces Series, Volume: 59, Pages 101-122, Springer, 2015.

[26] M. S. Cao, L. X. Pan,Y. F. Gao, D. Novák, Z. C. Ding, D. Lehký, X. L. Li, *"Neural network ensemble-based parameter sensitivity analysis in civil engineering systems,"*. Neural Computing and Applications, Volume 28, Issue 7, Pages 1583–1590, 2017.

[27] DG Sanchez, B Lacarrière, M Musy, B Bourges, *"Application of sensitivity analysis in building energy simulations: Combining first- and second-order elementary effects methods,"*. Energy and Buildings, Volume: 68, Pages 741-750, 2014.

[28] E Vanuytrecht, D Raes, P Willems, *"Global sensitivity analysis of yield output from the water productivity model,"*. Environmental Modelling and Software Volume: 51, Pages 323-332, 2014.

[29] C Yu, L Huang , *"A Web service QoS prediction approach based on time-and location-aware collaborative filtering,"*. Service Oriented Computing and Applications, Volume: 10, Issue: 2, Pages 135–149, 2016.

[30] L Bass, P Clements, R Kazman, *"Software architecture in practice,"*. Addison-Wesley, 2012.

[31] N Rozanski, E Woods, *"Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives,"*. Addison-Wesley, 2011.

[32] C. M. WOODSIDE, *"Software resource architecture,"*. International Journal of Software Engineering and Knowledge Engineering, Pages 407-429, 2001.

[33] ME Iacob, H Jonkers, L van der Torre, *"Architecture analysis,"*. Enterprise Architecture at Work, Pages 51-68, 2017.

[34] C Smith, A Van Moorsel, *"Mitigating provider uncertainty in service provision contracts,"*. in Proc. Workshop on Economic Models and Algorithms for Grid Systems, Pages 143-159, 2010.

[35] W Hussain, FK Hussain, O Hussain , *"Risk-based framework for SLA violation abatement from the cloud service provider's perspective,"*. The Computer Journal, Pages 1306–1322, 2018.

[36] G Di Modica, O Tomarchio, L Vita, *"Dynamic SLAs management in service oriented environments,"*. Journal of Systems and Software Volume: 82, Issue: 5, Pages 759-771, 2009.

[37] AFM Hani, IV Paputungan, MF Hassan, *"Renegotiation in service level agreement management for a cloud-based system,"*. Journal ACM Computing Surveys (CSUR) Surveys Homepage archive Volume: 47, Issue: 3, 2015.

[38] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia , *"A view of cloud computing,"*. in: Magazine Communications of the ACM , Volume: 53, Issue: 4, Pages 50-58, 2010.

[39] A. Saltelli, K. Chan, M. Scott, *"Sensitivity analysis,"*. Probability and statistics series. West Sussex: Wiley, 2000.

[40] M.D. Morris, *"Factorial sampling plans for preliminary computational experiments,"*. Technometrics, Pages 161-174, 1991.

[41] B. Sudret., *"Polynomial chaos expansions and stochastic finite element methods,"*. Risk and Reliability in Geotechnical Engineering, 2015.

[42] T. Hastie, R. Tibshirani and J. Friedman, *"Elements of Statistical Learning: Data Mining, Inference and Prediction,"*. Springer—Verlag, New York, Pages 693-694, 2009.

[43] R. Agrawal, R. Bayardo, D. Gruhl, and S. Papdimitriou. "A service-oriented architecture for rapid development of web applications,". *Computer Networks*, Pages 523-539, 2002.

[44] R. Allen and D. Garlan. "A formal basis for architectural connection,". *ACM Trans. Software Engineering Methodology*, Volume: 6, Issue: 3, 1997.

[45] C. Catley, D. Petriu, and M. Frize. "Software performance engineering of a web service-based clinical decision support infrastructure,". In *Proc. 4th international workshop on software and performance*, Pages 130-180, 2004.

[46] B. Bani-Ismail and Y. Baghdadi. "A literature review on service identification challenges in service oriented architecture,". *Knowledge Management in Organizations, L. Uden, B. Hadzima, and I. Ting, Eds. Cham: Springer International Publishing*, 2018

[47] S. Gokhale and K. Trivedi. "Reliability prediction and sensitivity analysis based on software architecture,". In *Proc. 13th International Symposium on Software reliability Engineering*, 2002.

[48] K. Goseva-Popstojanova and K. Trivedi. "Architecture based approach to reliability assessment of software systems,". *Performance Evaluation*, 2001.

[49] J. Grundy, G. Ding, and J. Hosking. "Deployed software component testing using dynamic validation agents,". *Journal of Systems and Software: Special Issue on Automated Component-based Software Engineering*, Pages 5-14, 2005.

[50] J. Grundy and J. Hosking. "Engineering plug-in software components to support collaborative work,". *Software: Practice and Experience*, Pages 983-1013, 2002.

[51] R. Heckel and M. Lohmann. "Towards contract-based testing of web services,". *Electronic Notes in Theoretical Computer Science*, Pages 145-156, 2003.

[52] C. Hicks. *"Fundamentals Concepts in the Design of Experiments,"*. Oxford University Press, 4th edition, 1999.

[53] N. Liu, J. Grundy, and J. Hosking. "A visual language and environment for composing web services,". In *Proc. 2005 ACM/IEEE International Conference on Automated Software Engineering*, Pages 321–324, 2005.

[54] C. Lung and K. Kalaichelvan. "An approach to quantitative software architecture sensitivity analysis,". In *Proc. 10th Internatioanl Conference on Software Engineering and Knowledge Engineering*, Pages 97-114, 1998.

[55] A. Mos and J. Murphy. "A framework for performance monitoring, modelling and prediction of component oriented distributed systems,". In *Proc. ACM 3rd International Workshop on Software and Performance*, Pages 235–236, 2002.

[56] V. Sharma and K. Trivedi. "Architecture based analysis of performance, reliability and security of software systems,". In *Proc. 5th international workshop on Software and performance*, Pages 217–227, 2005.

[57] M. Shaw and D. Garlan. *"Software Architectures: Perspectives on an Emerging Discipline,"*. Prentice-Hall, 1996.

[58] C. Smith and L. Williams. *"Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software,"*. Addison-Wesley, 2002.

[59] B. Spitznagel and D. Garlan. "Architecture-based performance analysis,". In *Proc. International Conference on Software Engineering and Knowledge Engineering*, 1998.

[60] X. Yuan, S. Duan, and T. Huang. "Analysis of service-oriented architectures with sensitivity analysis,". In *Proc. 2006 IEEE International Conference on Electro/information Technology*, 2006.

[61] D. Kluver, M. Ekstrand, and J. Konstan. "Rating-Based Collaborative Filtering: Algorithms and Evaluation,". Springer International Publishing, 2018.

[62] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. "Eigentaste: A constant time collaborative filtering algorithm,". Information Retrieval, Pages 133–151, 2001.

[63] F. Samreen. "Eigentaste: A transfer learning-aided decision support system for multicloud brokers,". Ph.D. dissertation, School of Computing and Communications, Lancaster University, 2017.

[64] B Li, Q Yang, X Xue. "Transfer learning for collaborative filtering via a rating-matrix generative model,". Proceeding ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning, Pages 617–624, 2009.

[65] K. Weiss, T. M. Khoshgoftaar, D. Wang. "A survey of transfer learning,". Journal of Big Data, Pages 1345-1359, 2016.

[66] X. Yuan and P. Cheng, *"An approach of sensitivity and metamodel-based analyses for SLA violation prediction,"*. in 2019 Canadian Conference of Electrical and Computer Engineering, 2019.

[67] E. Orta, M. Ruiz, N. Hurtado, and D. Gawn, *"Decision-making in IT service management: a simulation based approach,"*. in Decision Support Systems, Volume: 66, Pages 36-51, 2014.

[68] D. Gabi, A. Ismail, A. Zainal, and Z. Zakaria, *"Quality of service (QoS) task scheduling algorithm with taguchi orthogonal approach for cloud computing environment,"*. in Recent Trends in Information and Communication Technology, Pages 641-649, 2018.

[69] Thomas Erl, *"SOA Principles of Service Design,"*. in Prentice Hall, 1st ed, July 2007.

[70] D Serrano, S Bouchenak, Y Kouki, *"SLA guarantees for cloud services,"*. Future Generation Computer Systems, Volume: 54, Pages 233-246, 2016.

[71] J. Powell, *"Service Level Agreements Too IT-centric,"*. Forrester Report Warns, 2012.

[72] Di, Sheng, Derrick Kondo, and Walfredo Cirne, *"Host load prediction in a Google compute cloud with a Bayesian model,"*. Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press, 2012.

[73] TS Wong, GY Chan, FF Chua, *"A Machine Learning Model for Detection and Prediction of Cloud Quality of Service Violation,"*. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA , Volume: 10960, Springer, 2018.

[74] S Wang, Y Zhao, L Huang, J Xu, CH Hsu, *"QoS prediction for service recommendations in mobile edge computing,"*. Journal of Parallel and Distributed Computing, Available online, 2017.

[75] R. A. Fisher and F. Yates, *"Statistical Tables for Biological,"*,Agricultural, and Medical Research, 4th Ed. Oliver and Boyd, Edinburgh, 1953.

[76] B. Bani-Ismail and Y. Baghdadi. *"A literature review on service identification challenges in service oriented architecture,"*. Knowledge Management in Organizations, L. Uden, B. Hadzima, and I. Ting, Eds. Cham: Springer International Publishing, 2018.

[77] AFM Huang, CW Lan, SJH Yang, *"An optimal QoS-based Web service selection scheme,"*. Information Sciences Volume 179, Issue: 19, Pages 3309-3322, September 2009.

[78] L Bass, P Clements, R Kazman, *"Software architecture in practice,"*. Addison-Wesley, 2012.

[79] N Rozanski, E Woods, *"Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives,"*. Addison-Wesley, 2011.

[80] C. M. WOODSIDE, *"Software resource architecture,"*. International Journal of Software Engineering and Knowledge Engineering, 2001.

[81] Y Zhang, Z Zheng, MR Lyu, *"An online performance prediction framework for service-oriented systems,"*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Volume: 44, Issue: 9, Pages 1169-1181, 2014.

[82] TS Wong, GY Chan, FF Chua, *"A Machine Learning Model for Detection and Prediction of Cloud Quality of Service Violation,"*. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA, Volume: 10960, Pages 498-513, 2018.

[83] W Hussain, FK Hussain, O Hussain, *"Profile-based viable service level agreement (SLA) violation prediction model in the cloud,"*. in: 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing.

[84] W Hussain, FK Hussain, OK Hussain, E Chang, *"Provider-based optimized personalized viable SLA (OPV-SLA) framework to prevent SLA violation,"*. in: The Computer Journal, Volume: 59, Issue: 12, Pages 1760–1783, 2016.

[85] W Hussain, FK Hussain, OK Hussain , *"Risk Management Framework to Avoid SLA Violation in Cloud from a Provider's Perspective,"*. in: Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Pages 233-241, 2016.

[86] Z Zhou, J Abawajy, M Chowdhury, Z Hu, K Li , *"Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms,"*. in: Future Generation Computer Systems, Volume: 86, Pages 836-850, 2018.

[87] S Singh, I Chana , *"Q-aware: Quality of service based cloud resource provisioning,"*. in: Computers and Electrical Engineering, Volume: 47, Pages 138-160, 2015.

[88] S Singh, I Chana, R Buyya , *"STAR: SLA-aware autonomic management of cloud resources,"*. in: IEEE Transactions on Cloud Computing, 2017.

[89] S Huang, S Zeng, Y Fan, GQ Huang, *"Optimal service selection and composition for service-oriented manufacturing network,"*. Journal International Journal of Computer Integrated Manufacturing, Volume: 24, Pages 416-430, 2011.

[90] ZZ Liu, DH Chu, ZP Jia, JQ Shen, L Wang, *"Two-stage approach for reliable dynamic Web service composition,"*. Knowledge-Based Systems, Volume: 97, Pages 123-143, April 2016.

[91] F Chen, R Dou, M Li, H Wu, *"A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing,"*. Computers and Industrial Engineering, Volume: 99, Pages 423-431, September 2016.

[92] V Gaur, P Dhyani, OP Rishi, *"A Multi-Objective Optimization of Cloud Based SLA-Violation Prediction and Adaptation,"*. Information Technology and Computer Science, Pages 60-65, 2016.

[93] VE Unnamalai, JR Thresphine, *"Service-oriented architecture for cloud computing,"*. International Journal of Computer Science and Information Technologies, Volume: 5, Pages 251-255, 2014.

[94] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K. *"Surrogatebased analysis and optimization,"*. Progress in Aerospace Sciences, 41(1), 2005.

[95] Wang, G. G., and Shan, S, *"Review of metamodeling techniques in support of engineering design optimization,"*. Journal of Mechanical Design, 129(4), 2007.

[96] Forrester, A. I., and Keane, A. J., *"Recent advances in surrogate-based optimization,"*. Progress in Aerospace Sciences, 45(1-3), 2009

[97] S Wright, J Nocedal, *"Numerical optimization,"*. Springer Science, 1999

[98] DE Goldberg, *"Genetic algorithms,"*. Pearson Education India, 2006

[99] S Kirkpatrick, CD Gelatt, MP Vecchi , *"Optimization by simulated annealing,"*. Science: Volume: 220, Issue: 4598, May 1983.

[100] D Baş, IH Boyacı , *"Modeling and optimization I: Usability of response surface methodology,"*. Journal of Food Engineering , Volume: 78, Issue: 3, Pages 836-845, February 2007.

[101] WC Lee, S Yusof, NSA Hamid, BS Baharin, *"Optimizing conditions for enzymatic clarification of banana juice using response surface methodology (RSM),"*. Journal of Food Engineering Volume: 73, Issue: 1, Pages 55-63, March 2006.

[102] AV Schenone, LO Conte, MA Botta, *"Modeling and optimization of photo-Fenton degradation of 2, 4-D using ferrioxalate complex and response surface methodology (RSM),"*. Journal of Environmental Management Volume: 155, Pages 177-183, May 2015.

[103] T Shojaeimehr, F Rahimpour, MA Khadivi, *"A modeling study by response surface methodology (RSM) and artificial neural network (ANN) on Cu2+ adsorption optimization using light expended clay aggregate (LECA),"*. Journal of Industrial and Engineering Chemistry, Volume: 20, Issue: 3, May 2014.

[104] A Cicek, T Kivak, E Ekici, *"Optimization of drilling parameters using Taguchi technique and response surface methodology (RSM) in drilling of AISI 304 steel with cryogenically treated HSS drills,"*. Journal of Intelligent Manufacturing, Volume: 26, Issue: 2, Pages 295–305, April 2015.

[105] S Altarazi, L Hijazi, E Kaiser, *"Process parameters optimization for multiple-inputs-multiple-outputs pulsed green laser welding via response surface methodology,"*. Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference.

[106] JB Lasserre, *"Global optimization with polynomials and the problem of moments,"*. SIAM Journal on optimization,Pages 796–817, 2001.

[107] H Öktem, T Erzurumlu, H Kurtaran, *"Application of response surface methodology in the optimization of cutting conditions for surface roughness,"*. Journal of Materials Processing Technology ,Volume: 170, Issues: 1–2, Pages 11-16, December 2005.

[108] G Gary Wang, Z Dong, P Aitchison, *"Adaptive response surface method-a global optimization scheme for approximation-based design problems,"*. Journal Engineering Optimization Volume 33, Pages 707-733, 2001.

[109] H Jie, Y Wu, J Ding, *"An adaptive metamodel-based global optimization algorithm for black-box type problems,"*. Journal Engineering Optimization, Volume: 47, Pages 1459-1480, 2015.

[110] G Zhou, LB Duan, WZ Zhao, CY Wang, ZD Ma, *"An enhanced hybrid and adaptive meta-model based global optimization algorithm for engineering optimization problems,"*. Science China Technological Sciences, Volume: 59, Issue: 8, Pages 1147–1155, August 2016.

[111] Ranjit K. Roy, *" Design of experiments using the Taguchi approach: 16 steps to product and process improvement,"*. Wiley-Interscience, 2001.

[112] JH Friedman, *"Multivariate adaptive regression splines,"*. The Annals of Statistics Volume: 19, Issue: 1, Pages 1-67, March 1991.

[113] Y Zhou, H Leung, *"Predicting object-oriented software maintainability using multivariate adaptive regression splines,"*. Journal of Systems and Software, Volume: 80, Issue: 8, Pages 1349-1361, August 2007.

[114] JH Friedman, CB Roosen, *"An introduction to multivariate adaptive regression splines,"*. Statistical Methods in Medical Research,Volume: 4, Issue: 3, 1995.

[115] IN Vuchkov and NL Boyadjieva, *"Quality improvement with design of experiments : A response surface approach,"*. Springer Science and Business Media, 2001.

[116] RB Gramacy, HKH Lee, *"Cases for the nugget in modeling computer experiments,"*. Statistics and Computing, Volume: 22, Issue: 3, Pages 713–722, May 2012.

[117] AbbasHeiat, *"Comparison of artificial neural network and regression models for estimating software development effort,"*. Information and Software Technology, Volume: 44, Issue: 15,Pages 911-922, December 2002.

[118] AJ Smola, B Schölkopf , *"A tutorial on support vector regression,"*. Statistics and Computing, Volume: 14, Issue: 3, Pages 199–222, August 2004.

[119] Martin Krzywinski , Naomi Altman , *"Points of Significance: Classification and regression trees,"*. Nature Methods, volume: 14, Pages 757–758, 2017.

[120] Leo Breiman , *"Classification and Regression Trees,"*. New York: Routledge, 1984.

[121] Z Zheng, H Ma, MR Lyu, I King, *"Collaborative web service qos prediction via neighborhood integrated matrix factorization,"*. IEEE Transactions on Services Computing, Volume: 6, Issue: 3, Pages 289-299, July, 2013.

[122] Y Zhang, Z Zheng, MR Lyu, *"An online performance prediction framework for service-oriented systems,"*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Volume: 44, Issue: 9, Pages 1169-1181, 2014.

[123] K Su, B Xiao, B Liu, H Zhang, Z Zhang, *"TAP: a personalized trust-aware QoS prediction approach for web service recommendation,"*. Knowledge-Based Systems, Volume: 115, Pages 55-65, 2017.

[124] X. Yuan and C. Ji, *"Performance analysis of service-oriented architectures with multi-factor sensitivity analysis,"*. 2007 IEEE International Conference on Electro/Information Technology, 2007.

[125] M Jamil, XS Yang, *"A literature survey of benchmark functions for global optimisation problems,"*. International Journal of Mathematical Modelling and Numerical Optimisation, Volume: 4, Issue: 2, Pages 1-47, 2013.

Vita Auctoris

Peng Cheng has received BS and MS degrees in the PLA Information Engineering University, China, and Tongji University, China. He has 10 years of research and development experience in both the university and industry. He is currently a candidate for the Doctoral degree in Computer Science at the University of Windsor, Ontario and hopes to graduate in Fall 2019.