

WEB SERVICE SELECTION BASED ON QoS PARAMETERS

Ph.D. THESIS

by

LALIT PUROHIT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247667 (INDIA)
DECEMBER, 2018

WEB SERVICE SELECTION BASED ON QoS PARAMETERS

A THESIS

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

LALIT PUROHIT



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247667 (INDIA)
DECEMBER, 2018**



**©INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, ROORKEE-2018
ALL RIGHTS RESERVED**



INDIAN INSTITUTE OF TECHNOLOGY ROORKEE ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "**WEB SERVICE SELECTION BASED ON QoS PARAMETERS**" in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of Computer Science and Engineering of the Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from July, 2014 to December, 2018 under the supervision of Dr. Sandeep Kumar, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

(LALIT PUROHIT)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Sandeep Kumar)
Supervisor

The Ph. D. Viva-Voce Examination of, Research Scholar,
has been held on

Chairperson, SRC

Signature of External Examiner

This is to certify that the student has made all the corrections in the thesis.

Signature of Supervisor

Head of the Department

Date: _____

Acknowledgements

First of all, I would like to thank Almighty, for carving the path towards my dream institute Indian Institute of Technology Roorkee. I thank him to provide me an opportunity to learn, experience, improve myself, and live in an environment surrounded by exceptionally brilliant human beings.

I would like to express my deep gratitude to my supervisor Dr. Sandeep Kumar for accepting me as his Ph.D. student so that I had an opportunity to work under his kind guidance at the department of Computer Science and Engineering, Indian Institute of Technology Roorkee. His invaluable expertise, continuous support, valuable suggestions and patient guidance led to successfully complete my Ph.D. dissertation work. Dr. Sandeep Kumar has been an excellent advisor and a man par excellence. I am inspired by his ability and principle of conducting high quality and impeccable approach towards conduction of research work and breadth of his knowledge in the area of web services. I have benefitted tremendously from his professional approach during my course work and research conduction phase. I learn and improved my technical writing skills. He always influenced me as an active thinker. I am thankful to Dr. Sandeep Kumar for his ever ready support, facilities offered to conduct quality research and his keen sight to give me right direction throughout my Ph.D.

It is an honor for me to thank my committee members Dr. R. Balasubramanian, Dr. Durga Toshniwal and Dr. Dharmendra Singh for their valuable time and their every appreciated suggestion during my Ph.D. Their insightful comments and suggestions have significantly improved the quality of my work. My cordial acknowledgement to the head of department, Prof. Dharmendra Singh, for providing me the required infrastructure to conduct quality research. It is a great opportunity for me to also thank Prof. Manoj Mishra, to provide the learning environment and taking care of technical and non-technical requirements.

I would like to thank Indian Institute of Technology, Department of Computer Science and Engineering, and all professors and staffs, for giving me the chance of continuing my education in my very interested area, supporting me in every part of my study, and helping me throughout the whole duration of my Ph.D. here.

I wish to extend my deepest appreciation to my father Shri Jaikishan Purohit, who was always an encouragement and a powerful help angel for me in every part of my life, my mother

Smt. Gita Purohit who provided me a safest environment with her extreme support, my wife, Gunjan Purohit, who tolerated everything with love and kindness, and is a precious motivation in my whole life, my kids Aishwarya and Himani for their love and support during difficult times. Without their sacrifices, ever ready support and encouragement in all situations, from each of them, the completion of my Ph.D. would not be possible.

I have also been lucky to be a member of research scholar lab. I am indebted to many of my friends Dr. Santosh Rathore, Pravas Ranjan Bal, Dr. Satyendra Chouhan, Jayendra Barua, Deepak Kshirsagar, Ankush Agarwal, Avadh Kishore, Sandeep Shingade, Amar Dishilva, Vikas Chouhan, Vivek Raj, Suyash Shukla, Ajay Chaudhary, Manik Chandra, Sujata Swain for their supports anytime and anywhere they could.

Special thanks to QIP centre, IIT Roorkee and AICTE to provide financial support in the form of student assistantship. Many thanks to Director, SGSITS Indore and Head, Department of I.T., SGSITS Indore, to provide me leaves and other support required for completion of Ph.D.

Finally, I would like to thank everyone who directly or indirectly helped in achieving the successful completion of my Ph.D.

LALIT PUROHIT

Abstract

The potential of web service to automate machine to machine interaction has developed faith in using web service technology for building smart applications. It leads to increase in the interest of software developers to offer services in place of standalone software to the community of users. Multiple services offering identical functionality are available due to unabated uptake use of web service technology. In this situation, either the end user should manually select the web service or the system should have capability to identify the required web service satisfying the requirements of the user. The manual selection is not time efficient and does not results in the selection of optimal web service (low accuracy). In the other case, the selection process can be automated to identify the appropriate web service based on QoS or other parameters. With the enormous increase in number of web services offering identical functionality, the task of selection of web service itself become more difficult and challenging.

The task of selection of appropriate web services for the case of composition becomes more challenging. The challenge is in terms of selection of desired web services by maintaining the system performance along with due consideration to the end-to-end QoS. The performance of web service selection mechanism has an effect on the performance of system built using web services. Poor performance of a web service selection mechanism may result into poor and inaccurate system performance. Also, mostThe majority of the existing approaches to web service selection consider all of the candidate web services for selection without any preprocessing or filtering. Thus, these existing approaches do unnecessary processing for those web services which are far below the expectations of the end user. The extra processing leads to poor selection results and degraded system performance. Further, web services run in a very dynamic environment and are prone to failure or run time change in service quality. On the occurrence of failure or unavailability of any component service, the reselection of services is required. In case of composite web service, the reselection process is more complex and time consuming. In this thesis, we have tried to handle these issues.

We have studied the existing approaches to deal with above discussed issues and proposed improvements for selection of web services. A classification based approach for web service selection based on QoS parameters is presented. Initially, we evaluate the performance of eleven classification techniques for classifying web services. This analysis helped us to identify

top three classification techniques for achieving efficient classification of web services. Majority vote based classification model is developed by combining the output of top three classification techniques and is used to classify web services. The classification step is used to prefilter the candidate web services. An improved PROMETHEE method, we call it as PROMETHEE Plus, is presented and is applied to most eligible web services for selection of most suitable services. Maximizing Deviation Method based hybrid weight evaluation mechanism is adopted for weight evaluation of various QoS parameters.

We explore the use of clustering to determine similarity among candidate web services on the basis of QoS information. To identify the potential clustering technique, a systematic analysis is done to evaluate the performance of six clustering techniques based on efficiency and quality of clustering consideration. The best performing clustering technique is applied on candidate web services. Pruning along with clustering act as prefilter for candidate web services and this step identifies most promising set of web services with capability to meet the end user expectations. On the most prominent set of web services, proposed Skyline Plus method is applied for selection. Further, this thesis presents an efficient solution for replaceability of web services. Service selection is performed using modified genetic algorithm. To deal with the problem of service failure at run time, the replacement services are determined. In the presented approach, a web service selection mechanism by taking into cognizance the replaceability of service as the basis of selection is used. At the time of selection, the replaceability of a web service is determined using the proposed PROMETHEE Plus method. Further, the replaceability evaluation is enhanced by including IOPE based web service matchmaking. The matchmaking process is enhanced by using determinant method.

We have gone through the existing works and found two appropriate and highly popular labeled datasets in this area. One of the dataset is QWS dataset which is very popular QoS dataset measured on real-world web services with 364 labeled web services and 2507 unlabelled web services. To the best of our knowledge, this is the only available dataset consisting of a labeled set of QoS information measured on real-world web services. This dataset is highly used for experimentation in the area of Web Services Selection. Another dataset is generated using synthetic data generator, which is highly popular and highly cited. We have performed experimentation on this dataset as well to validate our results.

Based on the findings of the works presented in the thesis, it is observed that the use of combination of classifiers improves the classification results as compared to using individual classifiers for classification of web services. The efficiency of the selection system is improved by the use of classification mechanism as a prefilter to the selection process. The results of experimentations and Tukey test showed that the inclusion of end user request during evaluation of services by PROMETHEE Plus method generates improved selection results with enhanced end user satisfaction. Further, it is revealed from the presented works that the quality of clustering and stability measurement of clustering techniques are important parameter to find the suitability of clustering techniques. The use of clustering technique to identify QoS based similarity among web services along with pruning act as an excellent prefilter for candidate web services. Use of pruning and clustering based Prefilter ensure that the final set of skyline services closely meet the end user expectations. The results from various experiments and Tukey test confirm that Skyline Plus gives improved selection results. It is further concluded that the selection of web services with consideration to replaceability improves the system availability and fault tolerance capability.

Table of Contents

Acknowledgements

Abstract

List of Figures

List of Tables

List of Abbreviations

1. INTRODUCTION	1
1.1 Web Services	3
1.2 Web Service Selection	3
1.3 Commonly used Techniques for Web Service Selection	5
1.4 Motivation	8
1.5 Objectives of Study.....	9
1.6 Contribution of the work.....	10
1.7 Organization of Thesis	12
1.8 Conclusions	14
2. LITERATURE REVIEW	15
2.1 Introduction	16
2.1.1 Quality of Service model for Web Service	17
2.2 Classification of Web Services	19
2.3 Clustering of Web Services	22
2.3.1 Web Service Description Language	23
2.3.2 Semantic Information	24
2.3.3 Tags	24
2.3.4 User similarity	25
2.3.5 Quality of Service	25
2.4 Replaceability based Web Service Selection	27
2.4.1 QoS based Replaceability and Web Service Selection	28

2.4.2 IOPE based replaceability and Web Service Selection	50
2.5 Web Services Datasets	52
2.5.1 QoS dataset	52
2.5.2 OWL-S dataset of web services	54
2.6 Performance evaluation measures	55
2.7 Observations drawn from the Chapter	57
2.8 Conclusions.....	58
3. EMPIRICAL STUDY OF WEB SERVICE CLASSIFICATION TECHNIQUES	59
3.1 Introduction	59
3.2 Choosing the classification model.....	60
3.3 Web Service Classification Models.....	61
3.3.1 Tree based learning model.....	62
3.3.2 Function based learning model.....	62
3.3.3 Rule based learning model.....	63
3.3.4 Majority vote based learning model.....	64
3.4 Analysis and observations.....	65
3.4.1 Experimental setup.....	65
3.4.2 Analysis of six learning models	66
3.4.3 Evaluation of Majority Vote learning model	69
3.4.4 Discussion	70
3.5 Threats to validity	71
3.6 Conclusions	71
4. CLASSIFICATION BASED APPROACH FOR WEB SERVICE SELECTION	73
4.1 Introduction and Motivation	73
4.2 Proposed approach for Web Service Selection	76
4.2.1 Motivating Example	76
4.2.2 Web Service Selection Approach	78
4.2.3 The proposed CSS approach for Web Service Selection	80
4.3 Experimental Analysis	89

4.3.1	Experimental Setup	89
4.3.2	Results and Evaluation.....	92
4.4	Discussion	104
4.5	Conclusions	105
5.	EMPIRICAL STUDY OF WEB SERVICES CLUSTERING TECHNIQUES	106
5.1	Introduction	106
5.2	Web Service Clustering techniques	107
5.3	Analysis and Observations.....	110
5.3.1	Experimental setup	110
5.3.2	QoS parameter selection using Principal Component Analysis (PCA)	111
5.3.3	Performance evaluation parameters	111
5.3.4	Experimental study	113
5.4	Observations and Discussion.....	116
5.5	Conclusions	117
6.	CLUSTERING BASED APPROACH FOR WEB SERVICE SELECTION	118
6.1	Introduction and Motivation	118
6.2	Proposed approach for Web Service Selection	123
6.2.1	Web Service Selection	124
6.2.2	Web Service Selection Algorithm	130
6.3	Evaluation of the proposed approach	132
6.3.1	Performance parameters.....	133
6.3.2	The dataset and design of user queries	135
6.3.3	Results and Evaluation.....	136
6.4	Discussion	144
6.5	Conclusions	145
7.	REPLACEABILITY BASED APPROACH FOR WEB SERVICE SELECTION	146
7.1	Introduction and Motivation.....	146
7.2	Proposed approach for web service selection	151

7.2.1	Web Service Selection.....	151
7.2.2	QoS based replaceability of web services.....	154
7.2.3	IOPE matchmaking based Replaceability of Web Services	156
7.2.4	modified Genetic Algorithm (mGA)	163
7.3	Algorithm for Web Service Selection	165
7.4	Results and Observations	169
7.4.1	Evaluation of determinant method	169
7.4.2	Evaluation of the proposed WS Selection method	171
7.4.3	Comparison on replaceability factor	173
7.4.4	Performance comparison of the proposed $WSSR_p$, $WSSR_{pM}$ methods	174
7.4.5	Comparison on precision	175
7.5	Conclusions	177
8.	CONCLUSIONS AND FUTURE WORKS	179
8.1	Conclusions	179
8.2	Future works	182
	Publications	183
	Bibliography	184

List of Figures

Figure 1.1	Quality distribution of web services of hotel and weather domain from QWS dataset	4
Figure 2.1	Architecture of web service	16
Figure 2.2	Categorization of clustering criteria used for clustering WSs in existing works	23
Figure 2.3	Flowchart of Genetic Algorithm for WS Selection	32
Figure 2.4	Distribution of works based upon improvements in different steps of GA for WSS	33
Figure 3.1	Box-plot analysis for AAE and ARE measures for Six learning models	67
Figure 3.2	AAE and ARE based interval plot for difference of mean between LR, NNge, J48 and majority vote for classification of web services with confidence interval of 95%.....	70
Figure 4.1	Proposed system architecture for MSS approach	79
Figure 4.2	Proposed system architecture for HSS approach	80
Figure 4.3	Proposed system architecture for CSS approach	81
Figure 4.4	CSS algorithm for Web service selection	83
Figure 4.5	Classification based Prefiltering algorithm	84
Figure 4.6	PROMETHEE Plus algorithm for ranking of filtered web services and selection of Top-K Web Services	88
Figure 4.7	Selection of Top-K Web services from ranked list	89
Figure 4.8	Effect of variation of 'K' on satisfaction score	93
Figure 4.9	Measuring the effect of variation in the hardness of QoS query on the end user satisfaction score.....	94
Figure 4.10	Box plot analysis of mean Euclidean distance measure on 100 end user queries	95
Figure 4.11a	The mean plot for satisfaction score for five WS Selection approaches (K=3).....	95
Figure 4.11b	Box plot analysis of mean satisfaction score of 100 end user queries.....	95
Figure 4.12	Selection time comparison of five web service selection approaches.....	97
Figure 4.13	Effect of variation of 'K' on satisfaction score	99
Figure 4.14	Measuring the effect of variation in the hardness of QoS query on the end user satisfaction score.	99
Figure 4.15	Box plot analysis of mean Euclidean distance measure on 100 end user queries (K=3).....	100
Figure 4.16	The mean plot for satisfaction score for five WS Selection approaches (K=3)	100
Figure 4.17	Selection time comparison of five WS Selection approaches	102

Figure 5.1	Web service clusters created by performing QoS based clustering.....	108
Figure 5.2a	Silhouette coefficient based evaluation of clustering techniques on Silhouette coefficient evaluated using nine QoS parameters available from QWS	114
Figure 5.2b	Comparison of six clustering techniques on Silhouette coefficient evaluated using six QoS parameters obtained by applying PCA on QWS	114
Figure 5.3a	Comparison of six clustering techniques on Average distance between means (ADM) parameter evaluated using nine QoS parameters available from QWS	115
Figure 5.3b	Comparison of six clustering techniques on Average distance between means (ADM) parameter obtained by using six QoS parameters determined by applying PCA on QWS	116
Figure 6.1	Quality distribution of Web Services with identification of Skyline services using Documentation (z-axis), Reliability (x-axis) and Latency (y-axis) QoS parameter.....	122
Figure 6.2	Proposed system architecture for CPSky-FS approach	125
Figure 6.3	Conceptual view of proposed approach	126
Figure 6.4	CPSky-FS algorithm for web service selection	131
Figure 6.5	Algorithm for prefiltering web services (Prefilter layer)	132
Figure 6.6	SkylinePlus algorithm for top- <i>K</i> WSs selection (Selection layer	133
Figure 6.7	Effect of variation in ' <i>K</i> ' on satisfaction score	137
Figure 6.8	Satisfaction score vs query hardness (<i>K</i> =10)	138
Figure 6.9	Box plot analysis of satisfaction score for four web service selection approaches	140
Figure 6.10	Mean plot of Euclidean distance for four web service selection approaches.....	142
Figure 6.11	Time comparison of four web service selection approaches	143
Figure 7.1a	Web Service composition scenario	147
Figure 7.1b	Example composite web service	147
Figure 7.2	Pictorial representation of failure point in the composition.....	149
Figure 7.3	Example of web service selection for Composite web service	152
Figure 7.4	Architecture of the proposed $WSSR_P$ Approach.....	153
Figure 7.5	Architecture of the proposed $WSSR_{PM}$ Approach	155
Figure 7.6	Sorted list of services generated as output by applying PROMETHEE Plus method on concrete web-services	155
Figure 7.7	IOPE matching to determine WS similarity.....	157
Figure 7.8	A bipartite graph obtained after matchmaking $Input_{req}$ and $Input_{adv}$	159


Figure 7.9	Priority queue based on the Exact, Plugin, Subsume and Intersect type	162
Figure 7.10	Example service composition plan (WS_j) generated by random choice of concrete WS corresponding to each AWS	164
Figure 7.11	WSSR _{PM} algorithm for web service selection using mGA.....	165
Figure 7.12	Algorithm for replaceability evaluation.....	166
Figure 7.13	Determinant algorithm to obtain maximum matching for Semantic matching of web services...	167
Figure 7.14	Selection of Web Service from priority queues.....	168
Figure 7.15	Matchmaking time comparison for Hungarian and Determinant method for nine domains	170
Figure 7.16	Effect of increase in number of AWS and CWS.....	171
Figure 7.17	Performance comparison of WSSR _P with WSSR _{NN} on the basis of time for selection with variation in number of concrete WSs	172
Figure 7.18	Comparison of WSSP _R and WSSN _R on satisfaction score	174
Figure 7.19	Comparison of WSSR _{NN} , WSSR _P and WSSR _{PM} on means plot of satisfaction score.....	175
Figure 7.20	Comparison of three WS Selection approaches on Precision value	176

List of Tables

Table 2.1	Example of QoS attributes and QoS metric	17
Table 2.2	Recent work on web service classification	20
Table 2.3	Recent work on web service clustering	27
Table 2.4	Summary of Fitness functions.....	38
Table 2.5	Summary of comparison between GA and other suggested techniques	42
Table 2.6	Summary of genetic algorithm based web service selection approaches	45
Table 2.7	Comparison of Matchmaking techniques	51
Table 2.8	Web Service QoS parameter definition	53
Table 2.9	Sample entries for QWS dataset	53
Table 2.10	Details of dataset of 10000 web services generated using dataset generator	54
Table 2.11	Nine major domains and details of number of WSs in each major domain	55
Table 3.1	Comparative analysis of eleven learning models for web service classification using nine QoS parameters	61
Table 3.2	Labeled QWS dataset with class descriptions	66
Table 3.3	Performance evaluation parameters	66
Table 3.4	AAE and ARE analysis of six learning models for web service classification	67
Table 3.5	Tukey test for six web service classification models on AAE and ARE measures (95% Confidence Interval for Mean)	68
Table 3.6	Sheskin multiple comparison test	69
Table 3.7	AAE and ARE measure of LR, NNge, J48 and Majority vote learning models	69
Table 4.1	Example WSs with Cost And Response Time QoS	77
Table 4.2	User specified QoS parameters and their weights	78
Table 4.3	Summary of three variations of proposed WSS approach	79
Table 4.4	Comparison of QoS Weight calculation techniques	86
Table 4.5	User Queries with Hardness And Hardness Level	91
Table 4.6	PROMETHEE Parameters Evaluated Based On Maximum Satisfaction Score	92
Table 4.7a	Minimum, maximum and standard deviation value for mean satisfaction score measure of five WS Selection approaches	96
Table 4.7b	Tukey multiple comparison test for satisfaction score measure	96
Table 4.8	Details of dataset-2 consisting of 10000 web services	98

Table 4.9a	Minimum, maximum and standard deviation value for mean satisfaction score measure of five WS Selection approaches	101
Table 4.9b	Tukey multiple comparison test for Satisfaction score measure	102
Table 4.10	A comparison of existing state-of-the-art with proposed approach for web service selection ..	103
Table 5.1	Sample entries for QWS dataset	110
Table 5.2	QoS parameters with their min and max values	111
Table 6.1	Web services with three QoS parameters.....	122
Table 6.2	Example web services with four QoS parameter values	128
Table 6.3	The end user queries with different QoS demands	136
Table 6.4	Comparison of Maximum satisfaction score	139
Table 6.5a	Minimum, maximum, and standard deviation value for mean satisfaction score measure of four WSS approaches	140
Table 6.5b	Tukey multiple comparison test for measured satisfaction score	141
Table 6.6	A comparison of existing state-of-the-art with proposed approach for web service selection ..	143
Table 7.1	Match type defined in comparison of Input parameters	158
Table 7.2	Dataset used for evaluation	171
Table 7.3	Comparison of the proposed approach ($WSSR_p$) for WS Selection with an existing approach ($WSSR_{NN}$) on replaceability factor	173
Table 7.4	A comparison of existing state-of-the-art with proposed approach for web service selection	177

List of Acronyms



AAE	Average Absolute Error
A _b WS	Abstract Web Service
ACO	Ant Colony Optimization
ADM	Average Distance between Means
AHP	Analytical Hierarchy Process
ANP	Analytical Network Process
ARE	Average Relative Error
CBR	Case Based Reasoning
CWS	Concrete Web Service
Clara	Clustering for the large data set
FS	Feature Selection
GA	Genetic Algorithm
IOPE	Input, Output, Precondition, and Effect
LR	Logistic Regression
MCDM	Multi-criteria Decision Making
MDM	Maximizing Deviation Method
MIP	Mix Integer Programming
MTTF	Mean Time To Failure
MTBF	Mean Time Between Failure
NNge	Non-nested generalized exemplars
OWL	Ontology Web Language
PAM	Partitioning around medoids
PCA	Principal Component Analysis
QoS	Quality of Service
RF	Random Forest
SAW	Simple Additive Weight
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOAT	Self Organizing Tree Algorithm
UDDI	Universal Description, Discovery and Integration
WS	Web Service
WSS	Web Service Selection
WsRF	Web Service Relevancy Function
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

The web service technology enables us to develop software components programmatically accessible over the Internet. Owing to this, it is possible to integrate software components without boundaries [Hwang et al., 2015]. The web service based implementation of any system essentially satisfies the properties such as loosely coupled, platform independence and able to operate and interact in heterogeneous environment. With the increasing use of web services in day to day interaction of user's over Internet, service providers are motivated to implement web service. Due to technological advancements and evolution of new technologies, the use of web service and web service based softwares has been increasing continuously. A lot of web service providers are offering their business functionality as web service [Gangadharan et al., 2016]. This has resulted in an exponential increase in the number of functionally similar web services with diverse service quality. Hence, the selection of desired web service from multiple competing services offering same functionality is a difficult and challenging task [Kumar and Mastorakis, 2010].

The cost of manual selection of required web service is higher than the automatic selection. Further, the increase in the number of functionally identical web services raises the cost as well as the complexity of automated web service selection (WSS) system. The use of Quality of Service (QoS) parameters to select the required web service is one of the popular mechanisms that help in reducing the complexity of the selection process [Helal and Gamble, 2014]. The quality can be defined as "*the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied need*" [Hoyle, 2001]. Quality of web service is a combination of various qualities of web service and is a measure of how well the web service serve's the user request [Hoyle, 2001]. QoS parameters are used for describing the non-functional features of web service i.e. the service quality offered by the web service to the end user. The end user query specifies the expected quality from the web service. The WSS system matches the expected quality with the quality offered by the web service. In response to the end user query, WSS system selects

the web service offering desired functionality for a task to be performed and satisfying the quality requirements. The performance of WSS system is one of the majorly contributing factors which affect the quality of services offered and satisfaction level of the end user in using the web service.

The solution to the problem of web service selection by using QoS parameters is a well accepted solution. The available techniques uses QoS parameters associated with the web service to either prioritize web services on the basis of QoS score or identify web services with high quality offerings. An important concern about these techniques is the selection of web services with reference to the end user request. As the end user always prefer and feel comfortable in using the service closely matching her requirements. Another aspect of concern is the use of preference of QoS (in terms of values of weights of QoS) during evaluation of web services to generate priority ordering. The values of weights of QoS largely affect the quality of web services being selected. Further, the conflicting nature of QoS parameters has a profound effect on the results of service selection.

One of the popular solutions to the problem of WSS is the use of classification techniques. Earlier web service selection systems used a wide range of learning techniques and clustering techniques to predict the service quality. The results of these studies showed the limited capabilities of learning technique. The prediction accuracy of these techniques found to be considerably lower in the range from 75% to 80% [Mohanty et al., 2010, Patro and Patra, 2015]. An important concern related to web service selection is the selection of suitable learning technique. Moreover, the use of clustering technique to improve the results of WSS is also useful. However, the choice of clustering technique to be used to achieve services clustering is of prime concern. The results of studies made on web services clustering showed that the appropriate ground for choice made in selecting the clustering technique is absent. Furthermore, the run time failure of selected web service further degrades the performance of web service based software's. The selection of equivalent web service which can act as substitution for the failed web service without affecting the underlying expectations of the end user is important to be considered [Ezenwoke et al., 2017].

The chapter is organized as follows. Section 1.1 introduces web services. Section 1.2 elaborates the need of QoS based WSS. Section 1.3 presents various available techniques for WSS. The motivation behind the thesis work is presented in Section 1.4. The objectives of the thesis work and important contribution of the thesis are elaborated in Section 1.5 and 1.6, respectively. The

organization of the thesis has been presented in Section 1.7 followed by various conclusions drawn from the chapter in Section 1.8.

1.1 WEB SERVICES

Web Service (WS) is a software component accessible over Internet by other software's using XML standards (e.g. SOAP, WSDL). Web service technology enables two software components to interact with each other by allowing them to exist on different platform and written in any native language. WS is a piece of code which shares data, processes and business logic via an interface [Alrifai et al., 2011]. The potential of WS technology to automate machine to machine interaction has developed faith in using WS technology for building smart applications. It leads to increase in the interest of software developers to offer services in place of standalone software to the community of users. The offered services are consumed on pay per use basis. Overall, the end user is relieved from the burden of payment and maintenance of the whole software. Similarly, the service oriented model of software development is more useful for the service providers to promote their business quickly and in easy way.

1.2 WEB SERVICE SELECTION

Web services allow machine to machine interaction and machine to user interaction over the Web [Ezenwoke et al., 2017, Tewari et al., 2012a]. Using WS concept, machines can interact with each other without any human intervention. This is true when WSs to perform the intended task are already identified and known. Otherwise, the process of selection of WS(s) is to be performed. When the number of WSs is less, the task of selection is very trivial. However, with the increasing popularity of WSs and interest of the software developers, many WSs offering duplicate functionality are available over the Internet.

This has increased the complexity of selection process. Selection of most promising WS satisfying needs of the end user is a core and challenging task in Service Oriented Architecture (SOA). The selection of desired WS from multiple competing services offering same functionality is difficult. The task of selection is usually performed in two steps. First, from the WS pool, services offering identical functionality are fetched. In this regard, existing solutions such as AI planning based methods [Hwang et al., 2015] or semantic based approaches [Kumar and

Mastorakis, 2010, Purohit and Kumar 2016a, Kumar and Mishra 2008a] are efficient. Next, from the list of functionally similar WSs, a WS is to be selected. Usually, QoS parameters are used to differentiate among functionally equivalent WSs. The QoS parameters such as availability, response time, reliability, throughput, documentation, successability, best practices for web services, latency etc., are among the few popular QoS parameters [Masri and Mahmoud, 2007, Shehu et al., 2014]. The QoS parameters collectively characterize the quality a service offers. The QoS parameters such as availability, reliability, throughput, are considered better for high value. On the contrary, the QoS parameters such as latency, response time, cost are lower the better. Web services with same functionality may exhibit different quality. To understand this idea, we have identified few services of weather and hotel domain from QWS dataset [Masri and Mahmoud, 2007]. Weather service determines the current weather condition and hotel service offers the facility such as, hotel booking, the current booking status of hotel, etc. Based on the throughput, response time, and availability QoS parameters, quality distribution of few services of hotel and weather domain is drawn and shown in Figure 1.1. It can be analyzed from the figure that few of the WSs are best in all three QoS parameter values, while few other services are having low response time but have low availability and low throughput. Few of the WSs have moderate quality. From the figure, it is clear that the web services offer similar functionality, but differ in their QoS values. Therefore, QoS parameters can be regarded as a useful criterion by the WSS system to discriminate two WSs. Further, the WSS system also measures, how well the QoS requirements of the end user are satisfied by the candidate WSs. This measure is used to rank the candidate services.

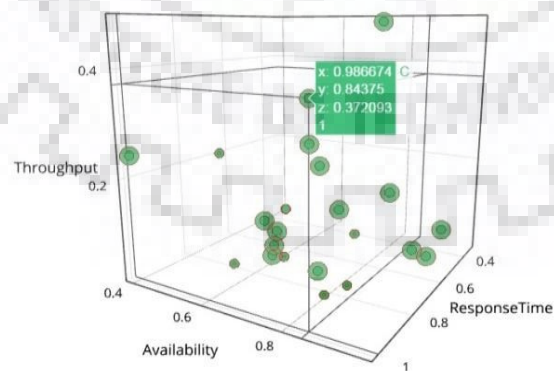


Figure 1.1. Quality distribution of web services of hotel and weather domain from QWS dataset [Masri and Mahmoud, 2007, Masri and Mahmoud, 2009].

Research Problem: The set $S = \{S_1, S_2, \dots, S_n\}$ is the set of candidate web services offering similar functionality and has some QoS offerings. Based on the offered functionality and non-functional characteristics, the WS is defined as, $S_i = \{I, O, QoS\}$ where 'I' is the input(s) required by the WS to process, 'O' is the output(s) generated by the WS. The 'I' and 'O' together illustrates the functionality provided by the WS. QoS is the multiple QoS attributes associated with the WS and describe the non-functional characteristics of the WS. If $S'\{UQoS\}$ is the QoS specifications of the desired WS by the end user, the problem of WSS is to select S_i such that, $S_i\{QoS\} \equiv S'\{UQoS\}$. In this work, it is assumed that the functionally similar WSs are available as a result from the execution of WS discovery operation based on 'I' and 'O' parameters. Moreover, to deal with the problem of runtime failure or change in QoS values of selected WS, an equivalent WS is to be selected to substitute failed WS.

1.3 COMMONLY USED TECHNIQUES FOR WEB SERVICE SELECTION

The process of creating the relative ranking of WSs based on QoS parameters can be reduced to the problem of decision making. The Multi Criteria Decision Making (MCDM) methods are suitable to solve such problems [Brans and Vincke, 1985]. The Preference Ranking and Organization METHod for Enrichment Evaluation (PROMETHEE-II) is a popular MCDM method used in past for WSS [Brans and Vincke, 1985, Herssens et al., 2008]. There are other available approaches for efficient WSS such as using the Probabilistic QoS based selection [Hwang et al., 2015], Skyline technique [Ouahad et al., 2015, Rhimi et al., 2015, Mobedpour and Ding, 2013], Utility function based selection [Alrifai et al., 2010], Mix Integer Programming (MIP) [Saleem et al., 2015, Cho et al., 2016], Case Based Reasoning (CBR) [Renzis et al., 2016], Collaborative filtering [Zheng et al., 2011], service bundling [Alrifai et al., 2011, R.-Zapata et al., 2011, R.-Zapata et al., 2015], Genetic Algorithm [Helal and Gamble, 2014], Semantic based approach [Kumar and Mishra, 2008a], classification approach [Wang et al., 2010, Yang and Zhou, 2015], Clustering approach [Xia et al., 2011], among others.

QoS is being used as an important parameter to distinguish among functionally similar WSs. It is increasingly difficult and time consuming task to obtain best WS due to conflicting QoS

parameters. An existing QoS based WSS approach treats QoS parameters as discrete random variables with probability mass functions [Hwang et al., 2015]. The service selection is performed by computing the utility function based on the probability of satisfying QoS constraints. A similar approach for WSS using utility function for sorting and selection of web services is available [Alrifai et al., 2011]. In this approach, the distance between the service candidate and the upper bound is obtained. Based on distance values, candidate WSs are sorted. Nevertheless, the approach does not consider the conflicting nature of QoS parameters during evaluation of the utility function, which we believe are very crucial for producing an accurate ranking of services. Web Service Relevancy Function (WsRF) can also be used for web service ranking and selection [Masri and Mahmoud, 2007]. Higher value of relevancy function indicates better service. Similar to our work, variety of solutions for QoS based WSS problem are available. However, these solutions do not consider the conflicting nature and interdependency among QoS parameters.

An effective method for comparison of alternatives based on the conflicting QoS attributes is PROMETHEE-II [Brans and Vincke, 1985]. It performs total ordering of alternatives. The PROMETHEE method for WSS is introduced by [Seo et al., 2005]. The QoS parameters are mapped to the actions and service providers are the alternatives. To define the criteria, appropriate choice of preference function is to be made. For qualitative criteria (performance, availability), usual/Gaussian type preference functions are suitable. Similarly, for quantitative criteria (cost, price, and power), V-shape/level/linear type preference functions are suitable [Brans and Vincke, 1985]. The preference index of services needs to be evaluated by pair wise comparison of alternatives. The net outranking flow for each candidate WS is calculated, which results in a ranking of WSs [Karim et al., 2011]. The PROMETHEE method does not include any predefined weight calculation scheme. The weight calculation is kept flexible.

WSs can be grouped together by performing classification task. Classification of WSs can be achieved by using information in WSDL [Yang and Zhou, 2015], OWL-S [Wang et al., 2010], QoS [Patro and Patra, 2015], and others. The available works on classification using WSDL and OWL-S performs classification at the time of service discovery. As it is performed at the time of service discovery, so, they are increasing the efficiency of discovery process, generating a set of functionally similar WSs. To further reduce the set of functionally similar WSs, in one of the work, we have performed classification of WSs based on associated QoS information. Thus, classification is applied in prefiltering step to reduce domain of search.

The Skyline based approach for WSS involves selection of best WS without considering any weight preference. The Skyline service computation is done using QoS parameters associated with the service. QoS data may be of different types such as independent, correlated, or anti-correlated. These types of data have an effect on the performance of the WSS methodology. Nonetheless, the Skyline technique has its own issues [Rhimi et al., 2015]. The Skyline approach selects same list of services irrespective of the end user requirements. It causes the same set of services to be presented before different end users, independent of their QoS requirements. Therefore, in addition of ignoring the end users requirements, it also causes problem of imbalance of load on a specific set of services [Wang et al., 2016]. The end users requirements during evaluation are also ignored in existing MCDM based WSS models [Heressens et al., 2008, Ouadah et al., 2015]. The Skyline technique is also useful to select WS in the case of composition. The number of tasks in the composition is represented using abstract WSs. For each of the abstract WS, a concrete WS is selected using the Skyline technique [Alrifai et al., 2010]. The method is not suitable for the case when the number of task in the system is very large. Also, the task of reselection of replaceable web service is not efficient. The existing approaches do not take into account the functional behaviour of services during the evaluation of replaceable WS.

An approach to cluster WSs based on the end user expectations of QoS and past quality ratings of the service is available in [Zhang et al., 2013]. In this approach, K-means based clustering with detection of outliers for WSS to improve the QoS prediction is presented. However, the approach requires a large number of comparisons in order to select the desired WS from the long list of WSs. The concept of clustering of WSs is useful in selecting alternative services to handle the case of unavailability/failure of selected WS [Xia et al., 2011]. In this work, the selection of service for composite WS is considered. The clustering approach is useful to group the WSs on QoS parameters and perform search and selection operations in the group. For each of the abstract WSs, clusters of concrete WSs are formed. The Optics algorithm is used to cluster WSs based on QoS information. At run time, various WSs corresponding to the abstract WSs are selected and executed. If a run time failure/unavailability of service is experienced, a WS from the same class with the highest utility value is selected as an alternative service to replace the currently executing WS.

1.4 MOTIVATION

The growing demand for high performance and efficient web service based software systems results in the increased complexity of selection process [Renzis et al., 2016]. Meanwhile, it is impractical to ensure that the run time failure of web services will not occur or the values of QoS parameters will not change in the present operating environment. Thus, the service selection mechanism can be enhanced to perform WSS by considering the capability of other web services to replace the failed/unavailable web service. This will help in narrowing the efforts required to reselect the web service upon occurrence of failure as well as ensure the end users satisfaction in using the web service.

Numerous approaches have been presented to perform QoS based WSS. However, there are some critical issues associated with the web service selection process, which needs to be resolved before developing/using the web service based systems. One of the important issues is to consider the conflicting nature of QoS parameters during selection process. QoS parameters such as availability, reliability, etc., are better for high values. On the contrary, QoS parameters such as cost, response time, etc., are better for lower values [Yang and Zhou, 2015]. It is often valuable to determine desired service by considering the conflicting nature of QoS parameters.

Another issue associated with WSS is the performance of WSS system in terms of time for selection and quality of web service being selected. The performance further deteriorates as more and more functionally equivalent web services are made available. Because, more efforts are required for selection of WS, hence, overall performance of WSS system degrades. An important concern for WSS system is to perform selection of web services with due consideration to the end user requirements. Few WSS techniques such as Skyline approach select same list of services irrespective of the end user requirements. It causes the same set of services to be presented before different end users, independent of their QoS requirements. Therefore, in addition of ignoring the end users requirements, it also causes problem of imbalance of load on a specific set of services [Wang et al., 2016].

Further, it is found that the end users provide numerical values of weights to represent the QoS importance [Hwang et al., 2015, Kumar and Mastorakis, 2010, Helal and Gamble, 2014]. The numerical value of weight is difficult to comprehend for a naïve user. The web service selection process uses weight values of QoS parameters to decide the ranking of web services and ultimately

lead to web service selection. Thus, the unrealistic values of weights have profound effect on the quality of service being selected. Also, the use of techniques such as AHP, ANP [Ouadah et al., 2015], etc., for QoS weight evaluation, requires input from domain experts. It makes the WSS system domain dependent and inflexible.

The WSs are operating in highly dynamic environment. The quality offered by the WS depends on many external factors such as network condition, load on network and machine, capabilities of hardware on which WS is deployed, client environment, etc. Thus, depending on these factors the QoS of WS may change at run time. Similarly, the WSs are prone to failure due to some uncontrolled factors. If run time failure occurs, the system needs to recover from the failure by selecting a new WS. The process of reselection further degrades the overall performance of WSS system.

From these motivations, we have identified problems as listed below.

- Large number of functionally similar WSs are available to selection process.
- QoS parameters are conflicting in nature.
- Non-consideration of end user requirement of QoS during WSS.
- Specifying the weight values of QoS parameters by a user is difficult.
- The WSs are operating in a highly dynamic environment and may experience run time failure.

These issues have encouraged us to undertake more investigations in the area of web service to explore the possibilities of improvements. In view of the above mentioned open research issues, we were motivated to work towards the following objectives discussed in the Section 1.5.

1.5 OBJECTIVES OF STUDY

The present research aims at developing the efficient web service selection system with capability of considering the end users request of QoS during selection and also deal with the case of web service failure / unavailability. This is accomplished through the following objectives:

1. To perform the detailed study of QoS based service selection approaches and analyze the advantages and limitations.
2. To conduct an empirical study of classification based learning models with reference to their

role in web service selection and to analyze the effect of use of majority vote classifier on the quality of classification of web services and web service selection.

3. To develop an approach for classification based selection of web services.
4. To develop an approach for automated assignment of weights to various QoS parameters.
5. To conduct an empirical study of various web services clustering techniques based on QoS and study the effect of QoS parameters selection on clustering results.
6. To develop a clustering based selection method for web services depending on the end users' requirements of QoS.
7. To analyze the effect of web service replaceability on web service selection.
8. To develop a web service selection approach with consideration to web service replaceability to deal with the run time failure of selected web service.

1.6 CONTRIBUTIONS OF THE WORK

The important contributions of the work are following:

The contributions of this research are as follows. Firstly, we perform the study to identify various QoS parameters used to perform web service selection and analyze the influence on the performance of selection process and quality of service selected. The study focuses on analyzing the reported works related to QoS based web service selection by using various QoS parameters such as reliability, availability, response time, throughput, etc. [Alrifai et al., 2010]. The availability of labelled dataset, known as QWS [Masri and Mahmoud, 2007] motivated us to explore the available approaches to classify web services. The dataset also consist of non-labelled set of QoS evaluated on real world web service. As part of determining the learning technique that has shown significant performance for web service classification, from literature survey, we have identified eleven most widely used WS classification techniques. These techniques are compared on various performance parameters and top six WSs classification techniques are identified. In order to further analyze the performance of WS classification techniques, we have performed empirical study on various performance parameters. Six techniques investigated are Logistic Regression (LR), Multilayer Perceptron, Non-nested generalized exemplars (NNge), JRip, J48

decision tree, and Random Forest (RF). From among these, top three techniques LR, NNge, and J48 are compared and evaluated against majority vote based classification.

Secondly, we have developed a two layer web service selection approach based on the results of classification. The first layer performs the job of prefiltering the web services by using the majority vote based learning model. The model is trained using labelled dataset (training dataset) and unlabelled dataset are classified. On the same ground, the end user request of QoS is also classified and the closest matching set of web services is determined. In this manner, we are able to reduce the number of candidate web services to be considered by selection process. Also, the most promising set of web services are participating in the selection process to improve the overall quality of web service finally selected by the system. We have also explored the use of Maximizing Deviation Method (MDM) for evaluation of weights of QoS parameters [Yin et al., 2016]. The weight information is utilized by the selection process. To perform web service selection an improved PROMETHEE approach called as PROMETHEE Plus is proposed. PROMETHEE Plus method is evolved by performing three modifications in the basic PROMETHEE method to improve the quality of selection and improve the user satisfaction. The experiments are performed by using QWS dataset and dataset generated using standard dataset generator [Borzsony et al., 2001]. Three variations of the proposed method, MSS, HSS, and CSS are tested on four performance evaluation parameters.

Further, we observe that the availability of labelled QoS dataset for web service is very difficult. This motivated us to further explore the use of some technique to identify most eligible set of web services with reference to the end user request. Therefore, in next set of experiment, the use of QoS based clustering of web services is studied in detail. An empirical study of six most popular web services clustering techniques is done. The performance of clustering techniques is compared by using measures such as quality and stability of clusters formed. The clustering techniques considered for empirical study are - Unweighted Pair Group Method with Arithmetic Mean (based on Hierarchical clustering), K-Means, Diana, Clustering for the large data set (Clara), Partitioning around medoids (PAM), and Self Organizing Tree Algorithm (SOTA). Also, the effect of features selection using principal component analysis (PCA) on quality and stability of clusters formed is analyzed.

Next, based on the results of empirical study, an approach consisting of two layers - Prefilter

and selection layer is proposed for web service selection. The task of prefiltering of web services is performed using pruning and clustering. The filtered set of web services represents those web services which have capability to closely meet the end user requirements of QoS. The filtered set of web services is passed to determine Skyline web services [Borzsony et al., 2001]. The selection of web services is done by the proposed Skyline Plus. Skyline Plus uses MDM method along with simple additive weight method to find top-k selection. Also, the system is enriched to determine replaceable web services in parallel to the task of selection of web service. The proposed approach is compared with existing state-of-the-art on four evaluation parameters and found to be better.

We have further observed that the failure/unavailability of web service is more critical in composite web service [Jatoth et al., 2017]. To solve this problem, we have developed a web service selection system with due consideration to replaceability. First, we have reviewed the literature to elicit the suitability of PROMETHEE method for prioritizing the web services. Subsequently, the findings are formalized and build a web service selection system with due consideration to replaceability. Further, the effect of Input, Output, Precondition, and Effect (IOPE) [Hu et al., 2017] on determining web service replaceability is critically examined by performing various experiments. The use of determinant method for IOPE based matchmaking is explored and its effect on the matchmaking process is observed. The proposed approach is compared with existing similar approaches and found to be better in terms of efficiency and precision.

1.7 ORGANIZATION OF THESIS

The thesis has been divided into eight chapters.

Chapter 1 presents the brief description of web service selection. The chapter introduces the underlying fundamentals of the proposed work. Basics of web services, web service selection process, QoS based selection mechanisms, their effect on the selection process are discussed. The chapter also elaborates various objectives of this work and the organization of the chapters in the report.

Chapter 2 presents the literature review of the works related to QoS based web service selection. We have reviewed the process of web service selection using QoS parameters and presented the achievements and improvements made in WSS. A detailed study of the existing approaches for

QoS based web service selection is presented. A comprehensive review of the works related to QoS based selection of web services over last decade is discussed. The review of the existing works is grouped into four classes: classification, clustering, semantic, and replaceability. For each of the class, observations drawn are also presented. The review and summary in tabular form is also shown. The research gaps identified are reported in this chapter, which act as a base for the work presented in the upcoming chapters.

Chapter 3 presents an empirical study of some classification techniques to perform web service selection. In order to find best classifier for performing classification of web services, we conducted an empirical study on six learning models. The learning models considered for empirical study are Logistic Regression, Multilayer Perceptron, Non-nested generalized exemplars, JRip, J48 decision tree, and Random Forest. The empirical study is done by using several performance measurement parameters. The study is extended by performing a comparative study of majority vote classifier with top three best performing learning models. In the subsequent chapter, we have utilized the results of this empirical study to build a two layer service selection model and perform efficient web service selection.

Chapter 4 proposes two layer architecture based on classification for web service selection. Three different variations MSS, HSS, and CSS for the proposed two layer architecture are formed and discussed at length. The details of Prefilter layer, selection layer and modifications in the basic PROMETHEE method to improve the web service selection results are presented. To evaluate values of weight of QoS parameters, the use of maximizing deviation method (MDM) along with hybrid weight evaluation mechanism are also discussed in this chapter. The chapter also compares the existing state-of-the-art with three variations of the proposed approach on satisfaction score, query hardness, Euclidean distance, and time for WSS as performance measures.

Chapter 5 explores the use of unlabelled QoS dataset for filtering web services. The empirical study on six clustering techniques for web service clustering is performed. The clustering techniques considered for empirical study are – UPGMA (Hierarchical), K-Means, Diana, Clustering for the large data set (Clara), Partitioning around medoids (PAM), and Self Organizing Tree Algorithm (SOTA). The web services clustering techniques are compared on quality and stability of clusters formed. The best performing technique ensures the best clustering results.

Chapter 6 proposes an approach based on clustering for web service selection. The details of Prefilter layer to filter candidate web services and the end user request of QoS using pruning and clustering are presented. The use of Skyline technique at selection layer is explored. The chapter also provides the details of obtaining top-k services from the set of Skyline services and the use of Pearson correlation coefficient to determine replaceable web services. The performance evaluation of the proposed approach with existing state-of-the art is performed using four performance measures.

Chapter 7 deals with the problem of run time failure of web services. Here, initially we have introduced the need for replaceability and possible ways to determine replaceability. Next, we have proposed the PROMETHEE Plus method for evaluating replaceability. Its effect on the quality of selection is analysed in detail. Further, we have explored the use of IOPE based matchmaking for finding web service replaceability along with QoS based replaceability. The effect of use of IOPE for service replaceability based selection is observed. We also investigated the effect of using determinant method for IOPE based matchmaking.

Chapter 8 concludes the thesis by summarizing the contributions made in the presented works. Further, possible directions for future research work, open challenges and limitations of the thesis work are also presented in this chapter.

1.8 CONCLUSIONS

In this chapter, we have introduced important basic concepts related to the domain of web services. Basics of web services, Quality of Service, web service selection, and commonly used techniques for web service selection have been introduced. We have elaborated the issues with the present approaches for web service selection. The chapter also elaborate the objectives of the work along with the contributions of the thesis. The thesis organization has also been summarized in this chapter. Next chapter presents the literature review in detail and the important observations drawn from the review work.

CHAPTER 2

LITERATURE REVIEW

The World Wide Web (WWW) is expanding at very fast rate. WWW is a place where information is published and accessed with the help of wide range of available tools [Singh et al., 2010]. Due to the popularity and widespread use of WWW, the web service technology based solutions quickly got acceptations. Web service selection is a very popular problem and has been studied in the past. When user makes a request for performing some task over Internet, for example - purchase book request, many online book sellers offers web service to enable the customer to purchase the book by placing the order. Since multiple book sellers are offering the web service to purchase the book, user has to make a choice of selecting the web service provider. At this point of time user may select any book seller randomly, but later on after placing the order user may face problem such as delay in order execution, receiving of item in bad condition due to poor handling etc. The chances of successful order execution by ensuring the on time delivery and other non-functional requirement can be increase by observing the ratings of service providers by past service consumers etc. This example shows that not only functional requirements but non-functional requirements for services are equally important from the view point of service consumer. Similarly, service providers also use QoS parameters to increase the web service consumption and thereby increasing the market business. Moreover, QoS based service selection facilitates the automation of web service selection task without the intervention of humans. Therefore, QoS based Web Service selection is an important topic for many researchers.

In the previous chapter, we have discussed fundamentals of web service, QoS, and selection of WSs. Further, we have reviewed the existing techniques for selection of WSs. The motivation behind the work presented in this thesis and important contributions are also discussed. In this chapter, we first explore various dimensions of web service selection process and analyze their influence on selection process. Second, the QoS model is described in detail. Third, we analyze various reported works on the web service selection based on QoS. These works are classified into three categories: classification based service selection, clustering based service selection, and replaceability based service selection. Replaceability based WSS

work is further sub-classified as QoS based replaceability & selection and semantics based service replaceability & selection. In each category of discussion, important features of the work, various characteristics, gaps identified, and comparison with the work presented in this thesis are discussed.

2.1 INTRODUCTION

Figure 2.1 gives an overview of the web service architecture and the place of web service selection process [Newcomer, 2002]. The architecture diagram shows that in order to use the web service, nine steps (represented as number on the arrow) are to be followed. The service provider implements the WS and register into the WS registry (step 1,2). The WS consumer fetches the functionally similar WSs by requesting the registry (also termed as discovery process) (step 3). The registry returns the list of functionally similar WSs back to the requester (step 4). From among the list of functionally similar WSs, the process of selection is to be followed. Usually, the process of selection of WS involves prioritizing the WSs on some criteria such as QoS followed by selection of the best WS (step 5). Once the desired WS is selected, the WSDL of the WS is fetched (step 6, 7). The details required for using the WS are obtained from the WSDL and the WS is requested to get the desired business functionality (step 8, 9). It can be observed from the Figure 2.1, that the task of selection plays a very vital role in describing the performance of the WS based software.

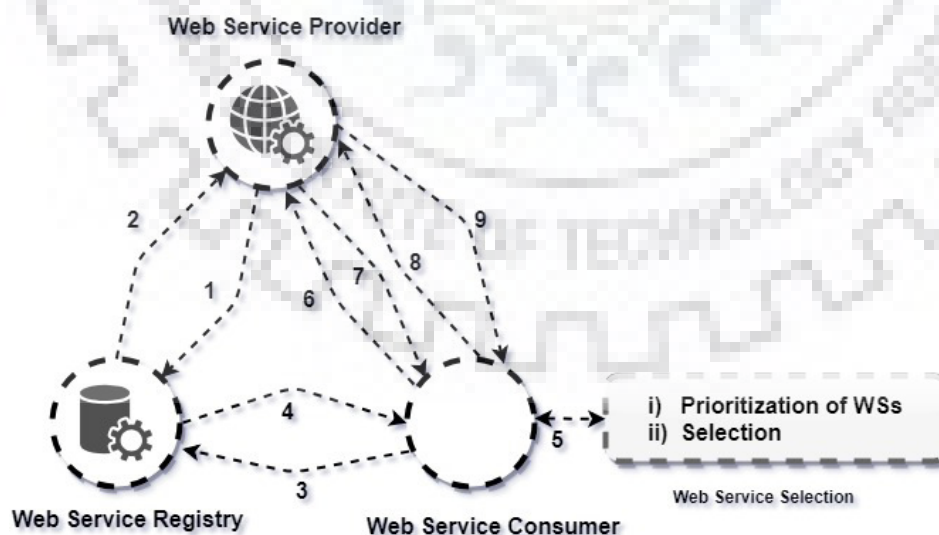
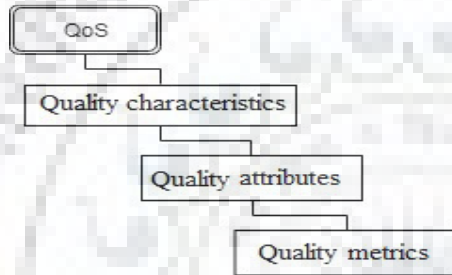


Figure 2.1: Architecture of web service

2.1.1 Quality of Service model for web service

As per the definition from Software Quality Standards, the quality models are helpful in specifying requirements, establishing measures and conducting quality evaluations [Oriol et al., 2014]. It can also be defined as a hierarchical decomposed set of quality attributes of software [Oscooei and Daud, 2014, Mishra et al., 2013]. Various general purpose Software Quality models are proposed such as ISO/IEC series of quality standards, specially in 9126 and its successor 25010, SQuaRE model [Oriol et al., 2014] etc. These models for software systems and may not be completely suitable for web service domain. Therefore, various research proposals [Kritikos and Plexousakis, 2009, Oriol et al., 2014, Oscooei and Daud, 2014, Antony et al., 2009, Ran, 2003] are available for QoS model. In general the QoS model for web service is represented as a hierarchy [Oriol et al., 2014] as follows:



QoS represents characteristics of web service quality in totality [Oriol et al., 2014].

Quality Characteristics represents groups of quality attributes [Oriol et al., 2014].

Quality Attributes are fine grained concepts which affects quality of a web service [Oscooei and Daud, 2014].

Quality metrics are quantitative measurements of quality attributes [Oscooei and Daud, 2014].

They are the mathematical way of defining quality values of Quality Attributes.

Example of few QoS Attributes and QoS metric are shown in Table 2.1.

Table 2.1: Example of QoS attributes and QoS metric [Oriol et al., 2014, Oscooei and Daud, 2014, Ran, 2003, Singh et al., 2016]

QoS Attribute	QoS Metric
Response time	Execution time + Waiting Time + Transmission Time
Reliability	1-failure rate (where failure rate = execution time / MTBF)
Availability	Uptime / (Uptime + Downtime) {or MTTF / MTTF + MTBF}
Reputation	$Rep_{w_{i,j}} = \frac{\sum_{i=1}^n w_i}{n}$

From the study of the research work [Kritikos and Plexousakis, 2009, Oriol et al., 2014, Oskooei and Daud, 2014, Antony et al., 2009, Ran, 2003], QoS model for web services can be defined by considering following six categories (Quality Characteristics) along with QoS attributes and QoS metric:

- (i) *Business Specific QoS* [Oskooei and Daud, 2014]: The foremost requirement of any web service is to fulfil the business requirements of the service consumer. Four business related QoS attributes are - Execution price, Transaction, Penalty rate, Compensation rate.
- (ii) *Performance related QoS* [Kritikos and Plexousakis, 2009]: This category of QoS refers to the performance of the web service based system under the stated condition. This refers to how efficiently users request can be served. Four performance related QoS attributes are :
 - Response time (Execution time + Waiting Time + Transmission Time),
 - Throughput (Number of Completed requests for service / Unit of time),
 - Reliability (1-failure rate; where failure rate = No. of failure / Time Period),
 - Availability(Uptime / (Uptime + Downtime))
- (iii) *Requesters Response specific QoS & Trust* [Oskooei and Daud, 2014, Antony et al., 2009]: The response specific QoS property is measured by feedback of service consumer. Reputation is one such QoS property. It is a measure of trustworthiness of web service. It represents the popularity of web service which depends on advertisement, financial and political interest and can be determined from the feedback of users and usage count of web service.

Trust is feature that one entity rely on the other entity to execute some set of operations. The trust can be measured by trust graph model.
- (iv) *Security* [Kritikos and Plexousakis, 2009]: Web services are resources available over Internet provided to be used by any other piece of software. Security Characteristics ensures the security attributes supported by web service. Various QoS attributes for security are - authentication, confidentiality, integrity, non-repudiation, authorization, traceability.
- (v) *Dependability* [Kritikos and Plexousakis, 2009, Ran, 2003] is the measure of capability of a computing system to deliver service that can justifiably is trusted. Attributes such as Accessibility, scalability, and Capacity can be used to specify dependability characteristic.
- (vi) *Failure Semantics* [Kritikos and Plexousakis, 2009, Ran, 2003] is the measure of specifications provided for the type of fault which may occur and how web service react to them. Failure masking, Operation semantics and Exception handling attributes may be used to represent Failure Semantics of web service.

2.2 CLASSIFICATION BASED WEB SERVICE SELECTION

The process of classification refers to the process of identifying class/category of a new object based on the training set of object whose class/category is known. Using the objects with known class, a learning model is created. Later on, the learning model is used to identify the class of the new object. The classification model for WSs can be created using labelled set of QoS dataset. The classification when applied to the set of unlabelled WSs, determines the similarity among the WSs.

In this section, most recent works on classification of WSs and QoS based selection of WSs are discussed. The problem of WS classification is studied in the past and is discussed in detail. Initially, we have performed an extensive study to find the relevant works on web service classification. Various learning models are used for web service classification. A comparative study of recent works on web service classification is presented in Table 2.2. We observed from the Table 2.2 that, majority of available works uses WSDL, OWL-S, and QoS information for classification. WSDL and OWL-S are used to describe functional features of web services and are mainly used during service discovery [Purohit et al., 2015, Kumar and Mishra, 2008b]. However, the QoS information is useful to perform the task of selection of web services. From among the available QoS parameters, few of the parameters are redundant and/or irrelevant in the presence of the strongly correlated QoS parameter(s). Therefore, the set of relevant QoS parameters should be identified. It can be performed by segregating these parameters using feature selection technique. Also, the feature selection process helps in reducing the training time, simplification of model, and handling the problem of data over fitting.

A very few of the available works uses feature selection technique, however, they lack in terms of stability and robustness of results. In addition, the existing work compares the performance of various classifiers using only efficiency parameter. Which sometime leads to unexpected classification. This is primarily because other performance parameters are neglected during evaluation of classifiers. Moreover, the classification model is build using single classifier. Nonetheless, the group decision for classification is observed to be more accurate than the decision from single classifier [Kittler et al., 1998]. The majority vote classifier can be useful to improve the classification decision by combining the output of multiple classifiers. Therefore, in this work we have studied web service classification based on QoS parameters along with feature selection. The empirical study is done by using several performance measurement parameters. The study is extended by performing a comparative study of majority vote classifier with top three best performing learning models.

Table 2.2: Recent work on web service classification

Work	Criteria	Learning models used	Dataset used	Tools Used	F.S.
[Patro and Patra, 2015]	QoS	FNN, FRNN,FRONN	QWS	WEKA	Y
[Mohanty et al., 2010]	QoS	PNN, BPNN, J48, GMDH , TreeNet, CART, SVM	QWS	WEKA, PM	Y
[Wang et al., 2010]	OWL-S	SVM	OWLS-TC4	WordNet	Y
[Yang and Zhou, 2015]	WSDL	SVM, NB, C4.5, BPNN	OWLS-TC4	WEKA	N
[Liu et al., 2016a]	WSDL	SVM, LDA-SVM	WS-DREAM	WEKA	N
[Mohanty et al., 2012]	QoS	NB, Markov Blanket, Tabu Search	QWS V1	WEKA	N
[Jie et al., 2012]	WSDL	NB, SVM, REPTree, AdaBoost	Collected from Internet	WEKA	N
[Vaadaala et al., 2013]	QoS	J48 decision tree	QWS	WEKA	N
[Mustafa, and Kumaraswamy, 2014]	QoS	FURIA, kNN, RIDOR, SVM	QWS	N.M.	N
[Mustafa, and Kumaraswamy, 2015]	QoS	MLP-LM, MLP-BPP, MLP- TS	QWS	N. M.	N
[Qamar et al., 2015]	WSDL	NB, DT, SVM	WSExpress	WEKA	Y
[Own and Yahyaoui, 2015]	QWS	BPNN, J48, CART, PNN, GMDH, TreeNet, SVM, RS	QWS V1	RSES	Y

Recent works on web service classification are summarized with details of classification model, dataset used, tools used, criteria for classification and whether feature selection (F.S.) is employed or not. NB=Naive Bayes, LDA=Linear Discriminant Analysis, SVM=Support Vector Machine, DT=Decision Tree, BPNN=Back propagation Neural Network, PNN=Probabilistic Neural Network, GMDH=Group Method for decision handling, RS=Rough Set, kNN=k nearest neighbor, PM = Predictive modeller, N.M. = Not Mentioned.

QoS is being used as an important parameter to distinguish among functionally similar WSs. It is increasingly difficult and time consuming task to obtain best WS due to conflicting QoS parameters. An existing QoS based WSS approach treats QoS parameters as discrete random variables with probability mass functions [Hwang et al., 2015]. The service selection is performed by computing the utility function based on the probability of satisfying QoS constraints. A similar approach for WSS using utility function for sorting and selection of web services is available [Alrifai et al., 2011]. In this approach, the distance between the service candidate and the upper bound is obtained. Based on distance values, candidate WSs are sorted. Nevertheless, the approach does not consider the conflicting nature of QoS parameters during evaluation of the utility function, which we believe are very crucial for producing an accurate ranking of services. Web Service Relevancy Function (WsRF) can also be used for web service ranking and selection [Masri and Mahmoud, 2007]. Higher value of relevancy function indicates better service. A variety of solutions for QoS based WSS problem are available such as – WSS based on constraint rules using MIP [Saleem et al., 2015], similarity rules in CBR [Renzis et al., 2016], predictions based on collaborative filtering [Zheng et al., 2011], etc. However, these solutions do not consider the conflicting nature and interdependency among QoS parameters.

WSs can be grouped together by performing classification task. The available works on classification using WSDL and OWL-S performs classification at the time of service discovery. As it is performed at the time of service discovery, so, they are increasing the efficiency of discovery process, generating a set of functionally similar WSs. To further reduce the set of functionally similar WSs, our work (discussed in detail in Chapter 3 and 4) is performing classification of WSs based on associated QoS information. Thus, classification is applied in prefiltering step to reduce domain of search. A similar approach based on bundling framework to reduce search domain is presented in [Alrifai et al., 2011, R.-Zapata et al., 2011, R.-Zapata et al., 2015]. The presented bundling framework uses complementarity indexes and user context to create service bundles. The customer and service providers interact with each other to identify the customers' need. Upon clearly identifying the needs of the end user, service bundles are generated dynamically. The service clusters are created based on the services offering similar functional consequences (FCs) [Alrifai et al., 2011, R.-Zapata et al., 2011, R.-Zapata et al., 2015]. This method achieves increased user satisfaction, social welfare and customer surplus. This method can be useful for efficient selection of WSs for composite service.

An effective method for comparison of alternatives based on the conflicting QoS attributes is PROMETHEE-II [Brans and Vincke, 1985]. It performs total ordering of alternatives. The PROMETHEE method for WSS is introduced by [Seo et al., 2005]. The QoS parameters are mapped to the actions and service providers are the alternatives. To define the criteria, appropriate choice of preference function is to be made. For qualitative criteria (performance, availability), usual/Gaussian type preference functions are suitable. Similarly, for quantitative criteria (cost, price, and power), V-shape/level/linear type preference functions are suitable [Brans and Vincke, 1985]. The preference index of services needs to be evaluated by pair wise comparison of alternatives. The net outranking flow for each candidate WS is calculated, which results in a ranking of WSs [Karim et al., 2011]. The PROMETHEE method does not include any predefined weight calculation scheme. The weight calculation is kept flexible. The modified Simos procedure for calculation of QoS weights is one such method to specify the QoS weight externally [Herssens et al., 2008].

The weight of the QoS parameter is one of the factors which guide the performance as well as output of WSS method, hence it cannot be ignored. In the literature, it is assumed that the end user has a clear idea about QoS preferences and is able to assign a scalar value to represent the QoS preferences [Alrifai et al., 2011, Masri and Mahmoud, 2007a, Rhimi et al., 2015,

Saleem et al., 2015, Lim et al., 2012, Stephen and Yin, 2011]. On the contrary, this is rather a big challenge for the end user to properly judge the QoS weight values and for WSS module to provide the best selection by incorporating the user judgment. In the available works, methods such as AHP [Herssens et al., 2008, Ouadah et al., 2015] and ANP [Karim et al., 2011] are used to represent the preference of the end user. However, the preference is determined by taking input from the domain experts. A more useful way to determine the linguistic weights of QoS parameters is based on a resolution process in which a group of participants' preferences is considered [Wang, 2009]. A more natural way to give preference to the user desired QoS parameters is to include them in the calculation. The skyline based approach for WSS does not require QoS preference from the end user explicitly [Yu and Bouguettaya, 2012].

2.3 CLUSTERING BASED WEB SERVICE SELECTION

The task of classification of WSs is very effective and efficient. However, the base requirement for generating QoS based WS classification model is the availability of labelled set of QoS dataset. The labelling of QoS dataset is very difficult and in real world scenario mostly the QoS dataset without any label is available. Thus, the grouping of WSs in such scenario is difficult. Clustering is an unsupervised learning technique designed to explore the natural structure of the data objects [Gajjar et al., 2017]. In other words, clustering is a mechanism to create groups of objects (WSs in our case) based on certain criteria. The relationship among the objects based on the object properties can be established. For the case of WSs, QoS information can be used to create WSs grouping. The clusters of WSs are formed by following the process of clustering based on QoS parameters. The clustering of WSs is a useful mechanism to explore the similarity among WSs.

In this section, most recent works related to clustering of WSs are discussed. It is revealed from the study that the clustering on WSs can be applied to boost up the performance of discovery as well as selection process. It is learnt that in the past most of the works have applied clustering on WSs to enhance the performance of discovery process. However, in our work we have applied clustering to improve the efficiency of selection process. The process of clustering is followed by using some criteria to identify similarity among objects. The summary of few important works on WS clustering with reference to criteria used for clustering are summarized in Figure 2.2.

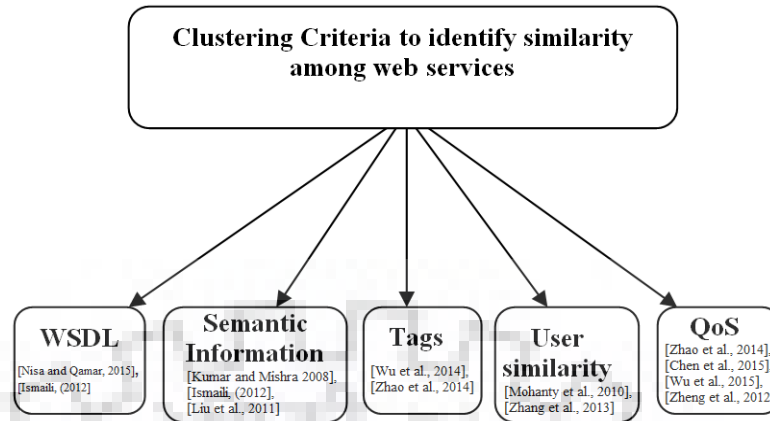


Figure 2.2: Categorization of clustering criteria used for clustering WSs.

2.3.1 Web Service Description Language (WSDL)

Service discovery phase identifies services with similar functionality. Employing service clustering during discovery phase enormously reduces service discovery time. The functional description of WSs based on three major components is available in WSDL file [Nisa and Qamar, 2015]. In place of choosing multiple features from WSDL file, relevant features based on observations are selected and cumulative score is used to group services. The maximum entropy algorithm is used for grouping services. Services falling in the same group are similar on the selected WSDL features. In [Ismail, 2012], a similar hybrid approach to cluster WSs using semantic and syntactic information is presented. The Growing Hierarchical Self-Organizing Map (GHSOM) is applied to generate self organizing maps of services. This reduces number of services to a reasonable number. The syntactic similarity using WSDL information of WSs is evaluated. With associated semantic information in OWL-S, semantic similarity is also calculated. The Euclidean distance between the query vector and cluster centre is obtained to evaluate the semantic similarity. Both the similarity scores are combined to formulate cumulative score. The cumulative score is utilized to search the desired WSs and the closest matching list of WSs is selected.

Another simple idea to perform web services clustering based on the WSDL features, i.e., Message, Port, Type, Service Name and Content [Yu, and Gen, 2010, Wu et al., 2014]. Based on the similarity observations, clustering of WSs is obtained. However, the clustering only based on WSDL level features results in low recall of web service discovery mechanism [Wu et al., 2014]. Therefore, clustering based on WSDL features is combined with tag based clustering to improve overall performance of WS clustering and recall, precision of WSS system [Wu et

al., 2014]. Tag based information improves the functional identification of WSs. The WSDL based clustering can also be combined with OWL-S based clustering [Yu, and Gen, 2010]. In this approach, the syntactic and semantic similarity among WSs is evaluated and based on the hybrid score, clusters are created. An approach to cluster WSs using ontology is presented in [Liu et al., 2011]. Each WS have associated ontology which carries information such as input, output, precondition, effect and QoS. The matching score of WSs is obtained on the ontology information and accordingly service clusters are generated. The presented approach is useful for WSs clustering, however, the process of matching and score generation is not elaborated in detail.

2.3.2 Semantic Information

The WSDL offers an easy way to describe any WS on different parameters. However, WSDL based mechanism is very limited to fully explore the features of WS. Ontology provides a systematic mechanism to fully describe and determine the functionality offered by the WS [Michalewicz and Vadis, 2012]. Service ontology is useful to assist search for needed WS over Web [Masri and Mahmoud, 2009, Biot and Academy, 1977]. From the ontology, information such as interface similarity, service name, capability, QoS similarity, etc., can be used to form clusters of WSs. The work presented in [Michalewicz and Vadis, 2012] uses environment ontology to cluster WSs. Further, it is observed that the cognitive parameters are effective in service selection and should not be ignored [Kumar and Mishra 2008a]. A new model based on various cognitive parameters such as intension, trustworthiness, desire, reputation, capability, etc. is presented in [Kumar and Mishra 2008a]. Considering these parameters during service selection produces more reliable selection results. Using cognitive parameters, ontology is generated for each WS. The cognitive information is used by service selection process to select WSs for executing composite task.

2.3.3 Tags

Tag is a phrase which contains small descriptions to describe the property of object and make it more meaningful to the outside world. Capabilities of a WS can be described using tags with strong correlation [Zhao et al., 2014b]. Tagging of WS enables external world to understand the features along with functional and non-functional capabilities of WS. It leads to correct identification of WSs as desired by the service consumer. Tag information can be obtained by scanning WSDL file, from Internet [Zhao et al., 2014b], or from the user [Wu et al., 2014]. The WS page is accessed using URL of WS stored in the WSDL file of the service. To collect and extract tag information, the vector space model (VSM) based structure analysis is used in [Zhao

et al., 2014b]. Once the content extraction using VSM model is completed, the process of mining is started. The mining process combines the mining of service related tags with traditional WSDL based processing to boost the service characteristics presentation. In [Wu et al., 2014], a similar approach is presented for WS clustering by utilizing tag based information. The hybrid tag recommendation strategy *WSTRec* recommends tag by calculating semantic relevance. The similarity among services is evaluated using two similarity scores - tag level similarity and WSDL level similarity. The similarity values are utilized to cluster WSs.

2.3.4 User similarity

The performance of service discovery mechanism can be improved by performing service discovery on the basis of user similarity. The service recommendation system using user based clustering is another powerful idea to identify WSs of interest [Zhang et al., 2013]. In this approach, the past quality ratings of the service and QoS requirements of the end user forms the basis of clustering. The similarity among users is identified on the basis of interest of end user and service usage pattern. User similarity coefficient is calculated and user similarity is accessed.

2.3.5 Quality of Service

The QoS parameters based clustering is a handy mechanism to identify similarity among WSs on the basis of non-functional characteristics [Purohit and Kumar, 2016b] of services. QoS clustering approaches are then subsequently used to perform selection. A useful approach to cluster WSs is available in [Zheng et al., 2012a]. In this approach, the access situation of the user forms the basis of clustering. This work uses KMeans clustering. Once the clusters are formed, the services are matched against the end user request. Matching of services is done in two steps. At first, the service similarity is determined using basic service description. If the matching score is at par with the minimum threshold, then the QoS matching of the service is performed. By using this two step mechanism, the improvement in the precision of service discovery is observed. On the negative side, a large number of comparisons are required to discover the desired services. The clustering of WSs can be useful to identify substitutable WSs. The substitute WS is needed because the WS execution environment is very dynamic and sometime leads to runtime failure of WSs. In order to handle this situation of failure, available closely matching services can be helpful [Ismaili, 2012]. To implement this idea, clustering of WSs based on QoS is done and extended to the case of service composition. The clustering approach is useful to perform search and selection operations in the group. For the composite

WS, clusters of concrete WSs are formed for each of the abstract WSs. The Optics algorithm is employed to cluster WSs which allows clusters of any shapes. The task of abstract WSs is performed by concrete WSs. If any of the concrete service experience run time failure, other concrete WSs from the same cluster is chosen as a replacement. The choice of alternative WS is made based on the utility value of the service. Both these techniques improves the service selection efficiency, however, does not consider fuzzy based clustering. Collaborative filtering (CF) is an effective technique for QoS prediction. The prediction is performed on historical QoS data furnished by similar services and users. But, the CF suffers from the sample matrix data sparsity problem. Using QoS based clustering prior to applying CF improves the overall prediction accuracy [Chen et al., 2015]. The CF based QoS prediction technique does not consider the detection of unreliable data from untrustworthy users. The unreliable data significantly affects the performance of prediction system. To deal with unreliable QoS data, CAP approach is applied [Wu et al., 2015]. The K-Means based clustering of unreliable services is done using untrustworthy index calculation. In the next phase, the users are clustered based on their index.

Clustering of WSs using QoS criteria is used in the past for QoS prediction [Zhang et al., 2013]. In user request based clustering, the services able to serve the user requests are identified and logged [Zheng et al., 2012]. For a fresh user request, the similar user requests (using similarity coefficients) and the WSs used to serve these requests are identified from the log. Based on the identified WSs, the appropriate WSs for fresh user request are recommended. This improves the quality of service discovery process.

For composite task, the service selection using clustering concept is useful to improve the performance of selection mechanism [Xia et al., 2011, Tripathy et al., 2014]. The clustering of web services using K-means [Tripathy et al., 2014, Tewari et al., 2012b] and Optics algorithm [Xia et al., 2011] for selection of WS for composite task is employed. In [Xia et al., 2011], the service clusters for each concrete service is obtained and based on best utility value, top most services are selected. This approach will lead to saving in composition time, however, the end user satisfaction is not achieved to its best level. In [Tripathy et al., 2014], the service cluster graph algorithm based on bellman ford's algorithm is employed to select a best service composition. Due to clustering, the process of identification of concrete WS takes less time. However, the composition process will consume significant time due to the large number of composition possibilities [Xiang et al., 2015]. The clustering to reduce the search domain is a useful technique.

When user is not in position to specify expected numerical values of QoS parameters, fuzzy values are useful to represent QoS expectations of the end user. In [Mobedpour and Ding, 2013], the services are clustered into three clusters representing fuzzy values – poor, medium, and good. The symbolic clustering mechanism is used to perform task of clustering of WSs. The clustering is done based on the Euclidean distance of each candidate WS. Two levels of selection mechanism is adopted to obtain the closest matching WS.

A comparative study of recent works on web services clustering for WS discovery/selection is shown in Table 2.3. It can be observed from Table 2.3 and the available literature that the existing approaches performs clustering using functional features of web services such as WSDL, OWL-S, Tags, User preference based, etc. It is also noted that the existing web service clustering approaches are used during service discovery. Further, the existing state-of-the-art is not using feature selection to clustering WSs.

Table 2.3 Recent works on web service clustering

Work	Clustering Criteria	Clustering Technique used	F.S.	Applied for Service Discovery (D) or Web Service Selection (S)	WSS for atomic (A) or composite (C) service
[Zheng et al., 2012]	USC	K-Means	NU	D	N.M.
[Yu, and Gen, 2010]	OWL-S	SOM	NU	D	N.M.
[Wu et al., 2014]	WSDL	K-Means	NU	D	N.M.
[Liu et al., 2011]	Ontology	NM	NU	D	N.M.
[Tripathy et al., 2014]	Customer	K-Means	NU	D	C
[Zhang et al., 2013]	Customer usage	K-Means	NU	S	A
[Xia et al., 2011]	QoS	Optics	NU	S	C
[Mobedpour and Ding, 2013]	Fuzzy values	SCLUST	NU	S	A

Recent works on clustering of web services are summarized with details of criteria used for clustering, clustering technique applied, dataset used for clustering, clustering performed to select WSs for atomic or composite task and whether feature selection (F.S.) is employed or not. USC=User similarity coefficient, NU=Not Used, N.M. = Not Mentioned.

2.4 REPLACEABILITY BASED WEB SERVICE SELECTION

The WS replaceability is the ability of a WS to replace another WS without affecting the underlying expected functionality and the offered service quality. The functional and non-functional characteristics of the WS form the basis of two criteria for obtaining service

replaceability. The functional similarity among WSs is determined on the basis of IOPE matching. Similarly, the similarity on the basis of non-functional characteristics is obtained by using QoS matching among WSs. The similarity among WSs can be used to determine service replaceability. The replaceability of WSs is useful to deal with the case of run time failure of WS. The failed WS can be replaced with the closest matching WS. The problem of service failure is fundamentally linked to the service selection problem due to its effect on the efficiency of the process of selection. Thus, most recent works related to service substitution/ replacement are discussed in detail.

2.4.1 QoS based Replaceability and Web Service Selection

Web service composition is used frequently to offer value added services to the end user. The criteria used for composition include QoS parameters [Helal and Gamble, 2014, Xia et al., 2011, Yin et al., 2009], Input/ output [Yin et al., 2009, Ernst et al., 2006], owl-s [Liu et al., 2011, Wijesiriwardana et al., 2012], ontology [Yan et al., 2015, Bhuvanewari and Karpagam, 2012, Kumar, 2012], etc. The QoS based selection of services for composite task is a popular method [Helal and Gamble, 2014, Purohit et al., 2015]. The task of composition run on the top of predetermined composition plan. Petrinet model is useful to design composition plan [Singh and Rajput, 2018, Singh et al., 2014]. The Petrinet act as best tool for dependency representation. The dependency among WSs involved in composition can also be modelled using Petrinet. The QoS based replaceability using nearest neighbour is effective to handle the run time failure of WSs. The GA is applied for selection of WSs. In each iteration of GA, the fitness of population is evaluated by using penalty based fitness function. The replaceability of each plan is calculated and the plan with higher replaceability is selected for further consideration. However, the approach is time-inefficient due to nearest neighbour search in every iteration of GA. Further, the approach do not consider the service similarity during replaceability evaluation.

To ensure that the end-to-end QoS requirements of the composite WS are satisfied, the GA based solution is preferred. In [Purohit et al., 2015], the GA is applied on the set of candidate WSs to select most optimal composition. In each iteration of GA, the population is evaluated using a fitness function. A closely related GA based approach for WSS is proposed in [Yan et al., 2015]. In this approach, the service selection for composition problem is modelled as Multi-Constrained Optimal Path (MCOP) problem. The business rules between service providers are considered along with QoS parameters for carrying out the task of composition. Business rules with dependency, conflict and positive inference category are used to achieve service selection.

The proposed solution is optimal, however, business rules are difficult to establish in generalized form. The basic GA used for service selection has performed reasonably well. The global optimization is achieved using the randomization step. The population diversity handling can be achieved in more effective way by using simulated annealing and harmonic search to improve the mutation operation [Yilmaz and Karagoz, 2014, Parisi et al., 1999]. The above available methods of service selection for composition follow bottom-up approach by searching possible combinations of concrete WSs to determine optimal composition. A new way of achieving composition is to break the global QoS constraints into local constraints [Liu et al., 2016b]. The number of candidate WSs are reduced by selecting top-k composition schemes and the candidate web services are obtained from the component services of the selected schemes. The main emphasis of available works as discussed above is on determining services involved in optimized composition. The run-time failure of any of the component WSs may lead to serious damage due to unavailability of services (and application).

The application uptime (or availability) can be improved by determining a substitutable WS [Bhuvanewari and Karpagam, 2012]. One of the ways to determine substitutability of a web service is to compare the inputs of one WS with the input of other [Yin et al., 2009, Ernst et al., 2006]. Similarly, outputs of both the services are compared. If the maximum matching for input/output comparison is resulted, than one service can substitute other. In addition to input/output based service substitutability, QoS parameters based comparison is done to match the end user requirements of QoS [Yin et al., 2009]. However, this approach [Yin et al., 2009] for QoS based matching and substitution is based on directed acyclic graph and is slow and less useful for real time applications based on web services. A similar approach to obtain dependency among services is presented in [Wijesiriwardana et al., 2012]. This approach considers the pre and post condition to evaluate dependency among services. To rank the services, along with dependency, the correlation among the services is also taken into account. The failed service is replaced by the service with the next highest rank. Another work in [Kumar, 2012, Kumar and Mishra, 2008c] has explored the use of ontology to check the compatibility of WS. If any component WS fails during execution, the information in the ontology such as *Task*, *Domain*, *QoS*, etc., are used to determine service compatibility [Bhuvanewari and Karpagam, 2012]. The process of search for compatible WS is repeated till a compatible WS is obtained. The method is useful to get a matching WS, however, it lacks in checking all possibilities of compatible WS and leads to locally optimized replaceable WS.

Clustering of web services is another useful criteria to identify similar structures among WSs [Xia et al., 2011, Liu et al., 2011, Tewari et al., 2012b]. In [Xia et al., 2011], the clustering of candidate WSs is done based on the QoS properties. The web services part of same clusters shares the QoS properties closely. On the contrary, the services in the different clusters differ on QoS parameters. Based on this property of clustering, it can be observed that the services in the same clusters can act as replacement for other services in the same cluster [Xia et al., 2011]. However, this approach fails to identify close or exact replacement of services, if available. This is due to the fact that the identification of replacement service is not done in close vicinity of failed service.

The composite web service is more prone to failure. This is due to the fact that number of services is involved in composition. The successful execution of composite WS is dependent on successful execution of services involved in composition. If any of the service involved in composition fails, then whole composition is failed [Gupta and Bhanodia, 2012]. Thus, it is important to evaluate substitutable WS in advance. The recovery mechanism to overcome the failure situation is based on subset replacement. This model recovers from failure dynamically, but has low time efficiency.

In the case of composite WS, the use of genetic algorithm (GA) is suitable to perform selection of WSs [Liu et al., 2016b, Helal and Gamble, 2014]. GA based WSS is one of the highly explored work and is performing better than other approaches [Purohit and Kumar, 2018c, Jatoth et al., 2017, Garriga et al., 2015, Yilmaz and Karagoz, 2014, Fister et al., 2013, Fethallah, 2012]. In the work presented in Chapter 7, we have used replaceability of WS as the basis of selection of WSs when selection of WSs is done using genetic algorithm (GA). Therefore, the GA based selection of WSs is explored in detail to observe the possible places of improvement in GA to enhance the WSS quality.

The task of WSS guided with replaceability factor is more useful to realize the real time system implemented using WS technology. In such critical systems, during the occurrence of run time failure of any component service, the replaceable WS is immediately made available. One such approach for selection of WSs based on replaceability factor is presented in [Helal and Gamble, 2014]. The genetic algorithm (GA) is used to perform WSS. During each iteration of GA, the fitness of each composition plan is evaluated based on replaceability factor. The replaceability factor is obtained by nearest neighbour in each iteration of GA for each composition plan in the population. Using this approach, the service with highest replaceability factor is preferred for selection. Along with the selection of WS, replaceable WSs are

determined which can be useful at run time. However, the approach is slow in convergence. Also, the method does not take into cognizance the conflicting nature of QoS parameters to determine replaceable WSs.

GA serves as an efficient solution to the problem of WSS. The problem of WSS can be modelled as a single objective optimization problem (SOOP) [Michalewicz and Vadis, 2012, Yu, and Gen, 2010, Fethallah, 2012, Sharifara et al., 2014, Bandyopadhyay and Saha, 2013] as well as a multi-objective optimization problem (MOOP) [Sharifara et al., 2014, Wang and Hou, 2008, Yao and Chen, 2009, Zhang and Ren, 2011, Sasikaladevi and Arockiam, 2012, Savic, D. A., 2002]. In single-objective optimization, the single objective function lumps all the objectives together, and a good solution (maximum/minimum) is to be obtained. On the other hand, in multi-objective optimization, the multiple objectives are to be achieved with due consideration to tradeoff among objectives. The compromised solution is obtained by combining different objectives together [Fister et al., 2013, Savic, D. A., 2002]. For the problem of selection of WS modelled as a single-objective optimization problem, solution such as calculus-based techniques, enumerative techniques, random techniques, etc., are suitable [Bandyopadhyay and Saha, 2013]. For multi-objective optimization modelling of a WSS problem, the multi-objective versions of the meta-heuristic techniques are developed [Bandyopadhyay and Saha, 2013]. A different way to achieve WSS is by decomposing end-to-end constraints into local constraints and then using the local optimization technique to find an optimal concrete WS for each abstract WS [Liu et al., 2016b, Liu et al., 2013, Liu et al., 2015].

Genetic Algorithms are the family of a computational model grounded in the natural evolution process of selection and genetics [Singh et al., 2012, Ganesh et al., 2005]. The basic steps of GA in solving the WSS problem are shown as flowchart in Figure 2.3 [Zhi-peng et al., 2009, Yilmaz and Karagoz, 2014, Jaeger and Muhl, 2007, Ma and Zhang, 2008, Kher et al., 2009]. Six major steps of GA for WSS identified from Figure 2.3 are – encoding scheme (ES), population initialization (IP), fitness function based evaluation (FF), selection operation (SO), crossover operation (CO), and mutation operation (MO). Simple GA start by randomly selecting MAX_POPULATION of valid composition plans represented using a data structure called chromosomes. Each gene on the chromosome represents a Concrete WS. For each composition plan (chromosome) in the population, fitness is calculated. Based on the goodness of composition plan, composition plans with higher fitness values are given preferences for reproduction in the subsequent iteration as compared to composition plan with weaker fitness value. The process is repeated till either MAX_ITERATION (MI) or termination condition is

achieved. After GA finishes the execution, the optimized composition plan is produced (C_{best}) as an output.

The use of GA to resolve the problem of WSS is introduced by [Canfora et al., 2005]. In this work, the proposed WSS approach is in line with the algorithm steps of flowchart shown in the Figure 2.3. GA uses different parameters such as mutation rate, crossover rate, maximum population in each iteration, the choice of fitness function, etc. A fine tuning of these parameters in the context of application have a significant effect on the algorithm performance [Zhang et al., 2006b, Wang et al., 2007]. The GA is one of the well known technique for finding a solution to the optimization problem. User specified constraints are handled distinctly by GA. There are two major ways to handle constraints by GA [Ai and Tang, 2008a]. First, by applying penalty to the infeasible solutions at the time of fitness evaluation of chromosomes and this type of GA is known as Penalty GA (P-GA) [Ma and Zhang, 2008, Jian-hua et al., 2008]. Second, by applying the repairing concept to the population such that it is always ensured that individual in the population is always feasible. This is achieved by applying domain specific knowledge on the infeasible solutions. This kind of GA is called repair GA (R-GA) [Tan et al., 2014].

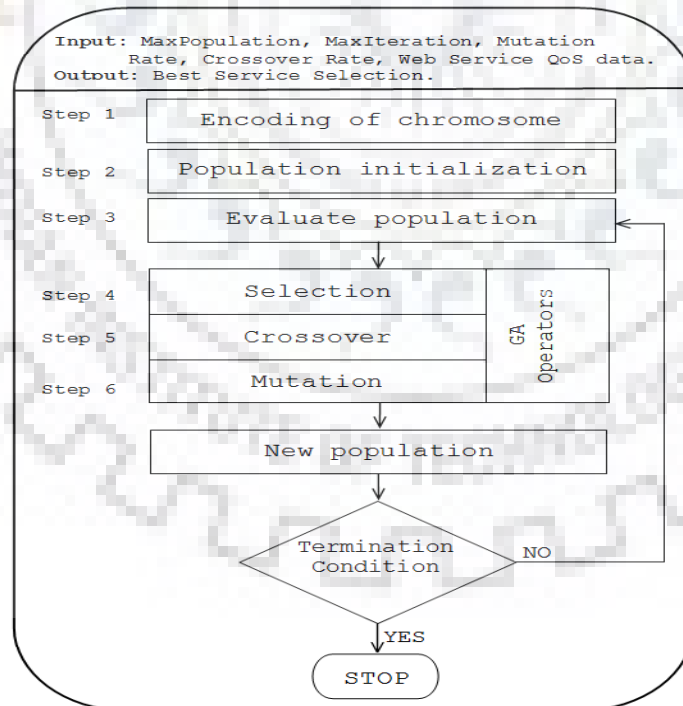


Figure 2.3: Flowchart of Genetic Algorithm for WSS [Zhi-peng et al., 2009, Yilmaz and Karagoz, 2014, Jaeger and Muhl, 2007, Ma and Zhang, 2008].

Based on six important steps identified in Figure 2.3, we have grouped the available works on GA based WSS around these six steps. Forthcoming subsections discuss the available works grouped based on these six steps. The grouping of the available works on GA based WSS is done with reference to their contribution in any of the six steps of GA. So, if a work possesses its major contribution at a step, then it is categorized into corresponding group. In few of the papers, significant contribution in the multiple steps of GA is observed. A total of sixty three papers were found that discuss GA based WSS. Figure 2.4 represents the bar-chart indicating research work held on improvements in identified six steps of GA and other improvements held for efficient WSS.

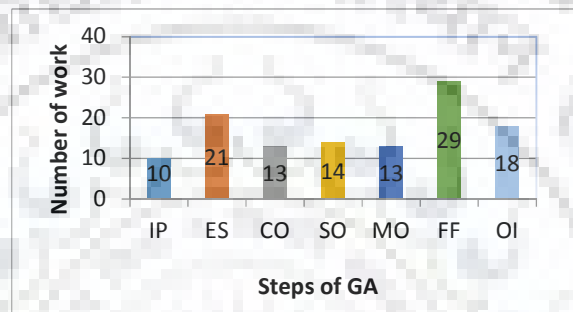


Figure 2.4: Distribution of works based upon improvements in different steps of GA for WSS

Total forty seven out of sixty three research papers has suggested improvements in one or more steps of the algorithm. Remaining sixteen papers are on the application of GA in WSS and composition. It can be observed from Figure 2.4 that maximum of these works emphasize improvements in fitness function. It is to be noted that 65.52% research works use penalty based fitness function and remaining 34.48% works have employed GA for WSS without using penalty based fitness functions.

2.4.1.1 Encoding of chromosomes for efficient selection of WS

In this section, we have discussed the work on GA based WSS which have major contribution in encoding scheme. Twenty one papers have significant contribution in encoding scheme step. As mentioned earlier, one of the important factors affecting the functioning of GA is encoding scheme used to represent WS composition plan. Four most promising encoding schemes are - binary [Zhang and Ma, 2009a], value, permutation, and tree encodings [Arockiam and Sasikaladev, 2012]. A simple strategy for encoding is based on arrays of pointers [Canfora et al., 2005]. Each element of array points to another array of size equal to the number of concrete WS for that service class. Another simple idea is to use an integer array to represent the WSC [Tang and Ai, 2010, Lin et al., 2012, Wang et al., 2013, Buqing et al., 2013, Seghir and

Khababa, 2018]. Each element of integer array stores the index of concrete WS of that service group. At any instance, the values in the integer array represent a sequence of concrete WS involved in composition. However, the encoding is not able to represent all models of composition flows. To handle this, a new chromosome encoding known as a relation matrix coding scheme (CoDiGA) with a population diversity handling mechanism can be used [Zhang and Ma, 2009a, Zhang et al., 2007, Jaeger and Muhl, 2007, Ma and Zhang, 2008, Wang et al., 2007, Chen et al., 2009, Zhang et al., 2006b, Zhang and Lin, 2009]. The relation matrix stores the information about concrete WS and how they are associated with other concrete WS. This encoding scheme also keeps track of all the feasible paths representing service composition. With this coding scheme, an improved and optimized composition plan of WSs is obtained. A similar relation matrix coding scheme, known as a triangular relation matrix coding, is used to represent multifold information in a single matrix [Jian-hua et al., 2008]. The coding scheme is suitable to be used with pseudo-parallel genetic algorithm (PGA), which explores valid composition paths in parallel. A special tree traversal sequence (TTS) coding scheme is able to represent all composition flows in much simpler ways and is comparable to the relation matrix coding scheme in the representation of information [Shuang et al., 2009]. The post-order traversal is used to encode/decode concrete WS in chromosomes. Other important encoding schemes include Differential Evolution (DE) based encoding for complex search spaces [Pop et al., 2011], an auxiliary encoding scheme [Tan et al., 2014] with variable length of chromosomes to represent the forward and backward path, etc. A two dimensional encoding scheme and collaborative learning based learning operator is devised for GA with Cultural Algorithm (CGA) [Liu et al., 2015]. One dimension specifies the service class (representing abstract WS) and the other dimension specifies quality levels. Table-2.6 summarizes these works along with their contributions and research gaps.

2.4.1.2 Effect of the initial population generation scheme on Web Service Selection

In this section, we have discussed the works on GA based WSS which have major contributions in initialization step. Out of total reviewed paper, ten papers have significant contribution in initialization step. Initial population generated in GA based WSS system represents a set of different choices of concrete WS forming different executable compositions. The choice of initial population has a pervasive effect on GA to determine optimized concrete WS satisfying global end to end constraints. The random choice of the initial population usually exhibits substantial variation in running time and convergence time [Canfora et al., 2005, Zhang and Ma, 2009b, Tang and Ai, 2010, Ai and Tang, 2008a, Su et al., 2007]. The convergence time to

find an optimal WS depends on the fitness level of the initial population [Wang et al., 2007]. One of the ways to improve the fitness level of the initial population is to use ant colony optimization (ACO) to select initial population representing WSC [Liu et al., 2010]. ACO performs learning through feedback mechanism that leads to the better global optimization [Dasgupta et al., 2016]. On the selected plans, GA is applied to obtain the optimized composite WSs. The use of ACO in the initial stage improves the overall convergence time of GA. Similarly, the QoS of candidate WSs of each service class can be adaptively divided into a different quality collection known as quality scale [Yuan et al., 2013]. The choice of initial population is made based on the quality scale meeting the global end-to-end QoS constraints. The initial population chosen in this manner results in better fitness of the population and ultimately quick divergence [Yuan et al., 2013]. In order to improve the search capability of GA, WSs from different composition sequences can be chosen to generate an initial population [Wang et al., 2007]. Further, to mitigate the possibility of arriving to local optimality, the Enhance Initial Population Policy (EIPP) probability concept is suitable [Tan et al., 2014]. This eventually leads to the recovery plans with a higher probability of success. The convergence of GA for WSS can be improved by selecting one fifth of the initial population by using skyline and randomly choosing the remaining four fifth populations [Tan et al., 2014]. However, major challenge with this approach is to enumerate skyline services due to the cost involved in computation and problem in threshold selection. The local optimization approach for initial population selection is promising approach to ensure quality results [Seghir and Khababa, 2018]. Local optimizer works in two steps. In first step, the local score is calculated and in the next step, tournament selection is used to ensure that the best individual is selected as part of initial population. Table-2.6 summarizes these works along with their contributions and research gaps.

2.4.1.3 Evaluate population using fitness function

In this section, we have discussed the work on GA based WSS which have major contribution in fitness evaluation of each composition plan in the population using fitness function. Out of the total reviewed papers, twenty nine papers have significant contribution in fitness function step. The fitness function in GA based WSS algorithm measures the goodness of WSC to satisfy end-to-end QoS constraints. The idea of the fitness function in a system for WSS is to filter out unfit executable WSC (chromosomes). Usually QoS parameters are used to characterize the fitness function, so it indirectly represents the QoS requirements of the end user. A fitness function with dynamic penalty factor is more effective and useful for efficiently

solving the WSS problem [Helal and Gamble, 2014, Fethallah, 2012, Ma and Zhang, 2008, Zhang et al., 2006a, Shuang et al., 2009, Su et al., 2007, Mardukhia et al., 2013]. The penalty based evaluation imposes penalty on fitness of the infeasible solution whose fitness is below the threshold. For WSS problem represented as multi-objective optimization, multi-objective genetic algorithm (MOGA) with multiple fitness functions is required [Wang and Hou, 2008]. The cumulative result of multiple fitness functions represents fitness of composition plan. To deal with QoS constraints based on dependency and conflicts between WSs, the repair genetic algorithm (R-GA) is suitable [Ai and Tang, 2008b, Tan et al., 2014, Ai and Tang, 2008a]. The repair GA does not use an explicit fitness function. A repair operator is used to recursively repair the infeasible solution. The repair process is based on the hill-climbing concept.

In [Chen et al., 2009], the dynamic evaluation of the population using relative and absolute fitness function is used to improve the convergence as well as diversity handling of the population for effective WSS. The Relative and Absolute Penalty Technology (RAPT) with the calculation of ' α ' factor is suitable to decide the type of penalty to impose. The fitness of a chromosome can be obtained by calculating the variance of fitness value [Lin et al., 2012]. In this case, two fitness values are obtained, $E(\text{Fitness})$ representing the average value of fitness of the individual plans in the population and $D(\text{Fitness})$ representing the variance of individual plans in the population. To deal with WS failure at run time, the replaceability factor can be evaluated for each chromosome and can be used as one of the parameters for fitness evaluation [Helal and Gamble, 2014].

A number of different fitness functions are presented in the different research works to improve the performance of GA for efficient selection of WSs. A summary of various fitness functions used to improve selection of WSs in different ways is shown in Table 2.4. The fitness functions listed in Table 2.4 can be grouped into four categories. First category (Cat1) includes fitness functions at serial number 1, 3, 5, 6, 12 and 14. Second category of fitness function (Cat2) incorporate fitness functions defined at serial number 4, 9, 11, 15 and 17. Category 3 (Cat3) encompasses the fitness functions at serial number 2, 7, 10, 13 and fitness functions at serial number 8, 16 are in category 4 (Cat4). The fitness functions in Cat1 are similar on treating the QoS parameters as either increasing type or decreasing type. The increasing parameters are kept in numerator and decreasing parameters are kept in denominator to maximize the fitness function score. On the contrary, the Cat2 fitness functions do not treat the QoS parameters differently. But, they are similar to Cat1 in imposing penalty by reducing the fitness values of individual web services involved in the composition. Fitness functions of Cat3

involve the computation of fitness values by including factors apart from QoS values such as current iteration, number of competing services and/or path information, etc. For fitness functions of Cat4, the fitness values for favourable solutions are given incentives and unfavourable solutions are penalized. This ensures the proper distance between two solutions (favourable/unfavourable).

2.4.1.4 Effective Selection operator and selection schemes for efficient Web Service Selection

In this section, we have discussed the works on GA based WSS which have major contribution in selection schemes. Out of total reviewed papers, fourteen papers have significant contribution in selection step. The selection operation regulates the advancement of chromosomes to the next level. The goodness / ability of superior potential of reproduction of each executable composition plans is tested and the premier set of executable composition plans is promoted for consideration in subsequent iterations. During the operation of selection of most prominent executable composition plan, the selection is based on the common strategy such as tournament selection [Wang et al., 2013, Claro et al., 2005] and a probabilistic selection such as roulette wheel selection [Beran et al., 2012], proportionate selection, ranking selection [Palanikkumar and Kousalya, 2012], etc. One improvement favourable for improving the quality of the population is to use an elitism strategy [Zhang and Ma, 2009a, Wang et al., 2013, Seghir and Khababa, 2018, Ai and Tang, 2008a, Ludwig and Schoene, 2011],. Elitist strategy has also been improved further by combining the population of the previous iteration and population generated by crossover operator. For this, sorting of population is done and top 'Max Population' executable composition plans are selected as the population for further consideration [Buqing et al., 2013, Gupta et al., 2015]. While merging two populations, the replacement of old population with new population using SA improves the convergence and prevent locally optimized WSs to get selected [Jaeger and Muhl, 2007]. One refinement in GA is proposed by incorporating memetic algorithm (MA) in each iteration of GA for performing local search of best composition plans. Local search with MA improves overall fitness of all composition plans obtained at the end of every iteration of GA [Ludwig, 2011]. If fittest composite WS instances are selected multiple times, the produced executable sequence of concrete WS will also have higher aggregate QoS value. Based on this observation, a new selection strategy with probability associated with each group is also presented [Arockiam and Sasikaladev, 2012]. The variable probability of selection of WSC relative to the other plans has improved chances of optimal selection of WS [Yuan et al., 2013]. Summary of these works along with their contributions and gaps are shown in Table 2.6.

Table 2.4: Summary of Fitness functions

S. Study No	Fitness Function	Parameter description	Penalty
1 [Helal and Gamble, 2014]	$\text{Fitness}(W_j) = \frac{w_1 * \text{Availability}(W_j) + w_2 * \text{Reliability}(W_j) + w_3 * \text{Replaceability}(W_j)}{w_2 * \text{Cost}(W_j) + w_2 * \text{ResponseTime}(W_j)} - P(W_j)$	W_j is the WSC. w_1, w_2, w_3, w_4, w_5 are weights of QoS. $P(W_j)$ is penalty function.	Yes
2 [Fethallah, 2012]	$P(C) = -t * \sum_{k=1}^R (D_k)^2(C), \text{ Where,}$ $D_k(C) = \begin{cases} 0 & \text{if } Q^k(C) \geq \text{cons}(k) \\ Q^k(C) - \text{cons}(k) & \text{otherwise} \end{cases}$	't' is current iteration, $Q^k(C)$ is k^{th} QoS attribute of composite service c. P is penalty function.	Yes
3 [Sharifara et al., 2014, Jin et al., 2008]	$F = \frac{w_1 * \text{Availability} + w_2 * \text{Reliability}}{w_3 * \text{Cost} + w_4 * \text{ResponseTime}}$	F is fitness function, $w_1, w_2, w_3,$ and w_4 are user supplied weights of QoS attributes.	No
4 [Liu et al., 2016b, Liu et al., 2013, Liu et al., 2015]	$F(X) = \sum_{j=0}^n F(\text{QLC}_{ij})$	QLC_{ij} is i^{th} quality level combination for service class s_{ij} .	No
5 [Canfora et al., 2005, Jian-hua et al., 2008]	$F(\text{gen}, g) = \frac{w_1 \text{Cost}(g) + w_2 \text{ResponseTime}(g)}{w_3 \text{Availability}(g) + w_4 \text{Reliability}(g)} + w_5 D(g) * \frac{\text{gen}}{\text{maxgen}}$	'F' is fitness function of gene 'g' with 'gen' generation, penalty D(g), and w_1, w_2, w_3, w_4, w_5 are weights of fitness factors.	Yes
6 [Jaeger and Muhl, 2007]	$f_{\text{fitness}}(s) = \left(P(s) * \frac{\text{avail}(s) * \text{reputation}(s)}{\text{cost}(s) * \text{time}(s)} \right)^k$	P(S) is penalty factor. avail, reputation, cost and time are QoS parameters.	Yes
7 [Zhang et al., 2006a, Ma and Zhang, 2008, Chen et al., 2009, Su et al., 2007, Shuang et al., 2009]	$\text{Fit} = f - \lambda \sum_{j=1}^n \left(\frac{\Delta P_j}{R_{j\text{Max}} - R_{j\text{M}}} \right)^2 \text{ where, } \Delta P_j = \begin{cases} P_j - R_{j\text{Max}}, & \text{if } P_j > R_{j\text{Max}} \\ 0, & \text{if } R_{j\text{Min}} \leq P_j \leq R_{j\text{Max}} \\ R_{j\text{Min}} - P_j, & \text{if } P_j < R_{j\text{Min}} \end{cases}$	$R_{j\text{Min}}$ and $R_{j\text{Max}}$ are min. and max. value of j^{th} QoS attribute, λ is factor to adjust penalty value, f is a objective function and P_j is sum of j^{th} QoS.	Yes
8 [Ai and Tang, 2008a, Ai and Tang, 2008b, Tang and Ai, 2010, Lin et al.,	$\text{Fitness}(X) = \begin{cases} 0.5 + 0.5 * F(X), & \text{if } V(X) = 0, \\ 0.5 * F(X) - \frac{V(X)}{V_{\text{Max}}}, & \text{Otherwise} \end{cases}$	X is individual in the population, F(X) is total QoS of CWS, V(X) is no. of constraints that X is not in accord with, V_{Max} is total no. of constraints.	Yes

2012]				
9 [Yuan et al., 2013]	et al.,	$F = \sum_{i=1}^r S_i * P_i * W_i^T$	P_i is context matrix and W_i^T is context attribute weight coefficient of service class S_i .	No
10 [Wang et al., 2007]	et al.,	$F(W) = \begin{cases} W - C_{min}, & W > C_{min} \\ 0, & \text{Others} \end{cases}$ $F(W) = \frac{1}{1+c-W}, c \geq 0, c-W \geq 0$	C_{min} is minimum evaluation of 'W' and 'c' is the limit value of conservative evaluation.	No
11 [Wang et al., 2013]	et al.,	$F(CS) = \sum_{k=1}^2 \frac{(Q_k^{max} - Q_{CS}^k)}{(Q_k^{max} - Q_k^{min})} * W_k$	Q_k^{max} and Q_k^{min} are summation of maximum and minimum values of k^{th} QoS among all possible solutions, W_k is weight of k^{th} QoS parameter.	Yes
12 [Pop et al., 2011]	et al.,	$F(y) = w_1 * R + w_2 * A + \frac{w_3}{T} + \frac{w_4}{C}$	w_1, w_2, w_3, w_4 are weights of QoS attributes Reliability (R), Availability (A), Time (T) and cost (c), respectively.	No
13 [Mier et al., 2010]	et al.,	$Fitness = w_1 * \left(\frac{\sum_i^{ Obj } \frac{1}{ DO_i+1 } + \frac{ I_{root}^n \cap I_{obj} }{ Obj } \right) + w_2 * \frac{1}{runPath} + w_3 * \frac{1}{\#atomicProcess}$	O_{obj} are the expected outputs, DO_i is the distance of the individual to the i^{th} required output, I_{root}^n are the necessary inputs of the root node, I_{obj} are the inputs provided to solve the composition, $runPath$ is the execution time of the composite service, $\#atomicProcess$ is the number of atomic processes in the tree, and w_k are weight values of each criterion.	No
14 [Mardukhia et al., 2013]	et al.,	$Penalty(G,P) = (\sum_{i=1}^r W_c[i] * v[G,i]) + \frac{p}{P_{max}}$ $Fitness(G,P) = \begin{cases} F(G), & G \text{ is feasible} \\ F(G) - Penalty(G,P), & G \text{ is infeasible} \end{cases}$	$F(G)$ is objective function for individual G, $W_c[i]$ is importance of constraint i, $v[G,i]$ is 1 or 0, respectively, if the constraint is violated by G or not, p and P_{max} is the current and maximum number of generations, respectively.	Yes
15 [Wang et al., 2015a]	et al.,	$f(CS) = \sum_{i=1}^o \alpha_i * \zeta_i(CS)$	ζ is normalized QoS and α is user priority.	No
16 [Ding et al., 2015]	et al.,	$f(x) = \begin{cases} \eta F(x) + \rho, & \text{if flag} = \text{True} \\ \eta F(x), & \text{if flag} = \text{False} \end{cases}$	$F(x)$ is QoS value, η is an incentive factor, parameter ρ is a penalty-factor, and flag is a transactional label.	Yes
17 [Yan et al., 2015]	et al.,	$Fitness = \lambda_r * \lambda_c * \sum_{i=1}^m \omega_i * f_i(Q_r^i, q_g^i)$	λ_r, λ_c are penalty factors, ω_i is weight assigned to each QoS, and $f_i(Q_r^i, q_g^i)$ indicate whether aggregate QoS value of the plan q_g^i satisfies the requirement Q_r^i in the i^{th} dimension.	Yes

2.4.1.5 Impact of crossover operator on Web Service Selection

In this section, we have discussed the works on GA based WSS which have major contribution in crossover operation step. Out of total reviewed papers, thirteen papers have significant contribution in crossover operation step. Once the encoding of the chromosome is done, new individuals in the population are generated by applying the crossover operator. The crossover operator uses two chromosomes for mating operation [Zhang and Ma, 2009a, Wang et al., 2013]. The crossover operation determines the goodness of newly generated WSC. It is also responsible for the global search capability of GA to search globally optimized concrete WS. The crossover operation is in general either single point crossover [Zhang and Ma, 2009b, Wang et al., 2013] or multi point crossover [Yilmaz and Karagoz, 2014, Buqing et al., 2013]. Generally, the crossover point is randomly selected, however, the chaotic sequence can also be used to generate sequence for selecting crossover points [Zhang and Ma, 2009b, Zhang and Lin, 2009]. The chaotic sequence improves the convergence capability of GA for selection of WSs. This idea is carried forward in the work [Shuang et al., 2009, Zhang and Ren, 2011], and an adaptive crossover strategy is implemented in AdGA [Zhang, 2011] and adaptive genetic algorithm (AGA) [Zhang and Ren, 2011]. The adaptive capability enables the algorithm to use variable crossover probability. The probability factor can be adjusted according to the diversity of the population and fitness of individuals in the current population [Yuan et al., 2013, Zhang, 2011]. It results in improved search capability and optimized composition plan. A knowledge based crossover operator is useful to improve the search capability and convergence [Tang and Ai, 2010, Lin et al., 2012]. The knowledge based crossover operation is also performed based on the prior information such as priorities of mating parents [Tang and Ai, 2010], the importance of abstract WS [Lin et al., 2012], etc. In [Fethallah, 2012], a dynamic, time varying crossover operator is introduced, which keeps on changing with iterations. The time varying operator is useful to handle population diversity [Fethallah, 2012]. Table 2.6 summarizes these works along with their contributions and research gaps.

2.4.1.6 Mutation operator and various mutation schemes for effective Web Service Selection

In this section, the works on GA based WSS having major contributions in mutation operation are discussed. Thirteen papers have significant contribution in mutation operation. Randomness improves the ability of GA to ensure that the global optimization is achieved [Canfora et al., 2005, Mardukhia et al., 2013]. This randomness is introduced by the mutation operation. Randomly a chromosome from the population is selected and with mutation probability, the

gene representing concrete WS in the chromosome is inverted [Canfora et al., 2005, Zhang et al., 2007, Tang and Ai, 2010]. Due to mutation operation, a new sequence of WSs representing executable composition plan is generated. The old execution sequence is replaced with the mutated sequence of WSs [Canfora et al., 2005, Zhang et al., 2006b]. The classical mutation operation does not check whether the mutated composition plan has higher fitness than the original composition plan. It results in either slow convergence of the algorithm or the overall fitness of the solution decrease. A modified mutation policy is proposed in the work [Zhang et al., 2007], in which after applying a mutation operation, two executable composition plans, mother and mutated, are compared. The plan with higher fitness value is considered and other one is discarded [Zhang et al., 2007].

The mutation operation can be controlled using SA. The SA ensures that the fitness of composition plan resulted after applying the mutation operation is improved [Yilmaz and Karagoz, 2014]. Further, harmonic search (HS) has also been used as a replacement for mutation operation. The HS results in better fitness of composition plans as compared to the composition plans obtained after applying a classical mutation operation [Yilmaz and Karagoz, 2014]. A dynamically varying mutation probability based on the fitness of an individual [Lin et al., 2012, Yuan et al., 2013], and with iteration [Fethallah, 2012] has been presented that shows a positive impact on the efficiency of WSS. The work in [Zhang and Ma, 2009b, Zhang and Lin, 2009] shows that the randomness in the selection of a gene for replacement can be avoided by using chaotic law. It improves the convergence of GA for effective selection of WS. The local selection strategy is also helpful in the quick convergence of GA. The selection of concrete WS from the pool can be done in two ways, either based on utility value [Wang et al., 2013] or based on the probability to create different paths from the mutated path [Chang, 2012]. Summary of these works along with their contributions and gaps are shown in Table 2.6.

2.4.1.7 Miscellaneous works on Web Service Selection

In this section, the miscellaneous works (eighteen papers out of total reviewed papers) on GA based WSS are discussed, which uses other algorithms along with GA to improve the performance of selection mechanism. The algorithms compared with simple GA and important outcomes of these works are summarized in Table 2.5. In [Ai and Tang, 2008b], the repair based GA that uses minimum-conflict hill-climbing (MCHC) as repairing operator is introduced. However, MCHC operator takes extra time to repair, but improves efficiency of WSS. Further, a work in [Jin et al., 2008] has presented a new genetic based ant algorithm (GBAA) approach to solve WSS problem by first generating sub-optimal solutions using GA

and then using a sub-optimal solution to initialize the pheromone in max-min ant system (MMAS) [Jin et al., 2008]. In majority of work the WSS problem is modelled as single objective optimization problem. In papers [Claro et al., 2005, Yao and Chen, 2009], the WSC problem is modelled as MOOP and a modified non-dominated sorting based GA (NSGA-II) is employed. The NSGA-II algorithm ranks and sorts each competing individual based on non-domination level. In addition to NSGA-II algorithm, a new crowded comparison operator is applied in [Yao and Chen, 2009]. The crowded comparison operator creates a new pool of offspring and to calculate the crowded distance of each member in the pool. This results in increased population fitness and improved quality of services are selected.

Table 2.5: Summary of comparison between GA and other suggested techniques

Study	Techniques compared with GA	Important outcomes of the research
[Zhang and Ma, 2009b, Zhang and Lin, 2009]	Chaotic GA with relation matrix coding	Optimal selection of WSs, however, performance degradation in the GA is observed.
[Su et al., 2007, Arockiam and Sasikaladev, 2012]	Simulated Annealing	The time complexity of GA is less than SA. SA has better search capability than GA. A Hybrid approach consists of GA and SA can be followed for WSC and is useful.
[Canfora et al., 2005, Wang et al., 2013]	Integer Programming	IP is fast for small number of A_b WS, however, not scalable. GA is scalable, but slow in convergence.
[Yilmaz and Karagoz, 2014]	GA-SA & GA-HS	GA-SA and GA-HS has better convergence time than simple GA, however, complex in nature.
[Ma and Zhang, 2008]	GA with population diversity handling	Population diversity handling mechanism improves the convergence of the algorithm.
[Ai and Tang, 2008a]	GA with penalty based fitness function	Experimental results demonstrate that penalty-based GA is more effective and scalable.
[Beran et al., 2012]	Random walk, blackboard algorithm and GA with blackboard algorithm	Blackboard algorithm produces slightly better quality results for small number of deployment and is more scalable than GA. However, it takes more time to converge.
[Mardukhia et al., 2013]	GA-quality constraint decomposition (QCD) and ILP	GA-QCD outperforms ILP to decompose global QoS constraints to local constraints.
[Ding et al., 2015]	TQoS using GA and exhaustive search	TQoS using GA shows improvements in the time taken to find C_n WS, as compared to exhaustive search and ACO.
[Palanikkumar and Kousalya, 2012]	PSO	PSO converges faster than GA for single and multiuser service selections.

Apart from QoS parameters, two parameters - transactional properties [Ding et al., 2015a], Quality of Experience (QoE) [Zhi-peng et al., 2009], can act as parameter for optimal selection and composition of WSs. Using the transactional properties and transaction composition rules, a GA based transactional QoS aware service selection algorithm, known as TQoS [Ding et al.,

2015a] and TGA [Ding et al., 2015b] can also be employed. In [Zhi-peng et al., 2009], customer expectation and environment is used as one of the parameters for acceptability of composition. In this paper [Zhi-peng et al., 2009], a new parameter called as Quality of Experience (QoE) ameliorates GA by combining learning capability of SA with GA. Furthermore, the work also deals with the problem of pre-mature convergences of GA. The single and multiuser WSS problem can be envisaged as an optimization problem.

In few of the work [Palanikkumar and Kousalya, 2012, Yu et al., 2015, Arockiam and Sasikaladev, 2012] the hybrid solution based on GA and other algorithms is proposed to solve WSS problem. It is reported that a hybrid solution using GA and particle swarm optimization (PSO) [Palanikkumar and Kousalya, 2012], GA with ACO [Chifu et al., 2014, Yu et al., 2015] and also by using a hybrid of SA and GA [Arockiam and Sasikaladev, 2012] performs better than pure GA for WSS. In hybrid solutions, the advantages of all participating algorithms is combined which covers disadvantages/limitations of all participating algorithms. This helps in enhancing the capability of GA and improves the performance of WSS algorithm. On the contrary, hybrid solutions are complex and need to be designed carefully to leverage full advantage of all participating algorithms.

Performance of GA for selection of desired WS largely depends on initial population generated [Beran et al., 2012]. Population with higher fitness leads to best quality optimal solution at the end of GA iterations. Therefore, the multiple composition plans generated as a result of the random walk with exhaustive search act as a useful starting point for GA based WSS and the blackboard algorithm based WSS. The blackboard methodology uses two concepts useful for efficient selection of WS. First, a knowledge base to guide the search in a systematic manner [Beran et al., 2012]. Second, a decision tree based on cost estimation of visited paths [Beran et al., 2012]. Similarly, the initial WSC can be generated automatically with the help of Genetic Programming (GP) due to its benefit of using a tree structure to represent individuals [Mier et al., 2010, Xiao et al., 2012, Ma et al., 2015, Silva et al., 2016, Zhao and Li, 2014].

All the WSS algorithms studies so far, solve the service selection problem by considering end-to-end QoS requirements. This enforces the selection algorithm to check whether end-to-end QoS are being satisfied by combining QoS of all the participating services. A different way to solve the WSS problem is proposed in [Liu et al., 2015] by decomposing end-to-end QoS requirements using cultural algorithm with GA (known as CGA). After decomposing end-to-end QoS, local search techniques can be used to find WSs satisfying local constraints.

2.4.1.8 Research Gaps and Observations

With the changing business requirements and development of new technologies, the improvement in GA for efficient WSS is reviewed in the last section. Contribution of the works presented in last sections and research gaps are also identified and presented in Table 2.6. Based on the study of various works on GA based WSS, few important observations are drawn as specified below:

- Simple GA is capable of searching globally optimal solution for selection of WS. Simple GA performs better than Tabu search, branch and bound, bottom-up method and Global method. However, it is observed to be slow in searching the solution. Moreover, simple GA is stable and more scalable as compared to IP [Zhang et al., 2006a, Claro et al., 2005].
- Encoding scheme used to represent optimal selection of WSs has profound effect on efficiency of selection. Use of encoding schemes such as CoDiGA, tree based scheme, etc., improves the convergence and precision of WSS [Zhang et al., 2007, Zhang and Ma, 2009a, Jaeger and Muhl, 2007, Ma and Zhang, 2008, Wang et al., 2007] .
- The use of penalty based fitness function produces better average fitness of population as compared to GA with non-penalty based fitness function. Due to penalty based fitness evaluation, distance between optimal and non optimal solution becomes larger. This helps in producing better quality results leading to higher fitness of population. Though there is a risk that few of the solutions are not feasible. Dynamic penalty based fitness function is more effective than static fitness function [Canfora et al., 2005, Jian-hua et al., 2008].
- Use of local search technique such as SA, MA, HS, etc. improves the search precision in WSS. This is due to the fact that the local search operates in every iteration of GA. So, the quality of search results are improved. Moreover, local search strategy helps in avoiding premature convergence of GA [Arockiam and Sasikaladev, 2012, Yilmaz and Karagoz, 2014, Wang et al., 2013].
- Using GA along with other evolutionary techniques, for instance, PSO, fruit-fly optimization, MMAS, ACO, etc., produces high fitness of the solution and improved population divergence handling [Liu et al., 2010, Ludwig, 2011, Seghir and Khababa, 2018]. However, the resulting method is more complex and sometime not scalable. Further, in such cases it is difficult to handle run-time failure of WSs. Use of MCDA methods with GA iterations improves the utility score and hence quality of selected WSs [Jian et al., 2016].

Table 2.6: Summary of GA based Web Service Selection approaches

Study	Basis of approach for WSS	Contribution	Tools / Language	Research Gaps / limitations	Improvement in GA step
[Zhang et al., 2006b]	CoDiGA + population diversity handling.	Improved average fitness and convergence over traditional GA.	Java	Complex Encoding scheme. Less adaptive capability.	ES
[Tan et al., 2014]	R-GA + EIPP	In case of failure, recovery is possible without starting from scratch (Enhanced reliability compared to standard GA approach)	Apache ODE	Penalty based fitness evaluation may result more fruitful results.	ES, IP, FF
[Wang et al., 2007]	CoDiGA + population diversity handling.	Improved average fitness and convergence.	Java	Complex Encoding scheme. Less adaptive capability.	ES, IP, FF
[Tang and Ai, 2010]	GA + knowledge-based crossover operator.	The average fitness of the population is improved.	MS VC# 2005	Unstable performance, deteriorate with an increase in the number of constraints. Higher convergence time as compared to standard GA based WSS.	ES, IP, FF, CO, MO
[Canfora et al., 2005]	GA + Dynamic fitness function.	Scalable as compared to IP.	Java and lpsolve	Not suitable for re-planning, fail to meet global QoS constraints in few cases. Slower than IP.	ES, IP, FF, MO
[Seghir and Khababa, 2018]	GA and fruit-fly optimization	High fitness value of the solution, improved convergence rate.	MATLAB	As number of abstract services increases, execution time also increases.	ES, IP, SO
[Zhang and Ma, 2009a]	Multi-choice, Multi-dimension genetic algorithm (MMGA)	Pareto Optimal solution within a certain generation.	Undefined	Not generalized. Not suitable for large number of WSSs.	ES, IP, SO, CO
[Liu et al., 2015]	CGA + improved case-based reasoning (CBR).	Enhanced reliability of Composite WS.	C++	Prediction accuracy is low over traditional GA due to possibility of reuse of poor quality cases.	ES, FF
[Ma and Zhang, 2008]	GA + population diversity handling mechanism.	Improved and stable fitness value. Prematurity problem is removed.	Java	Lack of adaptive capability feature.	ES, FF
[Chen et al., 2009]	Dynamic GA + relation matrix coding scheme + RAPT.	Effective handling of diversity, fast convergence.	Java	Comparative analysis of the multidimensional encoding scheme with other evolutionary techniques can be done.	ES, FF

Study	Basis of approach for WSS	Contribution	Tools/Language	Research Gaps / limitations	Improvement in GA step
[Pop et al., 2011]	GA + LongDE	Constant behavior w.r.t. variation in problem complexity & change in evolutionary parameters.	Ecj tool	Not tested with real world WS data. The objective function can be improved.	ES, FF
[Jian-hua et al., 2008]	PGA + triangular relation matrix coding mode.	Improved performance, dynamic fitness function.	Undefined	PGA Adaptive parameter adjustment is not done.	ES, FF
[Jaeger and Muhl, 2007]	DiGA + SA	Improved Convergence and premature convergence are avoided.	Java	Not able to handle the WS failure situation. Self adaptation is absent.	ES, FF, SO
[Wang et al., 2013]	GA + local selection strategy	Scalable and effective than Integer Linear Programming (ILP) and random selection approach.	lpsolve	Needs to be compared with other variants of GA and evolutionary techniques.	ES, FF, SO, CO, MO
[Shuang et al., 2009]	GA + TTS coding scheme + Tree Composite Pattern (TCP).	Simple encoding scheme, better optimal choice of WSs.	Undefined	Space complexity is high for a large number of C_n WS.	ES, FF, CO
[Lin et al., 2012]	GA + adaptive crossover and mutation strategy.	Termination by computing variance value of fitness.	Undefined	Poor performance. Only sequential workflow is considered.	ES, FF, CO, MO
[Arockiam and Sasikaladev, 2012]	SA + GA	As the number of groups increases, SA performance for WSS also improves over standard GA based selection.	C++, MS VC++	Conflicting QoS parameters need to be considered.	ES, SO
[Buqing et al., 2013]	Catacly-smic mutation trustworthy service composition method (CHC-TSCM) + GA	Higher service composition success rate, smaller decline trend of the service composition success-rate, and enhanced stability	Java	Complex algorithm, less adaptive.	ES, SO, CO
[Zhang et al, 2009, Zhang et al., 2009]	GA + chaotic time series + relation matrix coding.	As compared to standard GA based selection enhanced precision of optimal WSS.	Undefined	Reduced time efficiency compared to traditional GA..	ES, CO, MO
[Zhang et al., 2007]	GA + relation matrix encoding scheme.	Enhance convergence of GA and can get more excellent composite service plan.	Java	Self adaptation of GA may be achieved.	ES, MO
[Liu et al., 2010]	GA + ACO.	The average fitness is higher and better convergence than WSS using ACO.	Undefined	Not implemented for real world WSs. Not suitable to handle WS failure.	IP
[Su et al., 2007]	GA + SA	Better fitness evaluation than standard GA for WSS.	Undefined	Not tested with real world WS.	IP, FF

Study	Basis of approach for WSS	Contribution	Tools / Language	Research Gaps / limitations	Improvement in GA step
[Ai and Tang, 2008a]	GA + Penalty based fitness function.	GA with penalty based fitness function is more effective, scalable, extensible and efficient compared to standard GA.	MS-visual studio .Net	No comparison with other evolutionary techniques and GA alternatives is done.	IP, FF, SO
[Yuan et al., 2013]	GA + Service co-relation matrix.	Improved Optimal selection of WS and user satisfaction as compared to standard GA based approach for service selection.	Undefined	Interdependency among WSs may be considered for achieving optimal WSS.	IP, FF, SO, CO, MO
[Helal and Gamble, 2014]	GA with Replaceability.	Calculation of fault tolerant plan considering Replaceability.	Undefined	Replaceability is calculated using only availability and cost as QoS.	FF
[Sharifara et al., 2014]	GA, Fuzzy Logic and NSGA	Improved processing speed when large number of services are available over NSGA-II.	Java	Optimal solution generation within certain generations only. Mapping of service composition to multi objective problem is not clearly mentioned.	FF
[Liu et al., 2016b, Liu et al., 2013]	CGA + CBR	Improved composite service reliability over CMMAS.	C++	Recovery from failure is difficult and time consuming compared to rGA.	FF
[Wang and Hou, 2008]	MOGA + dominance relation.	Design of fitness function is simplified.	Undefined	Inconsistency in the dimension of the attributes.	FF
[Ai and Tang, 2008b]	Repair GA + MCHC	Improved GA scalability and effectiveness over GA without repairing.	MS VC#	Slow convergence as compared to standard GA.	FF
[Jin et al., 2008]	GBAA + MMAS+ divide and composite + and rank sort.	GBAA can find the optimal WSS with less number of iterations (fast convergence).	Java	Not suitable for multipath and large number of WSs.	FF
[Mier et al., 2010]	GP	Fast convergence compared to ILP based approach, automatic generation of WSC plans.	Java	Dependency on context free grammar (CFG), higher space complexity.	FF
[Wang et al., 2015a]	GA + skyline technique	Improved optimality and computation time over exhaustive search, random selection and standard GA.	Java	High computation cost, problem in threshold selection. Skyline set for each service to be maintained by brokers.	FF
[Ding et al., 2015a]	TQoS + GA	Improved and effective selection of services compared to transactional service selection algorithm.	Java	Slow convergence in comparison with traditional GA.	FF

Study	Basis of approach for WSS	Contribution	Tools / Language	Research Gaps / limitations	Improvement in GA step
[Fethallah, 2012]	Penalty based GA.	In comparison with integer linear programming algorithm, penalty based GA has improved average population fitness, Adaptive Penalty function .	Undefined	Comparison with other approaches is not done. MOOP in place of SOOP.	FF, CO, MO
[Zhang et al., 2006a]	GA	GA based WSS outperforms other heuristic techniques - branch and bound, bottom-up method and Global method.	SENE CA	No implementation for real world WSs is available.	FF, MO
[Mardukhia et al., 2013]	GA + quality constraints decomposition (QCD).	Running time is comparable with TGA and better than mixed integer program. Less complex.	Microsoft Visual C# .NET	Running time is slightly on the higher side when the number of A _s WS increases.	FF, MO
[Beran et al., 2012]	GA+ random walk + Blackboard algorithm	Improved WSS quality over sequential GA.	Python, Google App Engine	Comparative study with existing approaches not done.	SO
[Gupta et al., 2015]	GA with population sorting	Improved fitness of population over traditional GA.	Undefined	Higher execution time and premature convergence.	SO
[Ludwig, 2011]	Munkres algorithm + GA + MA + Random Search	Improved time efficiency, good selection scalability over traditional GA, random selection and MA approach for selection of WS.	Java	Comparative study only. Less efficient with increasing P-C pair.	SO
[Palanikkumar and Kousalya, 2012]	GA + PSO	Few parameters to adjust, easier to implement.	MS Visual C++, BPEL4 WS	Domain specific Implementation, user preference is not considered. GA produces better result than PSO. Particle velocity is not controlled.	SO
[Claro et al., 2005]	NSGA-II + crowded comparison operator.	Simplified fitness function, multiple feasible solutions.	Java	Slow convergence, generation of optimal solution within certain generation only.	SO
[Zhang and Ren, 2011]	AGA	AGA model can be used to achieve MultiObjective Optimization.	Undefined	Complex, Performance degradation with increase in number of QoS.	CO

Study	Basis of approach for WSS	Contribution	Tools / Language	Research Gaps / limitations	Improvement in GA step
[Zhang, 2011]	AdGA + population diversity tracking + adaptive crossover.	Achieves better search ability and faster convergence speed.	Undefined	Comparative study with other evolutionary techniques is missing. Internal structure of WSC is not considered.	CO
[Yilmaz and Karagoz, 2014]	GA + SA + HS.	Improved search precision compared to traditional GA and GA+ACO approach.	BPEL4WS	Higher convergence time as compared to standard WSS.	CO, MO
[Chang, 2012]	GA	Stable and optimal degree of above 90%.	Undefined	Slow Convergence, less scalable compared to PSO and ACO based selection.	MO
[Purohit et al., 2015]	GA + Ontology	Improved WSS compared to QoS based selection using traditional GA.	Java	Suffer from time complexity due to ontology processing compared to non-ontology based approaches.	App
[Yao and Chen, 2009]	NSGA-II + crowded comparison operator.	Simplified fitness function, multiple feasible solutions.	Java	Slow convergence, generation of optimal solution within certain generation only.	App
[Sasikaladevi and Arockiam, 2012]	GA with MOOP + Population Based Service Selection Algorithm (PBSSA).	Stable optimality results for PBSSA are obtained.	C++, MS VC++	Performance is not tested with other conflicting parameters.	App
[Zhi-peng et al., 2009]	QoE/QoS driven SA based GA (QQDSGA)	Better than GA and SA for WSS in terms of average fitness of population. More satisfactory results with feedback (QoE).	C++, MATL AB	More fruitful results can be obtained by QoE quantification.	App
[Ding et al., 2015b]	GA using transactional property	Study of effect of transactional property on QoS. Selection efficiency and accuracy is enhanced.	Java	Transactional properties are difficult to measure. Higher cost.	App
[Jian et al., 2016]	GA and MCDA	The fitness of the solution and utility score is high as compared to brute force approach (called EEP).	MATL AB	Time complexity is high over standard GA.	App
[Liu and Weng, 2012]	GA and workflow	Optimal selection of workflow based composition.	Undefined	With the increase in number of subtasks, fitness and hit rate is reduced.	App
[Amiri et al., 2013]	GA	Simple algorithm. Improved computation time than Tabu search.	Java	Unstable results are obtained. Real time requirements are difficult to handle.	App

ES→ Encoding Scheme, IP→ Initialize population, FF→ Fitness function, SO→ Selection operation, CO→ Crossover operation, MO→ Mutation operation, App→ The papers have application of GA for WSS.

2.4.2 IOPE based replaceability and Web Service Selection

The functional description of WS can be specified by the Input, Output, Precondition, and Effect (IOPE). The Input specifies the details of input required by the WS. The inputs are processed by the WS to generate desired output. The expected output from the WS after execution is specified by the Output parameter. The Precondition parameter states the condition to be fulfilled by the WS before start of the execution. The effect parameter describes the after effects when WS execution is completed. The IOPE matching among WSs determines the ability of one WS to match the functionality offered by the other WS. The IOPE based matchmaking of WS is introduced by [Paolucci et al., 2002] and studied in the past. Many solutions for web service discovery and selection based on IOPE information are available [Bellur and Vadodaria, 2008, Guo et al., 2011].

The idea of capability matching of WSs is not sufficient and does not produce accurate results. The matchmaking based on Input/Output (IO) parameters is observed as an improvement over capability matching [Paolucci et al., 2002, Paolucci et al., 2002]. The matchmaking is performed between Input parameters of required WS with input parameters of advertised WS. Similarly, the matchmaking of output parameters is done. The matchmaking process involves semantic matching between parameters. Depending on the result of matchmaking, a matching score is assigned. The matchmaking algorithm in this work [Paolucci et al., 2002] is based on greedy strategy and suffer from the problem of generating incorrect outcome. To overcome this problem, a new matchmaking algorithm based on Hungarian method is proposed [Bellur and Kulkarni, 2007]. The work [Bellur and Vadodaria, 2008] reported that the input/output based matchmaking results can be further improved by including precondition and effect (PE) parameters along with input/output parameters. By including precondition based matchmaking, the precondition requirements specified at the service providers end are fulfilled before service execution. Similarly, on the client end, effect is needed. The IOPE based matchmaking methods enforces end user to specify the IOPE requirements having context similar to that of the IOPE information available in owl-s. The end user may not have exact domain knowledge and therefore, the exact WSs may not appear in the output. Therefore, wordnet based matchmaking is available [Paulraj et al., 2011].

The use of parameters other than IOPE is essential to determine service compatibility. Sometime, location constraints are essential to be considered for finding service compatibility [Hu et al., 2017]. Without considering the location constrain on WSs the correct and desired

output from composite WS is difficult to achieve. Therefore, along with IO matchmaking, the location constraint matchmaking is a useful solution. In order to acquire the location constraints of WSs, the tags are collected. Based on the tags classification in the Location category, the location constraints are determined. The use of similarity/dissimilarity measure further improves the matchmaking results [Shirazi, H., 2018].

The core idea of IOPE matchmaking rests on the bipartite graph matching. The bipartite graph matching determines the similarity level. The bipartite matching problem can be solved using brute force approach [Bellur and Kulkarni, 2007], Ford-Fulkerson algorithm [Paulraj et al., 2011], Hungarian method [Bellur and Vadodaria, 2008, Bellur and Kulkarni, 2007], the determinant method using recursive algorithm [Mucha and Sankowski, 2004]. In the work [Bellur and Kulkarni, 2007], the problem of matchmaking is represented as a bipartite graph matching problem and Hungarian method is used to evaluate the matching score. A modified Hungarian method is proposed in [Khanam et al., 2013]. The summary of the comparison of brute force approach/Ford-Fulkerson algorithm/Hungarian algorithm/determinant method is presented in Table 2.7. ' n ' is number of parameters to be compared.

Table 2.7: Comparison of Matchmaking techniques

Method Used	Time complexity
Brute force Approach	$O(n^n)$
Ford-Fulkerson algorithm [Paulraj et al., 2011]	$O(n^3)$
Hungarian method [Bellur and Vadodaria, 2008, Bellur and Kulkarni, 2007]	$O(n^4)$
The determinant method using recursive algorithm [Mucha and Sankowski, 2004]	$O(n!)$
The determinant method using Gaussian elimination method [Ivan et al., 2011]	$O(n^w)$

The weighted path based semantic service similarity is presented in [Pukkasenung et al., 2010]. The work considers the semantic similarity evaluation based on Input, Output, Precondition and Effect (IOPE). Two vectors having IOPE information for advertised and required web services are formed. The cosine of the angle between the two vectors is evaluated which gives a semantic similarity measure between advertised and required web service.

The work related to IOPE matching for WSs, mostly uses Hungarian method. However, the Hungarian method has its own limitations in terms of time complexity. As the number of IOPE parameters increases, Hungarian method takes large time.

2.5 WEB SERVICES DATASETS

The dataset which represents functional and non-functional (QoS) properties of web services are used to perform experimentations related to web service selection. The functional description of WSs are useful to identify functionally similar WSs. Whereas, the non-functional details are useful to differentiate among functionally similar WSs.

The functionality offered by the web services essentially includes description from the service provider about offered service, type and number of parameters, output generated by the service, any pre-requisites for smooth service execution, changes made by the service after completion of execution, etc. The functional properties of web services are usually presented as either using ontology or as owl-s descriptions.

2.5.1 QoS dataset

A dataset of 2507 WSs (also known as QWS dataset) is most popular QoS dataset [Masri and Mahmoud, 2009]. It consists of records of 2507 WSs with nine QoS parameters namely, Response Time (R_t), Reliability (R_l), Availability (A_v), Compliance (C_o), Throughput (T_h), Successability (S_u), Documentation (D_o), Best Practices (B_p), and Latency (L_a) are used. Table 2.8 provides an introductory definition of each of the QoS parameters. Table 2.8 also details the unit of each QoS parameter, min value, max value, mean, standard deviation and type of QoS parameter. For the QoS parameter having the higher value as better is known as increasing type and the QoS parameter with lower value as better is termed as decreasing type QoS parameter.

The currently available version of the QWS includes a set of 2,507 web services with QoS measurement of each parameter. The dataset is generated by using the web service broker framework [Masri and Mahmoud, 2007]. The dataset of 2507 web services is available in a file with 2507 rows and 11 column entry each separated by commas. Each of the row corresponds to the QoS parameter value of the individual candidate web service. Some of the example entries from QWS dataset are as shown in the Table 2.9.

Table 2.8: Web Service QoS parameters definition [Masri and Mahmoud, 2009]

QoS parameter	Definition	Unit	Min Value	Max Value	Type	Mean	Std. Deviation
Response Time	The amount of time a WS takes to serve the user request. Usually, it is calculated by finding the difference between the time of receiving the response from the WS and the time at which the request is issued to the WS.	ms	37	4989.67	Decreasing	383.866	564.472
Reliability [Singh et al., 2016]	For the given duration of time, the probability of WS to respond to the request of user without any error.	%	33	89	Increasing	69.782	8.577
Availability	At any instant of time, the probability of a web service to respond to the given request.	%	7	100	Increasing	81.142	18.704
Compliance	The percentage of the extent to which the WSDL documents associated with the WS follows the specifications of WSDL.	%	33	100	Increasing	88.438	10.026
Throughput	A measure of the number of WS invocations occurs in a given time period.	Req/min	0.1	43.1	Increasing	9.036	7.732
Successability	The ratio of the total number of response sent by a WS with the number of request messages received by the WS.	%	8	100	Increasing	83.885	19.906
Documentation	The measure of the number of description tags in the WSDL document of the WS.	%	1	97	Increasing	31.32	31.516
Best Practices	It is the measure of the extent to which the non-proprietary specifications are followed by the WS.	%	50	95	Increasing	79.307	7.818
Latency	The amount of time taken to process the request for the WS by the server on which the WS is deployed.	ms	0.25	4140.35	Decreasing	56.614	191.729

Table 2.9: Sample entries for QWS dataset [Masri and Mahmoud, 2009]

R _t	A _v	T _h	S _u	R _l	C _o	B _p	L _a	D _o	Service Name	URL of WSDL
56	97	9	99	73	78	84	1	35	BookInfoService	http://edi.btol.com/bookinfoservice/bookinfoport.asm?wsdl
459	98	9.5	100	73	89	80	3	36	LDAPService	http://www.epfl.ch/ws/ldap.wsdl
469	98	13.1	100	67	78	77	0.67	93	XigniteArchive	http://www.xignite.com/xArchive.asmx?wsdl
516.67	63	7.2	63	83	89	91	10.67	33	sms	http://www.info-me-sms.it/ws.php?wsdl

The first nine values in the table are the QoS measurement of each of the nine QoS parameters. Next column represents the name of the service. The value in the last column is the URL of the WSDL of the web service.

The first nine values in the table are the QoS measurement of each of the nine QoS parameters. Next column represents the name of the service. The value in the last column is the URL of the WSDL of the web service.

Another QoS dataset (we call it as Dataset-2) consists of QoS measures of 10000 web services generated using [Borzsony et al., 2001]. The details of nine QoS parameters used for evaluation of WSS methods are presented in Table 2.10. The *type* field represents whether the parameter is of increasing or decreasing type. Increasing type of QoS parameter represents ‘higher the better’ trend, such as, throughput, and decreasing type indicates the ‘lower the better’ trend such as cost.

Table 2.10 Details of dataset of 10000 web services generated using dataset generator

Parameter	R _t	A _v	T _{hr}	S _{uc}	R _l	C _o	B _{pr}	L _a	D _o
Type	D	I	I	I	I	I	I	D	I
Sample Values	492	84	10	84	72	61	89	11.72	64

R_{tm} = response time, A_{va}=availability, T_{hr}=throughput, S_{uc}=successability, R_{el}=reliability, C_{om}=compliance (to WSDL description), B_{pr}=best practice (by following WS-I), L_{at}=latency, D_{oc}=documentation, I=Increasing, D=Decreasing.

2.5.2 OWL-S dataset of Web Services

The OWL-S dataset for WSs is available from [Sem, 2016]. The OWLS-TC4 dataset contains 1083 OWL-S files to describe the functional capabilities of WSs. The dataset also contains 42 queries with their relevance services sets. Both of the OWL-S files and queries are well annotated by 38 domain ontologies. The major categories along with number of OWL-S files in each major categories are summarized in Table 2.11. The owl-s file contains semantic descriptions of WSs in terms of Input, Output, Precondition and Effect (IOPE). These owl-s files are written in XML and can be read by owl-s XML parser. The owl-s description is useful to determine the functional characteristics of any WS. Further, the IOPE based matchmaking

among WSs can be performed. Matchmaking process is helpful in determining semantic similarity and/or semantic composability among WSs [Bhuvanewari and Karpagam, 2012].

Table 2.11: Nine major domains and details of number of WSs in each major domain.

S. No.	Domain	Sub-domains	Number of WSs
1.	Communication	Media, video, film, etc.	58
2.	Economy	Vehicle price, Book price, Electronic item price, eadable item price, etc.	359
3.	Education	Book/journal search, publication search, university professor in academia, geographical region, company profession, etc.	286
4.	Food	Food store, drug stores, grocery store, food service, etc.	34
5.	Geography	Get distance between location, google map, get traffic information, mile-to-kilometre conversion, traffic information, zip code, location, etc.	60
6.	Medical	Check cost and healing plans, hospital, medical flights, medical records of patients, etc.	73
7.	Simulation	Door lock/unlock, switch on/off, light, mess module on/off, etc.	16
8.	Travel	Weather, hotel, geographical, Sports, surfing, etc.	165
9.	Weapon	Weapons information (lending range, financing range, funding agency service, etc.	40

2.6 PERFORMANCE EVALUATION MEASURES

Four performance parameters are used to evaluate proposed and existing approaches. The hardness level, satisfaction level, Euclidean distance, and Time for selection of WSs parameters are used for evaluation purpose and defined in the following section. We have selected these parameters because these parameters are widely used in the existing similar works for the evaluation of WSS approaches [Cho et al., 2016, Seo et al., 2005, Lim et al., 2012, Stephen and Yin, 2011].

Hardness: The hardness of a QoS constraint is the ratio of the required QoS to the maximum value of the corresponding QoS parameter for the available web services. In order to determine hardness of a web service, the hardness of associated QoS parameters can be used. The average value of hardness of all QoS parameters forms the hardness of a web service and defined using equation 2.1 [Purohit and Kumar, 2018].

$$S_{i_{hardness}} = \frac{1}{n} \sum_{j=1}^n S_{i\{QoS_j\}hardness} \quad (2.1)$$

Where,

$$S_{i\{QoS_j\}hardness} = \frac{S\{QoS_{(j)}\}_{max} - S_{i\{QoS_j\}}}{S\{QoS_{(j)}\}_{max}} * 100 \quad (2.2)$$

Here, $S\{QoS_{(j)}\}_{max}$ signify the maximum value of the j^{th} QoS parameter among all candidate WSs and $S_{i\{QoS_j\}}$ is the j^{th} QoS parameter of i^{th} web service S_i .

Satisfaction: The service satisfaction is defined as a measure of how well the QoS of service meets the users' concerned QoS requirements. The service satisfaction can be obtained from satisfaction score of individual QoS parameter of the service using equation 2.3 and 2.4 [Stephen and Yin, 2011].

$$Sat_{i,j} = \begin{cases} 0.5 * (2 * QoS_{i,j} - 1)^{1-w_j} + 0.5, & \text{if } QoS_{i,j} > 0.5 \\ 0.5 * (-2 * QoS_{i,j} + 1)^{1-w_j} + 0.5, & \text{if } QoS_{i,j} \leq 0.5 \end{cases} \quad (2.3)$$

$$Sat_{WS_i} = \sum_{j=1}^n w_{wt_j} * Sat_{i,j} \quad (2.4)$$

Where, $Sat_{i,j}$ is the satisfaction score for j^{th} QoS parameter of i^{th} WS, Sat_{WS_i} is the satisfaction score of i^{th} WS and w_{wt_j} is the weight of j^{th} QoS parameter.

Euclidean distance: The Euclidean distance (ED) between two web services is the ordinary distance between WSs defined in Euclidean space. The ED between web service S_i and any service S_k is defined using equation 2.5 [Mobedpour and Ding, 2013].

$$ED = \sqrt{\sum_{j=1}^m w_{wt_j} * (|S_{k,j} - S_{i,j}|)^2} \quad (2.5)$$

Where, w_{wt_j} is the weight of j^{th} QoS parameter, m is number of QoS parameters, $S_{k,j}$ is j^{th} QoS parameter of k^{th} WS, and $1 \leq i \leq n$.

Time for selection of WS: The time required to perform task of selection of web services by following a systematic procedure such that the set of selected services meet the requirements specified by the end user. The service selection time is calculated after the services are discovered.

2.7 OBSERVATIONS DRAWN FROM THE CHAPTER

Web service selection is an active area of research. The WSs finds their applications in each of the upcoming new technologies [Gajjar et al., 2016]. For many of these technologies, WSs act as backbone to offer value added services. The practical significance of WSs has attracted huge amount of research in the area of SOA. The efficiency requirements of the new applications being developed using WSs has inspired researchers to introduce the application of area such as MCDM approaches, skyline technique, machine learning techniques, semantic based techniques, etc. for selection of WSs and draw general conclusions [Phartiyal et al., 2018]. However, the insufficient methodological details are provided by the large part of earlier reported studies. Thus, makes it difficult to understand the process of web service selection and adopt the existing solutions to handle the challenges offered by new upcoming technologies. We observe that the end user requirements are not taken into consideration while making the service selection decision. Moreover, the scalability issue with the existing WSS approaches is observed i.e. with the large increase in the number of candidate WSs, the performance of WSS process is reduced largely.

The more practical approaches of WSS should find the mechanism to consider only an appropriate set of WSs to be considered by selection process. We have further analyzed that the weight of QoS parameter plays a vital role in selection process. Before developing a WSS system, we need to pay attention on three important concerns. First, due to the upcoming demands from new technologies, more and more WSs are made available over Internet and are increasing continuously. The system should have capability to deal with the increasing number of services. One need to apply some pre-processing technique to identify the useful set of WSs with reference to the end user requirements. Secondly, the end user finds difficulty in expressing the values of weights of QoS parameters. The system should have capability to evaluate values of weights by observing the quality of services offered by WSs available from various service providers. Moreover, the system should also deal with the partial weight values specified by the end user. The existing approaches for WSS fail in addressing this issue. Thirdly, the run time failure of the web service has a deep impact on the efficiency of the WSS

system. The problem has bigger impact on the performance of WSS system in case of composite WS.

2.8 CONCLUSIONS

This chapter briefly discusses the process of Web Service Selection and introduces the quality models for Web Service. We have reviewed the existing available work on QoS based Web Service Selection. The important observations and research gaps are highlighted. The identified research gaps are further used to design the objectives undertaken in this thesis work. Based on the literature review conducted, it is observed that the candidate Web Services can be processed in two steps – firstly by applying prefiltering on candidate Web Services with reference to the end user requirements and in the second step prioritization followed by selection can be performed. The classification and clustering are useful to achieve Web Services prefiltering. We have initially performed empirical study of various classification approaches. The results of the empirical study are further used to develop the classification based Web Service Selection approach. Similarly, the empirical study of various clustering techniques is conducted. The results of the empirical study are used to develop clustering based Web Service Selection approach as discussed in the forthcoming chapters of the thesis. It is observed that replaceability plays a very important role in Web Service Selection. To handle the Web Service failure, a new approach based upon replaceability is discussed in the thesis. The works handling other research gaps are presented in the forthcoming chapters. The work presented in this chapter has been published as [Purohit and Kumar, 2018c].

CHAPTER 3

EMPIRICAL STUDY OF WEB SERVICES CLASSIFICATION TECHNIQUES

The review work presented in the previous chapter showed that to deal with large number of candidate WSs, existing approaches are not efficient. Further, it is observed that the classification technique is a very good candidate for prefiltering WSs. Therefore, in this chapter we have presented an empirical study to determine best performing classification model to classify WSs. There are many important parameters such as error rates, F-measure, kappa statistics, and other performance measures which are considered for comparing and evaluation of various classification models. Further, the majority vote technique is used to perform WS classification. The objective of this chapter is to perform empirical study of various learning models namely logistic regression, multi layer perceptron, NNge, JRip, J48, and random forest. The top three learning models with best performance are compared with majority vote based learning model.

The problem of WS classification is introduced in Section 3.1. The choice of learning models to perform empirical study and their details are discussed in Section 3.2 and 3.3, respectively. Section 3.4 presents a detailed analysis on the performance of various learning models and important observations drawn. Various threats to the validity of the presented approach appears in Section 3.5 followed by conclusions drawn from the chapter in Section 3.6.

3.1 INTRODUCTION

The web service offering smart services to realize Cloud computing, smart city applications, IoT concept, etc. has attracted many service providers. As a result, a lot of functionally similar web services are available to perform the desired task. In order to ensure the efficiency of applications developed using web services, the task of web service selection must be efficient. QoS information can be used to compare, prioritize, and select web services. Moreover, when number of functionally similar web services is significantly large in number, the task of

selection of desired web service based on QoS also become difficult and time consuming. This lead to defeat the purpose of applications developed using web services to offer value added services in real time [Crowley et al., 2016]. Therefore, the web service classification can be employed for efficient selection of web service. To efficiently perform the classification of WSs, a learning model is to be selected from among the available popular learning models. To achive this, the empirical study of learning models is conducted. The empirical analysis help in choosing the best learing model for WS classification, as presented in forthcoming chapter. In order to realize this, a dataset of real world web services with associated QoS information is desired. One such standard dataset which can be useful to realize the web service based applications is available [Masri and Mahmoud, 2009]. The dataset contains QoS information for the real world web services. Few of the web services available in the dataset are, service for obtaining current temperature (TemperatureService), service for weather information (ForecastServiceImplService), service to book hotel (K4THotelAvailWS), service to make payment (SmartPayments), etc.

In this chapter, we address the problem of web service classification. The empirical study of various classifiers is performed. Following are the important contributions of the chapter:

- i) Study of available labeled QoS dataset useful to generate learning model for WS classification.
- ii) An empirical study of learning models is conducted on the basis of error rate and other performance parameters.
- iii) Comparative analysis of best three web service learning models with majority vote classifier is done.

3.2 CHOOSING THE CLASSIFICATION TECHNIQUES

Based on the past study and the reported performance analysis, eleven learning models are selected. The learning models considered are, Logistic Regression (LR), Multilayer Perceptron (MLP), Non-nested generalized exemplars (NNge), PART, Decision Table (DT), JRip, J48 decision tree, Random Forest (RF), Decision Stump (DS), CART, and Support Vector Machine (SVM) [Kundu et al., 2011]. We have conducted a performance based analysis of these eleven learning models. 10-fold cross validation based training and testing of each of the learning model is performed using QWS dataset. The web service classification models are compared on various parameters such as - accuracy, average absolute error (AAE), average relative error

(ARE) along with kappa statistics and visual analysis using model performance chart, etc., as shown in Table 3.1. The best performance parameter values are shown in bold. Based upon the observation, top two performing models from each of the categories, i.e., function based, rule based and tree based classifiers are selected. The six learning models considered for further analysis are - LR, MLP, NNge, JRip, J48, and RF.

Table 3.1: comparative analysis of eleven learning models for web service classification using nine QoS parameters

Performance Measurement parameters	Learning model										
	Rule Based			Decision Tree based			Function Based				
	PART	JRip	NNge	DT	RF	J48	Cart	DS	MLP	SVM	LR
Accuracy	68.68	71.4286	71.4286	65.9341	82.143	69.506	68.1319	43.96	81.319	78.297	85.714
Kappa statistic	0.5666	0.6024	0.6044	0.5298	0.752	0.5806	0.558	0.224	0.7422	0.6961	0.802
Average absolute error (AAE)	0.166	0.1692	0.1429	0.3451	0.1667	0.1626	0.1721	0.324	0.1097	0.1085	0.09
Root mean squared error	0.383	0.3409	0.378	0.3358	0.2697	0.3744	0.3716	0.405	0.2706	0.3294	0.2284
Average relative error (ARE)	45.8	46.7747	39.4944	67.7556	46.098	44.952	47.579	89.69	30.334	30.005	24.889
Root relative squared error	90.08	80.1721	88.8938	88.9842	63.429	88.067	87.4083	95.24	63.635	77.476	53.721

3.3 WEB SERVICE CLASSIFICATION MODELS

In this subsection we have discussed six learning models used for classification of web services.

3.3.1 Tree based learning model

The decision tree based learning model generates decision tree based on the attributes. Attributes are considered in the priority order one after the other. In this category two learning models are considered for study – J48 decision tree classifier and Random Forest.

3.3.1.1 J48 decision tree

The decision tree generated using J48 based learning model is used to take decision about classifying unknown inputs. The decision tree formed must be as balanced as possible. For a training set S and an item $X \in S$, the number of bits needed to decide whether X is positive or negative is estimated using entropy. If $\rho_{(+)}$ and $\rho_{(-)}$ are percentages of positive and negative examples, respectively, in S , then entropy on training set can be defined using equation 3.1 [Mohanty et al., 2010, Goh and Singh, 2015].

$$H(S) = -\rho_{(+)} \log_2 \rho_{(+)} - \rho_{(-)} \log_2 \rho_{(-)} \text{ bits} \quad (3.1)$$

The gain is measured by obtaining the difference between entropy before the split and after the split. The expected drop in the entropy is obtained using equation 3.2 [Mohanty et al., 2010].

$$\text{Gain}(S,T) = H(S) - \sum_{v \in \text{Value } s(T)} \frac{|S_v|}{|S|} H(S_v) \quad (3.2)$$

Where, v is all possible value of T and S_v is the subset for which $X_T = v$. During decision tree generation, attributes are considered in the order of gain in the entropy, i.e., the attribute with the highest gain is given priority over other.

3.3.1.2 Random Forest (RF)

Random forest model starts by taking randomly chosen subset S_r , of training data and randomly chosen subset, d , of attributes. The information gain is evaluated on S_r instead of full training set. A new data point D is classified using each of the trees S_1, S_2, \dots, S_m and majority vote is applied to take the consensus decision [Schapire, 2001, Darbar and Samanta, 2015].

3.3.2 Function based learning model

The function based learning model has a utility function associated with them. The function needs to be optimized for taking decision with high precision and accuracy. Two popular function based learning models are logistic regression and multi-layer Perceptron model.

3.3.2.1 Logistic regression (LR)

Logistic regression is a technique to establish relationship between a dependent variable Y and one or more independent variables. For a given x the expected value of Y is represented as $E(Y|x)$. The conditional mean of Y given x is $\pi(x)$. The logistic regression model used to determine the conditional mean is represented using equation 3.3 [Hosmer and Lemeshow, 2000].

$$\pi(x) = \frac{1}{1+e^{-\alpha}} \quad (3.3)$$

Where, α is a vector representing the weighted combination of independent variables. A logistic regression model is of two types [Hosmer and Lemeshow, 2000]. (i) Univariate regression model has one independent variable to determine dependent variable. (ii) Multiple regression model, also known as multivariate regression model, has more than one independent variables to determine dependent variable. For web service classification, we have used multiple regression model.

3.3.2.2 Multi Layer Perceptron (MLP)

This classifier is a neural network (NN) model based on supervised learning method called back-propagation algorithm. The neural network model has input layer, hidden layer and output layer. Each of the neuron in the hidden layer transforms weighted input x_i from previous layer using equation 3.4 [Jie et al., 2012].

$$z = x_1 * \omega_1 + x_2 * \omega_2 + \dots + x_n * \omega_n \quad (3.4)$$

For an unknown input y , the predicted output d is a function of input (\bar{y}), weight (\bar{w}), and threshold (\bar{T}), i.e., $\bar{d} = g(\bar{y}, \bar{w}, \bar{T})$, where the weight vector \bar{w} is initialized randomly, and adjusted iteratively using error (E) prediction using equation 3.5 [Mustafa, and Kumaraswamy, 2015].

$$E = |z - d| \quad (3.5)$$

The weights are adjusted as per the equation 3.6 [Mustafa, and Kumaraswamy, 2015].

$$w_i = w_i + E * m * y_i \quad (3.6)$$

Where, m being the learning rate of the neural network.

3.3.3 Rule base learning model

Rule based classifiers are based on a list of rules of the form IF C_1 AND C_2 AND C_3 AND THEN class X [Mustafa, and Kumaraswamy, 2015]. The rules are ordered based on the

rules quality or based on class. Rule based classifiers uses the optimized set of rules to take classification decision. Two rule based models considered for study are Non-nested generalized exemplars and JRip.

3.3.3.1 Non-nested generalized exemplars (NNge)

NNge is a hybrid learner which combines the advantages of induction and instance based learners. Examples are combined (merged) and most generalized form is kept in the memory. The generalized exemplars represent more than one training example. If there are n attributes in the example in the training set, then each of the generalized exemplars represents a geometric figure of n -dimensional covering a finite area. For the classification of new example, in some of the cases rule determines the class and in others, nearest exemplar is used. The nearest exemplar is determined using Euclidean distance ($Euclid_{Ef}$) calculated by formula in equation 3.7 [Zaharie et al., 2011].

$$Euclid_{Ef} = W_f \sqrt{\sum_{i=1}^m \left(W_i \frac{E_i - f_i}{max_i - min_i} \right)^2} \quad (3.7)$$

Where, E_i and f_i are the i^{th} feature value in example and exemplar, respectively with exemplar weight W_f and feature weight W_i .

3.3.3.2 JRip

JRip is a method of extracting rules from data. JRip is based on Repeated Incremental Pruning to Produce Error Reduction (RIPPER). In the initialization phase, the initial rule set is obtained using incremental reduced error pruning (IREP) algorithm [Cohen, 1995]. The rules are added repeatedly and pruned in the building stage. The training records covered by the rule are removed and the process is repeated until there is no positive example or the rule searched by the IREP is having unacceptable error rate [Cohen, 1995].

3.3.4 Majority vote based learning model

It has been observed that different learning models result into complimentary information about classification of web services. By combining significantly different learning models, more accurate classification decisions are obtained [Kuncheva, 2004]. Voting is one such technique useful to improve the classification by combining individual opinion to derive a consensus classification decision. For L number of learners, and k number of output classes, the output of

learners is represented using k -dimensional binary vector $[c_{i,1}, c_{i,2}, \dots, c_{i,k}]^T \in \{0,1\}^k$, for $i=1, \dots, L$. The ensemble decision of the class ω_j will be resulted from the majority vote using equation 3.8 [Kuncheva, 2004].

$$\sum_{i=1}^L d_{i,j} = \max_{x=1}^k \sum_{i=1}^L c_{i,x} \quad (3.8)$$

The vote based combiner scheme will produce an accurate classification if at least $\lfloor \frac{L}{2} + 1 \rfloor$ learners produce a correct classification [Kuncheva, 2004].

3.4 ANALYSIS AND OBSERVATIONS

In this section, the details of environment used to perform various experiments are discussed. The analysis on the results and observations drawn from the results of experiments are presented.

3.4.1. Experimental setup

In order to classify web services, the experiments are conducted on a machine with Intel Core i7 CPU 3.4 GHz, Windows 7 platform. The QWS dataset with 364 labeled web services with nine QoS parameters along with web service relevancy function (WsRF) score are used to conduct the experiments. The details of QWS dataset are presented in Table 3.2. The QoS values of real world web services are measured by [Masri and Mahmoud, 2007] using a benchmarking tool. The WsRF score indicate the rank of web service quality calculated using six QoS parameters shown in Table 3.2. Based on the QoS values, [Masri and Mahmoud, 2007] have formed four service classes. Services with highest quality offerings have been kept in platinum category and the services with lowest QoS have been kept in bronze category. Gold and silver classes represent the intermediate levels of QoS. The class of the service represents the QoS level of the service. The dataset contains web services for, travel domain, hotel domain, smart payment gateways, weather and temperature determination, etc.

The empirical study for performance evaluation of six classifiers is performed using three cases. In case-1, web services with nine QoS parameters are considered. In case-2, to evaluate the effect of WsRF score, nine QoS parameters along with corresponding WsRF scores are considered. In case-3, the effect of feature selection is observed by choosing five QoS

parameters (out of total nine parameters) using Wrapper method based feature selection

Table 3.2: Labeled QWS dataset with class descriptions

Nine QoS parameters	WsRF	Class
Response Time (RT), Availability (AV), Throughput (TP), Successability (SU), Reliability (RL), Compliance (CM), Best Practices (BP), Latency (LT), and Documentation (DO)	A Web Service Relevancy function evaluated using six QoS attributes cost, RL, AV, TP, Accessibility and interoperability analysis.	(1) Platinum (high), (2) Gold, (3) Silver and (4) Bronze (low)

technique In this also, we performed experiments using these chosen five QoS parameters and corresponding WsRF scores. Same environment is used to conduct experiments for three cases. In each of the three cases, learning models are evaluated using seven performance parameters as detailed in Table 3.3.

Table 3.3: performance evaluation parameters

Parameter	Significance for web service classification
Accuracy	The measure of the quality or state of being correct in classification of web service.
Kappa statistics	Measure the classifier performance using expected accuracy and the observed accuracy of classification.
Precision (P)	Precision is the probability of a positive prediction being correct.
Recall (R)	Measures the proportion of web services belonging to the positive class and were correctly predicted as positive.
F-Measure	F-measure is the harmonic mean of precision and recall and evaluated as: $F\text{-measure} = \frac{2 * P * R}{P + R}$
AAE	Average of difference between actual class and predicted class of web service over all web service instances.
ARE	Average of ratio of difference between actual class and predicted class of web service to actual class for all web service instances.

3.4.2. Analysis of six learning models

An analysis of six learning models in each of the three cases is done. The error rate (AAE and ARE) measures are used to analyze six learning models as shown in Table 3.4. Table contains median values of absolute and relative error. In case-1, LR, NNge and J48 models perform

better than MLP, RF and JRip models in their respective category. For case-2, NNge model produces low prediction error and is slightly better than JRip in rule based category. J48 produced low error rate values as compared to RF model and LR has relatively low prediction accuracy than MLP in function based category. In case-3, LR, NNge and J48 models exhibit better prediction accuracy as compared to MLP, RF and JRip techniques, respectively in their categories.

Table 3.4: AAE and ARE analysis of six learning models for web service classification

		Case 1		Case 2		Case 3	
		AAE	ARE	AAE	ARE	AAE	ARE
Function based	LR	0.09	24.9	0.034	9.36	0.0167	4.61
	MLP	0.106	29.2	0.04	10.96	0.0227	6.29
Tree based	J48	0.163	45.0	0.012	3.40	0.0096	3.018
	RF	0.167	46.1	0.041	11.20	0.025	6.92
Rule based	NNge	0.140	35.6	0.012	3.018	0.0096	3.0175
	JRip	0.182	50.4	0.026	3.64	0.0096	3.0175

The box-plot analysis of AAE and ARE measures is conducted and presented in Figure 3.1. From box-plot analysis we measure the variation in samples for AAE and ARE values and few observations are drawn:

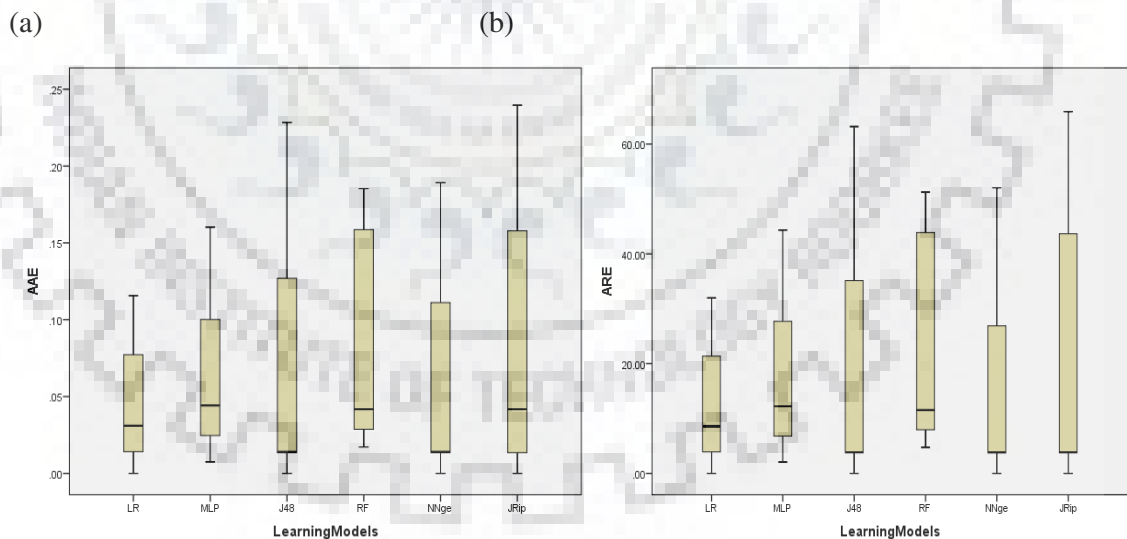


Figure 3.1: Box-plot analysis for AAE and ARE measures

- For AAE measure, NNge, LR and J48 decision tree model produced lowest minimum value of errors in their respective categories. Maximum value of LR model is minimum among all

models, whereas, JRip and J48 produced highest maximum value.

- For ARE measure, the minimum values are lowest for LR, J48 and NNge learning models. The MLP produced moderate and RF produced highest minimum value. J48 and NNge model produced lowest median value. The maximum value for ARE measure is minimum for LR model.
- Overall, for AAE and ARE values the LR model have performed better than any other learning model used for web service classification. In tree based category, J48 decision tree performed better than RF model. Similarly, NNge has outperformed over JRip in rule based category.

We have conducted Tukey post-hoc test to further analyze six learning models on AAE and ARE measures, for case-1. Similar analysis is obtained for case-2 and case-3. Tukey test is a post-hoc test to compare means of every treatment to the means of every other treatment. Based on comparison, Tukey test determine which mean values are significantly different. The mean value and standard deviation obtained for AAE and ARE measures are summarized in Table 3.5. During the test, significance value α is set to 0.05, i.e., 95% confidence interval.

Table 3.5: Tukey test for six web service classification models on AAE and ARE measures (95% Confidence Interval for Mean)

Learning Model	Performance Measure			
	AAE		ARE	
	Mean	Std. Deviation	Mean	Std. Deviation
LR	0.09	0.01498	24.88	4.13
MLP	0.1057	0.03296	29.22	9.10
J48	0.1626	0.01836	44.96	5.06
RF	0.1665	0.04614	46.03	12.78
NNge	0.1412	0.03510	35.58	9.64
JRip	0.1823	0.03836	50.38	11.17

We observed from Table 3.5 that,

- For AAE measure, LR model has lowest mean value and outperformed all other learning models. The JRip model has shown worst performance with respect to AAE measure.
- For ARE measure, LR model has achieved best performance and the worst values for performance are noted for JRip and RF model.
- Overall, the LR model is observed to have best performance. NNge and J48 models produced better prediction accuracy in their respective categories.

3.4.3. Evaluation of majority vote learning model

The majority vote model achieves a higher level of diversity by combining significantly different classifiers [Qamar et al., 2015]. From the analysis in previous section, we chose to combine top three models from each of the rule based, function based and tree based category, using majority vote model. For getting best from majority vote model, three base learners should be significantly different. In order to ensure this, we have performed Sheskin multiple comparison test on three base learners [Sheskin, 2011]. To compare multiple groups of alternatives and determine whether any group perform significantly different from the other group, Sheskins multiple comparison test is useful [Sheskin, 2011]. Sheskin multiple comparison test evaluate each pair of learning model with minimum required difference (MRD) (refer [Sheskin, 2011] for MRD evaluation). The result of multiple comparison test is shown in Table 3.6. From Table 3.6 we observed that three learning models, NNge, J48 decision tree, and LR are significantly different from one another.

Table 3.6: Sheskin multiple comparison test

Learning model	Case 1 (MRD 7.0417 %) DF=2	Case 2 (MRD 2.7903 %) DF=2	Case 3 (MRD 2.5988%) DF=2
(1) J48	(2)(3)	(2)	(2)(3)
(2) LR	(1)(3)	(1)(3)	(1)(3)
(3) NNge	(1)(2)	(2)	(1)(2)

The numbers in pair represents learning models which are significantly different from the learning model in column 1. Two models are significantly different if their absolute difference in proportion is greater than MRD. DF=Degree of Freedom.

The majority vote method uses three chosen learning models as base learner to categorize web services. The AAE and ARE measures for NNge, J48, LR and the majority vote method are measured separately and summarized in Table 3.7. We observed from Table 3.7 that AAE

Table 3.7: AAE and ARE measure of LR, NNge, J48 and majority vote learning models

	Case	LR	NNge	J48	Maj. Vote
AAE	#1	0.090	0.14	0.163	0.085
	#2	0.034	0.0124	0.0124	0.0096
	#3	0.0167	0.0096	0.0096	0.0096
ARE	#1	24.9	35.6	45.0	23.44
	#2	9.36	3.018	3.40	2.63
	#3	4.611	3.017	3.018	2.63

measure of majority vote is lowest in all three cases. Similarly, for ARE measure, the majority vote method outperform over other models in all three cases.

In order to test the performance of the majority vote method in comparison with three best performing models, we have conducted the Dunnett's test using AAE and ARE measures. Dunnett's test is a special case of Tukey test in which a pair wise comparison of single control group is made with multiple groups. The interval plot shown in Figure 3.2a and 3.2b, determines the difference of mean based on AAE and ARE measures between LR, NNge, J48, and majority vote, respectively. From Figure 3.2a and 3.2b, we observed that the use of majority vote technique reduces AAE and ARE measures of web service classification. Thus, majority vote performed better than LR, NNge and J48 model for classification of web services.

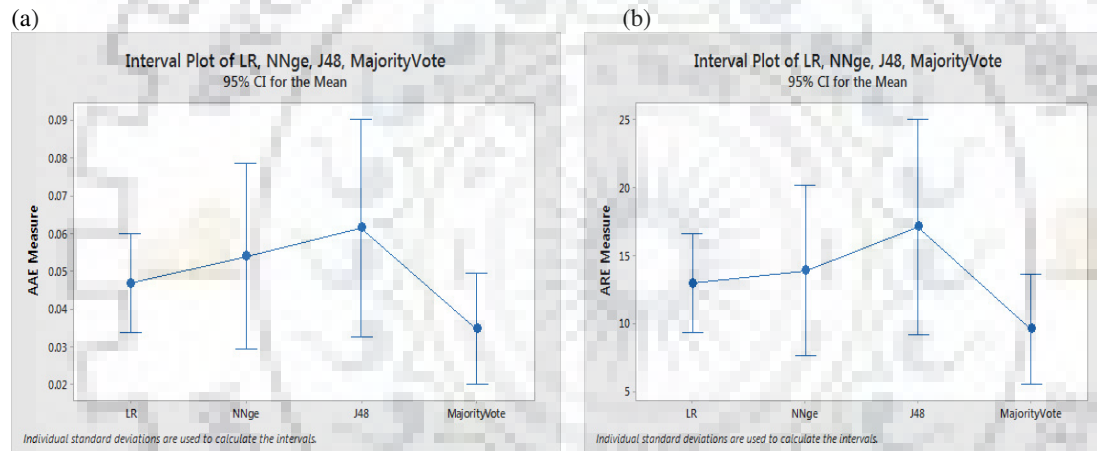


Figure 3.2. AAE and ARE based interval plot for difference of mean between LR, NNge, J48 and majority vote for classification of web services with confidence interval of 95% .

3.4.4 Discussion

With the enormous increase in number of web services, application of classification of web services has attracted a significant amount of research. Most of the earlier works on web service classification focused on web service classification using single classifier. In this chapter, we investigated the use of majority vote method for classification of web services using QoS information. An empirical study of six learning models used for web service classification is performed. As a result of empirical study, three best performing models, LR, NNge, and J48 decision tree, are chosen. A comparative analysis of individual classifier is done with majority vote learning model, using error rate measure. The result of empirical study

showed that the majority vote method produces improved results for web service classification in comparison with LR, NNge, and J48 decision tree classifiers.

In this work, a popular QWS dataset of real world web services is employed for performing experiments. The QWS dataset includes set of services from different domains, using which various applications can be developed.

3.5 THREATS TO THE VALIDITY

In this section, possible threats which may affect the validity of our experiments are discussed.

We evaluated the validity of six learning models used for web service classification in terms of accuracy and error rates using only single web service dataset. To the best of our knowledge, no other QoS based web service dataset with class labels assigned to each of the web service is available.

The QoS values are assumed to be static which does not change with time. However, in the real world scenario the QoS parameters of web services may vary continuously and may have effects on the system.

From among the selected web service classification models, few of the models such as SVM require tuning of kernel related and other control parameters. Various parameters of learning models are tuned to best of our knowledge and efforts. However, with the use of different sets of QoS parameters and change in the set of web services, further tuning of parameters for learning models may be required.

3.6 CONCLUSIONS

In this chapter, an empirical study of learning models based on error rates and other performance parameters has been conducted. We have considered six learning models LR, MLP, NNge, JRip, J48, and RF for evaluation. Based on AAE, ARE measures and various statistical tests, the top three models for web service selection are - LR, NNge and J48 in the order of decreasing performance. Based on the empirical study the chapter concludes that the majority vote technique further improves the classification accuracy by reducing error rates. Thus, majority vote based learning model can also be used for web service classification and subsequently web service selection decision can be improved.

The results obtained from this analysis are further used to build an approach for selection of WSs. The majority vote classifier helps in classifying WSs and hence identifies similarity among WSs. The results of this chapter help in identifying base learning techniques for majority vote classifier to efficiently classify WSs. In the next chapter, the results of this chapter are utilized and a two layer service selection model is developed. The work presented in this chapter has been published as [Purohit and Kumar, 2018b].



CHAPTER 4

CLASSIFICATION BASED APPROACH FOR WEB SERVICE SELECTION

In Chapter 3, an analysis of WS classification using various learning models is presented. Based on the analysis it is found that the performance of three learning models - logistic regression (LR), Non-nested generalized Exemplars (NNge), and J48 is found better. Further, it is found that the majority vote based learning model is performing relatively better and hence we have used it in this chapter. Objective of this chapter is to present an efficient web service selection approach by using learning model for prefiltering of WSs. Also, to develop an approach to automatically evaluate weight values of QoS parameters. The PROMETHEE Plus method is proposed to perform selection and maximizing deviation method is utilized for evaluation of values of weights of QoS parameters. The problem of WSS is introduced in Section 4.1. Proposed approach and motivating example appears in Section 4.2. In Section 4.3, the developed approach is evaluated and compared with existing state-of-the-art on four performance measures. The discussion in detail is presented in Section 4.4 followed by chapter conclusions in Section 4.5.

4.1 INTRODUCTION AND MOTIVATION

Selection of most promising Web Service (WS) satisfying needs of the end user is a core and challenging task in Service Oriented Architecture. The task of selection is usually performed in two steps. First, from the WS pool, services offering identical functionality are fetched. In this regard, existing solutions such as AI planning based methods [Hwang et al., 2015] or semantic based approaches [Kumar and Mastorakis, 2010, Purohit and Kumar, 2016a, Kumar and Mishra, 2008a] are efficient. Next, from the list of functionally similar WSs, a WS is to be selected. Usually, QoS parameters are used to differentiate among functionally equivalent WSs. The QoS parameters such as availability, response time, reliability, throughput, documentation, etc., are among the few popular QoS parameters [Masri and Mahmoud, 2007]. The QoS parameters collectively characterize the quality a service offers. The web services with same

functionality may exhibit different quality [Atrey et al., 2008]. The process of creating the relative ranking of WSs based on QoS parameters can be reduced to the problem of decision making. The Multi Criteria Decision Making (MCDM) methods are suitable to solve such problems [Brans and Vincke, 1985]. The Preference Ranking and Organization METHod for Enrichment Evaluation (PROMETHEE-II) is a popular MCDM method used in past for WSS [Brans and Vincke, 1985, Herssens et al., 2008]. There are other available approaches for efficient WSS such as using the Probabilistic QoS based WSS [Hwang et al., 2015], Skyline technique [Ouadah et al., 2015, Rhimi et al., 2015, Mobedpour and Ding, 2013], Utility function based selection [Alrifai et al., 2011], Mix Integer Programming (MIP) [Saleem et al., 2015, Cho et al., 2016], Case Based Reasoning (CBR) [Renzis et al., 2016], Collaborative filtering [Zheng et al., 2011], and service bundling [R.-Zapata et al., 2011, Alrifai et al., 2011, R-Zapata et al., 2015].

The QoS parameters are conflicting and difficult to compare. These conflicts among QoS parameters lead to interdependency between them, which cannot be ignored during service selection. Existing solutions [Hwang et al., 2015, Rhimi et al., 2015, Saleem et al., 2015, Zheng et al., 2011] lack in terms of considering the conflicting nature of QoS parameters, which should be included in the selection step to produce accurate results. Moreover, the performance in terms of time of the existing techniques including available MCDM techniques [Brans and Vincke, 1985, Herssens et al., 2008, Karim et al., 2011] deteriorates as more and more functionally equivalent WSs are made available. Because, more efforts are required for selection of WS and hence overall performance of WSS system degrades. To improve the performance of WSS system, the selection is done in two steps [Ouadah et al., 2015, Mobedpour and Ding, 2013]. Firstly, WSs having higher end QoS is obtained using a skyline technique. The skyline services are prioritized in second step. Nonetheless, the skyline technique has its own issues [Rhimi et al., 2015]. The skyline approach select same list of services irrespective of the end user requirements. It causes the same set of services to be presented before different end users, independent of their QoS requirements. Therefore, in addition of ignoring the end users requirements, it also causes problem of imbalance of load on a specific set of services [Wang et al., 2016]. The end users requirements during evaluation are also ignored in existing MCDM based WSS models [Herssens et al., 2008, Ouadah et al., 2015]. Further, the end user specifies importance of QoS by using weight values. The existing solutions such as [Hwang et al., 2015, Alrifai et al., 2011, Mobedpour and Ding, 2013, Cho et al., 2016] allows end users to provide numerical values to represent the QoS importance. The numerical value of weights are difficult to comprehend for a naïve user. Also, the use of

techniques such as AHP [Ouadah et al., 2015], ANP [Masri and Mahmoud, 2007], etc., for QoS weight evaluation, require input from domain experts. It makes the WSS system domain dependent and inflexible.

Based on the gaps discussed above, the research problem can be summarized as follow:

Research problem: Let, $S = \{S_1, S_2, \dots, S_n\}$: Set of 'n' discrete web services offering same functionality; $QV = \{qp_1, qp_2, \dots, qp_m\}$: represents a QoS vector with set of 'm' conflicting QoS parameters for decision making; WS_{QoS} : is a matrix representing the set of all QoS parameters for S; $User_{QoS}$: is a Web Service representing the QoS requirements of the end user. The challenge is to select most optimal WS S_r from the set S such that $S_r \equiv User_{QoS}$ with three considerations. Firstly, the size of set S is increasing as more and more WS providers are offering WSs with the same functionality. Secondly, the QoS parameters are conflicting in nature. Third, the system should support a mechanism to evaluate weight of QoS parameters using mathematical model. Moreover, the weight evaluation mechanism should allow the end user to specify partial preference of QoS. Therefore, an efficient technique is needed to solve the problem of WSS with due concern to above three criteria. Selection of top-k services require identification of K such services which closely meets $User_{QoS}$.

Research solution: The conflicting nature of QoS parameter can be considered during service selection step. A PROMETHEE based solution can be helpful in providing consideration to conflicting nature of QoS parameters [Brans and Vincke, 1985]. To improve the end user satisfaction of QoS, PROMETHEE method for WSS can be modified in three ways. Firstly, the end user request of QoS can be included during the evaluation phase of PROMETHEE. Second, those WSs which are close to $User_{QoS}$ can be identified. Third, the choice of evaluation function can be made as per the type of QoS parameter. To improve the performance of service selection system, functionally similar WSs can be preprocessed. Two conditions must be satisfied by the preprocessing based WSS system. Firstly, the end user requirements must be considered to avoid non-compliance of the user requirements. Secondly, the preprocessing system should be independent of selection system and should filter out only non-eligible services. Eligible WSs should remain in the system for further processing by selection module. PROMETHEE method offers flexibility to specify weight of QoS parameters externally. MDM based mathematical model is useful to evaluate weight values [Chen et al., 2010]. The weight preference specified by the end user can be combined with the QoS weight obtained from mathematical model.

The motivational example to better explain the motivation underlying our proposed design is discussed in Section 4.2. Following are contributions of this chapter:

1. For ranking of web services with due consideration to conflicting QoS parameters, an improved PROMETHEE approach called as PROMETHEE Plus is proposed. This improves the end user satisfaction and also the quality of WS(s) to be selected.
2. A hybrid QoS weight evaluation scheme based on Maximizing Deviation Method (MDM) is suggested to be used with PROMETHEE Plus method. It causes the WSS system to be domain independent.
3. Classification based prefiltering technique is developed to improve the performance of WSS algorithm.
4. Statistical analysis is done to observe the system behaviour and compared with existing approaches of WSS.

Performance of the proposed system is measured using Euclidean distance, query hardness, satisfaction score and time for web service selection. The experimental analysis performed using these parameters shows that the proposed CSS approach has outperformed over existing similar approaches. It is observed that the use of PROMETHEE Plus approach provides improvement in the end user satisfaction by including the end user request during service evaluation and handling the conflicting QoS parameters. Further, the use of classification based prefilter layer improves the performance of WSS system and reduces overall selection time. The use of MDM based weight evaluation approach makes WSS system more robust and domain independent.

4.2 PROPOSED APPROACH FOR WEB SERVICE SELECTION

In this section, initially a motivating example is discussed. Subsequently, the proposed system architecture and algorithms are elaborated in detail.

4.2.1 Motivating Example

With the fast proliferation of WS based technologies, WSs offering replicated functionality are continuously increasing. For the set 'S' of 'n' discrete services, the service selection module compare and rank WSs based on their QoS information. During the selection phase, all of the 'n' WSs each of which have 'm' QoS parameters are considered. As number 'n' increases, more efforts are required in the selection phase and efficiency of selection process is affected. By including a preprocessing step before the selection phase, the service selection time can be

reduced significantly [Ouadah et al., 2015]. Moreover, the conflicting nature of QoS parameters also has an effect on the quality of services being selected [Yu and Bouguettaya, 2012]. Therefore, the selection decision should also take into account the conflicting nature of QoS parameters. It was the major motivation of using classification based prefiltering.

Further, the existing systems for QoS based WSS require minimum two user inputs: (1) QoS preference (2) Weight of QoS parameters. In most of the cases, end user experiences problems in specifying (1) and (2). The end user can form the query to specify the QoS preference by using user centric system [Mobedpour and Ding, 2013] or using the QoS browser [Ding et al., 2009]. However, weights of QoS parameters need to be judged by the end user. The imprecise specification of QoS weights may result into missing the user desired services.

The significance and effect of weights of QoS parameters for WSS can be understood with the help of an example. Consider that there are 10 candidate WSs, S_1 to S_{10} , with given cost and response time (RT in msec) as shown in Table 4.1. The cost is measured on the scale of 1 to 10 and RT on the scale of 0 to 1. Lower the values of cost and RT better is the service quality. Consider three user requests arrives in the system with query “Search the WS with $\text{Cost} \leq 6$ and $\text{RT} \leq 0.5$ with QoS weights as shown in Table 4.2. In this case, QoS requirements of three users’ are same. The response to the query contains WSs satisfying the user query also shown in Table 4.2. The list of WSs satisfying user query is determined using Simple Additive Weight (SAW) [KalisZewski and Podkopaev, 2016]. Although, QoS requirements for all three users are same, but they differ in weight values of QoS parameters. The list of WSs satisfying the end user request is different for all three users. User-1 and User-2 minutely differ in their QoS weight values, but the order of services returned to them is different (service S_2 and S_6). For User-3, weights of cost and RT parameters are largely different as compared to User-1 and 2.

Table 4.1: Example WSs With Cost And Response Time QoS

QoS	Web Services									
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Cost	3	5	9	3	5	4	7	6	1	8
RT	0.95	0.4	0.37	0.93	0.6	0.97	0.96	0.83	0.9	0.45

So, the resultant list of WSs is very different for User-3. Hence, it shows that the response of WSS system is largely dependent on weight-values of QoS specified by the end user. We could analyze from this example that the performance and results of WSS method are highly dependent on the QoS weight values.

Table 4.2: User specified QoS parameters and their weights

Users	Weight of QoS		WSs returned in response to query (SAW)
	Cost	RT	
User-1	0.35	0.65	S ₉ , S ₄ , S ₁ , S ₆ , S ₂ , S ₅
User-2	0.37	0.63	S ₉ , S ₄ , S ₁ , S ₂ , S ₆ , S ₅
User-3	0.11	0.89	S ₂ , S ₉ , S ₅ , S ₄ , S ₁ , S ₁₀ , S ₆ , S ₃ , S ₈ , S ₇

4.2.2 Web Service Selection Approach

In the present scenario, the WSS system performs the task of selection in two steps. In first step, the relative ranking of WSs is done. In second step, the selection is accomplished from the ranked list of WSs. As discussed in Section 4.1, the basic PROMETHEE method is helpful in obtaining QoS based ranking of WSs. In this work, we have proposed three variations of WSS system. All three variations are based on an improved PROMETHEE method, PROMETHEE Plus, for ranking followed by top-k selection of WSs. The PROMETHEE Plus method is derived from original PROMETHEE with three modifications. First, the basic PROMETHEE algorithm for WSS is improved by including the end-user's request of QoS as part of PROMETHEE evaluation. This improvement will ameliorate the ranking results by segregating the WSs into two groups. One group will include WSs which exactly satisfy the required QoS and/or have QoS close to the required QoS. The other group includes the WSs having QoS below the QoS expectations of the end user. Second modification is done as part of ranking process. The higher preference is given to the WS with QoS score closely matching with the requested QoS. This will increase the user satisfaction in using the system [Masri and Mahmoud, 2007]. Third, the Gaussian and level type preference functions are used during QoS parameters evaluation. PROMETHEE Plus is discussed in more details in Section 4.3.2.

In the traditional WSS system, user could specify the preference of QoS parameters using weight values. The user must provide the precise weight values for all QoS parameters. However, in actual practice, two variations are possible. Firstly, no weights of QoS parameters are provided by the end user. Secondly, partial weights are specified by the end user. In second variation, system should be flexible enough to incorporate the weight values provided by the end user/expert and weights calculated using mathematical model [Chen et al., 2010]. In the proposed approach, the first variation is realized by using MDM method for weight evaluation in presented PROMETHEE Plus based WSS (MSS) and is shown in Figure 4.1. A hybrid scheme that combines the user specified weight preference with the output of MDM method can be used to support partial preference of weights from the end user. This variation using

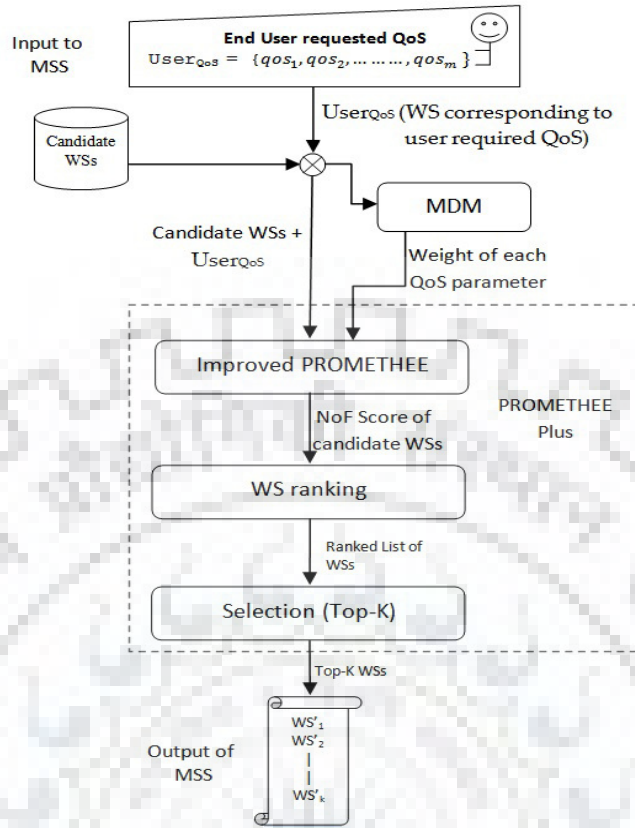


Figure 4.1: Proposed system architecture for MSS approach

Hybrid scheme of weight evaluation in PROMETHEE Plus method for WSS (HSS) is presented in Figure 4.2. Proposed approaches, MSS and HSS, are practical approaches and useful in scenario where user is a naïve and is unable to specify the QoS weight values.

Moreover, if candidate WSs are very large in number, the complexity of the system will also be high. Therefore, a mechanism such as prefiltering of services using classification is proposed as a third enhancement. The Classification based prefilter with PROMETHEE Plus for WSS (CSS approach) is obtained by adding a prefilter layer over HSS approach. The details of CSS approach is discussed in next section. A summary of three variations for PROMETHEE Plus based WSS is presented in Table 4.3.

Table 4.3: Summary of three variations of proposed WSS approach

Approach	Features
MSS	Weight evaluation using mathematical model, PROMETHEE Plus.
HSS	Hybrid weight evaluation, PROMETHEE Plus.
CSS	Prefiltering based on classification, Hybrid weight evaluation, PROMETHEE Plus.

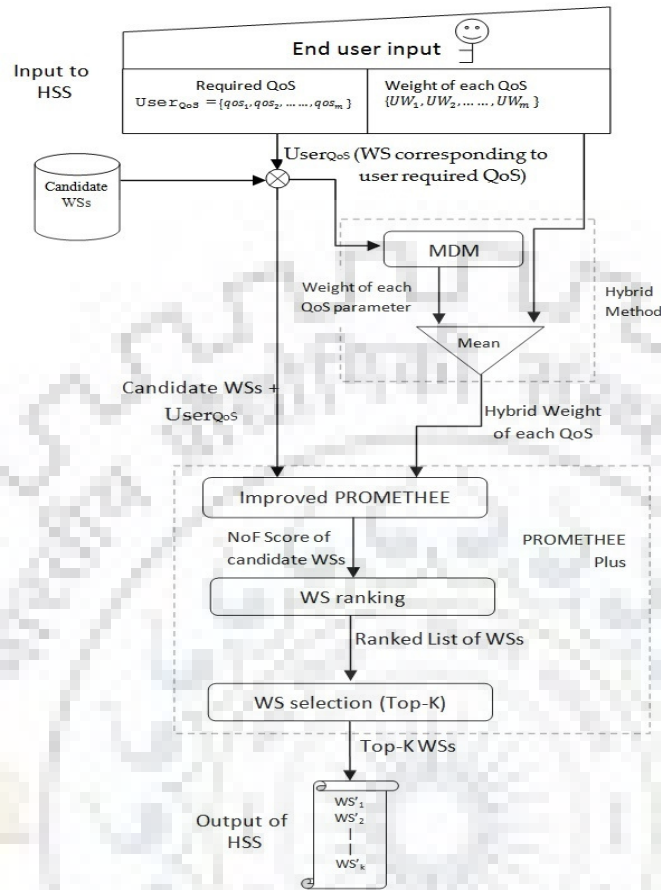


Figure 4.2: Proposed system architecture for HSS approach

4.2.3 The proposed CSS approach for Selection of Web Service

Web service can be labelled using the associated QoS information. Each label associated to the web service represents a pre-defined category. This categorization leads to identification of WSs with similar QoS offerings. The process is referred as classification of WSs. The process of classification starts by creating the learning model using a set of labelled WSs. In the later stage, built model is used to categorize unlabelled WSs. The process of classification of web services helps in reducing the search space. The system architecture of proposed CSS approach is shown in Figure 4.3. It consists of two layers. The upper layer is a prefilter layer and bottom layer is selection layer. The aim of prefilter layer is to filter out irrelevant web services and keeps only those WSs capable of meeting the QoS needs of the end user. In order to achieve the task of prefiltering, user requirements of QoS is collected and functionally similar candidate WSs are retrieved from WS-database. Functionally similar web services can be made available

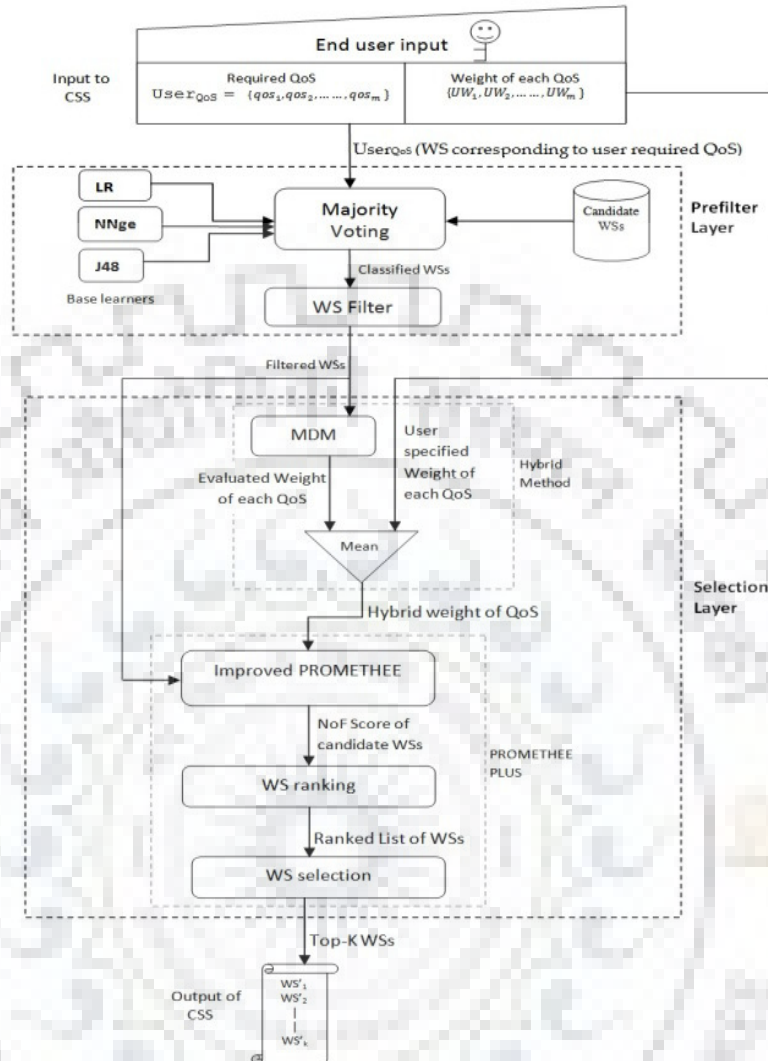


Figure 4.3: Proposed system architecture for CSS approach

using any of the available approaches such as [Kumar and Mastorakis, 2010, Purohit and Kumar, 2016a, Kumar and Mishra 2008a] or others.

The prefilter layer uses classification for filtering web services. To perform classification task, the majority vote based technique with Logistic Regression (LR), Non-nested generalized exemplars (NNge), and J48 decision tree as base learner is used. This step takes constant time and act as a prefilter for functionally similar web services. The filtered set of WSs is passed to the selection layer for further processing. At selection layer, the hybrid method evaluates weight of each QoS parameter using filtered set of WSs. Next, the proposed PROMETHEE Plus method is applied to obtain a relative ranking of WSs by taking the end user QoS

specifications as reference. The web services are sorted on the score obtained from PROMETHEE Plus method and top-k web services are selected by the system. Selected top-k web services consist of two types of web services – exact match, and closest match. In this way, number of web services to be processed at selection layer is reduced due to prefilter layer. Only relevant WSs are processed which ensure improved performance of selection in terms of time and quality of selection. The algorithm for CSS approach based selection of top-k web services is shown in Figure 4.4. The call to CSS algorithm is placed by passing five parameters as input. The CSS algorithm in turn calls prefilter function to prefilter large list of candidate WSs (line 2). The filtered WSs are passed to selection layer for ranking and selecting top-k WSs (line 3). Finally, the top-k WSs obtained as a result of selection are return back to the end user (line 4).

In next two sections, details of CSS approach are discussed. The prefilter layer and selection layer along with the details on PROMETHEE Plus are presented.

4.2.3.1 Prefilter Layer

The prefilter layer performs the task of prefiltering by ensuring that irrelevant web services are removed from the set of web services. Only those services which are promising enough to meet the end user expectations of QoS are passed to the selection layer. For applying the prefiltering, we are considering that all the input web services to the prefilter layer are functionally equivalent. So, the prefiltering step is not considering the functional parameter but only the QoS parameters. The upper layer performs prefiltering using majority vote based classification [Kittler et al., 1998]. In classification, a set of systematic steps is followed to create a model to predict the class of data object whose class is unknown [Saleem et al., 2015]. For this purpose, it uses a training data set (objects whose class label is known). The labeled QWS dataset [Masri and Mahmoud, 2007] of 364 WSs is used as training dataset. For WSS, the classification is performed on candidate web services using the QoS information. The candidate web services along with user requests ($User_{QoS}$) are classified to four predefined classes - Platinum (highest quality), Gold, Silver and Bronze (lowest quality) [Masri and Mahmoud, 2007]. The classification step results into groups of WSs, with services belonging to same group are assigned same identification number (class). The class to which the $User_{QoS}$ is classified is tracked and all of the candidates WSs classified in that class are determined. All these identified WSs have approximately analogous QoS requirements as that of $User_{QoS}$. The steps of the proposed approach for the prefiltering are summarized below:

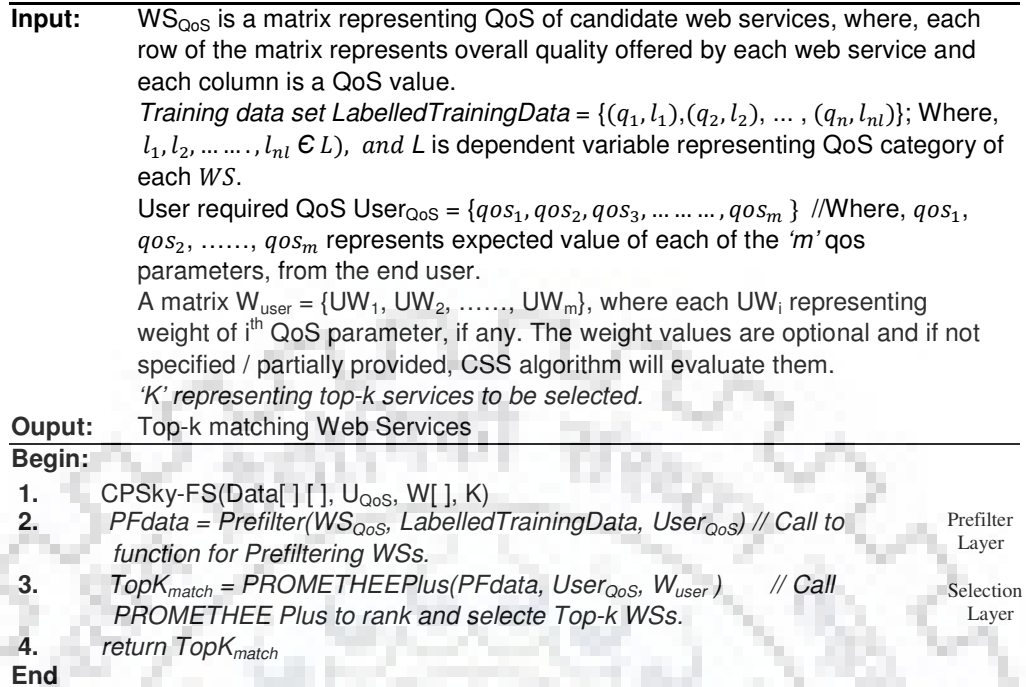


Figure 4.4: CSS algorithm for web service selection

- i. Accept functionally similar WSs (unlabelled WS instances) and QoS need of the end user as input.
- ii. Create majority vote based combiner using LR, NNge, and J48 decision tree as base learners and train the classifier using labelled dataset of WSs to generate model *Model*.
- iii. Using trained model *Model*, classify unlabelled WS instances and the service represented as $User_{QoS}$.
- iv. Identify web services classified in the same group to which the service represented by $User_{QoS}$ is classified.
- v. All the candidate services in this group (*matchServiceDataSet*) identified in step 4, have QoS matching with the QoS requested by the end user.
- vi. All these services in the group *matchServiceDataSet* are passed to selection layer for further processing and WSs in the other non-matching classes are simply discarded. In this way, the bigger set of candidate WSs is filtered and WSs closely matching with QoS requirements of the end user are identified.

The web service prefiltering algorithm based on classification is presented in Figure 4.5. The algorithm starts by creating an object of type vote to represent majority vote based method with J48, NNge, and LR as base learner (line 2). 10 fold cross validation is applied to train and validate the model (line 3). To generate the training data, stratified sampling is used. The trained model is used to classify instance of unlabelled web service dataset WS_{QoS} and $User_{QoS}$

(line 4-6). Those services are identified from the list, after classification, whose label matches with the label of $User_{QoS}$ (line 7-10). The resulting filtered dataset includes most promising set of WSs whose QoS offerings are closely matching with the expected $User_{QoS}$.

Input: The matrix representing QoS of candidate web services WS_{QoS} , where each row of the matrix represents quality offerings of a web service and each column is a QoS value.
Training data set $LabelledTrainingData = \{(q_1, l_1), (q_2, l_2), \dots, (q_n, l_{nl})\}$;
 $l_1, l_2, \dots, l_{nl} \in L$, Where, L is dependent variable representing QoS category of WS.
 User required QoS $User_{QoS} = \{qos_1, qos_2, qos_3, \dots, qos_m\}$, Where $qos_1, qos_2, \dots, qos_m$ represents expected value of each of the 'm' qos parameters, by the end user.

Output: Prefiltered Web Services.

Begin:

1. *Prefilter*($WS_{QoS}, LabelledTrainingData, User_{QoS}$)
2. *Vote* $v = new Vote(J48, NNge, LR)$; // Object representing majority vote
 // is created with J48, NNge, and
 // Logistic regression as base learners.
3. *Model* = *TenFoldCV*($v, LabelledTrainingData$) // A model is built using
 // labelled training data based
 // on ten-fold cross validation.
4. *for each*(*Instance* $I \in (WS_{QoS} \cup User_{QoS})$)
5. *clDataSet* = *Model.classifyInstance*(I) // Classify each instance of
 // unlabelled web service as well
 // as the end user request.
6. *end for*
7. *for each*($d \in clDataSet - \{User_{QoS}\}$) // Identify those web services
 // categorized to the same class as that of $User_{QoS}$
8. *if*($User_{QoS}.classLabel == d.classLabel$) (*end user request*).
9. *matchServiceDataSet* = *matchServiceDataSet* \cup 'd';
10. *end for*
11. *return matchServiceDataSet*

End

Figure 4.5: Classification based Prefiltering algorithm

4.2.3.2 Selection Layer

The selection layer performs two tasks. First task is to rank WSs based on the QoS score. Second task is to select top-k WSs from the ranked list. At selection layer, proposed PROMETHEE Plus method is employed for WSs ranking. PROMETHEE Plus method is a method to evaluate best among a set of alternatives based on a number of different criteria (may be conflicting). PROMETHEE Plus is a ranking method with a set S of 'n' alternative web services to be ranked and 'm' QoS conflicting criteria that are to be optimized. The outranking flow of each of the WS is evaluated based on the conflicting QoS values. ϕ^+ and ϕ^- represents positive and negative outranking flow. The ϕ^+ , ϕ^- , and ϕ^{Net} of candidate web services are obtained using equations 4.1, 4.2, and 4.3, respectively [Seo et al., 2005, Karim et al., 2011].

$$\phi^+(S_i) = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^n PF(S_i, S_j) \quad (4.1)$$

$$\phi^-(S_i) = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^n PF(S_j, S_i) \quad (4.2)$$

$$\phi^{\text{Net}}(S_i) = \phi^+(S_i) - \phi^-(S_i) \quad (4.3)$$

Where, 'PF' is a preference function, defines a preference of service S_i over all other services S_j . The net outranking flow (NoF), $\phi^{\text{Net}}(S_i)$, of the web service determines the ranking order. Higher value of ϕ^{Net} ranks the web service higher.

The PROMETHEE algorithm [Brans and Vincke, 1985] has the flexibility of specifying weights of alternatives (in our case QoS parameters) externally. The end user may want to target a particular WS which best suit her preference. Therefore, the end user wishes to include her QoS preference into the search criteria. Hence, in the basic PROMETHEE algorithm for WSS [Herssens et al., 2008], we have included the end user QoS requirements ($User_{QoS}$) in the list of candidate web service. With this modification two segregated Set_1 and Set_2 , with $User_{QoS}$ as line of separation, are created from among the candidate WSs as follows:

$$\begin{aligned} \forall ws_1 \in Set_1, \quad \text{such that, } \phi^{\text{Net}}(ws_1) & \left\{ \begin{array}{l} == \phi^{\text{Net}}(User_{QoS}), \\ \text{Exact Matching} \\ > \phi^{\text{Net}}(User_{QoS}), \\ \text{QoS of } ws_1 \text{ is higher than expected} \end{array} \right. \\ \forall ws_2 \in Set_2, \\ \text{such that, } \phi^{\text{Net}}(ws_2) & \left\{ \begin{array}{l} < \phi^{\text{Net}}(User_{QoS}), \\ \text{QoS of } ws_2 \text{ is not matching} \\ \text{with the end user requirements.} \end{array} \right. \end{aligned}$$

The proposed CSS approach gives highest priority to WSs in the set Set_1 having exact match. All the WSs in set Set_2 are not considered by the system. Weight of the QoS parameter represents the preference provided by the end user to be considered during WSS. The weight preference for each of the QoS parameter can be obtained by using various methods such as (i) directly from the end user, (ii) domain expert (AHP/ANP), (iii) by including user QoS preference into selection criteria [Yu and Bouguettaya, 2012], (iv) group consensus [Wang, 2009], (v) using MDM [Yin et al., 2016, Chen et al., 2010], etc. Specifying QoS weights using method i, ii and iii have certain limitations. The end user may not be able to provide the QoS preference due to lack of knowledge or partial preferences is obtained. At this point, the

domain experts can provide QoS weight preference on behalf of the end user. Nonetheless, the preference input from domain experts limit the use of system for a particular domain. The skyline approach does not require any explicit weight values from user, however, the use of skyline operation for WSS can be very expensive in terms of memory usage and computational time [Yu and Bouguettaya, 2012]. The aforementioned limitations can be overcome if the system determines the weights using QoS parameters itself [Yin et al., 2016]. But, the end user preference is not considered in such approach. Therefore, we have proposed a hybrid approach which considers the end user preference as well as weights of QoS parameters obtained using mathematical model. Also, if the end user provides partial weights or no preference, the system automatically copes with the issue. Pros and cons of three weight evaluation schemes - user/expert input, MDM method and hybrid scheme are presented in Table 4.4.

Table 4.4: Comparison of QoS Weight calculation techniques

Methods to obtain QoS Weight	Weight input	User Flexibility	Difficulty in expression of weight by the user	Domain Dependent
User Specified Weighting scheme / ANP, AHP	Depends on user / expert	Yes	Not taken care (Whether 0.4 or 0.41 is appropriate)	Yes
Maximizing Deviation Method	Calculated using Mathematical model	No	Taken care	No
Hybrid Scheme	Partly dependent on user / expert.	Yes	Taken care	Flexible

The user preference of weights is stored in the weight metric $W_{user} = \{UW_1, UW_2, UW_3, \dots, UW_m\}$, Where, $0 \leq UW_i \leq 1$ and $\sum_{i=1}^m UW_i = 1$. If partial preferences of weights are provided by the user, then for the unspecified preference, the average value is obtained as follows. If $u = \sum$ user specified QoS preference, where, $u \leq 1$, and $v = 1-u$. Then,

$$\text{Preference of unspecified values} = \frac{v}{\text{number of unspecified preferences}}$$

In order to evaluate weights of QoS parameters, Maximizing Deviation Method (MDM) is used. The impact of QoS criteria in decision making can be decided theoretically by looking at the following facts. The QoS parameter will be considered less important factor to differentiate web services if the QoS values are having small differences. However, if the QoS values are having obvious differences across all the WSSs, then that QoS parameter plays a very vital role in choosing the best web service [Yin et al., 2016, Chen et al., 2010]. Subsequently, as per the concept, the MDM is applied to evaluate the weight of each QoS parameter. Assume that there are 'm' QoS parameters to consider for each web service. The deviation method is used to

compute the difference of the performance values of each web service with respect to all QoS parameters. For each QoS parameter QoS_j , $1 \leq j \leq m$, the deviation of web service S_i to all the other web services can be defined using equation 4.4, and equation 4.5 [Chen et al., 2010].

$$D_{ij} = \sum_{r=1}^n \left(\delta(QoS_{ij}) - \delta(QoS_{rj}) \right)^2 \quad (4.4)$$

and

$$D_j(w) = \sum_{i=1}^n \sum_{r=1}^n \left(\delta(QoS_{ij}) - \delta(QoS_{rj}) \right)^2 \quad (4.5)$$

D_{ij} represents the deviation value of i^{th} service with respect to j^{th} QoS parameter. The $D_j(w)$ is the deviation value of all services to other services with respect to j^{th} QoS parameter. QoS_{ij} is the normalized QoS value of the j^{th} QoS parameter of service S_i . The non-linear programming model based on the MDM is used to calculate the weight of j^{th} QoS parameter (please refer equation 4.6) [Chen et al., 2010].

$$w_j = \frac{\lambda_k * \sum_{i=1}^n \sum_{r=1}^n (\delta(QoS_{ij}) - \delta(QoS_{rj}))^2}{\sum_{j=1}^m \sum_{i=1}^n \sum_{r=1}^n (\delta(QoS_{ij}) - \delta(QoS_{rj}))^2} \quad (4.6)$$

s.t. $w_j \geq 0$, $\sum_{j=1}^m w_j = 1$. Where, λ_k represents a weight factor for k^{th} QoS parameter, provided by the expert. In our experiment we have chosen equal weightage for all QoS parameters, i.e. $\lambda_k = 1/m$. Let, $W_{\text{mdm}} = \{W_1, W_2, W_3, \dots, W_m\}$, be the set of QoS weights obtained using MDM method. Once both the sets of weights are obtained, the hybrid scheme evaluates weights of QoS parameters using set in equation 4.7.

$$W_{\text{Hyb}} = \left\{ \frac{UW_1 + W_1}{2}, \frac{UW_2 + W_2}{2}, \dots, \dots, \frac{UW_m + W_m}{2} \right\} \quad (4.7)$$

PROMETHEE Plus algorithm for WSs ranking based on NoF is shown in Figure 4.6. The PROMETHEE Plus algorithm starts by evaluating the weight of each QoS parameter using hybrid scheme (line 3). The hybrid scheme uses weight matrix obtained from Maximizing Deviation Method (line 2) and user preference of QoS weights (input to the algorithm). Once the weight of each QoS parameter is obtained, the Net outranking flow for each web service is calculated by using positive and negative outranking flow (line 4 – 8). The NoF is used to generate the relative ranking of candidate web services. During the process of ranking, the candidate web service having a NoF equal to the NoF of $User_{QoS}$, is stored in List1 (Line 9-11). The List1 stores the WSs having QoS parameters exactly matching with the $User_{QoS}$. List2 stores web services having Net outranking flow $>$ Net Outranking flow of $User_{QoS}$ and sorted from lowest to highest NoF value (line 12-13). The rest of the other web services are not

Input: *PFdata* is prefiltered set of WSs obtained after performing classification based prefiltering. User required QoS $User_{QoS} = \{qos_1, qos_2, qos_3, \dots, qos_m\}$, Where, $qos_1, qos_2, \dots, qos_m$ represents value of m QoS parameters obtained from the end user. A matrix $W_{user} = \{UW_1, UW_2, \dots, UW_m\}$, where each UW_i representing weight of i^{th} QoS parameter.

Output: Top-k ranked list of Web Services.

Begin:

```

1.  PROMETHEEPlus(PFdata,  $User_{QoS}$ ,  $W_{user}$ )
2.   $W_{mdm} = \text{MaxDevMethod}(\text{PFdata} \cup User_{QoS})$  // Weight is evaluated for
    each
    // QoS parameter using MDM.
3.   $W_{Hyb} = \text{HybridWScheme}(W_{mdm}, W_{user})$  // Hybrid scheme combines the
    // weight evaluated using MDM and
    // complete or partial preference
    // obtained from the end user.
4.  for each( $ws \in (\text{PFdata} \cup User_{QoS})$ ) //  $User_{QoS}$  Added at the end
5.   $ws.PosOR_{FLOW} = \text{FindPFlow}(ws, W_{Hyb})$  // Calculate positive
    // outranking flow
6.   $ws.NegOR_{FLOW} = \text{FindNFlow}(ws, W_{Hyb})$  // Calculate negative
    // outranking flow
7.   $ws.NetOR_{FLOW} = \text{FindNetFlow}(ws.PosOR_{FLOW}, ws.NegOR_{FLOW})$  // Find
    net
    // outranking flow using positive and
    // negative outranking flows.
8.  end for
9.  for each( $ws \in \text{PFdata}$ )
10. if( $ws.NetOR_{FLOW} == User_{QoS}.NetOR_{FLOW}$ ) // If the net outranking flow
    // of any WS is matching with
    // end user specified QoS it
    // is Exact matching WS.
11. List1.add( $ws$ )
12. else if ( $ws.NetOR_{FLOW} > User_{QoS}.NetOR_{FLOW}$ ) // WS with higher NoF
    // than NoF of  $User_{QoS}$ .
13. List2.add( $ws$ )
14. else // else WSs cannot satisfy the end user requirements,
    // hence ignore such WSs ( $Set_2$ ).
15. Continue
16. end for
17.  $WS_{match} = \text{List1.append}(\text{List2})$  // Combine exact match and partial
18. // match WSs ( $Set_1$ ).
     $TopK_{match} = \text{SelectTopk}(WS_{match}, K)$  // Top-k services to be selected
    // from ranked list
19. return  $TopK_{match}$ 

```

End

Figure 4.6. PROMETHEE Plus algorithm for ranking of filtered web services and selection of top-k web services

considered (line 14-15). A single ranked list of WS (WS_{match}) is prepared where the exact matching services are kept ahead of partial/close matching WSs (line 17). The ranked list contains two types of services. Exact match and partial/close match services. From the ranked list, top-k services are selected using algorithm shown in Figure 4.7. The exact match services present in the list are retrieved first and if required, partial match services are also obtained (line 2-4). The top-k WSs are returned by the algorithm.

Input: WS_{match} : Ranked list of web services obtained after applying PROMETHEE Plus algorithm. The Ranked list is a matrix representing quality offered by WSs and WSs in the matrix are ranked according to increasing value of net outranking flow. The exact match WSs are ranked higher as compared to partial matching WSs.
'K' representing top-K services to be selected.

Output: Top-k Web Services.

Begin:

1. $SelectTopk(WS_{match}, K)$
2. for each($i \in (1 \text{ to } K)$) // For each ranked services, consider K services
3. $TopK_{match} = TopK_{match} \cup WS_{match}[i]$ // Select top-k WSs from the ranked list
4. end for
5. return $TopK_{match}$

End

Figure 4.7: Selection of top-k web services from ranked list

4.3 EXPERIMENTAL ANALYSIS

The proposed approach is tested by conducting different experiments to evaluate the performance and effectiveness in web service selection over existing approaches. In our proposed system, we have considered that all the WSs are functionally equivalent. So, the system needs to perform WSS based on QoS requirements only. The proposed WSS approach mainly emphasizes selection of service for atomic task where service equivalence is more important as compared to compatibility. Details of design of experiments and experimental results are discussed in the following section.

4.3.1 Experimental Setup

CSS algorithm is implemented in Java and Weka API is used to implement vote based classification. Experiments related to evaluation of the proposed WSS method and its variations

are conducted on machine with Intel Core i7 CPU @ 3.4 GHz, 8GB of RAM, Windows 7 platform and Netbeans installed on it.

4.3.1.1 The dataset and design of user queries

The QWS dataset (henceforth called as Dataset-1) of real world web services is used to conduct experiments [Masri and Mahmoud, 2007]. The dataset consist of 364 labelled and 2507 unlabelled web services as discussed in Section 2.5.1 of Chapter 2. The QoS parameters are evaluated using the QWS measures obtained from Web Service Broker Framework. Nine QoS parameters are used - response time (R_{tm}), availability (A_{va}), throughput (T_{hr}), successability (S_{uc}), reliability (R_{el}), compliance (to WSDL description) (C_{om}), best practice (by following WS-I) (B_{pr}), latency (L_{at}) and Documentation (D_{oc}). The last two values in Dataset-1 represent the service name and reference to the WSDL document. The dataset with 364 web services has additional information of WsRF score and class labels. The WsRF is web service relevance function representing relevance of web services based on QoS parameter values. Using WsRF score, each of 364 web services are assigned a class label out of bronze, silver, gold and platinum. The bronze class represents web services with lowest overall QoS value. Whereas, web services of platinum class have highest QoS value.

The system for web service selection bears the responsibility of responding to user query as efficiently as possible. The WSS system responds to users differently as per the user query. In order to test the system behavior and response, hundred queries are generated randomly. Query represents QoS concerns of the end user. Each of the hundred query belongs to a hardness level from one (L_1 – least hard) to ten (L_{10} - hardest). The hardness level L_1 represents range from 1-10 (in %), L_2 has range of hardness from 11-20, and so on. One query from each level is selected and presented in the Table 4.5. The hardness of each query is obtained using equation 2.1 and equation 2.2 (in Chapter 2, Section 2.6).

To validate the results, we have also performed the experimentation on another popular dataset generated using publicly available dataset generator [Borzsony et al., 2001] (henceforth called as Dataset-2).

4.3.1.2 Performance parameters

Three performance parameters are used to evaluate proposed and existing approaches. The hardness level, satisfaction score, Euclidean distance and time for WSS parameters are used for evaluation purpose as defined in Chapter 2. We have selected these parameters because these parameters are widely used in the existing similar works for the evaluation of WSS approaches [Cho et al., 2016, Seo et al., 2005, Lim et al., 2012, Stephen and Yin, 2011].

Table 4.5: User Queries With Hardness And Hardness Level

Query	QoS Need	Hardness (in %)	Hardness Level
Q1	4950.1,10,4.5,15,40,42,56,4029.3,10	7.8	L ₁
Q2	3939.024,18,10,22,45,47,62,3601,19	18.89	L ₂
Q3	3381.3,28,17,29,48,51,64,3295,27	27.59	L ₃
Q4	3209.58,16.44,12.52,8.57,42.77,91.97,55.9,2082.35,76.69	35.77	L ₄
Q5	813.8,76.58,9.97,12.78,72.01,37.80,75.27,3595.30,71.37	44.97	L ₅
Q6	164.27,93.64,2.04,53.33,40.82,99.82,76.76,540.33,3.52	56.78	L ₆
Q7	2177.05,64.73,18.1,83.72,63.83,95.95,88.18,2034.31,54.81	65.10	L ₇
Q8	452.75,88,4.5,96,67,100,72,142.25,88	76.12	L ₈
Q9	91.4,86.8,41.9,86,78.7,85,86.8,8.4,82	88.5	L ₉
Q10	44.2,93,41,97,85,93,89,8,92	94.74	L ₁₀

In addition to service satisfaction defined using equation 2.3 and 2.4 (in Chapter 2, Section 2.6), the satisfaction score for each user query is obtained by finding weighted sum of satisfaction scores of top-k WSs as defined using equation 4.8.

$$Sat_{Query_i} = \sum_{j=1}^k wt_j * Sat_{WS_j} \quad (4.8)$$

Where, Sat_{WS_j} is the satisfaction score of j^{th} top-k service and wt_j is the weight of j^{th} top-k service obtained using rank-sum method [Hala and Mohamed, 2012].

4.3.1.3 PROMETHEE Plus parameters evaluation

The PROMETHEE Plus method has six predefined functions to evaluate QoS parameters for comparison among the candidate web services. Preference function determines preference of one web service over others. For QoS parameters which are quantitative type, such as cost, price, documentation, compliance, etc., level/linear type preference function can be used [Seo et al., 2005]. The parameters which are qualitative in nature, such as reliability, availability, etc., Gaussian type of preference function is suitable [Seo et al., 2005]. The type of function used has influence over the precision in WSS. The suitable values of preference criteria, threshold and preference/indifference are obtained after conducting experiments multiple times. The indifference parameter ' Q ' is the maximum value of difference function $d(x)$ for which alternatives are indifferent. The preference parameter ' P ' is the minimum value of difference function $d(x)$ for which the alternatives are different.

In order to obtain the optimized value of P and Q parameter, hardness and satisfaction score are used. The hardness of each of the candidate web service is determined using equation 2.1 and equation 2.2 (in Chapter 2, Section 2.6). Eight web services from the dataset of 2507 web services, one from each of the hardness level L₃ to L₁₀ are selected. These web services ensure to cover all range of QoS hardness. The optimized value of preference parameter P and

indifference parameter Q is selected for which the satisfaction score is maximized. The satisfaction score for i^{th} WS (Sat_{WS_i}) is evaluated using equation 2.3 and 2.4 (in Chapter 2, Section 2.6). Once the value of P and Q parameters is obtained, the parameter σ which represents point of inflexion for Gaussian curve is obtained as mean of P and Q parameters [Brans and Vincke, 1985]. Table 4.6 represents the values of preference (P), indifference (Q) and σ parameters obtained.

4.3.2 Results and evaluation

We have performed experimentation using Dataset-1. To further validate the results, we have performed evaluation with Dataset-2 as discussed. The results and evaluation using Dataset-1 are presented in this subsection. Results and evaluation using Dataset-2 are also discussed and presented in the next subsection. The main contribution lies in the proposed PROMETHEE Plus approach, use of MDM for weight evaluation of QoS parameters and the use of classification method for prefiltering.

4.3.2.1 Evaluation of the proposed approach using Dataset-1

The major objective of this section is to show the improvements caused by the PROMETHEE Plus, weighting method and classification method. Existing approach of BPS [Herssens et al., 2008] is representing the results of PROMETHEE and EPS [Karim et al., 2011] represents the

Table 4.6: PROMETHEE parameters evaluated based on maximum satisfaction score

Parameter	R_{tm}	A_{va}	T_{hr}	S_{uc}	R_{el}	C_{om}	B_{pr}	L_{at}	D_{oc}
P	0.91	0.81	0.91	0.81	0.81	0.11	0.81	0.11	0.41
Q	0.81	0.71	0.81	0.71	0.71	0.09	0.71	0.09	0.31
σ	0.86	0.76	0.86	0.76	0.76	0.1	0.76	0.1	0.36

results by using the AHP based methodology along with PROMETHEE method. We have compared our work with these two approaches to show the improvements. In addition, we have presented the results of each of the variations of our approach MSS, HSS, and CSS to show the improvements caused by each of the contributions. We have also performed statistical analysis to compare our results. Each of the five algorithms (BPS [Herssens et al., 2008], EPS [Karim et al., 2011], MSS, HSS, CSS) is applied on the same set of WSs to select few top most services. The parameter ' K ' is used to specify how many top WSs are to be selected. The value of parameter ' K ' is specified externally and has effect on the satisfaction of QoS needed by the end user.

The effect of variation in the value of 'K' on satisfaction score is measured for five approaches and presented in Figure 4.8. The value of K=3, 5, 10 was chosen based on the works in [Zheng et al., 2011],[Hwang et al., 2015],[Rhimi et al., 2015]. The satisfaction score for 100 user queries is evaluated. For each user query, top-k web services are selected. Using equation 2.3 and 2.4 (in Chapter 2, Section 2.6), satisfaction score of each of the 'K' selected services is evaluated and the QoS satisfaction score for each user query is obtained from equation 4.8. The top-k selected services are assigned rank based on the NoF value. Using the rank information, weight w_{t_j} of each selected service is obtained using rank sum method [Hala and Mohamed, 2012]. Y-axis in Figure 4.8 represents mean satisfaction score measured over hundred queries and variation in 'K' values is presented on X-axis. It is analyzed from the figure that the proposed approach CSS has maximum satisfaction score for each value of 'K'. The satisfaction score obtained for each of the approach is increasing with increase in the value of 'K'. The satisfaction score of EPS is observed to be minimum among all approaches for each value of 'K'.

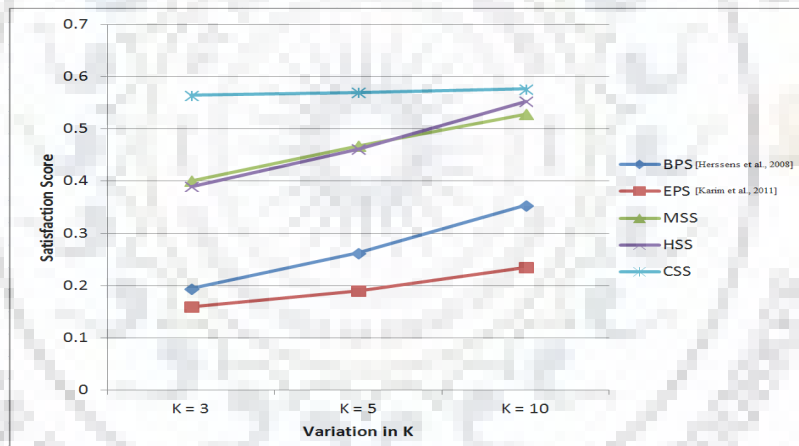


Figure 4.8: Effect of variation of 'K' on satisfaction score

Figure 4.9 shows the effect of increase in hardness of queries on satisfaction for each of the five approaches. Satisfaction score appears on Y-axis, whereas hardness levels are shown on X-axis. Experiments are conducted for obtaining top-3 web services ($K=3$). The BPS, EPS has no effect and MSS, HSS has a very little effect on satisfaction score with the increase in hardness of QoS queries. A slight improvement in satisfaction score for MSS approach over HSS is observed with the increase in hardness. The improvement in satisfaction score with increase in hardness of end user queries is observed for CSS approach. This is primarily due to the use of classification based prefiltering which identifies the closest matching set of web services during prefiltering phase.

Each of the web service selection approach may result in different selection of top-k web services. The Euclidean distance of the selected top-k web services from the end user QoS query can be used as a measure to compare the effectiveness in web service selection. If S_1, S_2, \dots, S_k are top-k web services selected by WSS approach and $User_{QoS}$ be the QoS requirements specified by the end user, then the Euclidean distance between $User_{QoS}$ and service S_i is obtained by equation 2.5 (in Chapter 2, Section 2.6).

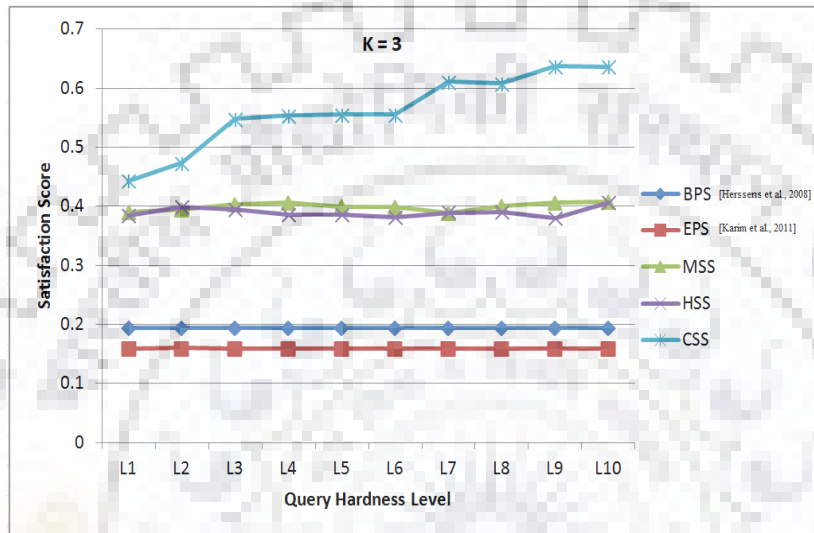


Figure 4.9: Measuring the effect of variation in the hardness of QoS query on the end user satisfaction score. Level L_1 represent least hardness and L_{10} is hardest level.

The Euclidean distance measures the effectiveness of WSS approach in selection. Lower value of Euclidean distance corresponds to more effective WSS approach. The box-plot for the mean Euclidean distance measured on each of the 100 queries for five WSS approaches is shown in Figure 4.10 (for $K=3$). It is observed from the figure that the best web service selection with smallest Euclidean distance is obtained using CSS approach. The lowest value of mean Euclidean distance is also achieved by CSS approach. The BPS and HSS has mean Euclidean distance on higher side. The EPS shows reduction in lowest and mean Euclidean distance as compared to BPS. On the contrary, the highest Euclidean distance for BPS is lowest among all approaches and highest for EPS. Overall, based on Euclidean distance based analysis, the CSS approach is found to be effective for selecting top-k web services.

The mean plot in Figure 4.11a shows that the QoS requirement of the end user is satisfied in best way by CSS approach. The EPS has lowest value of mean satisfaction score followed by BPS, HSS and MSS approach in increasing order of satisfaction score.

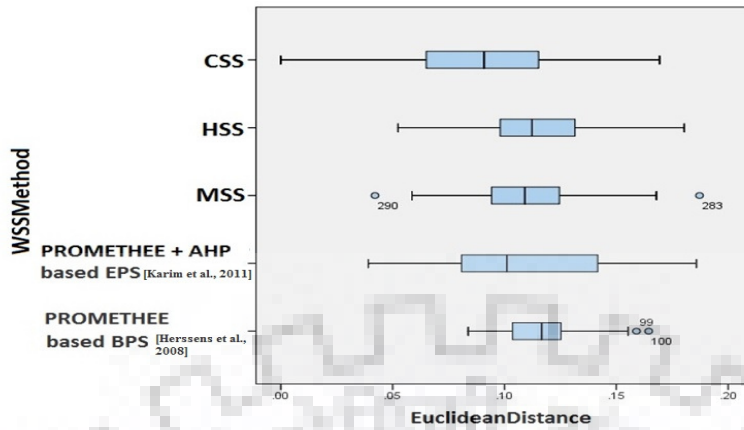


Figure 4.10: Box plot analysis of mean Euclidean distance measure on 100 end user queries

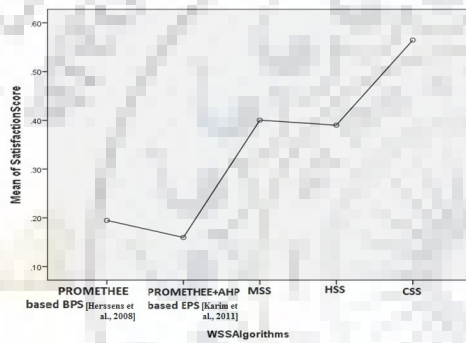


Figure 4.11a: The mean plot for satisfaction score for five WSS approaches (K=3)

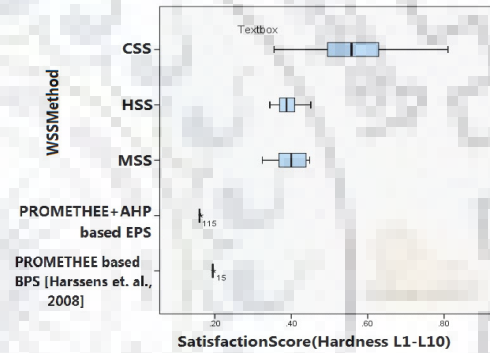


Figure 4.11b: Box plot analysis of mean satisfaction score of 100 end user queries

The lowest value of mean satisfaction is attained by EPS due to the fact that it uses V-shape preference criteria for all QoS parameters [Karim et al., 2011]. The QoS parameters are qualitative as well as quantitative type criteria. Use of V-shape criteria for evaluation of both type of QoS parameters is not suitable and hence the top-k services selected are not the best set of selected web services. On the other hand, the CSS utilizes Gaussian criteria [Brans and Vincke, 1985, Seo et al., 2005] to represent qualitative QoS and level criteria [Seo et al., 2005] for quantitative QoS. Moreover, the top-k web services selected by CSS approach are in accordance to the value of net-outranking flow closer to the NoF of $User_{QoS}$. The box-plot analysis of mean satisfaction score of 100 end user queries is presented in Fig. 4.11b. It is observed from the box-plot that the satisfaction score of CSS approach is found on the higher side. The BPA and EPS approach have satisfaction score on lower side and is constant w.r.t. query hardness.

The results of multiple comparison test is shown in Table 7a and 7b. In Table 7a, minimum,

maximum and standard deviation values obtained for mean satisfaction score measured for BPS, EPS, MSS, HSS, and CSS approach are shown. The confidence level of 95% is used and sample includes 100 values of satisfaction score for each of the five approaches. It is observed from the table that satisfaction score for BPS and EPS vary in a small range. On the contrary, large range of variations in satisfaction score for CSS approach is observed. The CSS approach has minimum mean satisfaction score far above than the maximum mean satisfaction score of BPS and EPS. The fact is also confirmed by observing absolute difference between HSS versus BPS and EPS in Table 4.7b.

Table 4.7a: Minimum, maximum and standard deviation value for mean satisfaction score measure of five WSS approaches

Approach	Observations	Minimum	Maximum	Standard deviation
BPS	100	0.19	0.20	0.00019
EPS	100	0.16	0.16	0.00034
MSS	100	0.32	0.45	0.03522
HSS	100	0.34	0.45	0.02456
CSS	100	0.36	0.81	0.10035

Table 4.7b: Tukey multiple comparison test for satisfaction score measure

Treatments	Abs. Diff*	p-value	Result
BPS vs EPS	0.03499	0.000	Sig. Diff.
BPS vs. MSS	0.20564	0.000	Sig. Diff.
BPS vs. HSS	0.19532	0.000	Sig. Diff.
BPS vs. CSS	0.36983	0.000	Sig. Diff.
EPS vs. MSS	0.24063	0.000	Sig. Diff.
EPS vs. HSS	0.23031	0.000	Sig. Diff.
EPS vs. CSS	0.40482	0.000	Sig. Diff.
MSS vs. HSS	0.01032	0.566	No Sig. Diff.
MSS vs. CSS	0.16419	0.000	Sig. Diff.
HSS vs. CSS	0.17451	0.000	Sig. Diff.

*The absolute difference is significant at the 0.05 level.

Sig. Diff. = Significant Difference, Abs. Diff. = Absolute Difference.

The p-value in Table 4.7b is obtained by finding test statistics (z) using equation 4.9 [Sheskin, 2011].

$$z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{MSE}{n_{group}}}} \quad 4.9$$

Where, \bar{X}_1 , \bar{X}_2 represents sample mean, MSE is mean square error, n_{group} is number of groups, and determine p-value. Null hypothesis is, “there is no significant difference between two

means of satisfaction scores”. Following observations are made from Table 4.7b.

1. The p-values indicate that there is significant improvement in the performance of CSS, MSS and HSS as compared to BPS and EPS.
2. The p-value for MSS and HSS approach is found to be above significance level. Therefore, no significant difference between MSS and HSS approach is observed on satisfaction score. However, both MSS and HSS approach perform significantly better from BPS and EPS.
3. CSS performs significantly better than MSS and HSS.
4. Overall, CSS approach performance is found to be best on satisfaction score and is significantly different from other WSS approaches.

The performance evaluation in terms of total CPU time taken to select top-k web services by each of the five WSS approaches is presented in Figure 4.12. The time taken by each approach is averaged over 100 queries and measured for variations in ‘K’ for values 3, 5 and 10. X-axis represents different variations in ‘K’ values and time taken (in msec) by each approach is shown on Y-axis of Figure 4.12. Few observations drawn from Figure 4.12 are as follows:

1. For K=3 and K=5, HSS and MSS approach takes highest time for selection of web services. EPS is the next higher time taking approach followed by BPS. The time taken by CSS approach is least among all five approaches. This is mainly due to the reason that the CSS approach prefilter the WSs and determines a candidate set of WSs on which selection is to be performed. The selection step now deal with a limited set of candidate WSs. The higher time taken by HSS and MSS is due to the time spent on weight evaluation before applying

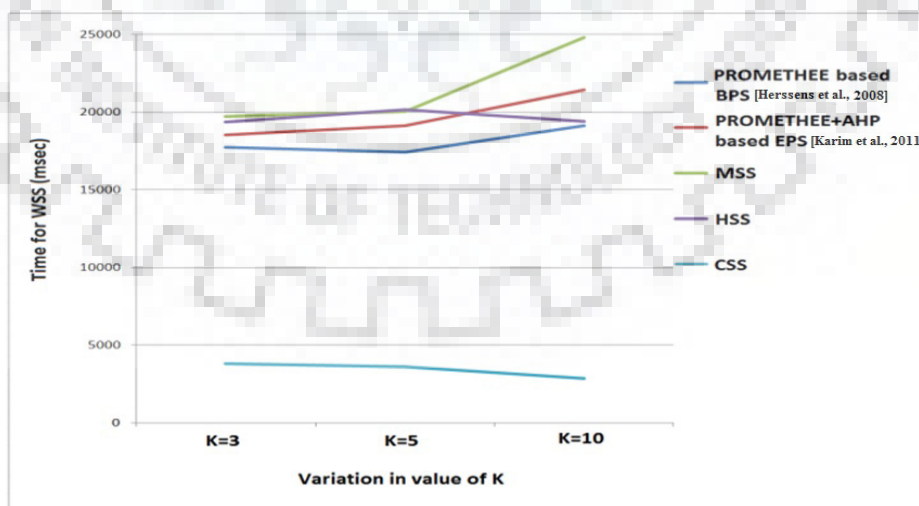


Figure 4.12: Selection time comparison of five WSS approaches

PROMETHEE Plus.

2. For $K=10$, the time taken by MSS approach is highest followed by EPS as next highest. WSS time for HSS and BPS is almost equal. The CSS approach has least time for WSS.
3. HSS and CSS show very little effect of variation in ' K ' on WSS time. This is true for small values of ' K '.

4.3.2.2 Evaluation of the proposed approach using Dataset-2

Dataset-2 consist of QoS measure of 10000 web services generated using [Borzsony et al., 2001]. The details of nine QoS parameters used for evaluation of WSS methods are presented in Table 4.8 and in Section 2.5.1 of Chapter 2. The type field represents whether the parameter is increasing or decreasing type. Increasing type of QoS parameter represents higher the better, such as, throughput and decreasing type is lower the better e.g cost. Using the QoS dataset of 10000 WSSs, various experiments are conducted. The details of each experimentation are discussed as follows.

Table 4.8: Details of dataset-2 consisting of 10,000 web services

Param	R _{tm}	A _{va}	T _{hr}	S _{uc}	R _{el}	C _{om}	B _{pr}	L _{at}	D _{oc}
Type	D	I	I	I	I	I	I	D	I

R_{tm} = response time, A_{va}=availability, T_{hr}=throughput,
 S_{uc}=successability, R_{el}=reliability, C_{om}=compliance (to WSDL description), B_{pr}=best practice (by following WS-I), L_{at}=latency, D_{oc}=documentation, I=Increasing, D=Decreasing.

The effect of variation in the value of ' K ' on satisfaction score is measured for five approaches (BPS< EPS, MSS, HSS, and CSSS) using Dataset-2 is presented in Figure 4.13. The satisfaction score for 100 user queries is evaluated. For each user query, top-k web services are selected. Using equation 2.3, and 2.4 (in Chapter 2, Section 2.6), satisfaction score of each of the ' K ' selected services is evaluated and the QoS satisfaction score for each query is obtained from equation 4.8. The top-k selected services are assigned rank based on the NoF value. Using the rank information, weight w_{t_j} of each selected service is obtained using the rank sum method [Karim et al., 2011]. Y-axis in Figure 4.13 represents mean satisfaction score measured over hundred queries and variation in ' K ' values is presented on X-axis. It is analyzed from the figure that the proposed approach CSS has maximum satisfaction score for $K=3$ and $K=5$, however, for $K=10$, the mean satisfaction score of CSS and MSS are comparable with satisfaction score of HSS approach. The satisfaction score of EPS is observed to be minimum among all approaches for each value of ' K '.

Figure 4.14 shows the effect of increase in hardness of queries on satisfaction for each of the five WSS approaches. The satisfaction score appears on Y-axis, whereas hardness levels (L_1 to L_{10}) are shown on X-axis of Figure 4.14. Experiments are conducted for obtaining top-3 web services ($K=3$). The BPS, EPS, MSS, and HSS approaches has no effect on satisfaction score with the variation in hardness of QoS queries. The improvement in satisfaction score with increase in hardness of end user queries is observed for CSS approach and is more than satisfaction score of other four approaches for hardness $\geq L_2$. This is primarily due to the use of classification based prefiltering which identifies the closest matching set of web services during prefiltering phase.

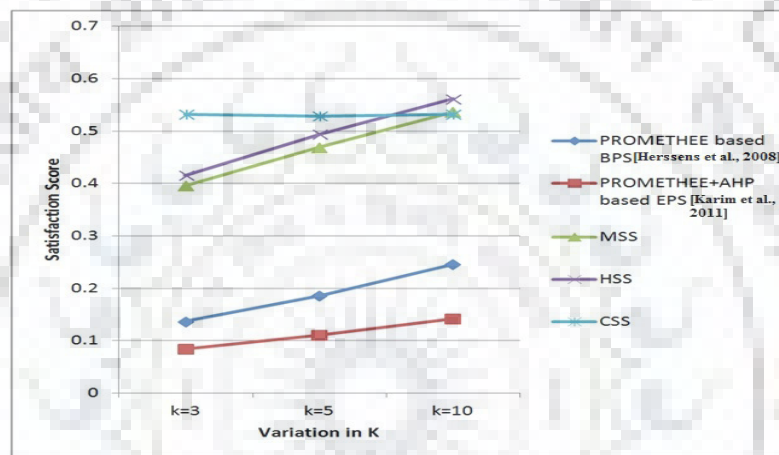


Figure 4.13: Effect of variation of 'K' on satisfaction score

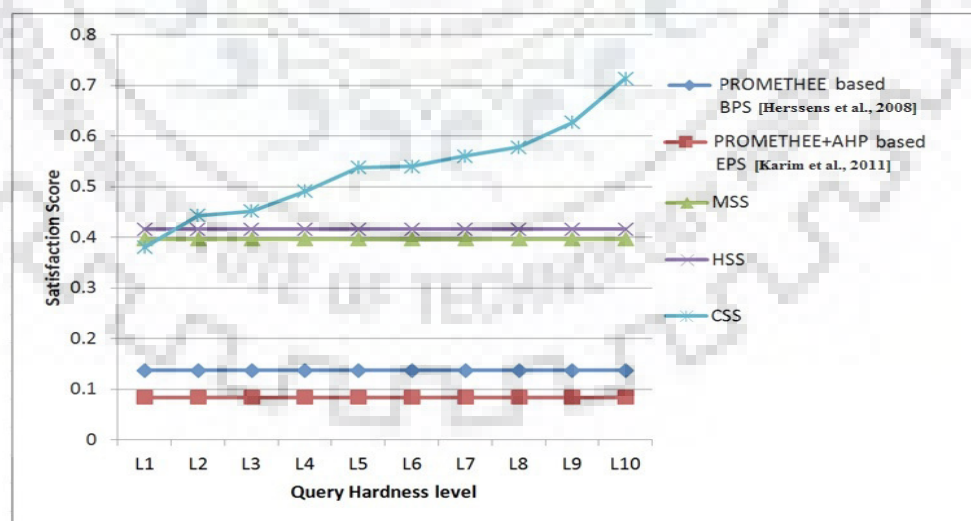


Figure 4.14: Measuring the effect of variation in the hardness of QoS query on the end user satisfaction score. Level L_1 represent least hardness and L_{10} is hardest level.

The box-plot for the mean Euclidean distance measured on each of the 100 queries for five WSS approaches is shown in Figure 4.15 (for $K=3$). It is observed from the figure that the best web service selection with smallest Euclidean distance is obtained using CSS approach. The lowest value of mean Euclidean distance is also achieved by CSS approach. The BPS and EPS has mean Euclidean distance on higher side. The HSS and MSS shows reduction in lowest and mean Euclidean distance as compared to existing EPS and BPS approach. The highest Euclidean distance for CSS is lowest among all approaches and highest for EPS. Overall, based on Euclidean distance based analysis, the CSS approach is found to be effective for selecting top-k web services.

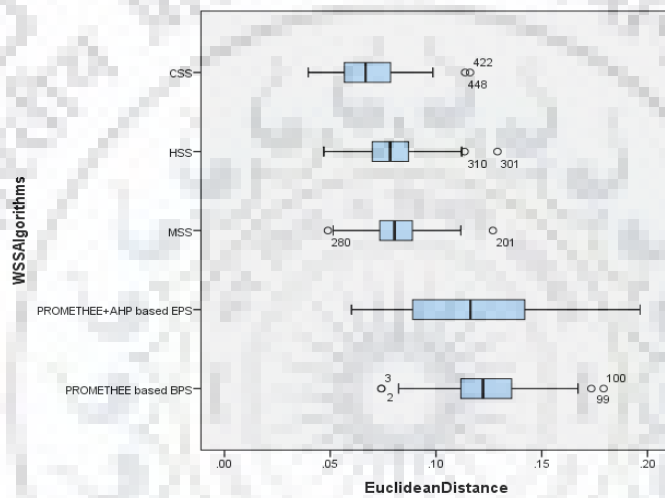


Figure 4.15: Box plot analysis of mean Euclidean distance measure on 100 end user queries ($K=3$).

The mean plot in Figure 4.16 show that the QoS requirement of the end user is satisfied in

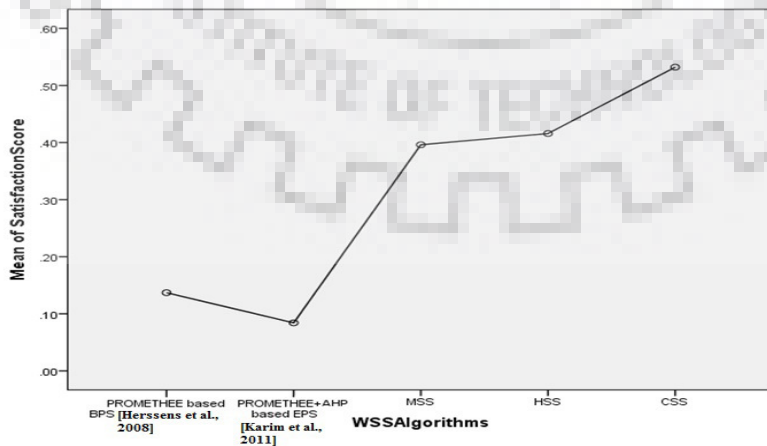


Figure 4.16: The mean plot for satisfaction score for five WSS approaches ($K=3$)

the best way by CSS approach. The EPS has lowest value of mean satisfaction score followed by BPS, HSS and MSS approach in increasing order of satisfaction score. The highest and lowest value of mean satisfaction score attained by CSS and EPS approaches respectively, due to the reason discussed in Section 4.2. Moreover, the top-k web services selected by CSS approach are in accordance to the value of net-outranking flow closer to the NoF of $User_{QoS}$.

The results of multiple comparison test is shown in Table 4.9a and 4.9b. In Table 4.9a, minimum, maximum and standard deviation values obtained for mean satisfaction score measured for BPS, EPS, MSS, HSS, and CSS approach are shown. It is observed from the table that satisfaction score for BPS, EPS, MSS, and HSS remains constant. On the contrary, large range of variations in satisfaction score for CSS approach is observed. The CSS approach has minimum mean satisfaction score far above than the maximum mean satisfaction score of BPS and EPS.

The fact is also confirmed by observing absolute difference between HSS versus BPS and EPS in Table 4.9b. Following observations are made from Table 4.9b.

1. The p-values indicate that there is significant improvement in the performance of CSS, MSS and HSS as compared to BPS and EPS.
2. The p-value for MSS and HSS approach is found to be above significance level. Therefore, no significant difference between MSS and HSS approach is observed on satisfaction score. However, both MSS and HSS approach perform significantly better from BPS and EPS.
3. CSS performs significantly better than MSS and HSS.
4. Overall, CSS approach performance is found to be best on satisfaction score and is significantly different from other WSS approaches.

Table 4.9a: Minimum, maximum and standard deviation value for mean satisfaction score measure of five WSS approaches

Approach	Observations	Minimum	Maximum	Standard deviation
BPS	100	0.14	0.14	0.00000
EPS	100	0.08	0.08	0.00000
MSS	100	0.40	0.40	0.00001
HSS	100	0.42	0.42	0.00000
CSS	100	0.29	0.83	0.10373

The performance evaluation in terms of total CPU time taken to select top-k web services by each of the five WSS approaches is presented in Figure 4.17. The time taken by each approach

is averaged over 100 queries and measured for variations in 'K' for values 3, 5 and 10. X-axis represents different variations in 'K' values and time taken (in msec) by each approach is shown on Y-axis of Figure 4.17.

Table 4.9b: Tukey multiple comparison test for Satisfaction score measure

Treatments	Abs. Diff.*	p-value	Result
BPS vs EPS	0.05282	0.000	Sig. Diff.
BPS vs MSS	0.25920	0.000	Sig. Diff.
BPS vs HSS	0.27888	0.000	Sig. Diff.
BPS vs CSS	0.39511	0.000	Sig. Diff.
EPS vs MSS	0.31202	0.000	Sig. Diff.
EPS vs HSS	0.33171	0.000	Sig. Diff.
EPS vs CSS	0.44793	0.000	Sig. Diff.
MSS vs HSS	0.01968	0.064	No Sig. Diff.
MSS vs CSS	0.13591	0.000	Sig. Diff.
HSS vs CSS	0.11623	0.000	Sig. Diff.

*The absolute difference is significant at the 0.05 level.
Sig. Diff. = Significant Difference, Abs. Diff. = Absolute Difference.

Few observations drawn from Figure 4.17 are as follows:

1. For K=3 and K=5, HSS and MSS approach takes highest time for selection of web services. EPS is the next higher time taking approach followed by BPS. The time taken by CSS approach is least among all five approaches. The higher time taken by HSS and MSS is due to the time spent on weight evaluation before applying PROMETHEE Plus.

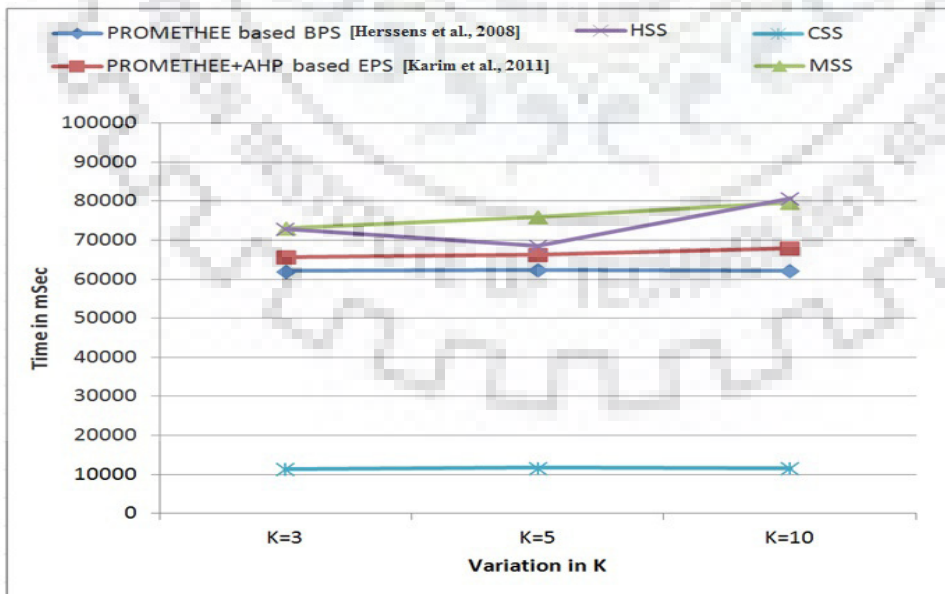


Figure 4.17: Selection time comparison of five WSS approaches

2. For $K=10$, the time taken by MSS and HSS approach is same. Time taken by EPS approach has next highest time followed by BPS. The CSS approach has least time for WSS due to the reason explained earlier.

3. CSS, BPS, and EPS approach show very little effect of variation in 'K' on WSS time. This is true for small values of 'K'.

Overall, based on the results obtained from above experimentation, the performance of CSS approach is found to be better than each of the MSS, HSS, EPS, and BPS. Moreover, performance of MSS and HSS approach are comparable and is found to be improved over EPS and BPS approach.

4.3.2.3 Comparative study of proposed approach with existing similar techniques

A comparative analysis of the proposed approach with the existing state-of-the-art is presented in Table 4.10. The points of comparison with state-of-the-art are: selection criteria used by each of them, whether prefiltering before selection is applied or not, the mechanism used to determine QoS weights, WSS technique used and the dataset used for evaluation purpose. From the Table 4.10 it can be observed that the proposed approach largely differs with state-of-the-art at three points. First, prefiltering technique is applied on candidate WSs before selection. None of the existing WSS techniques uses prefiltering before selection. Second, The QoS weight evaluation is done using MDM based mathematical model. The MDM based weight evaluation

Table 4.10: A comparison of existing state-of-the-art with proposed approach for web service selection

Reference	WSS Criteria	Prefiltering	QoS Weight Evaluation	WSS Technique	Dataset used
[Hwang et al., 2015]	QoS	×	User	Heuristic	QWS
[Herssens et al., 2008]	QoS	×	Simos	PROMETHEE	NM
[Ouadah et al., 2015]	QoS	×	AHP	Skyline + MCDM	QWS
[Rhimi et al., 2015]	QoS	×	NM	FDR	WSDream
[Mobedpour and Ding, 2013]	QoS	×	User	MIP	QWS
[Rhimi et al., 2015]	QoS	×	User	MIP	Synthetic
[Cho et al., 2016]	QoS	×	User	ASS+MIP	QWS
[R.-Zapata et al., 2011]	Ontology	×	NM	Clustering and Bundling	VCO
[Karim et al., 2011]	QoS	×	ANP	PROMETHEE	NM
[Lim et al., 2012]	QoS	×	User	IP	QWS
[Stephen and Yin, 2011]	QoS	×	User	PT	Dummy
CSS (Proposed approach)	QoS	√	MDM	PROMETHEE Plus	QWS

Recent works on web service selection are summarized with details of criteria used for selection of WS(s): whether prefiltering is used before selection, how weights of QoS parameters are obtained, technique used for selection of WSs, and dataset used. VCO=Vocational Competency Ontology, FDR=Fuzzy Dominance Relationship, MIP=Mixed Integer Programming, ASS=Adaptive Service Selection, CBR=Case Based Reasoning, CF=Collaborative Filtering, PT=Prospects Theory, IP=Integer Programming, N.M. = Not Mentioned.

enables us to apply hybrid weight evaluation. Third, PROMETHEE Plus technique is proposed for WSS. Use of PROMETHEE Plus method improves efficiency of WSS.

4.4. DISCUSSION

This chapter presents three approaches MSS, HSS and CSS for selection of WSs. MSS and HSS approach is useful for the case when labelled web services are not available for training. HSS approach is preferred in the case when preference of weights of all/partial QoS parameters is available from the end user. If no preference of weights is provided, the MSS approach is preferred. From the statistical analysis, the performance of MSS and HSS approach is found to be same on satisfaction score. For the case where few labelled WSs are available, the improved WSS results can be obtained using CSS approach. All three approaches uses proposed PROMETHEE Plus algorithm for selection of WSs. The PROMETHEE algorithm is improved in three ways – firstly, the end user request is included with candidate WSs to be processed by PROMETHEE algorithm. Secondly, exact match WS is given preference over the higher match. Thirdly, Gaussian and level type preference functions are used during QoS parameters evaluation.

Three proposed approaches are compared by conducting various experiments. The MSS approach has improved satisfaction as compared to BPS [Herssens et al., 2008] and EPS [Karim et al., 2011]. Moreover, the mean Euclidean distance of MSS is lower than BPS. The end user preference of QoS is combined with mathematical model based weight evaluation of MSS approach. This hybrid QoS weight evaluation mechanism based HSS approach has improved performance and satisfaction over EPS. The classification based prefiltering explores QoS based similarity among web services and classify in the group. The CSS approach has lowest mean Euclidean distance and overall time to select web service. The satisfaction score obtained for CSS approach is observed to be highest and is increasing with the value of K . The results and evaluation presented in Section 4.2 confirms that CSS approach has performed better than existing approaches for WSS. The hardness of query has no/minimum effect on BPS, EPS and HSS. The satisfaction score is observed to be increasing for MSS and CSS approach. Therefore, if labelled WSs are available to train the model, then the CSS approach can be employed for web service selection as is also evident from the experimental evaluation. However, if initial set of labelled WSs are not available, then MSS or HSS approach can be preferred for WSS.

4.5 CONCLUSIONS

In this chapter, a web service selection approach using classification for Prefiltering is proposed. The use of the hybrid weighting scheme brings uniformity in weight calculation and domain dependent inputs are minimized. An improved PROMETHEE method, PROMETHEE Plus, has also been proposed based upon which the presented selection approach works. Our proposed approach is tested by conducting experiments on QWS data set of real-world web services as well as using another dataset generated using available standard dataset generator. The experimental results based on Euclidean distance and satisfaction score show that proposed classification based web service selection approach has ability to find the web services which closely satisfy the end user expectations regarding QoS. The satisfaction score from classification based web service selection approach increases with increase in hardness and value of ' K '. Our proposed approaches for selection of web services outperforms over other existing PROMETHEE based selection approaches. The advantage of prefilter layer based selection approach is clearly observed over existing approaches.

The presented approach for WSS is based on classification process which can be applied when the labelled dataset is available. In the next two chapters, a clustering based web service selection approach is presented which can work for the unlabelled datasets of WSs. A systematic analysis of various clustering techniques for clustering web services is presented in the next chapter, the results of which are used to develop a clustering based web service selection approach as presented in Chapter 6. The work presented in this chapter has been published as [Purohit and Kumar, 2018a].

CHAPTER 5

EMPIRICAL STUDY OF WEB SERVICES CLUSTERING TECHNIQUES

A useful approach for Web Service Selection (WSS) to deal with large number of functionally similar web services is presented in Chapter 4. The proposed CSS approach is found efficient in selection of web services with enhanced end user satisfaction. The inclusion of end user request during web service selection process improves the quality of selection. The introduction of MDM based mathematical model for evaluation of values of weights of QoS parameters relieve the end user from specifying the values of weights of QoS parameters. Further, the use of Prefilter layer enhances the efficiency of WSS system. The approach presented in previous chapter is well suited in the scenario where labelled QoS dataset is available. However, if only the unlabelled dataset is available, then the clustering based service selection can be a more suitable approach. A number of clustering approaches for clustering WSs are available. The choice of clustering technique to be used has a profound effect on the selection results. Therefore, a rigorous analysis of web services clustering techniques is required. The objective of this chapter is to present an empirical analysis of various web service clustering approaches.

The problem of WS clustering is introduced in Section 5.1. The empirical study of clustering techniques and their details are discussed in Section 5.2. A detailed analysis on the performance of various clustering techniques is given. Important observations drawn are presented in Section 5.4 followed by conclusions drawn from the chapter in Section 5.5.

5.1 INTRODUCTION

In the recent years, many software providers have converted to service providers. Service providers offer their business functionality as WS. Using WS concept, machines can interact with each other without any human intervention. It is true when WSs to accomplish the intended task are already identified and known. Otherwise, the process of selection of WS(s) is to be performed.

With the increasing popularity of WSs and WS based applications, the process of WSS requires more care to select desired WS. Availability of web services providing duplicate functionality has an advantage of improved availability of service to the end user, upon occurrence of a failure. On the other hand, the service selection system needs to work more for selecting the desired service. Here, WSs are functionally similar, thus, mechanism to select WS based on criteria other than functional criteria, is needed. Quality of Service (QoS) parameter based selection is widely used criteria for selection of WSs [Zhang et al., 2013, Zheng et al., 2012b, Tripathy et al., 2014, Masri and Mahmoud, 2008]. WS selection system experience the performance degradation problem with the increase in number of candidate web services. In order to improve the performance of WSS system, the end user request of QoS should be taken into account. The goal here is to identify those services having QoS offerings close to the end user requested QoS. Thus, input to the WSS system is only the set consisting of WSs with closely matching QoS as required by the end user. In order to reduce the search domain of candidate WSs, the QoS information can be used to identify similarity among WSs. Multiple groups of candidate WSs are formed. Later on as per the user requested QoS, the appropriate group can be identified and desired WS can be searched. The clustering mechanism is useful in achieving this objective. The QoS parameter forms the basis of clustering.

In this chapter, the problem of web service clustering is addressed. An empirical study of different clustering techniques with reference to web services is performed. The important contributions of this chapter are:

- i) Study of existing web service clustering techniques is done. The details of each technique is explored with reference to the clustering technique used and the clustering criteria used for clustering.
- ii) Study the effect of QoS parameters (attributes) selection on clustering results. We have employed Principal Component Analysis (PCA) for attribute selection.
- iii) An empirical study of six clustering techniques on the basis of stability and quality of clustering is performed. The study reveals the best clustering technique for clustering of WSs using QoS parameters.

5.2 EMPIRICAL STUDY OF WEB SERVICES CLUSTERING TECHNIQUES

The clustering is an unsupervised learning technique to generate groups of objects based on certain properties [Kher et al., 2006]. In case of WSs, QoS parameters can be used to create

clusters of web services, as presented in Figure 5.1. Three clusters of WSs namely, ServiceCluster1, ServiceCluster2, and ServiceCluster3 are formed. The WSs with dissimilar QoS requirements are in different clusters and WSs having similar QoS requirements are clubbed into same cluster. The WSs in same cluster can act as a replacement for other WSs in the same cluster. Many clustering methods are available for performing the clustering process. The details of few popular clustering methods to cluster WSs based on QoS parameters are discussed next.

K-Means clustering: K-Means clustering is a centroid based clustering technique. The mechanism to add the unknown object to the cluster is by calculating the distance of the object from the centroid of all clusters. The object is added to the cluster from which the distance is found to be minimum [Tripathy et al., 2014, Maiti et al., 2014]. For WSs, k number of QoS parameters are used to evaluate the centroid. The clusters of WSs are formed such that the intra-cluster QoS similarity of WSs is high and inter-cluster QoS similarity is low. The criterion function is the mean square function represented by equation 5.1 [Tripathy et al., 2014, Saxena et al., 2017].

$$E = \sum_{i=1}^k \sum_{ws \in C_i} |ws - M_i|^2 \quad (5.1)$$

Where, E represents the mean-square-error of all WSs in the dataset, ws is the web service in the set and the mean of cluster i is represented using M_i .

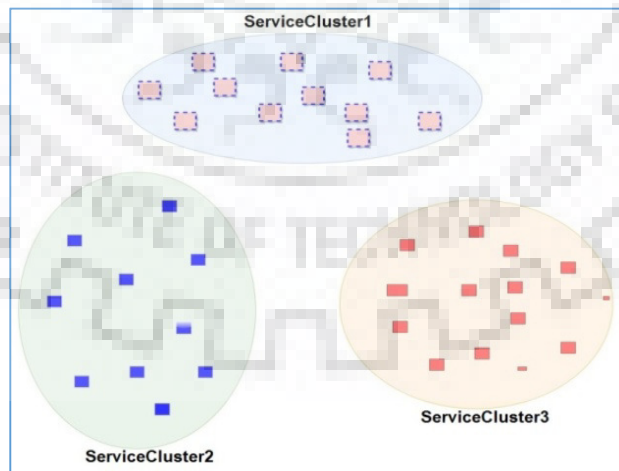


Figure 5.1: Web service clusters created by performing QoS based clustering.

Unweighted Pair Group Method with Arithmetic Mean (UPGMA): UPGMA is a hierarchical clustering method based on the agglomerative (also known as a bottom-up approach) paradigm to generate hierarchy [Zaki and Meira, 2014]. In this approach, each of the WS represents individual cluster. In each iteration, clusters are clubbed together based on the distance criteria among the clusters. If $Clust1$ and $Clust2$ are two clusters, x and y are two WSs such that $x \in Clust1$ and $y \in Clust2$, then the intra-cluster distance in agglomerative clustering is obtained from the equation 5.2 [Zaki and Meira, 2014].

$$Clust_{DIST} = \frac{1}{|Clust_1| \cdot |Clust_2|} \sum_{x \in Clust_1} \sum_{y \in Clust_2} [d(x, y)] \quad (5.2)$$

Partitioning around medoids (PAM): PAM clustering is based on k-medoid algorithm. The most optimal clustering is obtained using a greedy algorithm. The dissimilarity among the object (WS, in our case) of the same cluster is minimized. The robustness of PAM algorithm is more than the K-Means clustering. It is mainly due to the reason that K-Means uses only the Euclidean distance measure, whereas PAM has the ability to use Euclidean distance as well as other dissimilarity measure [Brock et al., 2008].

Self Organizing Tree Algorithm (SOTA): The SOTA is a divisive clustering method. The process of clustering starts from binary topology consisting of two leaves and a root node. The data is split into two cluster at each stage by using self organizing process [Brock et al., 2008]. The SOTA is independent of cluster size due to the fact that the resource value governs the growth of the network. The resource value is determined by taking mean of distance between the expression profile and a node.

Clustering for the large data set (Clara): The Clara is an extension of k-medoid method. Clara chooses a sample from the large data set and applies medoid on the obtained sample instead of the whole dataset of larger size. The PAM is applied to the sample to obtain medoids. The remaining elements are mapped to the clusters obtained from the sample. It leads to an improvement in the running time of the algorithm, even if there are large number of candidate WSs in the cluster [Zaki and Meira, 2014, Brock et al., 2008].

Diana: Diana is a divisive analysis of the hierarchical clustering algorithm. The Diana clustering starts by considering the entire set of candidate WSs as part of a single cluster. In each iteration, Diana successively divides the clusters until a stopping condition is met or each of the clusters contains a single WS [Zaki and Meira, 2014].

5.3 ANALYSIS AND OBSERVATIONS

In this section, initially the experimental setup and details of dataset used are discussed. The parameter selection using principal component analysis is discussed next. The experimental study conducted to evaluate six clustering techniques and important observations drawn from the experiments are presented at the end of this section.

5.3.1 Experimental setup

In order to generate clusters of web services, the experiments are conducted on a machine with Intel Core i7 CPU 3.4 GHz, Windows 7 platform. A QoS dataset based on real world WSs is used to conduct the experiments (henceforth called QWS). The currently available version of the QWS includes a set of 2,507 WSs with QoS measurement of each parameter [Masri, and Mahmoud, 2008]. The dataset of 2,507 WSs is available in a file with 2,507 rows and 11 column entry each separated by commas. Some of the example entries from QWS dataset are as shown in the Table 5.1. The details of the dataset are discussed in Chapter 2. The nine QoS parameters are values in the range (min and max) as shown in the Table 5.2.

Table 5.1: Sample entries for QWS dataset [Masri, and Mahmoud, 2008]

Service e	R _t	A _v	T _h	S _u	R _l	C _o	B _p	L _a	D _o	Service Name
S ₁	56	97	9	99	73	78	84	1	35	BookInfoService
URL of S ₁	http://edi.btol.com/bookinfoservice/bookinfoport.asm									
S ₂	459	98	9.5	100	73	89	80	3	36	LDAPService
URL of S ₂	http://www.epfl.ch/ws/ldap.wsdl									
S ₃	469	98	13.1	100	67	78	77	0.67	93	XigniteArchive
URL of S ₃	http://www.xignite.com/xArchive.asmx?wsdl									
S ₄	516.7	63	7.2	63	83	89	91	10.67	33	sms
URL of S ₄	http://www.info-me-sms.it/ws.php?wsdl									

The QWS dataset includes web services from various domains such as transport, smart banking, medical, hotel, weather, geographical, etc. For developing smart applications, these available web services or such similar web services are useful.

Table 5.2: QoS parameters with their min and max values

QoS Param	R _t	R _l	A _v	C _o	T _h	S _u	D _o	B _p	L _a
Min	37	33	7	33	0.1	8	1	50	0.25
Max	4989.67	89	100	100	43.1	100	97	95	4140.35
Type	Dec	Inc	Inc	Inc	Inc	Inc	Inc	Inc	Dec

Min = Minimum, Max = Maximum, Inc = Increasing, Dec = Decreasing

5.3.2 QoS parameter selection using Principal Component Analysis (PCA)

Quality of any object is described usually with various features associated with the object. Few features have minor contribution, whereas others have major contribution in characterizing an object. In case of web services, the non-functional characteristics of a WS are described using QoS parameters. To characterize any WS, the participation of few of the QoS parameters is high and that of few QoS parameters is very low, i.e., few QoS parameters with low contribution may be removed and still the overall quality of WS remains same. If such QoS parameters can be identified and processing based on high contributing QoS parameters is done then the overall quality of results can be retained and in turn fast processing and simplified execution of the operation can be carried out [Song et al., 2010]. This task of feature selection can be efficiently performed by principal component analysis (PCA). PCA uses the concept of eigenvector for covariance matrix. Let I_{vect} be the eigenvector of covariance matrix. The result of feature selection can be represented using equation 5.3 for arbitrary sample vector, η , of candidate WSs [Song et al., 2010].

$$P_k = \eta^T * I_{vect} = \sum_{i=1}^n \eta_i * I_{vect_i} \quad (5.3)$$

Where, n is number of WSs, a sample vector of candidate WSs $\eta = [\eta_1, \eta_2, \dots, \eta_n]^T$, and $I_{vect} = [I_{vect_1}, I_{vect_2}, \dots, I_{vect_n}]^T$. It can be observed from the equation 5.3 that the contribution of any i^{th} QoS parameter is statistically obtained using the absolute value of η_i . Thus, smaller value of η_i indicate lesser contribution of QoS feature component.

5.3.3. Performance evaluation parameters

The suitability of the clustering algorithm depends on the various algorithmic parameters and data on which it is to be applied. Therefore, the clustering algorithm suitability is to be accessed

before applying on the data of interest. The process of clustering validation and assessment evaluates the clustering suitability. We have performed clustering evaluation based on following two parameters [Zhu et al., 2010].

5.3.3.1 Performance evaluation of the quality of clustering (internal cluster quality measure):

The quality of clustering is the measure of goodness or quality of clustering for the clusters formed. The evaluation can be done by internal criteria such as the distance between inter/intra cluster WSs or using the characteristics of QoS parameters associated with WSs. Similarly, by varying clustering parameters, the output in each case can be compared. It establishes a mechanism of relative evaluation measure. In this work, internal measures are used to evaluate clustering. The measures such as compactness, connectivity, and separations of clustering partitions are used for evaluation [Brock et al., 2008]. We have used Silhouette coefficient to evaluate quality of cluster obtained from six different clustering methods. In this case, Silhouette coefficient defines the measure of the degree of confidence in assigning a WS to a cluster. It is measured by finding the difference of the average distance between WSs in the closest cluster and WSs in the same cluster [Zaki and Meira, 2014]. The coefficient value close to 1 refers to the well-clustered WS and poorly clustered WS keeps coefficient values near -1. For i^{th} WS instance, it is defined by using equation 5.4. The Silhouette coefficient value should be maximized .

$$S(i) = \frac{b_i - a_i}{\max(b_i, a_i)} \quad (5.4)$$

Where, the average distance between the i^{th} WS and all other WSs in the same cluster is represented as a_i , and b_i represents the average distance between the i^{th} WS and WSs in the nearest neighbouring Cluster.

5.3.3.2 Performance evaluation of stability measurement (external cluster quality measure):

The stability of clustering method is the measure of sensitivity of clustering method with reference to algorithmic parameters and type of data. The stability of clustering method can be measured by observing the effect of removing a column data from the dataset used for clustering. For a multi column dataset (such as QWS), clustering results obtained based on the full data is compared with the clustering results obtained by removing each column data, one at a time [Brock et al., 2008]. If the data are highly correlated, the stability

measures produce the desired results. The average distance between means (ADM) parameter is used to perform stability analysis of cluster obtained from six different clustering methods. The ADM measure is evaluated for the two variations i.e. Clustering with full QWS and clustering with single column removed. Finally, the average distance between cluster centres is obtained for web services placed in the same cluster [Brock et al., 2008].

5.3.4 Experimental study

In this section, an approach to access the performance of clustering methods for WS clustering is presented next. At first, various clustering methods are evaluated on QWS dataset with nine QoS parameters. The experiments are repeated with six parameters selected by applying parameter selection technique using PCA.

The summary of performance evaluation of the quality of clusters obtained using Silhouette Coefficient for six clustering methods on the QWS dataset is shown in Figure 5.2a. The results are obtained for clusters of sizes 2, 3, 4, 5, 6, 7 and 8 for each of the mentioned clustering methods. As per the definition and discussion done earlier, the value of silhouette coefficient is to be maximized. The experiments are repeated with six parameters selected using PCA, as shown in Figure 5.2b. From the Figure 5.2a it can be observed that –

- (i) The highest value of Silhouette coefficient is obtained when UPGMA clustering method with cluster of size two is used. For the same cluster size, the next highest value of Silhouette coefficient is attained for SOTA clustering algorithm.
- (ii) The lowest value of silhouette coefficient is attained by Clara for cluster size of three and next lowest is for Diana for cluster size of eight.
- (iii) The quality of cluster formed using UPGMA clustering is best for cluster size of two.

It is observed from Figure 5.2b that –

- (i) The introduction of PCA based parameter selection has improved the silhouette coefficient value for each of the clustering method.
- (ii) The highest value of silhouette coefficient is attained by UPGMA clustering for cluster of size two.
- (iii) The lowest value of silhouette coefficient is attained by Clara clustering method for cluster size of two.

Overall, the silhouette coefficient value is maximized by UPGMA clustering method for cluster size equals two. The introduction of feature selection based on PCA method for QoS parameters boost up clustering results. It in turn leads to the optimal selection of WSs and the overall quality of WSS can be improved.

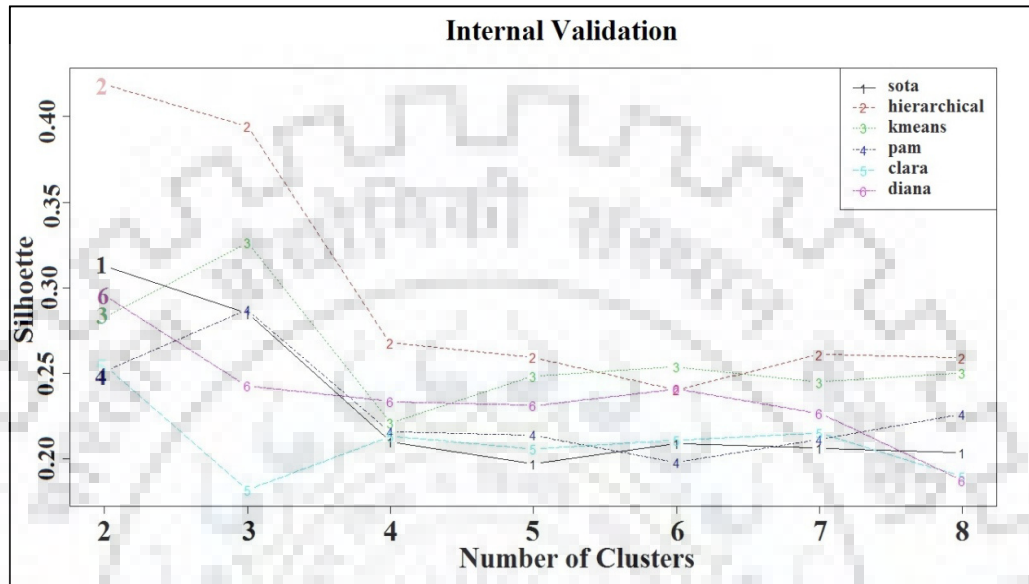


Figure 5.2a: Comparison of six clustering techniques on Silhouette coefficient evaluated using nine QoS parameters available from QWS [Masri and Mahmoud, 2009]

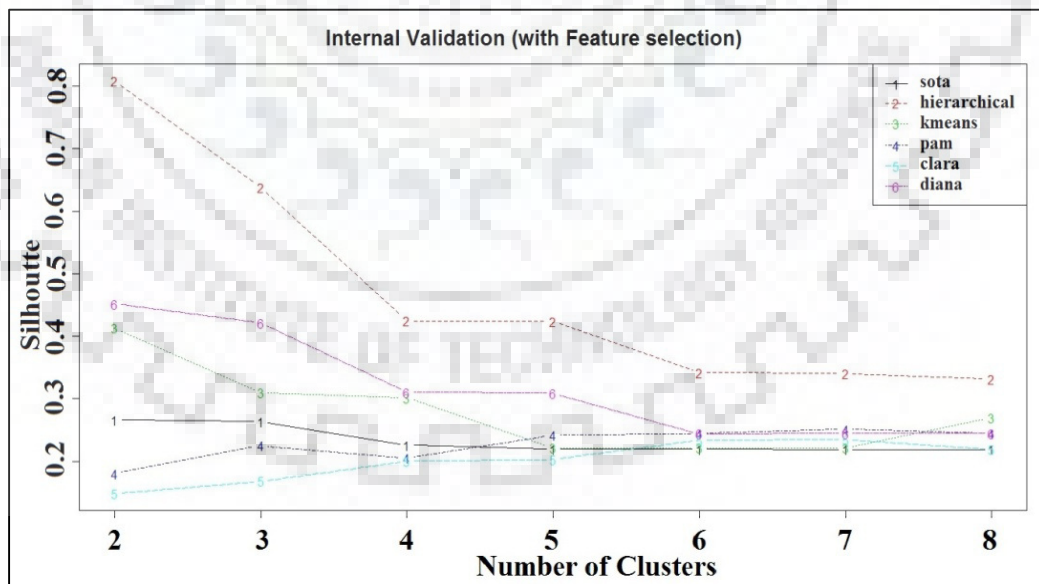


Figure 5.2b: Comparison of six clustering techniques on Silhouette coefficient evaluated using six QoS parameters obtained by applying PCA on QWS.

Figure 5.3a and 5.3b, represents the performance evaluation with reference to stability measurement for experiments performed using nine QoS parameters and six parameters selected by applying PCA method, respectively. The ADM parameter is used to conduct the stability validation. The value of ADM parameter is to be minimized for optimal results. From Figure 5.3a, it is observed that -

- (i) The minimum value of ADM parameter is achieved for number of clusters equals two by UPGMA clustering method.
- (ii) PAM and Clara have higher value of ADM parameter for number of cluster equals two. For number of cluster equals eight, higher values are observed for K-Means and PAM clustering.

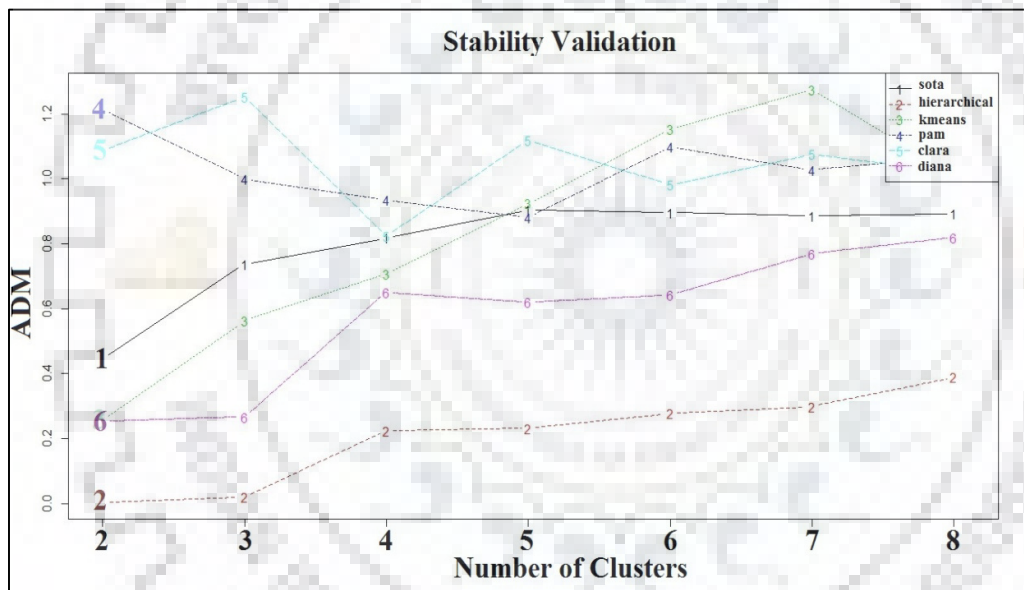


Figure 5.3a: Comparison of six clustering techniques on Average distance between means (ADM) parameter evaluated using nine QoS parameters available from QWS [Masri and Mahmoud, 2009]

It is analyzed from Figure 5.3b that –

- (i) The effect of PCA method based parameter selection has affected the ADM value of all clustering methods and overall reduction in ADM values is observed.
- (ii) For cluster size of two, the lowest and highest value is attained by the UPGMA and Clara clustering methods, respectively.
- (iii) For cluster size of two, PAM and Clara clustering shows degraded performance and for cluster size of eight Clara and K-Means shows weak performance.

Overall, the lowest value of ADM is produced by UPGMA clustering for cluster size of two and is found to be most stable clustering among six clustering methods.

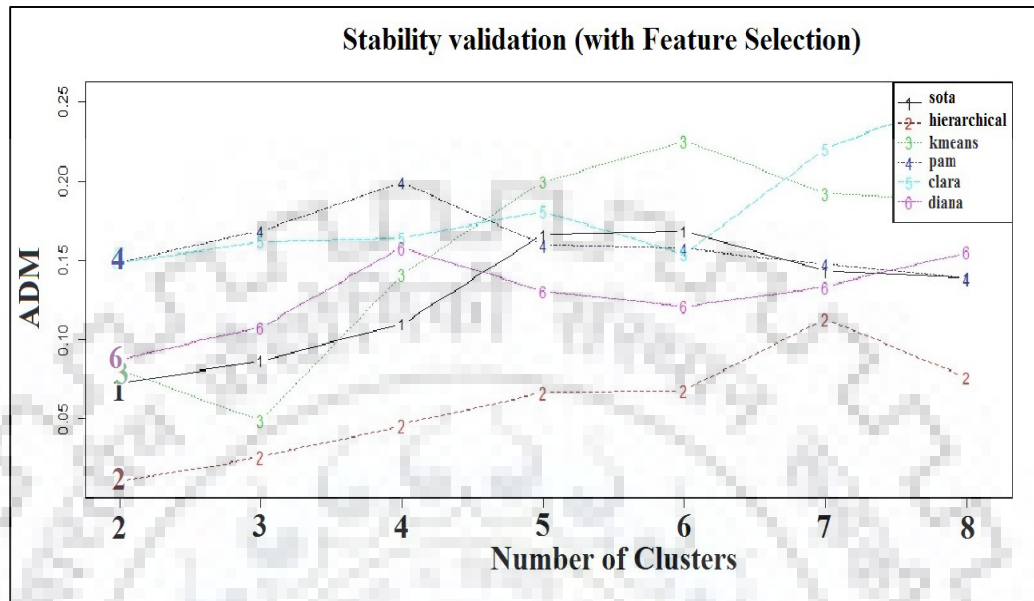


Figure 5.3b: Comparison of six clustering techniques on Average distance between means (ADM) parameter obtained by using six QoS parameters determined by applying PCA on QWS.

5.4 OBSERVATIONS AND DISCUSSION

The empirical study of various clustering techniques is carried out to judge the capability of clustering techniques to cluster WSs. Few of the important observations from the empirical study are as follows:

- (i) Performance evaluation of the quality of clustering shows that optimal quality is obtained with UPGMA clustering. The optimal results are obtained for cluster of size two. Further, the quality of clustering improves by reducing QoS parameters using PCA method.
- (ii) The Performance evaluation based on stability measurement shows that the UPGMA clustering produces clustering which is most stable among all other clustering techniques. Furthermore, the improvement in the stability is observed with the inclusion of PCA method for QoS parameters reduction.

The application of clustering in web service selection has shown promising results towards possibility in improvement of performance of selection mechanism. Further, the choice of clustering technique has multi-fold effect on the selection of promising WSs. Thus, in this

empirical study, we have evaluated six clustering techniques on quality of cluster generated and stability of clustering technique parameters. Also, the application of PCA as attribute selection technique is explored for QoS parameters selection. The selected parameters are further tested on the aspect of stability and quality of cluster generated. As a outcome of empirical study, UPGMA (hierarchical) clustering is found to perform exceptionally well. Moreover, the PCA based parameter selection generated improved results.

In order to conduct experiments, a popular QWS dataset consisting of QoS parameters measured on real world web services is chosen. The used dataset contains set of services from different domains, which finds their application in developing smart services.

5.5 CONCLUSIONS

The study of some clustering techniques for WSS is presented in this chapter. To improve the efficiency of the selection process, clustering of web services before selection is a useful step. Based on two performance measures - stability and quality of clustering, the UPGMA (hierarchical) clustering is found to be most efficient technique to cluster web services. The use of PCA based attribute selection further improves the quality of clustering. Thus, PCA based attribute selection can be used with UPGMA (hierarchical) clustering technique to cluster web services and subsequently web service selection process can be improved.

As reviewed in Chapter 2, there are many approaches available for selection of WSs. However, they do not scale well with the increase in number of candidate WSs. It was observed that classification and clustering based prefiltering can be a good solution for this scenario. Previous two chapters deals with the classification based WSS problem. The study of some clustering techniques for WSS is presented in this chapter. In the next chapter, the results of empirical analysis presented in this chapter are used to develop a clustering based approach for selection of WSs. The work presented in this chapter has been published as [Purohit and Kumar, 2018e].

CHAPTER 6

CLUSTERING BASED APPROACH FOR WEB SERVICE SELECTION

The selection of desired web services involves the matching of end user requirements with the service offered by web services. One such approach using the labelled dataset of QoS is presented in the Chapter 3 and Chapter 4. For unlabelled dataset, the classification based WSS approach presented in Chapter 3 and 4 is not efficient. In this scenario, a clustering based approach is most appropriate to perform selection. For this, in Chapter 5, a systematic analysis of various web services clustering techniques is presented. From the analysis, it is observed that the Hierarchical clustering based UPGMA technique perform well on quality and stability of clusters formed. Thus, in this chapter, Hierarchical clustering is employed for prefiltering WSs followed by pruning. In this chapter, a two layer selection model is proposed. The pruning process act as catalyst to improve the selection results by pruning out unmatching WSs. The Skyline based selection approach is followed to determine WS of interest. The traditional skyline technique always generates same set of skyline services without considering the end user requested QoS. Also, the skyline results in non-dominated set of services without any ordering of services. To handle these issues, a modified skyline, we call it as Skyline Plus, is proposed. The selection layer also identifies replaceable web services using pearson correlation coefficient.

An introduction to the need of WS clustering and Skyline concept is given in Section 6.1. A motivational example to illustrate the advantage of applying pruning and the proposed two layer model for WSS is discussed in Section 6.2. In Section 6.3 evaluation of the proposed approach is done by comparing it with existing state-of-the-art. The discussion carried out in Section 6.4 followed by important conclusions drawn from the chapter appears in Section 6.5.

6.1 INTRODUCTION AND MOTIVATION

Web service selection is the process of choosing relevant web service(s) with the capability to do the intended task as per the end user expectations [Zhao et al., 2014]. The primary task in

using the web service (WS) is to take the user input and select WS matching the end user needs [Oriol et al., 2014]. As a first step, functionally similar WSs are discovered from the pool of WSs. From among these, a most suitable WS is to be selected to perform the intended task. However, as discussed in Chapter 4, with the increasing use of WSs and WS based systems over the Web, a lot of WSs offering similar functionality are available [Huang, 2013]. The availability of functionally similar WSs has an edge of acting as a replacement in case of run-time failure of selected WS. Furthermore, the monopoly of service providers is also controlled. On the contrary, it intensifies the complexity of the task of selection of needed WS and presents a challenge for developing an efficient WS based system. The use of Quality of Service (QoS) as a second level criteria (after using functionality for first level filtering) for selection of the desired WS is a popular technique [Ouadah et al., 2015, Karim et al., 2011, Oriol et al., 2014, Huang, 2013]. The user is asked to provide functional as well as QoS specifications of the desired WS. The system uses these QoS specifications to select most suitable WS. Currently reported solutions for WSS using QoS includes - Skyline based techniques [Ouadah et al., 2015, Rhimi et al., 2015, Alrifai et al., 2010, Wang et al., 2016, Yu and Bouguettaya, 2013, Yang, 2015, Gang and Chunli, 2015, Benouaret et al., 2012, Kang et al., 2011], multicriteria decision making based techniques [Ouadah et al., 2015, Karim et al., 2011, Huang, 2013] clustering based techniques [Xia et al., 2011, Yu, and Gen, 2010, Wu et al., 2014], classification based WSS approaches [Purohit and Kumar, 2018a], negative selection algorithm [Zhao et al., 2014], Semantic Approach [Kumar and Mishra, 2008b], etc.

Generally, candidate WSs includes three categories of services. Firstly, the potential WSs which can meet all of the QoS requirements as specified by the end user. Second, WSs partly satisfying the QoS requirements of the end user. Third, WSs which do not satisfy any of the QoS requirements specified by the end user. The existing solutions [Purohit and Kumar, 2016a, Ouadah et al., 2015, Yu and Gen, 2010, Zhao et al., 2014, Huang, 2013, Gang and Chunli, 2015], do not differentiate between three categories of WSs and thus lacking in terms of performance as they consider all candidate WSs during service selection. The performance degradation is due to unnecessary processing of WSs in the third category. Therefore, the process of selection based on technique such as Skyline takes higher time. The Skyline technique used for service selection also ignores the end user expectations of QoS for generating sskyline services and the preference of QoS, if any [Rhimi et al., 2015, Wang et al., 2016]. The skyline technique gives a non-dominated set of WSs as output but no list of ranked WSs is available. Further, some of the existing *WS Selection* approaches [Alrifai et al., 2010, Zhao et al., 2014, Huang, 2013, Yu and Bouguettaya, 2013, Margaris et al., 2015], expect

minimum two user inputs to locate desired WS (i) Weight of QoS parameters (ii) preferred values of QoS parameters. It is very difficult for naïve user to clearly specify the values of (i) and (ii). The user centric approach [Mobedpour and Ding, 2013] as well as QoS browser [Ding et al., 2009] is helpful in such cases to guide the end user to specify the preferred values of QoS parameters. However, the QoS parameter weight still depends on the judgment of the end user. Values of weights of QoS parameters have a profound effect on the selection of WSs. The imprecise specification of values of weights of QoS may lead to low satisfaction score and/or missing of user desired services. For a WSS system to be useful, the system should be able to determine the values of weights using some mathematical model [Wu and Chen, 2007].

The work proposed in this chapter fulfils the above identified gaps. At first the proposed approach carries out prefiltering of non-compliant WSs and retains the WSs which are capable of meeting the end user requirements of QoS. This helps in improvement in service selection process by avoiding the non-compliant services to take part in selection process. Further, we have extended the traditional skyline approach by using simple additive weights (SAW) [KalisZewski and Podkopaev, 2016] and Maximizing Deviation Method (MDM) [Wu and Chen, 2007] to generate ranked list of skyline WSs, we call this approach as Skyline Plus. The MDM based hybrid approach to automatically evaluate the values of weights of QoS parameters is developed. This approach has two advantages. First, it releases the burden from the end user to judge the values of weights of QoS parameters. Second, the end user preference of values of weights is also considered, if end user is willing to provide. The change in QoS values or service failure is also handled by determining replaceable WS.

Based on the gaps identified, the research problem is summarized as below:

Research Problem: The set $S = \{S_1, S_2, \dots, S_n\}$ is the set of candidate web services offering similar functionality and has some QoS offerings. Based on the offered functionality and non-functional characteristics, the WS is defined as a, $S_i = \{I, O, QoS\}$ where 'I' is the input(s) required by the WS to process, 'O' is the output(s) generated by the WS. The 'I' and 'O' together illustrates the functionality provided by the WS. The 'QoS' is the multiple QoS attributes associated with the WS and describe the non-functional characteristics of the WS. If $S'\{UQoS\}$ is the QoS specifications of the desired WS by the end user, the problem of WS selection is to select the S_i such that, $S_i\{QoS\} \equiv S'\{UQoS\}$. In this work, it is assumed that the functionally similar WSs are available as a result from the execution of WS discovery operation based on 'I' and 'O' parameters [Purohit and Kumar, 2016a]. Moreover, to deal with the problem of

runtime failure or change in QoS values of selected WS, an equivalent WS is to be selected to substitute failed WS.

Research Solution: The desired web service can be obtained in three steps. First, all services S_i are identified such that $\exists k : S_i\{QoS_k\} \geq S'\{UQoS_k\}$. The pruning step is useful to identify such set of web services. Secondly, from the above set all those services S_i are to be determined such that $S_i\{QoS\}$ closely matching with $S'\{UQoS\}$ i.e. QoS offered by the candidate services should be exactly/closely matching with the QoS requirements specified by the end user. The clustering technique is useful to obtain the group of WSs [Yu and Gen, 2010, Zhang et al., 2013, Tewari et al., 2012b]. The QoS parameters can be used to cluster WSs. The clustering is suitable for the case when unlabelled data such as QWS dataset [Masri and Mahmoud, 2007] is available and a logical grouping is desired. As a result of clustering, each WS belongs to some cluster identified using cluster number C and candidate WS is now defined as a tuple $S = \{I, O, QoS, C\}$, Where $I, O, QoS,$ and C are as defined earlier. In order to obtain best matching service in the set, the Skyline technique is beneficial. The Skyline concept can be employed to obtain non-dominated set of WSs [Alrifai et al., 2010, Yu and Bouguettaya, 2013]. The Skyline technique is the most widely used technique for service selection scenario [Ouadah et al., 2015, Rhimi et al., 2015, Alrifai et al., 2010, Wang et al., 2016, Yu and Bouguettaya, 2013, Yang, 2015, Gang and Chunli, 2015].

The Skyline consists of non-dominated set of WSs. Services in the Skyline are obtained by firing Skyline query on the set of candidate WSs. The Skyline module selects those WSs which are non-dominated by any other WS in the set S of candidate WSs. For any two random WSs S_i and $S_j \in S$, we say S_i dominates S_j (denoted as $S_i \prec S_j$), if S_i is as good as S_j on all QoS parameters and strictly better than S_j on at least one QoS parameter. Using publicly available dataset generator [Borzsony et al., 2001], we have generated a QoS dataset of 25 WSs (S_1, \dots, S_{25}) with three QoS parameters – Documentation (D_{ocu}), Latency(L_{ate}), and Reliability (R_{eli}). From that set, consider five WSs, S_4, S_6, S_7, S_8, S_9 , as shown in Table 6.1. The QoS parameters are represented by float values normalized in the range $[0..1]$. For increasing type QoS parameters such as A_{val} and R_{eli} , the dominance $S_i \prec S_j$ is true, if, D_{ocu} and R_{eli} values of S_i is as good as that of S_j and strictly one of the D_{ocu} and R_{eli} values of S_i is better (higher) than that of S_j . Similarly, for decreasing type QoS parameter such as L_{ate} , the dominance $S_i \prec S_j$ is true, if, L_{ate} value of S_i is better (lower) than that of S_j . Based on this definition of dominance, for WSs in Table 6.1, it is easily observable that the services S_7 , and S_9 (represented as bold values) are Skyline services.

Table 6.1: Web services with three QoS parameters

Web Services	QoS parameters		
	<i>Documentation</i>	<i>Latency</i>	<i>Reliability</i>
S_4	0.3	0.516	0.89
S_6	0.54	0.342	0.67
S_7	0.88	0.0142	0.99
S_8	0.673	0.0879	0.7152
S_9	0.89	0.099	0.873

The availability and reliability parameters are of increasing type i.e. higher the better. The response time parameter is of decreasing type i.e. lower the better.

The pictorial representation of quality distribution of WSs in multi-dimensional space is shown in Figure 6.1. Figure 6.1 shows the distribution for a set consisting of 25 WSs. Using the definition of dominance relation, seven WSs S_5 , S_7 , S_9 , S_{15} , S_{19} , S_{20} , and S_{24} are identified as Skyline services and are shown in green color in Figure 6.1.

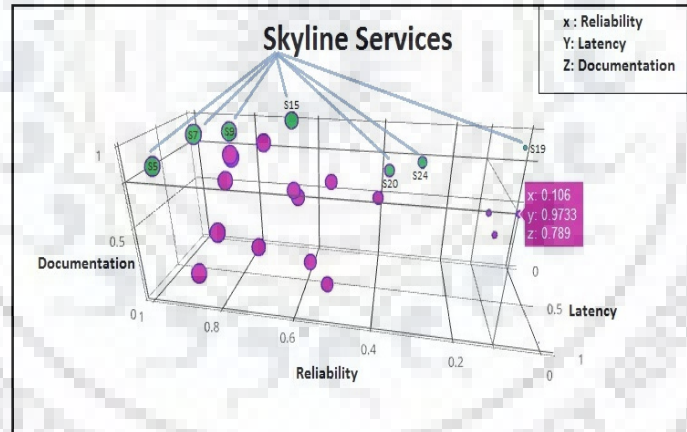


Figure 6.1: Quality distribution of Web Services with identification of Skyline services using Documentation (z-axis), Reliability (x-axis) and Latency (y-axis)

However, there are three limitations of the existing skyline technique when applied to WSS. First, for every user request, the skyline technique determines same non-dominated set of WSs. This causes imbalance in service access in terms of high load on few WSs and very little or no execution request for other services. Second, the skyline set of WSs is obtained without any consideration to the requested QoS and the QoS preference from the end user. Third, the skyline generates the non-dominated set of WSs without ordering them. The first and second

limitation can be overcome by the use of clustering based prefiltering technique. The third limitation can be handled by ordering the WSs in the non-dominated set and finding top-K WSs representing most optimal set. SAW method is the most appropriate method in this scenario [KalisZewski and Podkopaev, 2016]. The weight of each QoS parameter can be obtained using mathematical model such as MDM [Wu and Chen, 2007].

Using SAW and MDM together leads to avoid selecting those services in top-K having only one or two QoS parameters dominating. To deal with service failure or unavailability of WS, replaceable WS is to be determined. The Pearson correlation coefficient can be fruitful to find replaceable WS. An example to explain the underlying motivation of the proposed design is presented in Section 6.2.

Following are important contributions of this chapter:

- The pruning is applied to filter out WSs having low QoS offerings as compared to QoS demands of the end user. This step saves a large number of unnecessary calculations and improves the service selection efficiency.
- The use of hierarchical clustering (UPGMA) technique as a second step for prefiltering WSs is explored. The clustering step identifies WSs with similar QoS offerings as that of requested QoS.
- To select top-K WSs, an improved skyline, called as Skyline Plus is proposed. The skyline set of WSs are obtained by using a block nested loop paradigm. From the skyline set of WSs, top-K services are selected by using SAW and MDM method.
- For each of the top-K service, the replaceable WSs using Pearson correlation coefficient are derived.
- Statistical tests are performed to analyze the behavior of the proposed system and compared with the existing state-of-the-art on four performance parameters

6.2 PROPOSED APPROACH FOR WEB SERVICE SELECTION

In this section, the proposed approach for selection of optimal WSs for a given task is discussed. Two approaches for selection of WSs is proposed. Both the proposed approaches are based on the concept of prefiltering followed by selection of WSs.

6.2.1 Web Service Selection

The process of identifying a service or set of services having ability to provide desired business functionality by exactly/ closely satisfying the needs of the end user is known as web service selection. In the proposed CPSky approach, based on the attributes of datasets to be used for clustering, we have two possible variations. The first variation, we call it as CPSky-All Attributes (CPSky-AA), is realized by considering all QoS parameters for clustering candidate services. The CPSky-AA is useful in case when number of QoS parameters associated with WS is small in number and all parameters have major contribution for evaluation of service quality offered by WS. However, for the situation when number of QoS parameters associated with the WS is large in number and only few parameters contribute to determine service quality offered by WS, the majorly contributing parameters are to be identified. The second variation is realized by applying CPSky on QoS dataset obtained after feature selection, we call it as CPSky-FS.

The two layer architecture of CPSky-FS is presented in Figure 6.2. The model includes two layers – upper layer is prefilter layer and bottom layer is selection layer. The Prefilter layer of CPSky-AA includes only clustering and pruning, whereas an additional step of feature selection using PCA for QoS parameter selection performed before clustering step in the case of CPSky-FS.

The QoS values of the required WS are obtained from the user. The QoS dataset represents values of QoS parameters associated with candidate WSs. In the first phase, prefiltering is applied on the candidate WSs and the selection of top-K WSs is followed in the next step.

6.2.1.1 Prefilter Layer: In this section, the details of Prefilter layer are discussed. The task of prefilter layer is performed in three steps. At first, the feature selection using PCA is applied to identify major contributing QoS parameters. Secondly, the candidate WSs are clustered based on the QoS information. Third, on the clustered set of candidate WSs, pruning is applied to separate out WSs below the user expectations.

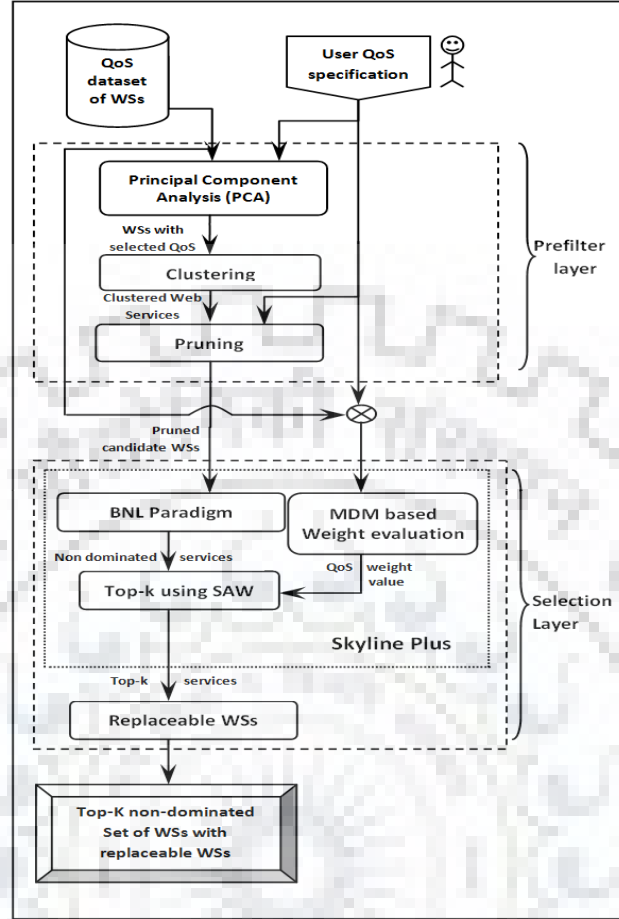


Figure 6.2: Proposed system architecture of CPSky-FS approach

a. QoS parameters selection using PCA

The PCA technique is applied on the set representing QoS values of WSs. Equation 6.1 is used to select QoS parameters. From equation 6.1, it is clear that smaller value of η_i indicate lesser contribution of QoS feature component. Therefore, removing such component η_k from equation 6.1, will not affect the result of feature selection. Moreover, it can be analyzed that if a QoS parameter is less important for feature selection, it is also less important in original space [Song et al., 2010]. Therefore, such QoS parameters can be ignored and remaining QoS parameters are selected by PCA for further processing. After applying PCA on available QoS dataset, six majorly contributing QoS parameters are returned. The selected QoS parameters are used by the subsequent steps to perform service selection.

$$P_k = \eta^T * I_{\text{vect}} = \sum_{i=1}^n \eta_i * I_{\text{vect}_i} \quad 6.1$$

Where, n is number of WSs, a sample vector of candidate WSs $\eta = [\eta_1, \eta_2, \dots, \eta_n]^T$, and $I_{vect} = [I_{vect_1}, I_{vect_2}, \dots, I_{vect_n}]^T$. It can be observed from equation 6.1, that the contribution of any i^{th} QoS parameter is statistically obtained using the absolute value of η_i .

b. Clustering of Web Services

The conceptual view of the proposed approach is shown in Figure 6.3. At first, the end user requested QoS parameters $UQoS_1, UQoS_2, \dots, UQoS_m$ are represented as WS S' (shown as double circle blue dot in ServiceCluster1). Thereafter, the clustering is applied on the set of candidate WSs along with S' based on the QoS information. The cluster is identified to which the S' is clustered. All of the other web services in this cluster will have similar QoS characteristics. These WSs are further processed by the skyline module and the set of skyline WSs, $SL_1, SL_2, SL_3, \dots, SL_g$ (shown as double circled red dot in ServiceCluster1), is determined.

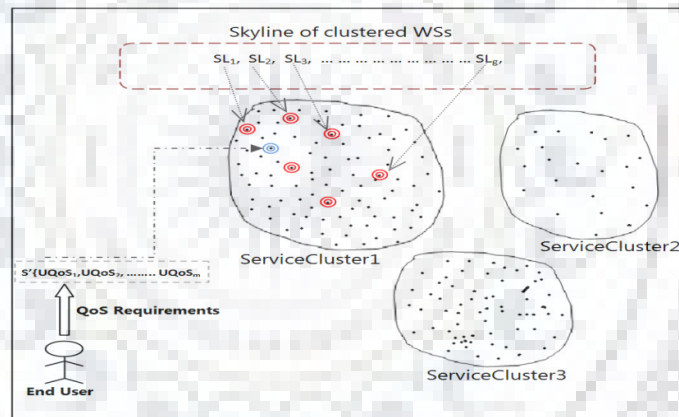


Figure 6.3: Conceptual view of proposed approach

The clustering step of the prefilter layer forms clusters of WSs according to the QoS parameter's value. Clustering is an unsupervised learning technique used to group elements that seem to fall naturally together [Witten et al., 2016]. The S' is also accepted as input by the algorithm. The cluster to which the S' is classified contains WSs having high QoS similarity with S' . WSs in this cluster are identified and the resulting WSs represented as tuple, as defined in Section 6.1. The procedure act as a prefilter for WSs and result into a set containing WSs, promising enough to meet the QoS requirements of the end user. The popular clustering methods include - K-Means clustering [Tewari et al., 2012b], Hierarchical clustering [Witten et

al., 2016], PAM [Zaki and Meira, 2014], SOTA [Zaki and Meira, 2014], Clara [Brock et al., 2008, Zaki and Meira, 2014], Diana [Brock et al., 2008], etc.

c. Pruning

The WSs which are not promising enough to meet the end user expectations of QoS are removed by pruning module. In order to perform the task of pruning, QoS parameters of the candidate WSs are compared with the QoS specifications provided by the end user. If there are two services $S_i(QoS_1, QoS_2, \dots, QoS_m)$ and $S'(UQoS_1, UQoS_2, \dots, UQoS_m)$, where, S' represents the WS corresponding to the user requested QoS and S_i is the i^{th} candidate WS under consideration, the pruning step removes S_i from the candidate list, if every $S_i\{QoS_k\} < S'\{UQoS_k\}$, where, $1 \leq k \leq m$. In other words, the pruning step compares the QoS values of each candidate WS against the user specified QoS. If any candidate WS has all QoS parameter values strictly less than the QoS values as desired by the end user, then the WS is removed from the candidate WS list. The basis of this step is that, if none of the QoS parameter value of the candidate WS is above the user expectations, the WS is having a very low probability of satisfying the end user expectations. Hence, such WS can be removed from the list.

The pruning improves the performance of WSS system by removing the services even if the services are potential candidates for the skyline. Consider, for example, the set of eight WSs from S_1 to S_8 with QoS values, as shown in the Table 2. The QoS parameters - Response Time (RT), Latency (LT), Error Rate (ER) and Availability (AV) values are also depicted. The skyline approach is used to obtain the set of WSs forming the skyline. The status of WSs, whether they are part of the skyline or not is also shown in the Table 2 (the last column). The skyline based approach returns services S_1, S_2, S_3, S_4 and S_5 as a set of services in response to the user request. Service S_3 is dominating S_6 and S_7 . Similarly, S_8 is being dominated by S_4 .

Now, if the end user request is for the WS having UQoS (5, 21, 30, 45) representing QoS parameter values for RT, AV, ER and LT, respectively, then the skyline based WS selection module should not return S_1 and S_3 in response to the requested QoS. However, for this example, the skyline based technique includes S_1 and S_3 services in the skyline. It is because the skyline technique does not consider the end user request during skyline computation. One solution to this problem is to apply the pruning process before applying skyline based WSS to remove all such WSs which are having all QoS values below the demanded QoS values. It will also help in improving the overall quality of the skyline.

The clustering and pruning step result in an improvement in the overall quality of selection in two ways. First, by considering only most relevant WSs by the selection module. Second, a significant reduction in the number of WSs to be considered by the selection module, which results in improved time efficiency of WS selection. Therefore, the WS selection with Prefiltering is a better solution as compared to WS selection without Prefiltering.

Table 6.2: Example web services with four QoS parameter values

WS	QoS parameter				Present in the Skyline
	RT (msec)	AV (%)	ER (%)	LT (msec)	
S ₁	6	20	40	50	Yes
S ₂	5	21	70	40	Yes
S ₃	6	16	34	70	Yes
S ₄	5	22	35	60	Yes
S ₅	4	20	40	60	Yes
S ₆	7	16	40	70	No
S ₇	11	15	50	80	No
S ₈	10	14	70	60	No

6.2.1.2 Selection layer:

The selection layer has a responsibility of identifying WSs meeting the end user expectations of QoS in the best way. This layer selects the optimal set containing WSs with best QoS offerings from the group.

Selecting ranked list of top-K services:

The task of service selection is performed using extended Skyline approach, we call it as Skyline Plus. The basic Skyline approach generates the unordered non-dominated set of WSs. The proposed Skyline Plus approach extends it to include the steps using SAW and MDM to generate ranked list of WSs.

The clustered set of services obtained from prefilter layer act as input to the selection layer. As a first step to perform selection of WSs, the proposed Skyline Plus technique is used to determine skyline WSs. Services in the skyline are non-dominated in terms of QoS parameter values and hence ensured to be best among the group. The cluster to which the S' is clustered,

only services in that cluster are considered by the skyline module. Thus, the skyline services are determined according to the QoS demand from the end user. After this step, we have a non-dominated set of WSs without their relative ranking. Now, these services used by SAW and MDM based process to generate list of top-K WSs based on QoS parameters. The weight values are used to signify the relative importance of QoS parameters. As mentioned earlier, the quality of selected WS depends on the weight values. If imprecise values of weights are specified, the desired WS may miss from the selection results. In order to reduce the system dependency on the end user to obtain the weight values, we have used a mathematical model based on Maximizing Deviation Method (MDM) [Wu and Chen, 2007]. Let there are 'm' number of QoS parameters associated with each web service. The difference of the performance values of each web service with reference to all associated QoS parameters is computed using deviation method. The deviation of any web service S_i to all other web services can be defined using equation 6.2 and 6.3 [Wu and Chen, 2007].

$$Dev_{ij} = \sum_{r=1}^n \left(\delta(S_i\{QoS_j\}) - \delta(S_r\{QoS_j\}) \right)^2 \quad (6.2)$$

$$Dev_j(w) = \sum_{i=1}^n \sum_{r=1}^n \left(\delta(S_i\{QoS_j\}) - \delta(S_r\{QoS_j\}) \right)^2 \quad (6.3)$$

Where, Dev_{ij} represents the deviation value of i^{th} WS w.r.t. j^{th} QoS parameter. The $Dev_j(w)$ is the deviation value of all WSs to other WSs corresponding to j^{th} QoS parameter. QoS_{ij} represents the normalized value of j^{th} QoS parameter of WS S_i . Equation 6.4 represents the non-linear programming model of MDM to evaluate the weight of j^{th} QoS parameter [Wu and Chen, 2007].

$$w_j = \frac{\lambda_k * \sum_{i=1}^n \sum_{r=1}^n \left(\delta(S_i\{QoS_j\}) - \delta(S_r\{QoS_j\}) \right)^2}{\sum_{j=1}^m \sum_{i=1}^n \sum_{r=1}^n \left(\delta(S_i\{QoS_j\}) - \delta(S_r\{QoS_j\}) \right)^2} \quad (6.4)$$

s.t. $w_j \geq 0$, $\sum_{j=1}^m w_j = 1$. Where, λ_k is the weight factor for k^{th} QoS parameter, and is specified by the expert. For conducting experiments we have chosen $\lambda_k = 1/m$. to signify equal weightage of each QoS parameter during this calculation. The hybrid weight evaluation scheme is proposed to calculate the values of weights of each QoS parameter. The hybrid weight values are determined by averaging the values of weights obtained from MDM and the weights specified by the end user, if any.

After the values of weight for each QoS parameter is evaluated, SAW method is used to determine the cumulative QoS score of each WS. Based on the score, top-K WSs are selected

by the system. Hence, with the help of MDM and SAW, the ranked list of top-K WSs is obtained.

b. Selecting Replaceable Service:

At runtime the possibility of change in QoS or failure of selected web service cannot be ignored. Thus, in advance the replaceable web services are also selected by the proposed system. For each of the top-K selected services, the replaceable WSs are obtained by using Pearson correlation coefficient. For each skyline WS the similarity with other skyline WSs using pearson correlation coefficient is determined. Equation 6.5 is used to find service similarity Sim between service S_i and S_r , ($S_i, S_r \in Skyline$), based on pearson correlation coefficient [Rhimi et al., 2015].

$$Sim(S_i, S_r) = \frac{m * \sum_{j=1}^m [S_i\{QoS_j\} * S_r\{QoS_j\}] - \sum_{j=1}^m S_i\{QoS_j\} * \sum_{j=1}^m S_r\{QoS_j\}}{\sqrt{[n * \sum_{j=1}^m S_i\{QoS_j\}^2 - (\sum_{j=1}^m S_i\{QoS_j\})^2] * [n * \sum_{j=1}^m S_r\{QoS_j\}^2 - (\sum_{j=1}^m S_r\{QoS_j\})^2]}} \quad (6.5)$$

Where, m is number of QoS parameters, $S_i\{QoS_j\}$ is j^{th} QoS parameter of i^{th} WS. The WS with highest value of correlation coefficient is considered as replaceable WS. The process of obtaining replaceable WS is repeated for each WS in the skyline.

6.2.2 Web service selection algorithm

The CPSky-FS algorithm for selection of WS is depicted in Figure 6.4. The algorithm starts by taking WSs QoS data ($Data[][]$), the QoS requirements of the end user ($UQoS[]$ forms the service S'), values of weights (optional) from end user ($wUser$), and ' K ' representing the number of top-K WSs to be retrieved as inputs (Line 1). The Prefilter layer is invoked by passing two parameters to the *Prefilter* algorithm (line 2). The Prefilter layer filters out those WSs which are not potent enough to meet the end users' expectations of QoS. The Prefiltered set of WSs are passed to the selection layer by calling SkylinePlus algorithm (line 3). At selection layer SkylinePlus algorithm identifies non-dominated set of WSs followed by top-K selection. For the top-K selected WSs, the replaceable WSs are obtained by using pearson correlation coefficient (line 4). The algorithm terminates by returning top-K selected WSs.

The algorithm to Prefilter WSs is presented in Figure 6.5. The *Prefilter* algorithm at selection layer accepts two parameters – the WSs QoS dataset ($Data[][]$) and QoS requirements of the end user ($UQoS[]$). The QoS dataset of WSs is normalized to keep the values in the range (0...1) (line 2). The Prefilter layer has three steps (line 3-7). The PCA technique is used to select majorly contributing QoS parameters (line 3). The UPGMA clustering algorithm is applied to cluster the WSs (line 4 & 5). UPGMA algorithm is an

agglomerative hierarchical clustering algorithm. QoS information of WSs obtained after feature selection is used during cluster formation (line 4). The service S' also participate in clustering process and the cluster to which S' is clustered, is marked as *userCluster*. All WSs in the cluster *userCluster* are stored separately in the *clusteredWS[][]* array (line 5). From the cluster, the index of WSs is determined and the WSs with all features are used from the normalized set (line 6). The feature selection is used only for clustering the WSs. The pruning step is applied with regard to S' . The pruning step prunes the normalized QoS dataset with reference to S' . The pruned WSs are stored in *prunedWSs[][]* array (line 7). Finally the algorithm returns pruned WSs (line 8). The filtering done at Prefilter layer ensure that the WSs which are congruous with the end user requested QoS are further processed.

Input: *Data[][]*: QoS data of WSs as array with column as QoS parameters and each row represents individual WSs.
UQoS[]: QoS requirements of the end user as 1D array.
wUser[]: The weight preference of the end user, if any (as array). The weight values are optional and if not specified / partially provided, CPSky-FS algorithm will evaluate them.
 'K': representing top-k services to be selected.

Output: Top-K matching Web Services

Begin:

1. CPSky-FS(*Data[][]*, *UQoS[]*, *wUser[]*, K)
2. *PFdata[][]* = Prefilter(*Data[][]*, *UQoS[]*) // Call to Prefilter function for Prefiltering WSs. Prefilter Layer
- // Call SkylinePlus to find non-dominated WSs followed by ranking and selection of top-K WSs. QoS dataset and UQoS. Selection Layer
TopK_{match} = SkylinePlus(*PFdata[][]*, *Data[][]*, *UQoS[]*, K, *wUser[]*)
- 3.
4. *findReplaceable(TopK_{match})*
5. *return TopK_{match}*

End:

Figure 6.4: CPSky-FS algorithm for web service selection

At selection layer, the Skyline Plus algorithm takes pruned set of WSs and finds top-K WSs. The algorithm is shown in Figure 6.6. The SkylinePlus algorithm is called by passing five parameters (line 1). The selection layer incorporates three functionalities - Determining skyline set of WSs, evaluating weight values of QoS parameters, and finding top-K WSs. The skyline set of WSs is obtained by using block nested loop paradigm (line 2). The skyline set consists of non-dominated set of WSs. No ordering on these non-dominated set of WSs is imposed by default. In order to find the priority ordering of WSs, we have MDM based hybrid weight

scheme with SAW (line 3-5). Once the WSs are prioritized, the top- K services are retrieved from the skyline set (line 6). The top- K services are returned by the algorithm (line 7).

In the proposed algorithm we have performed clustering first and then pruning. This is because the clustering is performed once and pruning is required to be done with every end user request.

Input: *Data* [][]: QoS data of WSs as array with column as QoS parameters and each row represents individual WSs,
UQoS []: QoS requirements of the end user as 1D array.

Output: Pruned set of WSs.

Begin:

1. *PreFilter*(*Data*[][], *UQoS*[])
 // Normalize the data and end user requested QoS.
2. (*NData*[][], *nUQoS*[]) = *norm.startNormalize*(*Data*[][] \cup *UQoS*[])
 // Apply PCA to select QoS parameters.
3. *FeatureData*[][] = *PCA*(*NData*[][])
 // Use hierarchical clustering, UPGMA, to determine WSs clusters.
 Identification number of cluster to which the end user request is mapped, is stored in *userCluster* variable. The *clusteredWS* matrix stores the clustered WSs and cluster identification number to which WSs belongs to.
4. (*ClustData*, *userCluster*) = *hcCluster.start*(*FeatureData*[], *nUQoS*[]);
5. *clustWSIndex*[][] = *readClusteredData*(*ClustData*, *userCluster*);
 // Replace with WSs with all attributes.
6. *clusteredWS*[][] = *findClusteredWSs*(*clustWSIndex*[][], *NData*[][])
 // Prune candidate WSs with reference to QoS requirements specified by the end user.
7. *prunedWSs*[][] = *prune*(*clusteredWS*[], *nUQoS*[]);
 // The pruned set of WSs are returned and act as input to the selection layer
8. return *prunedWSs*

End:

Figure 6.5: Algorithm for prefiltering web services (Prefilter layer)

6.3 EVALUATION OF THE PROPOSED APPROACH

In this section, an objective approach to assess the performance of the proposed approaches is presented. A machine with Intel Core i7 CPU 3.4 GHz, Windows 7 platform is used to conduct experiments related to evaluation of the proposed WSS approach. The CPSky-AA and CPSky-FS algorithms are developed using Java.

Input: *PFdata[][]*: Prefiltered set of web services obtained from Prefilter layer.
UQoS[]: QoS requirements of the end user.
wUser[]: The weight preference of the end user, if any. The weight values are optional and if not specified / partially provided, the algorithm will evaluate them using MDM based hybrid weight evaluation scheme.
K: representing top-k services to be selected
Data[][]: represents WSs QoS dataset

Output: Index of top-K WSs selected in the Skyline.

Begin:

1. *SkylinePlus(PFdata[][], Data[][], UQoS[], K, wUser[])*
// Determine non-dominated set of WSs using block nested loop paradigm.
2. *Skyline [] = BlockNestedLoop(PFData[][])*
// Evaluate weight of QoS parameters using MDM method.
3. *MDMwt [] = findWeightMDM(Data[][], UQoS[])*
// Evaluate Hybrid weight values of QoS parameters
4. *hybridWeight [] = findHybridWt(wUser[], MDMwt[])*
// Using SAW and MDM based hybrid weights of QoS parameters, determine
//QoS score of each skyline WS and return index of skyline WSs.
5. *skyServIndex[] = SimpleAditiveWeight(Skyline[], hybridWeight[]);*
// Find top-K WSs using SAW method
6. *topK[] = findTopK(skyServIndex []);*
7. *return topK []*

End:

Figure 6.6: SkylinePlus algorithm for top-K WSs selection (Selection layer)

6.3.1 Performance Parameters

In order to evaluate the performance of the proposed approaches and compare with the existing approaches, performance evaluation parameters such as, time for selection of WSs, hardness, Euclidean distance, satisfaction score, and user satisfaction are used. Time, Hardness and Euclidean distance performance parameters are explained in Chapter 2. The satisfaction score based performance parameter is redefined with MDM based weight evaluation.

1. Satisfaction Score: The satisfaction score of a service is the measure of ability of the service to meet the QoS requirements desired by the end user. The satisfaction score of a WS is obtained by evaluating satisfaction score of individual QoS parameter associated with the web service. The satisfaction score of individual QoS parameter and a web service is defined using equation 6.6 and 6.7 [Stephen and Yin, 2011].

$$SatScore_{i,j} = \begin{cases} 0.5(2 * S_i\{QoS_j\} - 1)^{1-w_j} + 0.5, & \text{if } S_i\{QoS_j\} > 0.5 \\ 0.5(-2 * S_i\{QoS_j\} + 1)^{1-w_j} + 0.5, & \text{if } S_i\{QoS_j\} \leq 0.5 \end{cases} \quad (6.6)$$

$$SatScore_{S_i} = \sum_{j=1}^n w_j * SatScore_{i,j} \quad (6.7)$$

Where, $SatScore_{i,j}$ is the satisfaction score for j^{th} QoS parameter of i^{th} WS, $SatScore_{S_i}$ is the satisfaction score of i^{th} WS and w_j is the weight of j^{th} QoS parameter obtained using hybrid weight evaluation scheme based on MDM method.

The query satisfaction score is obtained from the weighted sum of satisfaction scores of top-k WSs and is defined using equation 6.8.

$$SatScore_{Query_i} = \sum_{j=1}^k wt_j * SatScore_{S_j} \quad (6.8)$$

Where, $SatScore_{S_j}$ represents the satisfaction score of j^{th} WS from among the selected top-k services and wt_j is the weight of j^{th} top-k service. The weight of each of the top-k selected service is obtained using rank-sum method [KalisZewski and Podkopaev, 2016].

Users Satisfaction: The user satisfaction $User_{SS_{i,j}}$ is a measure of how well the end user requested QoS are satisfied by the selected web services. Usually, there are multiple QoS associated with web services and for each of the QoS parameter $UQoS_j$, for $1 \leq j \leq m$, the end user must specify her preference. To obtain the user satisfaction score, following three conditions must be fulfilled:

If the user does not specify any preference of QoS for

- i) $UQoS_j$, then, during ranking of services, services' qualities on $UQoS_j$ will not be taken into account by the system.
- ii) If user does specify QoS preference for each of the QoS parameter $UQoS_j$, the system should rank those services higher for which the QoS values are above the QoS specified by the end user.
- iii) Similarly, the service ranking will be low if few of the QoS values are lower than as expected by the end user.

Condition (ii) ensures that services which satisfy all of the QoS requirements are ranked higher than the web services satisfying partial requirements of the end user. As per the formula in (6.6), the services with value of QoS > 0.5 , are considered to be potent enough to meet the end user requirements of QoS. However, in actual practice this may not be true. For a WS with all

QoS parameters just above 0.5 (e.g. $QoS_j = 0.51$) is considered as WS with high satisfaction score. But, the user expectations of QoS may be much higher than 0.5. Further, the user satisfaction will be more for the service having QoS higher than the expected but close to the user requested QoS [Masri and Mahmoud, 2007]. Based on the above requirements and discussion, equation 6.9 and equation 6.10 are derived as an extension from equation 6.6 and equation 6.7 [Stephen and Yin, 2011]. From equation 6.9, it is clear that the services satisfying condition (ii) above and having QoS matching the QoS needs of the end user closely are ranked higher. Therefore, as this quantity $|QoS_{i,j} - UQoS_j| \approx 0$, the selected service closely matches the end user requirements and is ranked higher [Hao et al., 2012].

$$User_{SS_{i,j}} = \begin{cases} 0.5 * (2 * (1 - |S\{QoS_j\} - S'\{UQoS_j\}|) - 1)^{1-w_j} + 0.5, & \text{if } S\{QoS_j\} \geq S'\{UQoS_j\} \\ 0.5 * (-2 * (1 - |S\{QoS_j\} - S'\{UQoS_j\}|) + 1)^{1-w_j} + 0.5, & \text{if } S\{QoS_j\} < S'\{UQoS_j\} \end{cases} \quad (6.9)$$

$$SatScore_{WS_i} = \sum_{j=1}^n w_j * User_{SS_{i,j}} \quad (6.10)$$

Where, $User_{SS_{i,j}}$ is the user satisfaction score for j^{th} QoS parameter of i^{th} WS, $SatScore_{WS_i}$ is the satisfaction score of i^{th} WS, the weight of j^{th} QoS parameter w_j is obtained using hybrid weight evaluation scheme, and WS_{User_j} is the j^{th} QoS parameter of the web service corresponding to the QoS concern of the end user. Overall satisfaction score of a query can be obtained using equation 6.8.

6.3.2 The Dataset and design of user queries

To conduct experiments, the QoS dataset [Masri and Mahmoud, 2007], which is based on QoS measurements of real world web service is used. The currently available version of the QWS includes a set of 2,507 WSs with QoS measurement of each parameter. The dataset consists of 2,507 rows and 11 column entry each separated by commas. Each row corresponds to the QoS parameter value of the individual candidate WS. The first nine column values are QoS measurements of - Response Time (R_t), Availability (A_v), Throughput (T_h), Successability (S_u), Reliability (R_l), Compliance (C_o), Best Practices (B_p), Latency (L_a) and Documentation (D_o), QoS parameters, respectively. Next column represents the name of the service. The value in the last column is the URL of the WSDL of the WS. The details of QWS dataset is discussed in detail in Section 2.5.1 of Chapter 2. It is observed that each QoS parameter value is in different

range. Each of the QoS parameter is normalized in the range [0..1] using equation 6.11 [Zhao et al., 2014].

$$QoS'_j = \begin{cases} \frac{S\{QoS_j\}_{max} - S\{QoS_j\}}{S\{QoS_j\}_{max} - S\{QoS_j\}_{min}}, & \text{if } QoS \text{ parameter is decreasing type} \\ \frac{S\{QoS_j\} - S\{QoS_j\}_{min}}{S\{QoS_j\}_{max} - S\{QoS_j\}_{min}}, & \text{if } QoS \text{ parameter is increasing type} \end{cases} \quad (6.11)$$

The experiments are conducted using the set of hundred end user queries representing QoS demands from the end user. The queries are evenly distributed across the QoS range of the QWS dataset with hardness level from level 5 (L5, in the range 41- 50) to level 10 (L10, in the range 91-100). The hardness level of queries is obtained using equation 2.1 and 2.2 (Section 2.6 of Chapter 2). Six queries are chosen randomly from each level of hardness and shown in Table 6.3. The effectiveness of the proposed approach is assessed by conducting various experiments as discussed in next section.

Table 6.3: The end user queries with different QoS demands

Query No.	Query	Hardness Level (in %)
1	87.7, 96.5, 25.5, 349.9, 91.7, 100, 96.4, 12.7, 95.1	L ₁₀ (90-100)
2	6.13, 93.37, 41.8, 97.97, 80.3, 100.46, 87.75, 0.25, 93.56	L ₉ (80-90)
3	40.43, 90.4, 42.92, 97.86, 69.97, 100.2, 83.35, 0.52, 69.2	L ₈ (70-80)
4	144, 89, 3.3, 75, 80.8, 88.9, 77, 2.3, 67	L ₇ (60-70)
5	892.8, 41, 33.8, 44.4, 57, 64.2, 84.1, 1.9, 69.9	L ₆ (50-60)
6	139.14, 26.58, 34.7, 40.15, 42.3, 83.86, 65.5, 1238.95, 50.7	L ₅ (40-50)

6.3.3 Results and evaluation

The experimentations are performed by using QWS dataset. The main contribution lies in the proposed CPSky-FS approach, prefiltering using clustering followed by pruning method, and hybrid weight evaluation using MDM. The section demonstrates the improvements caused by the MDM, Clustering and pruning. The results of proposed approach variants CPSky-AA and CPSky-FS are compared with the baseline approach S-Sky [Yu and Bouguettaya, 2013] and its variant PSky. In S-Sky [Yu and Bouguettaya, 2013], the Skyline set of WSs is obtained and the best service from the Skyline is chosen in response to the end user request. The variant PSky is evolved from S-Sky by applying pruning before S-Sky. The results of PSky demonstrate the improvements caused by the use of pruning. Each of the four algorithms (S-Sky [Yu and

Bouguettaya, 2013], PSky, CPSky-AA, CPSky-FS) is applied and tested on the same QoS dataset to select few top most services. Number of topmost WSs to be selected is controlled by an external parameter 'K' and hence it has some effect on the satisfaction score of QoS needed by the end user.

Effect of variation in 'K': The effect of variation in the value of 'K' on satisfaction score is measured for each of the four approaches and is presented in Figure 6.7. For calculating the satisfaction score for an approach for a given value of 'K', the satisfaction score for a user query is determined as per the following steps:

Step 1: The user query is clustered and Skyline services are obtained, using algorithm in Figure 6.5.

Step 2: For each of the Skyline service, a cumulative QoS score using simple additive weight (SAW) method [KalisZewski and Podkopaev, 2016] is calculated.

Step 3: Based on the score, WSs are ranked and, top-k web services are selected.

Step 4: Weight of each of the top-k service is determined using rank sum method [Gang and Chunli, 2015].

Step 5: Satisfaction score for the user query is determined using (8), (9), and (10).

The satisfaction score value in Figure 6.7 is obtained by repeating the steps 1 to 5 for 100 user queries and averaged.

In Figure 6.7, variation in 'K' values is presented on X-axis and Y-axis represents mean satisfaction score measured over hundred queries. It is analyzed from the figure that the proposed approach CPSky-FS has maximum satisfaction score for all values of 'K'. The

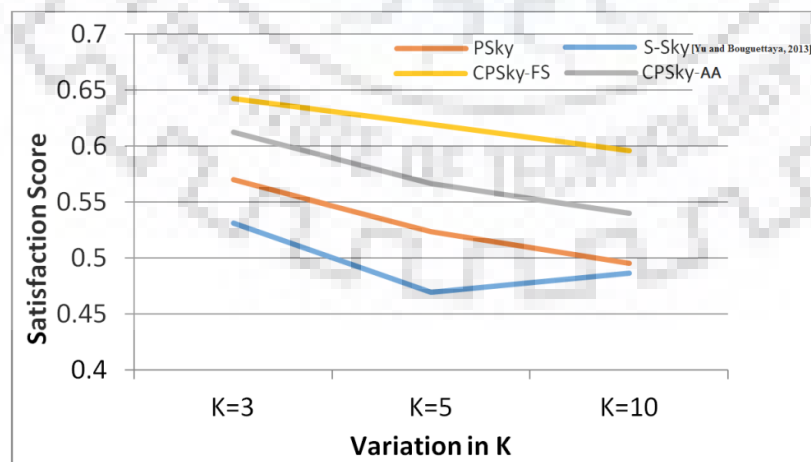


Figure 6.7: Effect of variation in 'K' on satisfaction score

satisfaction score obtained for each of the approach is decreasing with increase in the value of 'K'. The satisfaction score of S-Sky approach is observed to be minimum among all approaches for all values of 'K'.

Effect of variation in query hardness: In Figure 6.8, the effect of increase in hardness of queries on satisfaction scores of each of the four approaches is presented. Each satisfaction score represents the average value of satisfaction score obtained for all user queries in the different hardness level L_i ($5 \leq i \leq 10$). Hardness levels are shown on X-axis, whereas the satisfaction score appears on Y-axis. Experiments are conducted for obtaining top-10 web services ($K=10$). It is observed that generally S-Sky and PSky have similar satisfaction score, however, for hard queries with hardness level 80-100%, PSky shows slightly higher satisfaction score than the S-Sky approach. It is primarily due to the reason that the pruning step filters out WSS having values of all QoS parameters below the QoS expectations of the end user. As explained through an example in Section 6.2.1.1b, the pruning step does not consider the service if none of the QoS requirements specified by the end user is met by that service. Thus, it can be observed that the satisfaction score for CPSky-AA and CPSky-FS approach is more than the existing S-Sky approach. It is primarily due to the use of pruning and clustering based prefiltering which identifies the closest matching set of web services during prefiltering phase.

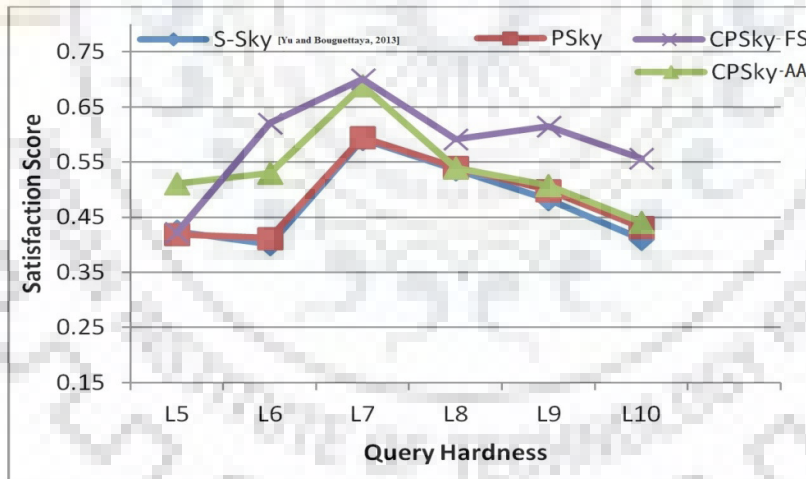


Figure 6.8: Satisfaction score vs query hardness (K=10)

Parameters for Comparison of proposed and existing Web Service Selection approaches

(i) **maximum satisfaction score:** We have compared maximum satisfaction score (MSS) achieved by proposed approaches, CPSky-AA and CPSky-FS, with MSS obtained for existing works WSS [Lim et al., 2012, Stephen and Yin, 2011, Wang et al., 2015b] (refer Table 6.4). In CPSky-AA and CPSky-FS, the MSS values are obtained for $K=10$. The satisfaction score for

100 user queries is calculated and the maximum value is chosen. In the work [Stephen and Yin, 2011], the satisfaction score is obtained by comparing QoS value of individual QoS parameters with the threshold value of 0.5. The selected WS with QoS value more than threshold is considered as WS with higher ability to satisfy end-user's requested QoS. The other methods which consider user requested QoS for calculation of satisfaction score is presented by [Lim et al., 2012, Wang et al., 2015b]. In [Lim et al., 2012], additional parameter such as revenue of slave WS is considered to determine satisfaction score. It is observed from Table 6.4 that maximum satisfaction score is attained by CPSky-AA (MSS 0.762545) and CPSky-FS (MSS 0.867598).

Table 6.4: Comparison of Maximum satisfaction score

Author	(MSS)	Threshold (Decided by)	WS QoS Dataset
[Stephen and Yin, 2011]	0.643	Fixed (0.5)	QWS [Masri and Mahmoud, 2007]
[Lim et al., 2012]	0.73	User request, Revenue	QWS [Masri and Mahmoud, 2007]
[Wang et al., 2015b]	0.693	Variable w.r.t user request	Randomly Generated
CPSky-AA	0.762545	Variable w.r.t user request	QWS [Masri and Mahmoud, 2007]
CPSky-FS	0.867598	Variable w.r.t user request	QWS [Masri and Mahmoud, 2007]

(ii) Mean satisfaction score: In Figure 6.9 (for $K=10$) a box-plot analysis for mean satisfaction score evaluated on 100 queries for four WSS approaches is presented. It is observed that the highest user satisfaction score is obtained for CPSky-FS approach. The mean value of the user satisfaction score is highest for the proposed CPSky-FS algorithm. The highest and lowest value of user satisfaction score is also higher for CPSky-FS approach. The CPSky-AA approach also shows mean user satisfaction score better than S-Sky and PSky approaches. The CPSky-AA approach has highest value of lowest user satisfaction score.

The fact is also confirmed by observing the results of multiple comparison test shown in Table 6.5a and 6.5b. In Table 6.5a, mean, maximum and standard deviation values of mean satisfaction score measured (over 100 observations) for S-Sky, PSky, CPSky-AA, and CPSky-FS approach are shown. From Table 6.5a, it is observed that the mean value of satisfaction score is obtained highest by using CPSky-FS approach followed by CPSky-AA approach. The

mean satisfaction score of S-Sky and PSky approach is on lower side. The maximum value of satisfaction score is attained by CPSky-FS approach where as the Maximum satisfaction score of S-Sky, PSky, CPSky-AA is almost similar and lower than CPSky-FS approach.

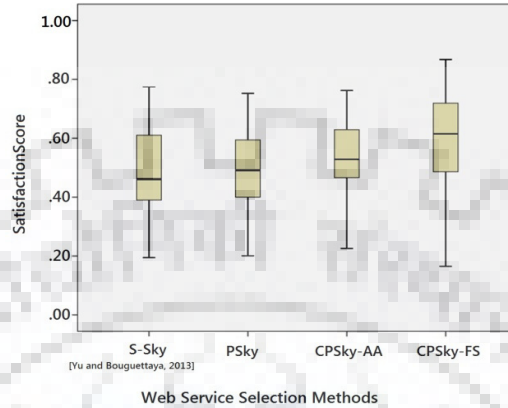


Figure 6.9: Box plot analysis of satisfaction score for four web service selection approaches

From the Table 6.5b based on the p-values, it is observed that

1. There is significant improvement in the performance of CPSky-AA over S-Sky and PSky.
2. A large improvement in the performance of CPSky-FS over S-Sky, PSky, and CPSky-AA is observed.
3. S-Sky and PSky approaches are not significantly different. Also, CPSky-AA and CPSky-FS are also found to be performing differently.
4. Overall, the CPSky-FS approach is found to be performing significantly better and is different than other WSS approaches.

Table 6.5a: Minimum, maximum, and standard deviation value for mean satisfaction score measure of four WSS approaches.

Approach	Observations	Mean	Maximum	Standard deviation
S-Sky	100	0.4863	0.77	0.13547
PSky	100	0.4955	0.75	0.12938
CPSky-AA	100	0.5399	0.76	0.13267
CPSky-FS	100	0.5958	0.87	0.17193

Table 6.5b: Tukey multiple comparison test for measured satisfaction score

Treatments	Abs. Diff*	p-value	Result
S-Sky vs PSky	0.00916	0.969	No Sig. Diff.
S-Sky vs CPSky-AA	0.00157*	0.000	Sig. Diff.
S-Sky vs CPSky-FS	0.10945*	0.000	Sig. Diff.
PSky vs CPSky-AA	0.04441	0.128	No Sig. Diff.
PSky vs CPSky-FS	0.10030*	0.000	Sig. Diff.
CPSky-AA vs CPSky-FS	0.05588	0.031	Sig. Diff.

*The absolute difference is significant at the 0.05 level. Sig. Diff. = Significant Difference, Abs. Diff. = Absolute Difference.

(iii) **Mean weighted Euclidean distance:** Each of the four approaches are evaluated on weighted Euclidean distance parameter. The mean plot is drawn and shown in Figure 6.10. The Euclidean distance is measured for 100 user queries calculated using (7). Each value of weighted Euclidean distance in Figure 6.10 is obtained by taking mean over 100 values. Lower the value of Euclidean distance better is the approach. It can be observed that, the CPSky-AA attains the lowest value of mean weighted Euclidean distance followed by CPSky-FS, S-Sky and PSky approach in increasing order of weighted Euclidean distance. The highest value of mean weighted Euclidean distance is attained by PSky due to the fact that it select WSs having minimum one QoS parameter value higher than the user requested QoS. The S-Sky and PSky approaches select non-dominated web services without considering the end user requested QoS. This leads to two possibilities. Firstly, the selected WSs are having QoS far away from the requested QoS. Secondly, the selected WS do not satisfies any of the QoS requirements as specified by the end user. In second case, although, the web service is part of the Skyline, it is not checked whether the selected web service satisfies the end user requirements or not. The possibility of arriving second case is more for the hard requirements of QoS by the end user i.e., the queries in the hardness level range of 70% to 100%. For CPSky-AA and CPSky-FS algorithms, the possibility of arising case one above is handled by using the concept of clustering. The selected web services are from the cluster to which the end user request is mapped. Similarly, the case two is avoided by the pruning step of Prefiltering stage. Pruning and clustering concept together select the WSs with high quality with due consideration to the end user requested QoS.

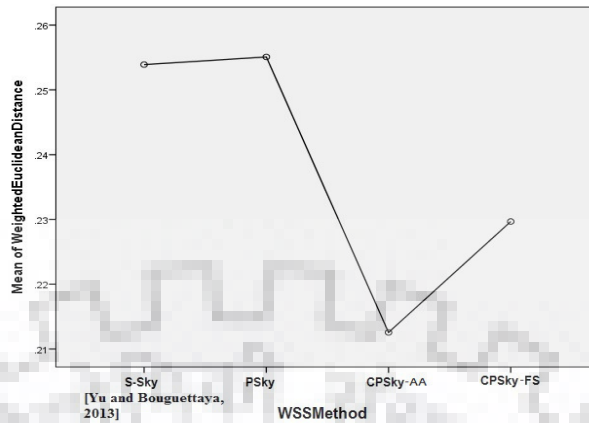


Figure 6.10: Mean plot of Euclidean distance for four WSS approaches.

(iv) *Time for selection of web services*: Figure 6.11 presents a comparative evaluation of four approaches based on the total CPU time taken for selection of top-k WS. The experiments on the time taken by each approach for top-k selection of WSs are conducted for hundred user queries. The average value of CPU time taken is obtained and experiments are repeated for variations in 'K' for values 10, 5, and 3. The variation in the values of 'K' are represented on X-axis and Y-axis represents the overall time taken (in msec) by each approach.

From Figure 6.11, few important observations drawn are as follows:

- i) The time taken by each of the four approaches is not much affected by the variation in the value of K .
- ii) The time taken by PSky and CPSky-AA approach is almost same. The PSky approach takes lesser time than S-Sky approach due to the pruning step. Pruning step avoids the unnecessary processing of WSs in selection step. Due to clustering of WSs in CPSky-AA approach, it takes slightly higher time than PSky approach.
- iii) The time for WSS is highest for S-Sky approach and is significantly less for CPSky-FS in all four approaches. The lowest time taken by CPSky-FS is due to significant reduction of number of services due to prefiltering and reduced number of QoS parameters.
- iv) For each value of 'K' it can be easily observed that among the four WSS approaches, the CPSky-FS has performed well and takes lowest time for selection of top-k WSs.

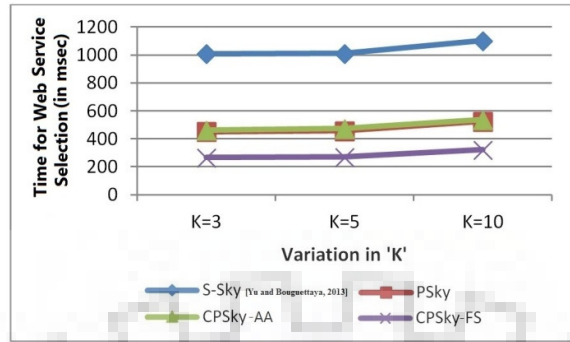


Figure 6.11: Time comparison of four web service selection approaches

A comparative study of existing state-of-the-art with proposed CPSky-FS approach is presented in Table 6.6. It is observed from the table that most of the existing approaches perform clustering during WS discovery, but the proposed approach uses clustering during WSS phase for atomic task. The majority of the available works uses WSDL, Ontology, user tags, etc., for performing clustering, however, the proposed CPSky-FS approach uses QoS information of WSs to cluster them. The existing state-of-the-art do not provide any in depth analysis of

Table 6.6: A comparison of existing state-of-the-art with proposed approach for web service selection

Author	Clustering done during		Clustering Criteria	Evaluation of clustering	Weight Evaluation using MM	Pruning	Prefiltering before selection
	Web Service						
	Selection	Discovery					
[Ismaili, 2012]		✓	WSDL	✗	✗	✗	✗
[Zheng et al., 2012b]		✓	Users preference	✗	✗	✗	✗
[Wu et al., 2014]		✓	WSDL, Tags	✗	✗	✗	✗
[Liu et al., 2011]		✓	Ontology	✗	✗	✗	✗
[Zhang et al., 2013]	✓		QoS Expectations, service rating	✗	✗	✗	✗
[Xia et al., 2011]		✓	QoS	✗	✗	✗	✗
[Margaris et al., 2015]	✓		Past usage of WS	✓	✗	✗	✗
CPSky-FS	✓		QoS	✓	✓	✓	✓

AT = Atomic task, CT = Composite task, MM = mathematical model

clustering techniques before using them. On the contrary, before selecting the clustering technique in CPSky-FS, various clustering techniques are evaluated on different performance parameters alongwith consideration to feature selection. None of the existing approaches make use of pruning and prefiltering to identify useful WSs in congruous with the end user expectations of QoS. The proposed CPSky-FS uses pruning and prefiltering for the stated purpose.

With variation in query hardness, the satisfaction score of CPSky-FS approach is found to be in the range of 0.422 to 0.701. The satisfaction score of CSS approach (presented in chapter 4) is found to varying in the range from 0.443 to 0.636 with query hardness from L_1 to L_{10} . in each hardness level. With the variation in the values of K (K=3, 5, 10), the time taken by CSS approach is higher than the time taken by CPSky-FS approach. However, the mean Euclidean distance of CSS approach (0.09, for K=3) is found to be much lesser than CPSky-FS These values are obtained by averaging the value of satisfaction score over 10 random user queries approach (0.228, for K=3). So, it is revealed from the experiments that the approach CPSky-FS is comparable with CSS approach. In the case when labeled dataset is not available, clustering based CPSky-FS approach can be followed.

6.4 DISCUSSION

This chapter presents an approach known as CPSky-AA and its variant CPSky-FS for selection of WSs. The CPSky-AA and CPSky-FS approach is useful for the case when the QoS dataset is unlabelled. CPSky-AA approach is preferable for the case when number of QoS parameters are less and all QoS parameters have major contribution. If number of QoS parameters are very large in number and only few of them are having major contribution, CPSky-FS approach is preferred. From the statistical analysis, the performance of CPSky-AA and CPSky-FS is found to be better than existing similar approaches. Both the approaches uses pruning and clustering to prefilter candidate WSs. The weight evaluation of each QoS parameter using MDM ensure that the top-k selected services does not includes those service having only one or two non-dominated parameters.

The performance evaluation of two proposed approaches is done by conducting various experiments. CPSky-AA and CPSky-FS approach has improved satisfaction score as compared to S-Sky approach [Yu and Bouguettaya, 2013]. This is due to the use of prefiltering layer before selection and the use of hybrid weight evaluation based on MDM method. Also, the mean satisfaction score of CPSky-AA and CPSky-FS is on higher side. Moreover, the mean

Euclidean distance of CPSky-AA is found to be better than CPSky-FS approach. On the basis of CPU time taken for WSS, the CPSky-FS outperforms all other similar approaches. It is evident from experimentation that CPSky-AA and CPSky-FS approaches have edge over the existing approaches and both approaches demonstrate better performance in selection of WSs.

6.5 CONCLUSIONS

The proposed CPSky-AA and CPSky-FS approaches are thoroughly tested by using QoS dataset of real world WSs. The experimental results based on Euclidean distance, satisfaction score and CPU time show that the proposed clustering based prefiltering used for web service selection is effective and it find the web services which closely satisfy the end user expectations of QoS. The improvements in the average Euclidean distance of WSs selected by CPSky-FS and CPSky-AA with reference to the end user request is observed. The proposed approaches, CPSky-AA and CPSky-FS, outperforms other existing skyline based web service selection approaches.

The approaches presented in Chapter 4 and Chapter 6 for WS Selection are based upon the classification and clustering, respectively. These works mainly deal with the problem of handling large number of functionally similar web services and consider the end user's requested QoS requirements during WSS. One of the other important factor, especially in the case of WSS for composite service, is replaceability of selected WSs. This factor is of utmost importance in the scenario when runtime failure of WS is of concern. So, in the next chapter we have presented a replaceability based WS Selection approach with due consideration to WS failure. The work presented in this chapter has been published as [Purohit and Kumar, 2016b, Purohit and Kumar, 2018e, Purohit and Kumar, 2018g].

CHAPTER 7

REPLACEABILITY BASED APPROACH FOR WEB SERVICE SELECTION

The web service based systems allows the Internet companies to scale their business. The success of business depends on how well the service perform and the service reliability. The web services operate in the environment where there are high chances of failure. The problem of unavailability/failure/change in QoS of web service is very critical and badly affects the performance of web service based systems. The previous chapters of this thesis mainly provide the solution in the form of prefiltering to get improved WSS and also presents the methods to provide WSS dependent upon the QoS requested by the end user. But, for the composite WS scenario, the failure of one service exploits the whole service. This may requires the rollbacking and restart of the process of selection, which is a costly operation. So, in this case selection of replacement WS for each of the selected WS can be highly useful. The objective of, this chapter is to deal with this problem by providing a replaceability based WSS approach. The prime advantage of the approach is that the web service selection is done in parallel to the selection of a similar web service. In case of any run time failure/change in QoS/unavailability experienced by the web service under execution, the equivalent web service act as a replacement. The replaceability is determined by using functional and non-functional parameters associated with the web service.

The problem of WSS for composite WS and need of replaceable WS is introduced in Section 7.1. Proposed approach and motivating example appears in Section 7.2. In Section 7.3, the algorithm for the developed approach followed by evaluation and compared with existing state-of-the-art is presented in Section 7.4. The Chapter conclusions appers in Section 7.5.

7.1 INTRODUCTION AND MOTIVATION

The required business functionality for a simple task is obtained directly from the available WS. If the available WS act as a non-decomposable single unit than in this case the WS is known as atomic web service. For complex tasks, two or more atomic WSs are required to be

combined in some logical order, are called non-atomic or composite WS. In other words, each composite WS can be visualized as number of tasks executing in some order. The non-functional characteristic and required functionality of each task is represented in abstract form as abstract WS (AWS). To meet the functional and non-functional requirements of each AWS, a large number of candidate services are competing. Each of the competing WS is known as concrete WS (CWS). The task of composition is to determine the most optimal combination of CWS to match the QoS constraints in the best way.

Let the composite web service internally comprise of 'n' AWS: $AWS_1 \dots AWS_n$. For each AWS, AWS_i ($1 \leq i \leq n$), maximum 'm' CWS are competing with each other. Each of the CWS, S'_j , ($1 \leq j \leq m$) has 'h' associated QoS parameters. Figure 7.1a represents a WS composition scenario. From each of the AWS_i , a CWS S'_j is to be selected such that the service composition together satisfies the end-to-end QoS requirements of the end user (uQoS). One example selection of CWS $S'_1 \dots S'_n$ is shown in Figure 7.1a. The selection of CWSs satisfying the uQoS in best way can be done by applying brute force approach. However, the approach is time inefficient [Helal and Gamble, 2014]. Thus, an approach to maintain balance between time taken for selection and optimality of solution is required.

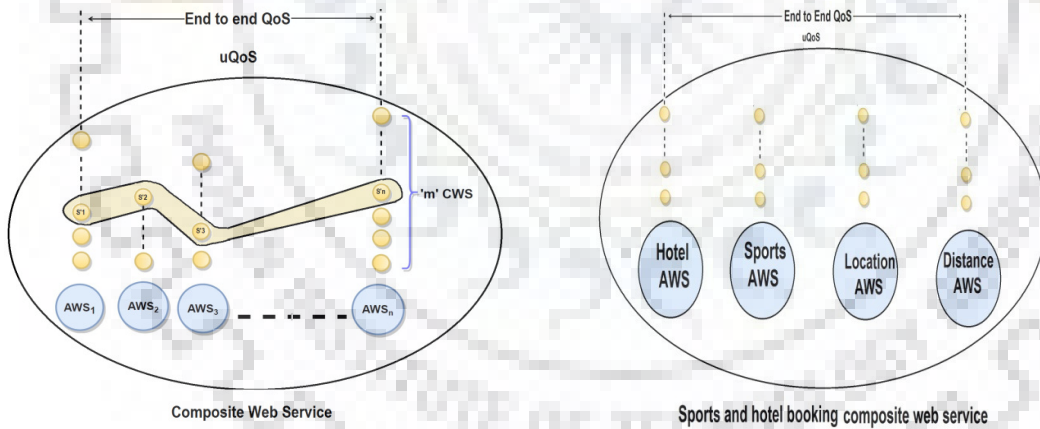


Figure 7.1a : Web Service composition scenario

Figure7.1b: Example composite WS

Consider for example, a person is looking for sport (surfing/hiking) activity. The city to which the sport activity is available, a hotel is to be booked in that city. The distance of the place of sport activity is to be determined from the hotel. The latitude/longitude information of the place of sport activity is available in advance. However, the latitude/longitude of hotel is to be obtained depending on the hotel booked. The composite WS for this situation can be

represented as shown in Figure 7.1b. Selection of most appropriate candidate WS each from hotel, sport, location and distance is to be done such that the end-to-end QoS requirements are satisfied.

There are few existing solutions for selection of services to perform composite task are - Skyline [Ouadah et al., 2015], Mixed Integer Programming (MIP) [Saleem et al., 2015], genetic algorithm (GA) [Helal and Gamble, 2014], Linear programming [Cho et al., 2016], Hill climbing [Ai and Tang, 2008], etc. The availability and reliability are two important building blocks to realize any web service based system. Due to the dynamic environment of web service execution, it becomes very challenging to ensure availability and reliability of web service based system. The operating environment of WSs may lead to run time failure of selected WS, change in the values of QoS offered by the service, unavailability of WS due to system or network failure/excess load, etc. The available solutions and their variants are efficient in selection of WSs. However, the issue of WS failure/unavailability is either ignored by the existing WSS methods or very few solutions are available which are not efficient.

Primarily, the available solutions to deal with WS failure obtain substitutable WS by considering QoS parameters only. The process of WS discovery determines all WSs in a specific domain. The discovered set of services provides the functionality related to the domain of search. But, the exact matching at the level of functional behaviour of the WS is usually ignored. The functional behaviour of the WS can be described using input, output, precondition and effect (IOPE) information. Thus, IOPE can be used as one of the factor to determine replaceable/ substitutable WS.

There are many external contributory factors on which the quality offered by the service depends. Few important among them are network load, load of server on which WS is deployed, hardware capabilities, network condition, client environment, etc. Therefore, the offered quality by WS changes at run time or service become unavailable. Similarly, due to uncontrolled factors, the WS experiences run time failure. Consider the service composition scenario in Figure 7.2. P is the point of failure in the composition. Part P_1 of the composition has been executed and part P_2 of the composition still remains to be executed. If a runtime failure of the WS at point P is experienced by the system, then the system can recover from the failure by selecting the equivalent WS or complete/partial composition. Four possibilities to recover from the WS failure are as below:

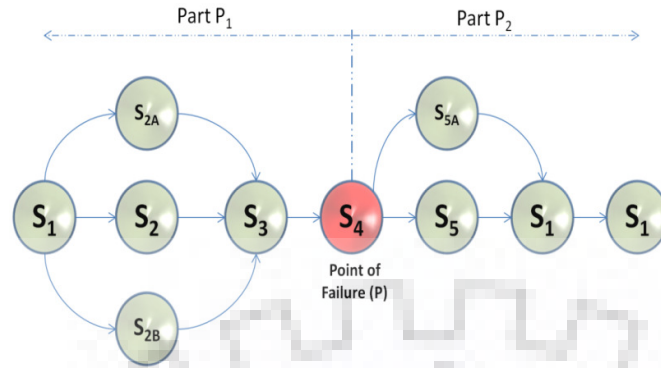


Figure 7.2: Pictorial representation of failure point in the composition. Part P_1 is completed. P is the point of failure in the composition and part P_2 of the composition is yet to be executed.

First, the execution of composition plan done so far (part P_1 in Figure 7.2) can be rolled back and a new composition plan is generated. This step is time consuming, as new composition plan is to be generated at run time, possibly, multiple number of times. Other problem is that WSs are accessed on pay per use basis. So, those services which are invoked so far involve cost and rolling back these services will be costly and complex.

Second, a backup composition plan can be prepared in parallel to main composition plan. However, the number of backup plans required cannot be decided unanimously for each service composition. Moreover, the roll backing of already executed part of the composition is still required.

Third, from the point of failure in the composition, the remaining part of the composition plan can be generated at run-time. In Figure 7.2, this situation is represented using point P and part P_2 of the composition. In this solution, no roll backing of part P_1 is required. However, it may be difficult to meet the end-to-end QoS with new composition. Moreover, the number of possibilities of various sub-compositions to be checked for part P_2 depends on the point of occurrence of failure. Number of service combinations to check is reduced, if occurrence of failure is towards the end of the composition. Whereas, if service failure is towards the starting point of composition, then number of possible combinations are relatively high. Thus, earlier occurrence of failure is worse than the occurrence of failure towards the end of the composition.

Fourth, the equivalent WS to replace failed service can be determined. Only, the failed WS is replaced and rest of the composition is followed as per the original selection. In Figure 7.2,

only service for P is replaced, part P_1 and part P_2 of the composition remains same. In this case, the replacement WS must be chosen with due consideration to end-to-end QoS parameters. This approach is followed in this work.

The existing works on finding replacement WS due to unavailability of any service at runtime can be divided into two categories. In first category, the algorithm determines the replacement service on the occurrence of the failure [Yin et al., 2009, Bhuvaneshwari and Karpagam, 2012]. This increases the overall response time of the WS. In other category, the replaceable services are determined during the selection of WSs [Helal and Gamble, 2014]. This increases selection time, but at the time of failure or unavailability of WS, immediate replacements are available. This can be done by considering replaceability factor during service selection process. The existing techniques for service selection [Yan et al., 2015, Yin et al., 2009, Bhuvaneshwari and Karpagam, 2012, Ernst et al., 2006, Wijesiriwardana et al., 2012], lacks in terms of considering replaceability factor during service selection. Further, the work on WSS using GA [Helal and Gamble, 2014] does not consider functional behaviour of the WSs for obtaining replaceable WS.

In this work, the problem of WSS for composite WS is considered. The failure of WSs at runtime is handled by the proposed system. The selection mechanism is modified such that along with the selection of concrete services as part of composition, backup services for each concrete WS is also obtained. Following factors are taken into account for obtaining replaceable WSs:

- (i) End-to-end QoS is not affected
- (ii) Replaceability factor of each concrete web service is counted towards replaceability of composition plan
- (iii) A hybrid fitness function based on end-to-end QoS and plan replaceability is used.

The important contributions of the chapter are addressed as follows:

1. The runtime failure of concrete WSs is handled using replaceable WSs.
2. Concrete WSs involved in composition are selected using modified genetic algorithm. Each iteration of GA evaluates fitness of individual in the population by considering modified fitness function.
3. The replaceability of individual service and cumulative services is assessed using the score obtained from PROMETHEE Plus proposed in Chapter 4 and IOPE based matchmaking.

This reduces the repetitive calculations required for obtaining replaceability count. The determinant method is used to perform IOPE matchmaking using proposed rank concept.

4. The proposed approach is compared with the existing similar approaches on satisfaction score, CPU time required to perform matchmaking and selection, replaceability factor, and precision parameters.

The proposed work improves system availability by suggesting the alternate equivalent WS as a replacement for any WS participating in composition and undergoing a run time failure. The repeated reckoning for obtaining replaceability count in every loop of the GA is avoided by introducing PROMETHEE Plus method [Purohit and Kumar, 2018a]. The experimental results also confirm that the proposed approach is better than existing state-of-the art.

7.2 PROPOSED APPROACH FOR WEB SERVICE SELECTION

The task of service selection is challenging and time consuming. The average time taken by selection process further increases due to run-time failure of services involved in the composition. This section discusses various components involved in the architecture of the proposed system to deal with run-time failure of services.

7.2.1 Web Service Selection

A composite WS consists of two or more than two WSs combined in some logical order. An example of composite WS is a tour planning WS (TPWS). The TPWS internally consisting of WSs from various domains such as Transportation (flight booking, cab booking), Hotel (hotel booking), Geomatic (distance calculation, location identification, map service), Payment (payment gateways), etc. The WSs belonging to these domains are selected and composed together to form TPWS. One example of WS selection for composite WS (CWS) is shown in the Figure 7.3. Services S_1 S_n , represents abstract WSs and $S_{1,1}$, $S_{1,2}$, ..., $S_{1,m}$ are concrete WSs. The process of WSS for composite WS require to select one candidate WS for each abstract WS by ensuring that the end-to-end QoS is satisfied by the selected set of WSs. Usually a bottom-up approach is followed to perform the selection. In this case, the use of genetic algorithm (GA) is suitable to perform selection of WSs [Liu et al., 2016b, Helal and Gamble, 2014]. GA based WSS is one of the highly explored work and is performing better than other

approaches [Purohit and Kumar, 2018c, Jatoth et al., 2017, Garriga et al., 2015, Yilmaz and Karagoz, 2014, Fister et al., 2013, Fethallah, 2012].

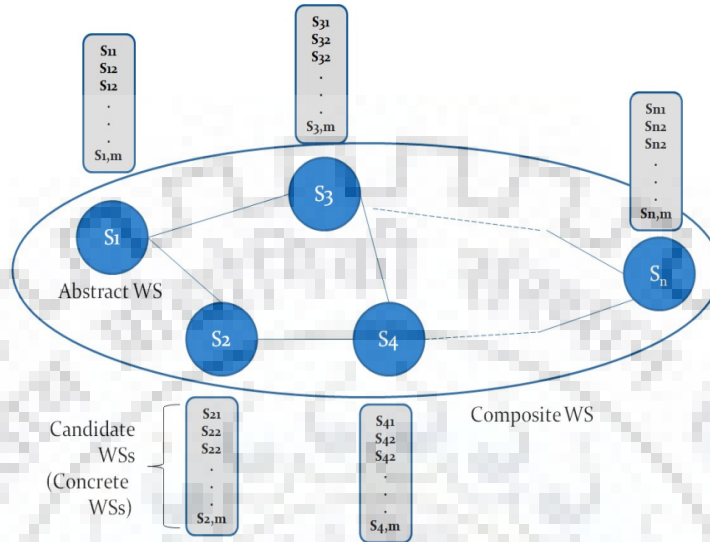


Figure 7.3: Example of web service selection for Composite web service

In this work, we have followed a bottom-up approach to perform selection using modified GA (mGA). The mGA is evolved from basic GA by introducing fitness evaluation based on QoS and replaceability factor. The proposed modified GA algorithm is explained in Section 7.2.4. The proposed PROMETHEE Plus [Purohit and Kumar, 2018a] approach is used to determine replaceability of each CWS. Based on replaceability factor, the fitness of each chromosome is determined and the chromosomes with highest fitness are considered further. We have proposed two variations of WSS system. First variation is obtained by performing WSS using GA and finding replaceable WSs using PROMETHEE Plus method ($WSSR_P$). $WSSR_P$ uses QoS values to obtain replaceable web service.

The architecture of the proposed system $WSSR_P$ is shown in Figure 7.4. It is assumed that functionally equivalent concrete WSs corresponding to each AWS are available as a result of discovery process [Liu et al., 2011]. QoS parameters of each concrete WS is fetched from the service database. The end-to-end QoS requirements are obtained from the end user. The task of normalization is performed to bring each QoS parameter in the range (0...1). This ensures uniformity among QoS parameters for further calculation. On competing concrete WSs, PROMETHEE Plus is applied to obtain a priority order of concrete WSs on NoF score. The

concrete WSs are stored in the corresponding list (L_1, L_2, \dots, L_n), where, 'n' is number of AWS. The concrete WSs in each list L_i are present in the order of priority. By taking lists $L_1 \dots L_n$ as input, the modified GA is applied on the candidate WSs to obtain the optimal composition. In each iteration of GA, the replaceability factor is calculated by utilizing the order of ranking of concrete WSs generated from PROMETHEE Plus. Based on replaceability factor and QoS parameters, the fitness of each chromosome is evaluated. With the multiple repetition of the mGA loop, with due consideration to replaceability factor, the most suitable set of WSs are selected.

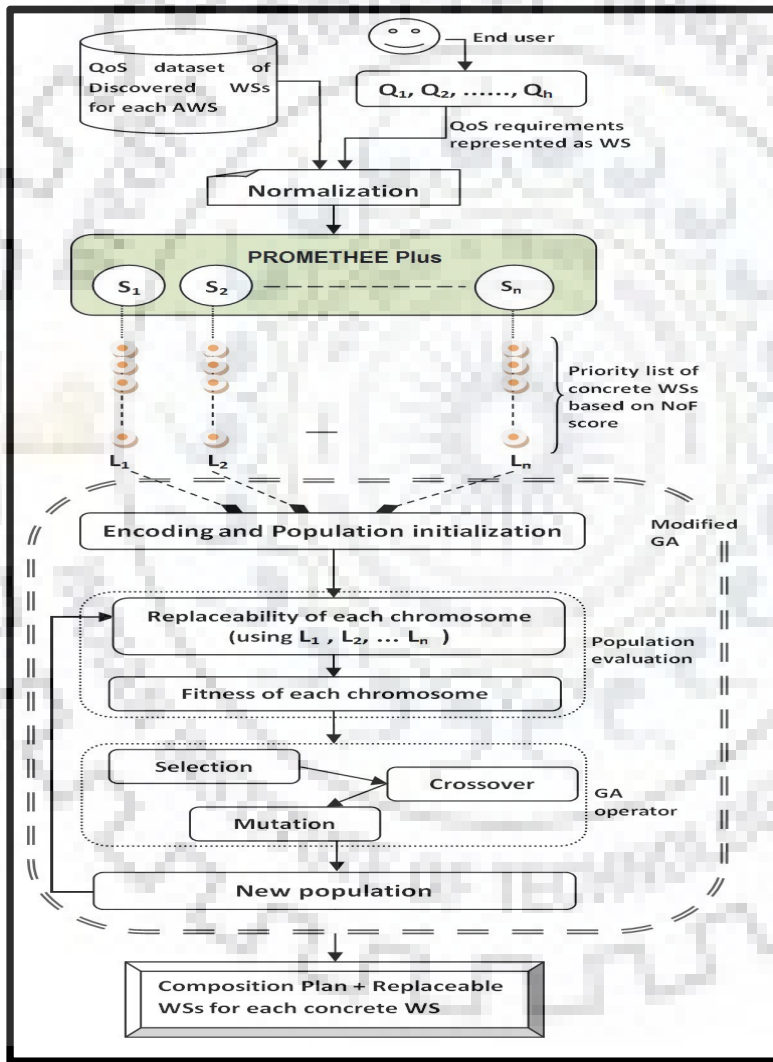


Figure 7.4 Architecture of the proposed $WSSR_p$ Approach.

In the second variation, GA is used to perform WSS as before and PROMETHEE Plus along with IOPE based matchmaking ($WSSR_{PM}$) is used to obtain replaceable WS. In Figure 7.5, architecture of $WSSR_{PM}$ approach is presented. In $WSSR_{PM}$ approach, replaceable WSs are obtained by combining the results of QoS based score obtained using PROMETHEE Plus and service similarity score obtained using IOPE based matchmaking. Based on these two scores, the service replaceability is determined. This replaceability information is utilized in each loop of mGA to determine the set of WSs matching the end-to-end QoS. The IOPE matchmaking is carried out using determinant method. Due to the time efficiency of determinant method, it is preferred over existing approaches of IOPE matchmaking.

7.2.2 QoS based Replaceability of Web Services

The replaceability is the degree of a service to be replaceable by other WS. To determine the replaceability of a WS, two web services are compared on their QoS properties. The comparison among QoS parameters is not trivial. The QoS parameters are conflicting in nature. Thus, to compare WSs on conflicting QoS parameters, we have used PROMETHEE Plus method. The replaceability factor of a WS is defined as the number of WSs which can replace the failed/unavailable WS. Applying PROMETHEE Plus method on concrete WSs results in a list of services sorted on Net Outranking Flow (NoF) score. An example result obtained from PROMETHEE Plus method is shown in Figure 7.6. The services on the left hand side and right hand side of the reference service WS_{ref} have NoF higher and lower than the NoF of reference service, respectively. The service(s) with NoF equals the NoF of reference service can be found in the list on LHS. The significance of services with higher or equal NoF score is that these services can replace reference service based on QoS score. Services with equal NoF indicate that they exactly substitute each other and services with higher NoF suggest the close replacement. While services with lower NoF cannot act as a replacement of reference service and hence should not be considered as substitute during replacement process. The replaceability factor can be obtained by counting number of services above the reference WS, i.e., number of WSs having NoF equal to or greater than NoF of reference WS.

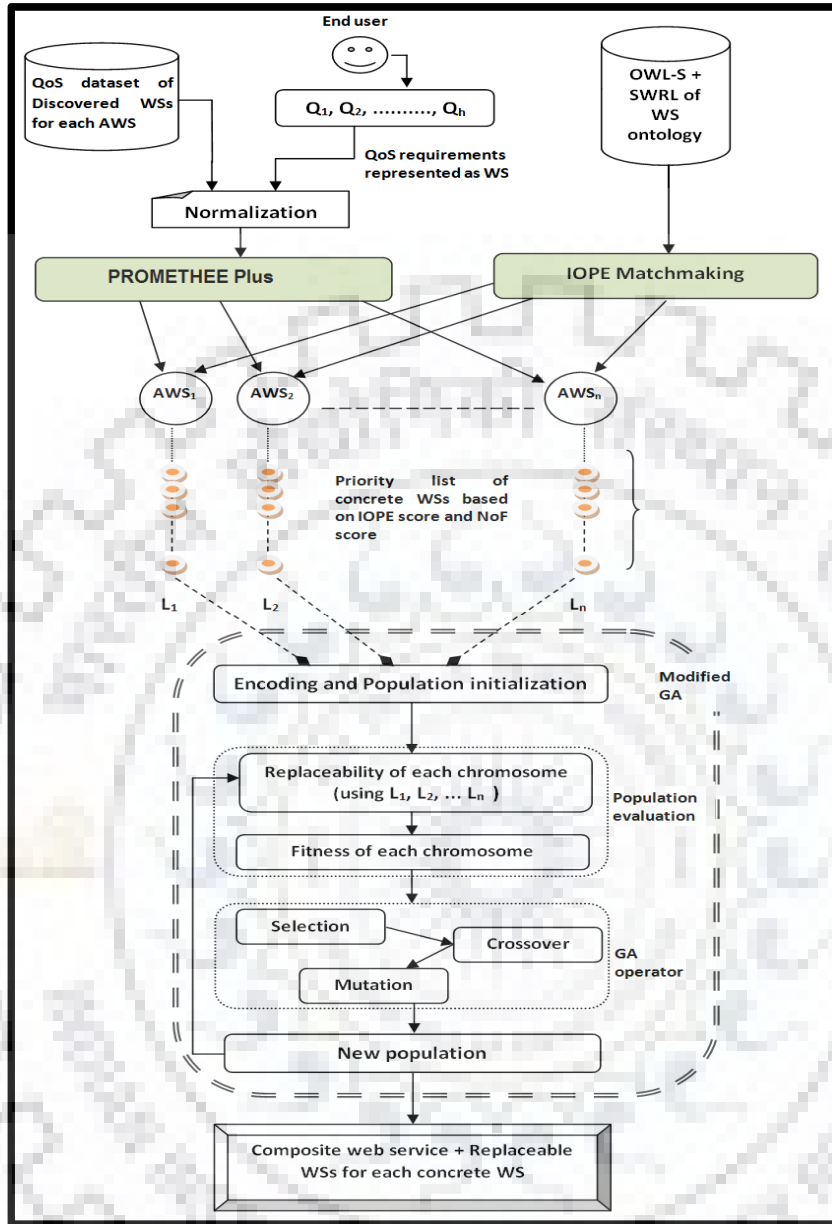


Figure 7.5: Architecture of the proposed $WSSR_{PM}$ Approach

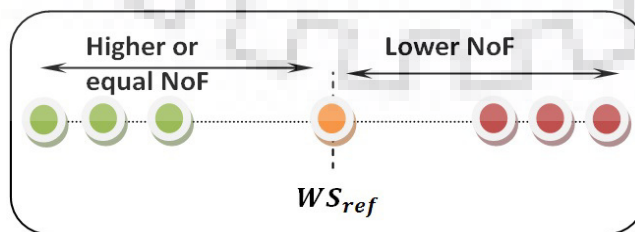


Figure 7.6: Sorted list of service generated as output by applying PROMETHEE Plus method on concrete WSs. NoF score is used to sort WSs.

7.2.2.1 PROMETHEE Plus method

The PROMETHEE method is a multi-criteria decision making (MCDM) based method which evaluates objects by considering conflicting criteria. The PROMETHEE Plus is evolved by performing three changes in the basic PROMETHEE [Purohit and Kumar, 2018a]. First, the end user request of QoS is included with candidate WSs to obtain NoF score. Second, the Gaussian function is followed to perform comparison among WSs. Third, the highest priority is given to the WS having closest match of NoF score with that of end user requested QoS. Further, the value of weights of each QoS parameter is determined by considering MDM model. The weight values of QoS parameters specified by the end user are also considered by this approach. A hybrid weight evaluation is done by taking into account the values of weights obtained from the end user, if any, and calculated using MDM method. The partial preference of weight values is also handled. The process of prioritization is repeated for set of concrete WSs associated with each AWS. As a result of applying PROMETHEE Plus method, a ranked list of concrete WSs is generated. The ranking of concrete WSs is done with respect to reference service. The highest rank is assigned to the concrete WS having net outranking flow (NoF) exactly matching with the NoF of *reference service*. The closest match concrete WS is assigned next higher rank, and so on. In our case, the *reference service* is the concrete WS selected by GA to take part in composition.

7.2.3 IOPE matchmaking based Replaceability of Web Services

The semantic description of a web service can be provided using service profile, service model and service grounding. The Ontology Web Language for Services (OWL-S) is useful in achieving this objective [Bhatt et al., 2013, Atrey et al., 2012]. Service profile gives a description of the web service about what a web service does. The service model describes how the stated functionality is achieved, i.e. how the web service does it. The service grounding part of the OWL-S details how to access the web service.

The service profile carries the semantic details of web services. These details are provided by the service provider. Among the other details, functional description in web service profile provides details of Input, Output, Precondition and Effect (IOPE) [Pro, 2016]. Input represents details of input to the web service. The output represents the details of the output generated by the web service. Preconditions are logical condition needs to be true before execution of the web service. The effect describes the changes take place after the web service execution is over. Consider the example of hotelReserve service. The service reserves hotel in the given city

of the given country for the specified time period mentioned as duration. These three act as input (I) to the web service. The details of hotel and other booking details are produced as output (O) of the service. The precondition (P) is that the city should be located in the country. As the web service execution is over, the hotel must be reserved as a result (E).

To determine the similarity among two WSs, the semantic matching between them can be performed. The semantic matching of WSs is primarily based on IOPE. The matchmaking module performs matchmaking between IOPE parameters of the candidate service (WS_{req}) and advertised WS (WS_{adv}). This situation is presented in Figure 7.7. As it is clear from Figure 7.7, all 'I' parameters of WS_{req} are compared with all 'I' parameters of WS_{adv} , and the best matching score is determined. Similarly, O, P, and E parameters are compared and best matching is obtained.

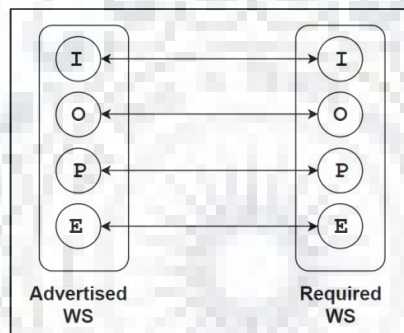







Figure 7.7: IOPE matching to determine WS similarity

Further, the Input(s) of desired WS is (are) identified and it forms a set $Input_{req}$. Similarly, Input(s) of advertised web service is (are) known as $Input_{adv}$. These two set forms the weighted bipartite graph. The weight values are obtained by applying semantic matchmaking on the elements of two sets on pair wise basis. Depending on the results of matchmaking, the match type is determined. Five match types are defined with numerical scores as shown in Table 7.1 [Pukkasenung et al., 2010]. Once the weight values are evaluated, the bipartite graph based maximum matching algorithm is applied to determine overall maximum matching between two sets.

Table 7.1: Match type defined in comparison of Input parameters

S. No.	Match Type	Set representation	Description	Score	
i.	EXACT	$Type_{adv} = Type_{req}$ if $\forall x[x \in Type_{req} \leftrightarrow x \in Type_{adv}]$	Advertisement and request are same	4	
ii.	PLUGIN	$Type_{req} \subset Type_{adv}$ if $\forall x(x \in Type_{req} \rightarrow x \in Type_{adv})$	The requirement is a subset of advertisement.	3	
iii.	SUBSUME	$Type_{adv} \subset Type_{req}$ if $\forall x(x \in Type_{adv} \rightarrow x \in Type_{req})$	Advertisement is a subset of requirement.	2	
iv.	INTERSECT	$Type_{req} \cap Type_{adv} = \{x \mid x \in Type_{req} \text{ and } x \in Type_{adv}\}$	Advertised and required web services have few parameters in common.	1	
v.	FAIL	$Type_{req} \cap Type_{adv} = \emptyset$.	No matching between advertisement and required WS.	0	

Definition 1: *The maximum matching.* Let $G=(U, V, E)$ is a weighted bipartite graph obtained after applying semantic matchmaking, where, U and V is a partition in graph G , such that $U \cap V = \emptyset$, E is edges set; a matching M of the bipartite graph G is a non-null subgraph of G , such that any edges of M are not adjacent in G . If any matching M' of the bipartite graph, $|M'| \leq |M|$, then, M is the maximum matching of G .

The best match algorithm maximizes the solution, i.e. summation of weights on the chosen edges of bipartite graph is maximized. Two parameters MatchScore and Rank of the solution are defined using equation 7.2 and 7.3.

$$MatchScore = \sum_{k=0}^n wt(v_k), \text{ where } v_k \in V \quad (7.2)$$

$$Rank = \min(v_k), \text{ where } v_k \in V \text{ and } k = 1 \dots n \quad (7.3)$$

Where, *MatchScore* is the sum of weights on the chosen edges of bipartite graph and Rank of the solution is decided by the minimum weight value of the edge involved in the evaluation of *MatchScore*. We have introduced the concept of rank to improve the overall matching of the solution.

Consider an example of bipartite graph generated after applying matchmaking algorithm as shown in Figure 7.8. Solution of maximum bipartite matching is $\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle, \langle u_3, v_3 \rangle$ with $MatchScore = 12$ and $Rank = 4$. The algorithm also ensures that highest ranked solution is chosen among the solutions with similar $MatchScore$. e.g. after applying the maximum matching algorithm on the bipartite graph of Figure 7.8, we have two solutions 'a' and 'b' such that the solution 'a' is having edges selected $(\langle u_1, v_2 \rangle, \langle u_2, v_3 \rangle, \langle u_3, v_1 \rangle)$ with the weights $(4, 4, 2)$ and solution 'b' is having edges selected $(\langle u_1, v_3 \rangle, \langle u_2, v_1 \rangle, \langle u_3, v_2 \rangle)$ with weights $(3, 3, 4)$. The solutions 'a' has Match Score of 10 and a rank of '2' and solution 'b' has Match Score 10 and rank '3'. Our approach gives preference to solution 'b' over solution 'a'. The ranking concept improves the overall matching precision of web services.

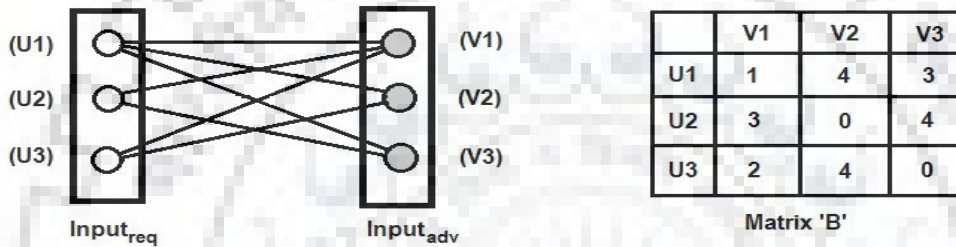


Figure 7.8: A bipartite graph obtained after matchmaking $Input_{req}$ and $Input_{adv}$.

Problem Formulation: The problem of matchmaking is modelled as maximum weight matching using bipartite graph. The weighted matching in bipartite graph problem can be defined as follows:

A weighted bipartite graph G represents semantic matching between elements of two sets, is a tuple $G = (U, V, E, w)$, where $|V| = 2 * k$. The sets U and V forms two vertex sets of bipartite graph G and are given by $U = \{u_1, u_2, \dots, u_k\}$ and $V = \{v_1, v_2, \dots, v_k\}$. U represents IOPE of the required WS and V represents IOPE of advertised WS. E denotes the edge set of bipartite graphs, $E \subseteq U \times V$. w is a weight function ascribes weights on the edges. W is the maximum weight in w . In the maximum weight bipartite matching problem, we determine matching 'M' in weighted bipartite graph G such that total weight

$$w(M) = \sum_{e \in M} w(e), \text{ is maximized [Sankowski, 2009] and the rank 'r' of the solution,}$$

$$r(M) = \prod_{\min} w(e), \text{ where } e \in M, \text{ is maximized.}$$

7.2.3.1 Determinant method

The determinant method can be used to determine maximum weight matching of a bipartite graph. There are few variants available to evaluate determinant. The recursive approach to evaluate determinant and hence finding a maximum matching can be done using formulae in equation 7.4 using Laplace rule [Quarteroni and Saleri, 2014].

$$\text{Det}(B) = \begin{cases} b_{11} , & \text{if } k = 1 \\ \sum_{j=1}^n \Delta_{ij} * b_{ij}, & \text{for } k > 1 \end{cases} \quad (7.4)$$

Where 'B' is the square matrix obtained by performing matching of I/O/P/E between $(I/O/P/E)_{\text{req}}$ and $(I/O/P/E)_{\text{adv}}$, respectively, and evaluating the weights as above. $\Delta_{i,j} = (-1)^{i+j} \det(B_{i,j})$ and $B_{i,j}$ is the matrix obtained by eliminating the i -th row and j -th column from matrix B. The recursive algorithm approach is going to take $O(k!)$ which is not suitable for large value of k . An efficient solution to the problem of obtaining maximum weight matching 'M' for bipartite graph using the determinant method is presented in [Sankowski, 2009]. The presented work assumes non-negative weights and works in $O(Wk^w)$, where 'w' is the matrix multiplication exponent and W ($=4$ in our case) is highest edge weight. The matrix 'B' of Figure 7.7 is used to define polynomial matrix B' for weighted bipartite graph $G = (U, V, E, w)$ as shown in equation 7.5.

$$B'(G, x)_{i,j} = \begin{cases} Z_{i,j} X^{w(v_i, v_j)}, & \text{if } v_i, v_j \in E \\ 0 & , \text{Otherwise} \end{cases} \quad (7.5)$$

Where, $Z_{i,j}$ corresponds to i^{th} element of set 'U' and j^{th} element of set 'V'. $w(v_i, v_j)$ corresponds to weight values on the edges from node $v_i \in U$ to node $v_j \in V$ in the bipartite graph defined above, where 'U' is $(I/O/P/E)_{\text{req}}$ and 'V' is $(I/O/P/E)_{\text{adv}}$, respectively. The matrix polynomial B' defined in equation 7.5 is used to find the best matching by evaluating the determinant of the matrix. As per the lemma 9 in [Sankowski, 2009], the degree of 'x' in determinant $(B'(G, x))$ is the weight of the maximum weight matching in a bipartite graph G. The determinant is defined as the sum over all possible permutations of the products along each permutation as shown in equation 7.6 [Sankowski, 2009].

$$\det(B'(G, x)) = \sum_{\pi \in \pi_n} \text{sgn}(\pi) \prod_{i=1}^n (B'(G, x)_{i, \pi_i}) \quad (7.6)$$

Where, operation \prod represents a permutation operation to be performed on polynomial matrix B' defined for a weighted bipartite graph $G = (U, V, E, w)$. Each non zero element of the sum corresponds to the matching in G. The maximum degree of 'x' corresponds to the maximum matching weight of the matching and the pair of vertices involved in the calculation of

maximum matching weights are the bipartite matching. All such maximum matching weights and the vertices is recorded in the set 'MM' = {mm₁, mm₂,....., mm_r}. Each element mm_i of the set 'MM' records MatchScore, rank of the solution and matching pair of vertices in bipartite graph, respectively. Each pair of vertex corresponds to the one vertex from set 'U' and other vertex of set 'V'.

For matrix 'B' in Figure 7.8, the polynomial matrix B' is defined using equation 7.5 as:

$$B' = \begin{array}{c|ccc} & v_1 & v_2 & v_3 \\ \hline u_1 & X^0 & X^4 & X^3 \\ \hline u_2 & X^3 & X^2 & X^4 \end{array}$$

The determinant of B' is evaluated using equation 7.6 as

$$\begin{aligned} \det(B'(G, x)) = & \langle u_1, v_1 \rangle X^0 (\langle u_2, v_2 \rangle X^2 * \langle u_3, v_3 \rangle X^2 - \langle u_2, v_3 \rangle X^4 * \langle u_3, v_2 \rangle X^4) \\ & - \langle u_1, v_2 \rangle X^4 (\langle u_2, v_1 \rangle X^3 * \langle u_3, v_3 \rangle X^2 - \langle u_3, v_1 \rangle X^4 * \langle u_2, v_3 \rangle X^2) \\ & + \langle u_1, v_3 \rangle X^3 (\langle u_2, v_1 \rangle X^3 * \langle u_3, v_2 \rangle X^4 - \langle u_3, v_1 \rangle X^2 * \langle u_2, v_2 \rangle X^2) \end{aligned}$$

$$\begin{aligned} \text{or } \det(B'(G, x)) = & \langle u_1, v_1 \rangle \langle u_2, v_2 \rangle \langle u_3, v_3 \rangle X^4 - \langle u_1, v_1 \rangle \langle u_2, v_3 \rangle \langle u_3, v_2 \rangle X^8 \\ & - \langle u_1, v_2 \rangle \langle u_2, v_1 \rangle \langle u_3, v_3 \rangle X^9 - \langle u_1, v_2 \rangle \langle u_3, v_1 \rangle \langle u_2, v_3 \rangle X^{10} \\ & + \langle u_1, v_3 \rangle \langle u_2, v_1 \rangle \langle u_3, v_2 \rangle X^{10} - \langle u_1, v_3 \rangle \langle u_3, v_1 \rangle \langle u_2, v_2 \rangle X^7 \end{aligned}$$

In the above determinant evaluation, maximum matching is the highest degree of 'X'. There are two such values which gives a maximum matching score of 10 (for the example quoted). These two solutions are recorded in set MM as

MM = {(10, 2, <u₁,v₂><u₃,v₁><u₂,v₃>), (10, 3, <u₁,v₃><u₂,v₁><u₃,v₂>)}. Each element in set MM is MatchScore, rank and maximum bipartite matching. Our algorithm chooses solution 2 i.e. (10, 3, <u₁,v₃><u₂,v₁><u₃,v₂>) as the solution 2 has higher rank than solution 1, with the same matching score. The MatchScore, rank and Maximum matching corresponding to input, output, precondition and effect are obtained separately. All the four solution sets are combined to obtain the MatchScore and Rank of all web services in the set. The discovered web services are arranged in different priority queues as per the rank values and are prioritized as per the overall MatchScore.

7.2.3.2 Creation of priority queue

In order to prioritize the web services, the MatchScore and the rank of the web service is used. The rank of web service is not the sufficient criteria to prioritize the web services. Two

services may have the same *rank*, however, the web services may differ at *MatchScore*. The web service with higher *MatchScore* must be preferred over the other web service with lower *MatchScore* and same *rank*. In equation 7.7, function $F^T(MS, r) \rightarrow [0, 1]$, is defined to calculate the similarity of web services based on IOPE of advertised and required web service. Lower values of similarity represent higher priority.

$$F^T(MS, r) = \begin{cases} 0, & \text{if } r = 0 \\ \frac{\sum_{i=T} MS^T_i}{\sum_{i=T} MS^T_i + \sum_{i=T} r_i} & \text{if } r \in \{1,2,3\} \\ 1, & \text{if } r = 4 \end{cases} \quad (7.7)$$

Where, '*MS*' and '*r*' is the match score and rank calculated after comparison of the required web service specifications with advertised web service on '*k*' parameters. '*T*' is any one of I, O, P, and E. In equation 7.8, the rank of the j^{th} advertised web service is obtained by finding the lowest rank evaluated among all four types '*T*'.

$$Rank_{WS}^j = \min\{r_T\} \quad (7.8)$$

For selecting the web service having best matching score, the advertised web services after matching are kept in different priority queues. Four priority queues based on four types viz EXACT, PLUGIN, SUBSUME and INTERSECT are created (refer Figure 7.9). Based on $Rank_{WS}^j$ value, the priority queue in which advertised web service, after matchmaking, is to be kept, is decided and the priority of web services belonging to same queue, is decided by similarity value obtained using equation 7.7. Services which are declared fail are not considered further.

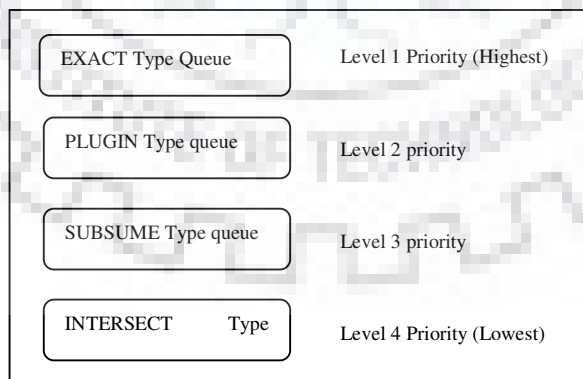


Figure 7.9: Priority queue based on the Exact, Plugin, Subsume and Intersect type

There are three advantages of creating queues. Firstly, web services with similar rank are prioritized separately. Secondly, the web service with higher rank gets assigned higher priority, automatically. Thirdly, if any run time failure of selected web service occurs, the replaceable web service can be obtained immediately.

7.2.4 modified Genetic Algorithm (mGA)

Survival of the fittest principle is the foundation for working of GA [Yan et al., 2015]. The principle applies to the individuals (known as chromosomes) in the population. The fit individual from the population is identified based on the fitness test. The fitness function uses certain properties associated with each chromosome to conduct fitness test. Chromosomes which are categorized as fit, are carried forward for further consideration. In this way, the best chromosomes among the population are identified. But, this step reduces number of chromosomes and does not ensure to further improve the quality. So, GA uses three operators, crossover, mutation and selection, to ensure that the global optimization is achieved [Tewari et al., 2012b]. The crossover operator considers two chromosomes and using crossover operation, a new chromosome is generated. The randomness is introduced by mutation operator which ensures population diversity and safeguard against local optima. The selection operation determines few top most fit chromosomes. Few popular and efficient selection techniques used to implement selection operation in GA are: roulette wheel selection, stochastic universal selection, tournament selection, rank selection, etc.

mGA is evolved from traditional GA by introducing new fitness function based on replaceability concept. mGA when applied to the problem of WSS, the population initialization is done by selecting random compositions of WSS. mGA loops over the population of fixed size for selection, crossover, mutation and fitness evaluation. The looping continues till a certain stopping condition is met, e.g. loop until no further changes appear in the fitness of top few chromosomes in the population. The QoS score generated from PROMETHEE Plus method (explained in Section 7.2.2.1) and WS similarity score obtained from determinant method based matchmaking are used to evaluate replaceability of each composition. To compute fitness of each composition, QoS parameters are used along with replaceability factor. The fitness function used in the proposed work is presented using equation 7.9.

$$Fit_{WS_j} = \frac{\sum((wt_k * QoS_{iType}), (wt_{10} * Replaceability(WS_j)))}{\sum(wt_k * QoS_{dType})} \quad (7.9)$$

Where, QoS_{iType} represents increasing type QoS parameters, i.e., parameters are considered better for higher value of QoS, QoS_{dType} represents decreasing type QoS parameters, i.e., lower the value better the parameter, w_{tk} is the weight of QoS parameter and represents the preference order of QoS parameter. The weight values are obtained from the end user. Replaceability (WS_j) is the replaceability of j^{th} composition plan (represented by j^{th} chromosome in the population). The value of fitness function is to be maximized. The objective function is determined by including the replaceability information in the objective function used in the work [Liu et al., 2016b], [Sharifara et al., 2014], [Helal and Gamble, 2014]. The higher the value of replaceability factor indicate higher possibility of getting replaceable WS (in case of any failure). Thus, the objective function should have higher value in this case. Therefore, the replaceability factor is kept in numerator part of the objective function. The goal of GA is to maximize the value of objective function.

7.2.4.1 Choosing the Reference Web Service

In every iteration of GA, chromosomes representing service composition is selected. Each gene of the chromosome represents concrete WS corresponding to each AWS. The concrete WSs are selected randomly. One such example chromosome is depicted in Figure 7.10. The chromosome has n number of gene represented as concrete WSs, $C_{1r}, C_{2r}, C_{3r}, \dots, C_{nr}$. Replaceability of i^{th} concrete WS, C_{ir} , is the number of WSs which can replace it, for $1 \leq i \leq m$. The replaceability count is obtained from QoS score determined by using PROMETHEE Plus method and WS similarity score determined using determinant method based matchmaking. The replaceability of j^{th} chromosome is obtained using equation 7.10.

$$\text{Replaceability}(WS_j) = \sum \text{Replaceability}(C_{ir}), \text{ for } 1 \leq i \leq m \quad (7.10)$$

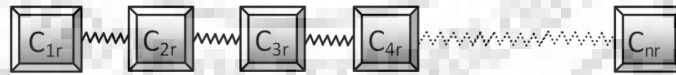


Figure 7.10: Example service composition plan (WS_j) generated by random choice of concrete WS corresponding to each AWS.

Thus, the reference WS is the concrete WS from the chromosome under consideration. In other word, the reference WS is one of the WS in the list L_i , where each L_i store WSs in the sequence obtained from PROMETHEE Plus in case of $WSSR_P$. For $WSSR_{PM}$ approach, the L_i store WSs in the sequence obtained from PROMETHEE Plus along with the service similarity score with reference to other CWS is also stored.

7.3 ALGORITHM FOR WEB SERVICE SELECTION

The $WSSR_{PM}$ algorithm is presented in Figure 7.11. The algorithm is called by passing five parameters as input (line 1). The QoS parameters of candidate concrete WSs are normalized along with QoS requested by the end user (line 2). The QoS parameters are brought in the range (0...1). The PROMETHEE Plus algorithm is called to obtain list of concrete WSs for each AWS arranged with reference to NoF score (line 3). Advantage of using PROMETHEE Plus to calculate replaceability is that the repeated calculations required are fully avoided. With each concrete WS in the arranged list, a count indicating number of WSs ahead of that concrete WS is kept. The count value can be fetched using *getAheadCount()* method. At the time of replaceability evaluation, the corresponding value of count is utilized. The result of QoS based ordering of CWS is stored in list L1.

Input: Set WS_{QoS} represent QoS values for candidate concrete WSs corresponding to each AWS. numAWS parameter indicate number of AWS involved in composition. Number of concrete WSs competing is specified as numCWS. Variable $WS_{enduser}$ is the web service corresponding to QoS requirements specified by the end user, $wt[]$ array specifies QoS preference of the end user

Output: Optimal composition and replaceable WSs for each concrete WS chosen to take part in composition

Begin:

1. $WSSR_{PM}(WS_{QoS}, numAWS, numCWS, WS_{enduser}, wt[])$
//Normalize QoS parameters
 2. $(WS'_{qos}, WS'_{enduser}) = \text{Normalize}(WS_{QoS}, WS_{enduser})$
// Apply PROMETHEE Plus algorithm to obtain the priority list of WSs based on
// the calculated NoF score
 3. List L1[] = PROMETHEEPlus($WS'_{qos}, WS'_{enduser}, numAWS, numCWS$)
// Find IOPE score of all concrete WSs for each AWS
 4. List L2[] = IOPE_Matchmaking();
 5. List L[] = prepareList(L1, L2)
// Apply modified GA on the set of candidate WSs to perform WSS using
// replaceability
 6. $(C_{PLAN}, WS_{repl}[], []) = mGA(WS'_{qos}, wt[], L[], WS'_{enduser})$ // Where, the array of lists
// $L[]$, in which each list stores the sorted sequence of concrete WSs obtained
// from PROMETHEE Plus
 7. return C_{PLAN} // Return optimal composition
- End**

Figure 7.11: $WSSR_{PM}$ algorithm for replaceability based WSS using mGA

The replaceability factor is evaluated using QoS and IOPE score. The IOPE based score for each CWS is determined using IOPE based matchmaking algorithm (line 4). The IOPE based

algorithm uses determinant method for performing matchmaking. The rank concept is also utilized and based on IOPE based matchmaking score, the WSs are prioritized and store in list L2. Using both the list L1 and L2, the final ordering of WSs is determined and WSs are enlisted in list L (line 5). The overall rank of the services is obtained by combining NoF score obtained from PROMETHEE Plus, IOPE score. The mGA algorithm is called, which after certain iteration returns globally optimal composition plan (C_{PLAN}) and list of replaceable WS(s) ($WS_{rep}[], []$) for each concrete WS in the plan (line 6). The optimized composition is returned by the algorithm (line 7). The replaceable WSs are utilized upon the occurrence of run-time failure or unavailability of any concrete WS. Clearly, neither the roll backing of composition plan is needed nor the selection process is repeated. This ultimately saves a lot of computations by avoiding reselection of services.

The algorithm for replaceability evaluation for a chromosome in the population is shown in Figure 7.12. The j^{th} composition plan, for which the replaceability count is to be obtained, is one of the parameter for algorithm. The array of lists, $L[]$, in which each list stores the sorted sequence of concrete WSs obtained as a results from PROMETHEE Plus and IOPE based matching, is the second parameter to the algorithm (line 1). The initial replaceability count for the plan is zero (line 2). The replaceability of composition plan is obtained from the replaceability of individual concrete WS in the composition. The calculation required to determine the replaceability of composition plan is simplified due to the use of PROMETHEE Plus. The replaceability of the composition is evaluated by summing up the replaceability of

Input: WS_j is the j^{th} composition plan represented by chromosome in the population. An array of list $L[]$ storing concrete WSs for each AWS in different list. The list contains WSs sorted on the basis of NoF score

Output: Replaceability count for composition plan WS_j

Begin:

```

1. findReplaceability( $WS_j$ ,  $L[ ]$ )
2.    $Replaceability_{WS_j} = 0;$ 
   // Calculate replaceability count for the composition plan by summing the
   // replaceability of each CWS
3.   for ( $i=0; i < WS_j.length$ )
4.     do
5.        $cws = WS_j[i];$ 
6.        $Replaceability_{WS_j} += L[i].get(cws).getAheadCount();$  // getAheadCount()
   // method returns number of WSs ahead of current WS on NoF score.
7.   end for
8.   return  $Replaceability_{WS_j}$ 

```

End

Figure 7.12: Algorithm for replaceability evaluation

each concrete WS (line 3 - 7). The algorithm returns the replaceability of composition. Time complexity of algorithm in Figure 7.12 is $\theta(n)$, where, n is the number of AWS.

Determinant method based matchmaking algorithm: The determinant evaluation using polynomial concept will yield the matching score of web services based on the semantic matching. The maximum matching for a weighted bipartite graph using the determinant method is solved using recursive algorithm presented in Figure 7.13. The algorithm in Figure 7.13 is called by passing MatrixMM type weighted bipartite graph (line 1). Det is a integer to store the matchmaking score and is initialize (line 2). If there is a single node in the graph, the determinant calculated is 0 (line 3-5). If there are two nodes in the matrix, then the edge with maximum weight is selected (line 6-10). Else the algorithm determines the edges with the best edge weights (line 11-13). While determining edges with best edge weights, rank is also evaluated.

Input: MatrixMM bGraph[][]: a two dimensional matrix representing weighted bipartite graph for either of I,O,P, or E. The weights of the graph are generated by OWL-S matchmaker using semantic matching.

Output: Determinant method based match making score

Begin:

1. CalculateDeterminant(MatrixMM bGraph [][])

// If only one node present then matrix determinant is evaluated as zero

2. Int Det = -1;

3. if(bGraph .length() == 1)

4. Det = bGraph [0][0].weight & store edge $u_i = 0$ and $v_j = 0$ // Determinant is 0.

5. return Det

// If there are two nodes then just evaluate the matrix based on determinant concept

6. else if (bGraph .length() == 2) {

7. Det = max(bGraph [0][0].weight + bGraph [1][1] .weight, bGraph [0][1] .weight + bGraph [1][0] .weight)

8. Rank = getRank()

9. store edge u_i and v_j representing max edge weights

10. return Det

// Else recursively evaluate the matrix.

11. else

12. Det

$$\text{Max}_{i=0}^{n-1} (bGraph[0][i] + \text{CalculateDeterminant}(bGraph - (0^{\text{th}} \text{ row}, i^{\text{th}} \text{ column})))$$

13. Rank = getRank() // Find rank of solution(s) having same Det value.

//store rank and all vertices which contributes for calculation of Max Determinant.

14. return Det

End:

Figure 7.13: Determinant algorithm to obtain maximum matching for Semantic matching of Web Services.

The algorithm in Figure 7.13 based on memoization. It explores all possible combinations and select the *max* value among the selected combinations. Due to the use of memoization technique, the repeated calculations are avoided. Once the *MatchScore* and *rank* of the maximum matching is obtained, the rank value of the solution determines the match type of the web services. The numerical value of the rank is mapped to any of the five match types mentioned earlier. The match score for Outputs matching and Inputs matching is evaluated separately for required web service and advertised web services.

The algorithm for functionally equivalent services to be selected from various queues in the order of priority is presented in Figure 7.14. The algorithm in Figure 7.14 is called by passing the AWS for which the WS is to be selected (line 1). The WSs in EXACT type queue is selected first (line 2-5). If no WSs found in this queue, next queue is searched for most compatible candidate web service. In the PLUGIN (line 6-9), SUBSUME (line 10-13) and

Input: AWS_i is the i^{th} abstract web service for which the corresponding CWS needs to select functionally equivalent WS. The functionally equivalent WS is obtained from the appropriate queue.
 Output: The index of the selected WS is returned, if a WS is available in the queue else a "0" is returned to signify that no functionally equivalent WS is present

Begin:

1. procedure SelectSemanticSimilarWS(AWS_i)
2. if(! ($AWS_i.EXACT_Type_Queue.empty()$))
3. Randomly chose WS from EXACT_Type_Queue
4. Remove the selected WS from queue
5. Return the selected WS
6. else if(! ($AWS_i.PLUGIN_Type_Queue.empty()$))
7. Select WS from PLUGIN_Type_Queue with highest similarity score
8. Remove the selected WS from queue
9. Return the selected WS
10. else if(! ($AWS_i.SUBSUME_Type_Queue.empty()$))
11. Select WS from SUBSUME_Type_Queue with highest similarity score
12. Remove the selected WS from queue
13. Return the selected WS
14. else if(! ($AWS_i.INTERSECT_Type_Queue.empty()$))
15. Randomly chose WS from INTERSECT_Type_Queue with highest similarity score
16. Remove the selected WS from queue
17. Return the selected WS
18. Else
19. print "No Suitable Matched WS is found"
20. Return "0"

End:

Figure 7.14: Selection of Web Service from priority queues

INTERSECT (line 14-17) type queues, the WSs with highest matching score is chosen. The WS selected for execution is determined and the queue from which the web service was selected is also recorded. At run time, if, due to some reason service failure is experienced, than the replaceable WS can be found from the priority queue from where the WS was selected earlier. If WS from none of the queue is available, then a error message is displayed back to the user (line 18-20).

7.4 RESULTS AND OBSERVATIONS

In this section, results obtained by conducting various experiments are discussed. The experiments are conducted to firstly evaluate the replaceability using PROMETHEE Plus and IOPE based matchmaking. Two variations of the proposed approach $WSSR_p$ and $WSSR_{PM}$ are compared with the existing similar technique for selection of WSs ($WSSR_{NN}$) [Helal and Gamble, 2014]. In $WSSR_{NN}$, the replaceability is evaluated using nearest neighbour technique. To conduct experiments, a QoS Dataset-2 is used. The dataset include measurement for nine QoS parameters namely, Reliability (R_l), Availability (A_v), Compliance (C_m), Throughput (T_p), Successability (S_u), Documentation (D_c), Best Practices (B_p), Latency (L_t), and Response Time (R_t). To perform IOPE based matchmaking, the OWL-S dataset with nine domains and 1083 service record is chosen [Sem, 2016]. The details of both the dataset are presented in Section 2.5.1 and 2.5.2, respectively, of Chapter 2. The machine with Intel core i7 CPU and 8 GB of RAM is used to conduct the experiments. The algorithms for PROMETHEE Plus and mGA based web service selection are implemented in Java.

7.4.1 Evaluation of determinant method

Consider the required web service WS_{req} and advertised web service WS_{adv} having ' k ' inputs, outputs, precondition and effects, each. The problem of matchmaking is firstly expressed as a bipartite graph matching problem. For each of the nine domains, the time taken for matchmaking using Hungarian method [Bellur and Kulkarni, 2007] and Determinant method is recorded. The comparison chart of both the techniques for nine domains is presented in Figure 7.15. It can be observed from Figure 7.15 that the time complexity of Determinant method is lower than the Hungarian method and is found suitable for performing IOPE based matchmaking.

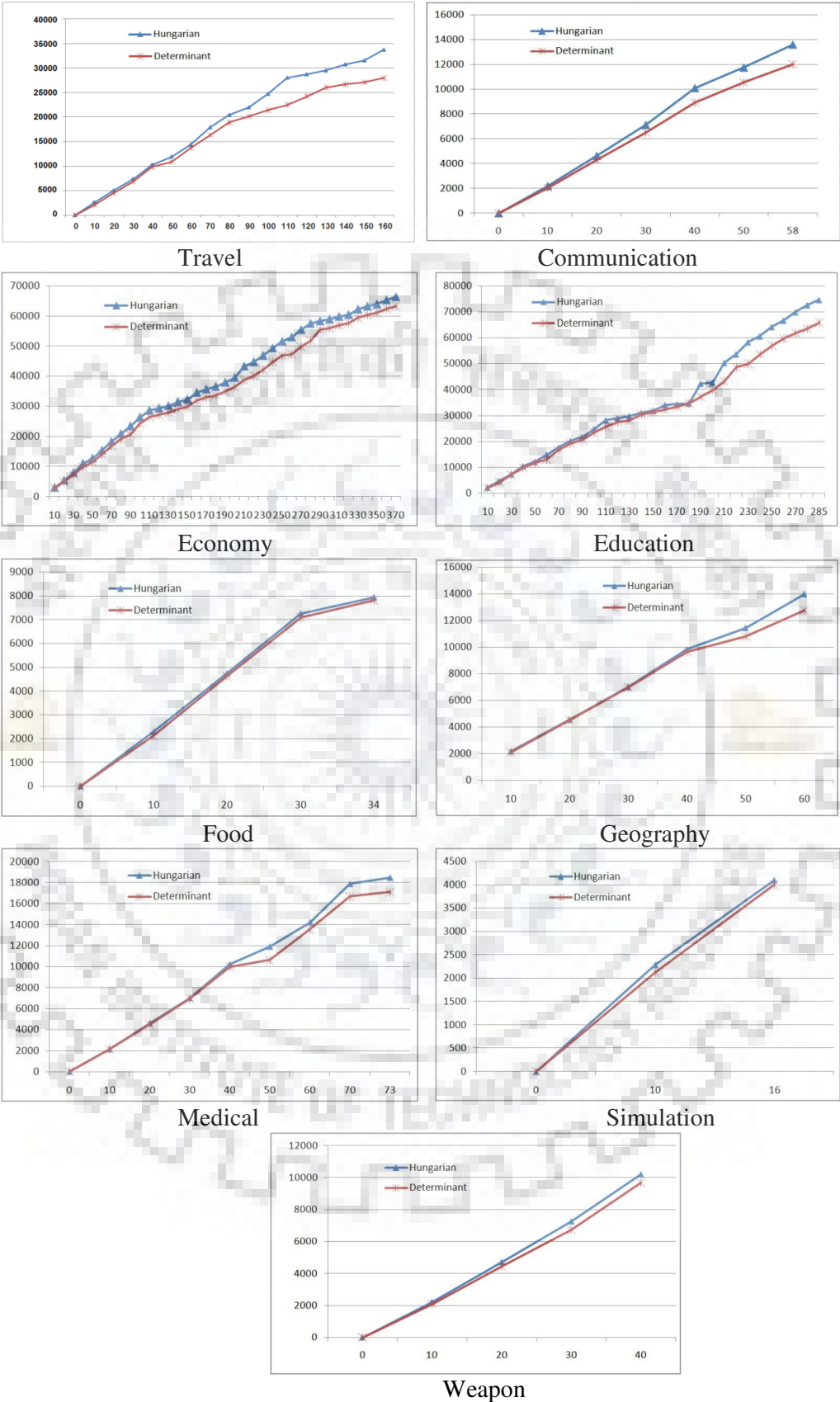


Figure 7.15: Matchmaking time comparison for Hungarian and Determinant method for nine domains

7.4.2 Evaluation of the proposed Web Service Selection approach

The effect of introducing PROMETHEE Plus method is observed by comparing with nearest neighbour [Helal and Gamble, 2014] for evaluating replaceability. Two approaches $WSSR_P$ and $WSSR_{NN}$ are compared by using 12 different datasets with different number of AWS and CWS [Helal and Gamble, 2014]. The details of dataset is presented in Table 7.2. Here, number of AWS is decided by the available domains according to OWLS-TC4 dataset. The OWL-S records of CWS are generated by repeating the available OWL-S records for the particular domain.

Table 7.2: Dataset used for evaluation

Dataset	No. of AWS	No. of CWS	Dataset	No. of AWS	No. of CWS
1A	3	2	7A	9	10
2A	4	4	8A	9	25
3A	6	4	9A	9	50
4A	6	6	10A	9	100
5A	7	10	11A	9	500
6A	8	10	12A	9	5000

The time taken by the proposed $WSSR_P$ and $WSSR_{NN}$ [Helal and Gamble, 2014] approaches is measured and presented in Figure 7.16. The x-axis represents dataset used with varying number of AWS and CWS. It is observed that, the time taken by $WSSR_P$ is almost constant and time taken by the existing $WSSR_{NN}$ is increasing with the increase in number of AWS and CWS. This is primarily due to the fact that $WSSR_P$ uses PROMETHEE Plus based solution to determine replaceable WSs. However, $WSSR_{NN}$ uses nearest neighbour to evaluate

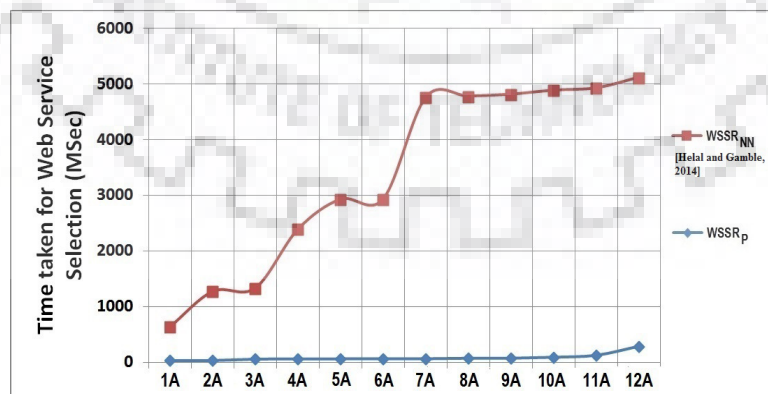


Figure 7.16: Effect of increase in number of AWS and CWS.

replaceability score of each selected WS in each iteration of GA. This consumes a lot of time for determining replaceable WS with repeated calculations.

In order to analyse the effect of variation in the number of concrete WSs on $WSSR_P$ and $WSSR_{NN}$ [Helal and Gamble, 2014] another experimentation is performed by keeping the number of abstract WSs fixed to four. The result of experiment is shown in Figure 7.17. The average value of time is measured by repeating the experiment 100 times. The variation in number of concrete WSs is depicted on X-axis and measured time (in msec) for selection from each approach is depicted on Y-axis of Figure 7.17.

It can be observed that, with an increase in the number of concrete WSs, the time taken for selection also increases for both the approaches. However, relatively higher increase in selection time is observed for approach $WSSR_{NN}$ in [Helal and Gamble, 2014]. It is due to the fact that the $WSSR_{NN}$ approach follows the nearest neighbour algorithm to obtain replaceability count. With increase in the number of concrete WSs, the nearest neighbour algorithm is required to search in the larger domain of candidate services to determine nearest neighbour. However, in the proposed approach $WSSR_P$, the priority order of services is determined using PROMETHEE Plus. Based on this priority order, number of replaceable WSs are determined and stored with each concrete candidate WS as *AheadCount*. Whenever, the replaceability count for a concrete WS is to be determined, it can be directly obtained from *AheadCount*. It saves the computational time.

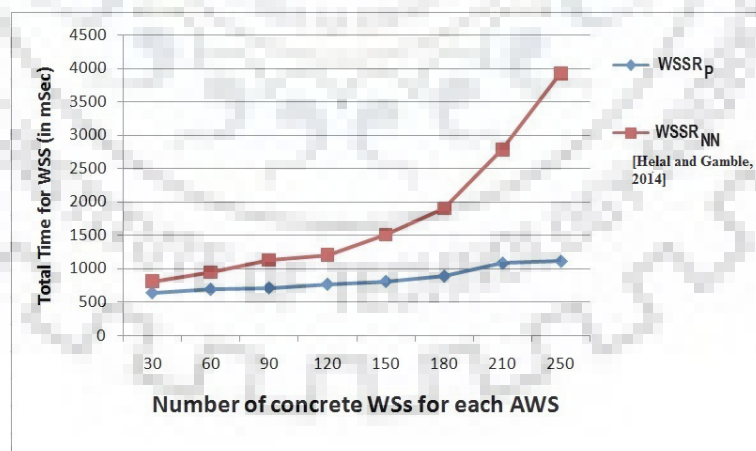


Figure 7.17: Performance comparison of $WSSR_P$ with $WSSR_{NN}$ [Helal and Gamble, 2014] on the basis of time for selection with variation in number of concrete WSs.

7.4.3 Comparison on replaceability factor

In this experiment, the example of hotel booking with sports activity (discussed in Section 7.1) is considered. The proposed approach $WSSR_P$ and $WSSR_{NN}$ [Helal and Gamble, 2014] are compared on number of replacements for each concrete web service. The statistical results obtained by performing this experiment on both the approaches are listed in Table 7.3. The experiment is repeated for both the approaches by varying number of concrete WSs and keeping the number of AWSs fixed to four.

It can be analysed from Table 7.3 that the improved replaceability factor is resulted from the proposed approach as compared to the existing approach [Helal and Gamble, 2014] (column 2 and 3). It is primarily due to the use of PROMETHEE Plus for replaceability evaluation. The PROMETHEE Plus ensures that the services contributing towards replaceability evaluation are compared with each other by considering the conflicting nature of QoS parameters. The conflicting nature of QoS parameters is ignored by the nearest neighbour method.

Table 7.3: Comparison of the proposed approach ($WSSR_P$) for WSS with an existing approach ($WSSR_{NN}$) [Helal and Gamble, 2014] on replaceability factor

<i>No of concrete WSs per AWS (for 4 AWS)</i>	<i>No of replacements (For 4 AWS)</i>		<i>Detection of exact matching replaceable WS</i>	
	Proposed Approach	Existing approach in [Helal and Gamble, 2014]	Proposed Approach	Existing approach in [Helal and Gamble, 2014]
5, 5, 5, 5	2,0,2,1	1,0,1,2	N	N
10, 15, 10, 5	2,2,3,1	1,1,0,2	Y	N
20, 30, 10, 7	3,1,3,2	2,1,0,2	Y	N
30, 40, 15, 7	4,4,3,2	2,2,1,2	Y	N
36, 53, 15, 7	4,4,3,2	2,2,1,2	Y	N

Number of concrete web services for each abstract web service is assumed to be same, however, in actual practice this may not be the case

Further, $WSSR_P$ approach also determines the exact matching replaceable WS(s) (column 4, in Table 7.3), if any. The system responds back to the requester with the optimal composition plan (consisting of a set of concrete WSs). At run time, if any of the concrete WS becomes unavailable or fails, the selection system uses the set of replaceable WSs. At this point, the exact match WSs are given preference over closely matching WSs.

7.4.4 Performance comparison of the proposed $WSSR_P$, $WSSR_{PM}$ methods

The replaceability evaluation in $WSSR_P$ is further enhanced by including IOPE based matchmaking ($WSSR_{PM}$). The proposed approaches $WSSR_P$, $WSSR_{PM}$ and existing approach $WSSR_{NN}$ are evaluated on satisfaction score. The dataset presented in Table 7.2 is used to perform experiment. The number of abstract web services and concrete web services are varied and average satisfaction score over random 100 end user queries for 10 run is obtained. The variation in satisfaction score with number of WSs is shown in Figure 7.18. It can be observed from Figure 7.18, that the proposed $WSSR_P$ technique outperforms over $WSSR_{NN}$. The

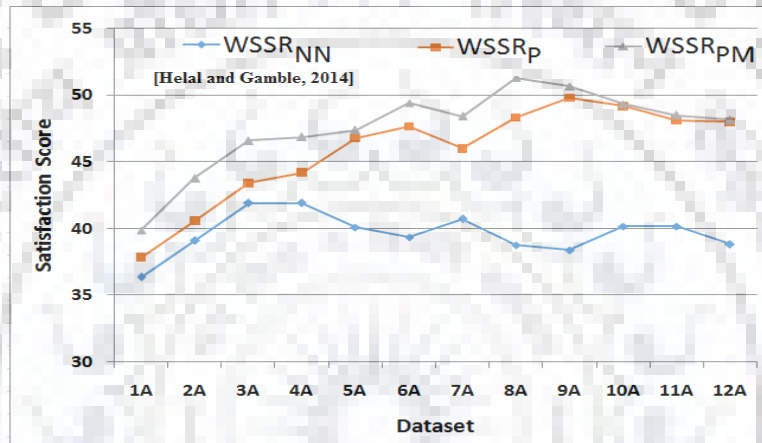


Figure 7.18 Comparison of $WSSR_P$ and $WSSR_{NN}$ on satisfaction score

satisfaction score for each variation is observed to be higher for $WSSR_P$.

The means plot representing the mean value of satisfaction score calculated on 100 end user queries over 10 run, is presented in Figure 7.19. The y-axis represents the mean satisfaction score obtained. The x-axis represents the WSS method being compared. It can be observed that, for each of the 12 dataset (Table 7.2), the mean satisfaction score of $WSSR_P$ and $WSSR_{PM}$ is found to be higher than $WSSR_{NN}$. The $WSSR_{PM}$ further improves the mean satisfaction score. For dataset 1A, 2A, 4A, 5A, and 6A, almost linear increase in the mean satisfaction score is observed. Further, for dataset 9A, 10A, 11A, and 12A, the mean satisfaction score of $WSSR_P$ and $WSSR_{PM}$ is almost similar.

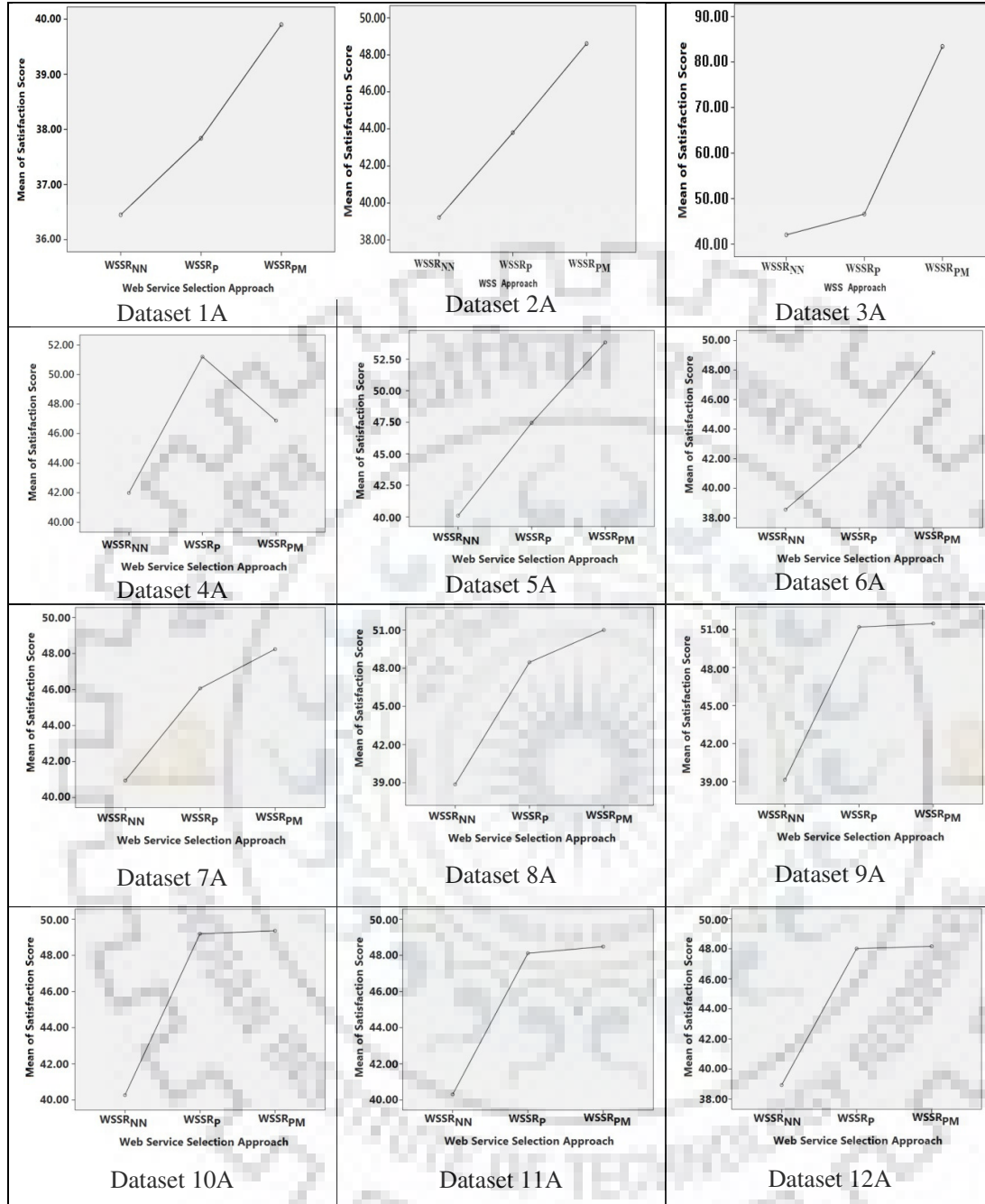


Figure 7.19 Comparison of WSSR_{NN} [Helal and Gamble, 2014], WSSR_P and WSSR_{PM} on means plot of satisfaction score

7.4.5 Comparison on precision

The performance of the proposed approaches WSSR_P, WSSR_{PM}, and existing approach WSSR_{NN} is tested using precision performance measure. The experiment is performed by

taking example discussed in Section 7.1. Web services from four domains are considered i.e. hotel, sports, geographical distance, and location. The precision values are measured by performing experiments by taking average over hundred runs. The precision values for each of the three approach for four domains is presented in Figure 7.20. It can be observed from Figure 7.20 that due to the introduction of IOPE based matchmaking to find service similarity, the precision in selection of WSs is improved. The results are obtained by repeating the experiments 100 times and taking average. The precision of $WSSR_{PM}$ is found better than $WSSR_P$ and $WSSR_{NN}$ approaches. Further, the precision of $WSSR_P$ approach is found better than $WSSR_{NN}$ approach.

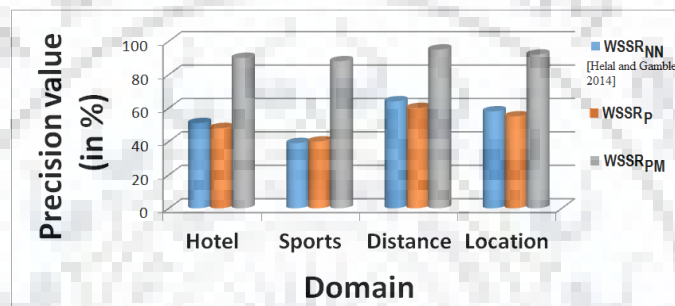


Figure 7.20: Comparison of three WSS approaches on Precision value.

A comparison of the proposed approach $WSSR_{PM}$ with existing state-of-the-art is given in Table 7.4. The comparison shows that most of the existing works uses IO for matchmaking, whereas our proposed approach uses IOPE for matchmaking. Further, it can be observed from the Table 7.4 that to perform matchmaking, existing works uses Greedy, Hungarian and other methods, whereas, $WSSR_{PM}$ is using Determinant method which is efficient than the existing matchmaking algorithms. Most of the existing works applies matchmaking at the time of WS discovery, but $WSSR_{PM}$ applies matchmaking during selection process. Also, most of the existing works do not consider replaceability during WSS. A very few of the works determines replaceable WSs based on QoS parameters only. The proposed $WSSR_{PM}$ improves the WSS precision by using IOPE based WS similarity along with the QoS parameters for replaceability evaluation.

Table 7.4: A comparison of existing state-of-the-art with proposed approach for web service selection

Reference	Matchmaking Criteria	Matching Technique	Web Service Discovery (D) /Selection (S)	Replaceability	Replaceability Criteria
[Helal and Gamble, 2014]	-	-	D	✓	QoS
[Paolucci et al., 2002]	IO	Greedy	D	✗	-
[Hu et al., 2017]	IO + Location Constraints	NM	S	✗	-
[Bellur and Kulkarni, 2007]	IO	Hungarian	D	✗	-
[Bellur and Vadodaria, 2008]	IOPE	Hungarian	D	✗	-
[Paulraj et al., 2011]	IO + WordNet	NM	D	✗	-
[Khanam et al., 2013]	IO	Improved Hungarian	D	✗	-
[Pukkasenung et al., 2010]	IOPE	Semantic Distance	D	✗	-
[Shirazi, H., 2018]	Similarity/Dissimilarity measure	NM	D	✗	-
Proposed Approach (WSSR _{PM})	IOPE	Determinant	S	✓	IOPE + QoS

IO: Input, Output; IOPE: Input, Output, Precondition, Effect; NM: Not Mentioned

7.5 CONCLUSIONS

The operating environment of web services is unpredictable and does not ensure the availability and non-failure of web services at run time. Moreover, few of the QoS parameters are dynamic in nature and dependent on external factors such as network, load on machines, and other parameters. Thus, backup service which can replace the failed/unavailable service is required. In this chapter, QoS based evaluation and IOPE based similarity is used for replaceability assessment. The task of selection of web services to accomplish the composite task is performed by providing due consideration to the replaceability factor. The modified genetic algorithm is applied to select optimal composition plan. The set of replaceable Web Services

for each concrete web service involved in the chosen composition are also selected. Various experiments are performed and results obtained from the comparison of proposed approach with existing similar approaches are presented. It is analyzed from the obtained results that the proposed approach $WSSR_{PM}$ is better than the existing similar approaches for selection of WS. The $WSSR_{PM}$ has improved time efficiency and precision in selection of WSs. With the increase in the number of either concrete web services or abstract web services, the $WSSR_{PM}$ performs reasonably well and has improved the replaceability factor.

The work presented in this chapter has been published as [Purohit et al., 2015, Purohit and Kumar 2016a, Purohit and Kumar, 2018f].



CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

This chapter summarizes the conclusions drawn from various works presented in the thesis (section 8.1) and the further directions of the research on QoS based web service selection (section 8.2).

8.1 CONCLUSIONS

In this thesis, we have mainly proposed the improvements in the web service selection. The proposed improvements primarily concentrate on refinements in the performance of the web service selection system in terms of improving time efficiency of selection process, improving user satisfaction level, closely matching the user's expectations, and dealing with run time failure of web services. Another important issue handled in this thesis work is the evaluation of weight values of QoS parameters without user involvement. This makes system domain independent. The system adopt automatically if partial values of weights are provided by the end user. The works presented in this thesis have shown that the performance of majority vote based technique for web service classification and hierarchical clustering for web services clustering is well suited for improving the overall performance of web service selection system. Overall, we have presented three web service selection approaches to perform efficient web service selection.

The specific conclusions drawn from various works presented in this thesis are discussed below.

The literature review of the works related to QoS based web service selection is presented in *Chapter 2*. The chapter concludes that the problem of WSS is studied in the past and a lot of work has been done on QoS based selection of web services. Major contributions of the past research works lies around the applications of techniques such as MCDM approaches, machine learning, semantic based techniques, Skyline techniques, among others. It is concluded from literature review that the existing web service selection approaches suffered from the problem

of scalability, i.e., with the large number of candidate web services, the performance of web service selection approach is reduced significantly. Also, the existing solutions failed in determining the desired web service with due consideration to the end user request of QoS. This can be handled by considering the end user request of QoS as one of the candidate web service. Further, it was concluded that the weight values of QoS parameters largely affect the selection results. The use of mathematical models can be helpful in evaluation of weight values. The chapter also concludes that the existing approaches of web service selection ignore the run time failure of web services. The service selection can be done with due consideration to service replaceability. It was observed that the number of candidate WSs can be reduced by applying filtering before selection step. The classification technique is useful to identify similar WSs based on their QoS properties. Many classification techniques are available to fulfil this objective. Thus, we have performed an empirical study on few popular and efficient classification techniques.

An empirical study on number of web service classification techniques is conducted in Chapter 3. Initially, eleven learning models - Logistic Regression (LR), Multilayer Perceptron (MLP), Non-nested generalized exemplars (NNge), PART, Decision Table (DT), JRip, J48 decision tree, Random Forest (RF), Decision Stump (DS), CART, and Support Vector Machine (SVM) are compared. The performance of these learning models is evaluated on various performance measurement parameters such as, accuracy, average absolute error (AAE), average relative error (ARE) along with kappa statistics and visual analysis using model performance chart, etc., Based on the empirical study it is concluded that three learning models – LR, NNge, and J48 outperformed over all other learning models. LR, NNge, and J48 learning models were compared with majority vote based learning model. The result of Tukey test confirms that majority vote based learning model beats other learning models. Based on the findings, we have developed a classification based WSS approach.

Based on the findings of empirical study, we have proposed a two layer model for classification based web service selection in Chapter 4. We have proposed PROMETHEE Plus method to perform WSS. Three different variations MSS, HSS, and CSS are proposed. The proposed approach is compared with the existing state-of-the-art. The proposed approach was found better on user satisfaction, Euclidean distance, and time for WSS parameters. The results of Tukey test also confirms that the proposed approach improves the selection results. Thus, to deal with large number of candidate WSs, the classification based WSS approach can be

adopted. In the next chapter, we have presented the approach to deal with unlabelled dataset based WSS.

We have performed the empirical study on six web services clustering techniques – K-Means, UPGMA (based on Hierarchical clustering), PAM, Diana, SOTA, and Clara, in *Chapter 5*. Two performance measures – stability (external) and quality (internal) of clustering, are used to judge the strength and suitability of each clustering technique. Based on the empirical study, it is concluded that the UPGMA (based on Hierarchical) clustering outperform over other clustering techniques. The clustering results are further improved by using PCA for feature selection. Based on the findings of this work, it is concluded that, UPGMA clustering outperform over other clustering techniques. Observations drawn from the chapter helps in developing a clustering based WSS approach.

We have introduced a new web service selection model in Chapter 6 based on clustering, pruning and skyline technique. To filter out those candidate WSs which are far below the expectations of the end user, pruning is proposed. We have used and proposed Skyline Plus method to perform WSS. Two variations of the proposed approach, we call it as, CPSky-AA and CPSky-FS, respectively, are presented. The proposed approach is compared with the existing similar works. Based on the results of experimentation, the proposed approach is found to be better than the existing state-of-the-art on Euclidean distance, user satisfaction and CPU time parameters. From this work, it is concluded that the proposed prefiltering based web service selection model is effective and can be used to perform WSS. One important factor, specially in case of composite WS, is service replaceability of selected WS. In the next chapter we have presented a WSS approach based on service replaceability.

The problem of run time failure of web services is handled in Chapter 7. Two variations of the replaceability based WSS approach are proposed. To evaluate QoS based equivalence of WSs, the use of PROMETHEE Plus method is proposed. Also, to determine functional equivalence of WSs, the use of determinant method to perform IOPE based matchmaking is proposed. Based on the results of QoS and IOPE based matchmaking, WS replaceability is evaluated using modified genetic algorithm (mGA). The proposed approach for replaceability based WSS is compared with existing state-of-the-art. From the results of experimentation it is concluded that the proposed approach is better than existing similar approaches on user satisfaction, CPU time for WSS and precision in selection of WSs parameters.

8.2 FUTURE WORKS

Based on the literature review presented and experimental study performed in this thesis work, we envisage the few directions in which few works can be performed and are summarized as follows:

- The present study uses a limited dataset for conducting various experiments. This is primarily due to lack of availability of QoS dataset based on real world web services. In future, the work done in this thesis can be extended if new datasets are available. In addition, on the availability of datasets from different domains, further studies can be performed.
- The proposed CSS and CPSky-FS approaches have limitation that they perform selection of web service for atomic task. In future, the proposed approaches can be extended to perform selection for composite tasks. Also, the behaviour of web service selection process can be investigated by using the proposed two layer selection models based on clustering and classification for achieving composition.
- The work presented for replaceability based web service selection can be further enhanced by including parameter such as matching of web services based on description of services provided by the service providers. The work can further be extended if a common dataset having QoS as well as OWL-S description of WSs is made available.
- The selection process plays a very vital role in selection of WSs. The quality of services being selected and the time taken for selection is dependent on the selection methodology used. For composite WS, the selection decision is affected by the other services which are part of composition. Thus, composability checking during the selection time is useful. Presently, the proposed system has limitation of evaluating replaceability without considering composability parameter. The work presented in this thesis for replaceability based WSS can be further extended by considering IOPE based composability of web services.

PUBLICATIONS

- [1] Lalit Purohit, Sandeep Kumar, "A Classification Based Web Service Selection Approach", *IEEE Transactions on Services Computing*, Vol. 11, Feb-2018. doi: 10.1109/TSC.2018.2805352 (SCI Impact Factor: 4.418)
- [2] Lalit Purohit, Sandeep Kumar, "Web Services in the Internet of Things and Smart Cities", *IEEE Consumer Electronics Magazine*, Vol. 8(2), 2019. DOI: 10.1109/MCE.2018.2880808 (SCI Impact Factor: 1.434)
- [3] Lalit Purohit, Sandeep Kumar, "A Study on Evolutionary Computing based Web Service Selection Techniques", submitted to *Artificial Intelligence Review*, Springer, 2018.
- [4] Lalit Purohit, Sandeep Kumar, "Web Services Clustering in Smart city context", submitted to *IEEE Computer*, 2018.
- [5] Lalit Purohit, Sandeep Kumar, "CPSky: A Multi-Layer model for Web Service Selection", submitted to *IEEE Transactions on Services Computing*, 2019.
- [6] Lalit Purohit, Sandeep Kumar, "QoS and Semantic Similarity based model for Web Service Selection", Submitted to *IEEE Transactions on Systems, Man and Cybernetics: System*, 2019.
- [7] Lalit Purohit, Sandeep Kumar, "Towards an Approach for Replaceability based Web Service Selection", revised and submitted to 19th IEEE International Conference on Web Services (ICWS), 2019, ERA-A Conference.
- [8] Lalit Purohit, Sandeep Kumar, "Clustering based Approach for Web Service Selection using Skyline Computations", submitted to 19th IEEE International Conference on Web Services (ICWS), 2019, ERA-A Conference.
- [9] Lalit Purohit, Sandeep Kumar, "Exploring K-Means Clustering and skyline for Web Service Selection", In Proceedings of 11th IEEE International Conference on Industrial and Information System (ICIIS), pp. 603-607, 2016. (H-Index: 10).
- [10] Lalit Purohit, Sandeep Kumar, "Web Service Selection using Semantic Matching", In Proceedings of ACM International Conference on Advances in Information Communication Technology & Computing (AICTC-16), pp. a1-a6, August 2016.
- [11] Deepak Kshirsagar, Sandeep Kumar, Lalit Purohit, "Exploring Usage of Ontology for HTTP Response Splitting Attack", In Proceedings of IEEE International Conference on Next Generation Computing Technologies (NGCT), pp. 1123 - 1126, 2015.
- [12] Lalit Purohit, Sandeep Kumar, Deepak Kshirsagar, "Analyzing genetic algorithm for web service selection", In Proceedings of IEEE International Conference on Next Generation Computing Technologies (NGCT), pp. 999 - 1003, 2015.
- [13] Lalit Purohit, Sandeep Kumar, "QoS aware Web Service Selection using Nature Inspired Algorithm", Presented in *Uttarakhand State Science and Technology Congress (USSTC) 2015-16*, pp. 155, Dehradun, India, Feb. 2016.

BIBLIOGRAPHY

- [Ai and Tang, 2008a] Ai, L., and Tang, M., (2008). A penalty-based genetic algorithm for qos-aware web service composition with inter service dependencies and conflicts. In Proceedings of the International Conference on Computational Intelligence for Modeling Control and Automation, pages 738–743.
- [Ai and Tang, 2008b] Ai, L., and Tang, M., (2008). QoS-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm. In Proceedings of the IEEE International Conference on eScience, USA, pages 119–126.
- [Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T., (2010). Selecting skyline services for QoS-based web service composition. In Proceedings of the WWW'10, NY, USA: ACM, pages 11–20.
- [Alrifai et al., 2011] Alrifai, M., Risse, T., and Nejdil, W. (2011). A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints, *ACM Transactions on the Web*, 1(1): 11:1–11:30.
- [Amiri et al., 2013] Amiri, M. A., Derhami, V., and Ghasemzadeh, M., (2013). QoS-based web service composition based on genetic algorithm. *Journal of AI and Data Mining*, 1(2): 63-73.
- [Antony et al., 2009] Antony, D., Mello, D., and Ananthanarayana, V. S., (2009). Dynamic selection mechanism for quality of service aware web services. *Enterprise Information Systems*, 4(1): 23–60.
- [Arockiam and Sasikaladev, 2012] Arockiam, L., and Sasikaladev, N., (2012). Simulated annealing versus genetic based service selection algorithms. *International Journal of u- and e-Service, Science and Technology*, 5(1): 35–50.
- [Atrey et al., 2008] Atrey, P. K., Hossain, M. A., and Saddik, A. E., (2008). Association-based dynamic computation of reputation in web services, *International Journal of Web and Grid Services*, 4(2): 169-188.
- [Atrey et al., 2012] Atrey, P. K., Ibrahim, H., Hossain, et al., (2012). Determining trust in media-rich websites using semantic similarity. *International Journal of Multimedia Tools and Applications*, 60(1): 69-96.
- [Bandyopadhyay and Saha, 2013] Bandyopadhyay, S., and Saha, S., (2013). Unsupervised classification: similarity measures, Classical and meta-heuristic approaches and applications. Springer-Verlag, Berlin publications.

- [Bellur and Kulkarni, 2007] Bellur, U., and Kulkarni, R., (2007). Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In Proceedings of the IEEE International Conference on Web Services (ICWS 2007), pages 86-93.
- [Bellur and Vadodaria, 2008] Bellur, U., and Vadodaria, H., (2008). On Extending Semantic Matchmaking to Include Preconditions and Effects. In Proceedings of the IEEE International Conference on Web Services (ICWS 2008), pages 120-128.
- [Benouaret et al., 2012] Benouaret, K., Benslimane, D., and Hadjali, A., (2012). WS-Sky: An Efficient and Flexible Framework for QoS-Aware web service selection. In Proceedings of the Service Computing Conference, pages 146-153.
- [Beran et al., 2012] Beran, P. P., Vinek, E., Schikuta, E., Leitner, M., and Zhang, C., (2012). An Adaptive heuristic approach to service selection problems in dynamic distributed systems. In Proceedings of the International Conference on Grid Computing, pages 66–75.
- [Bhatt et al., 2013] Bhatt, C., Atrey, P. K., and Kankanhalli, M. S., (2013). A reward and punishment based approach for concept detection using adaptive ontology rules. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, 9(2): 1-21.
- [Bhuvanewari and Karpagam, 2012] Bhuvanewari, A., and Karpagam, G. R., (2012). Discovering Substitutable and Composable Semantic Web Services for Web Service Composition. *International Journal of Computer Applications*, 48(8): 1-8.
- [Biot and Academy, 1977] Biot, M. A., and Academy, R., (1977). *Data Mining and Analysis: Fundamental concepts and Algorithms*. volume 22, pages 183–198.
- [Borzsony et al., 2001] Borzsony, S., Kossmann, D., and Stocker, K., (2001). The Skyline operator. In Proceedings of the 17th International Conference on Data Eng, pages 421–430. doi:10.1109/ICDE.2001.914855.
- [Brans and Vincke, 1985] Brans, J. P., and Vincke, P., (1985). A Preference Ranking Organization METHod. *Lecture notes in The Institute of Management Sciences*, pages 647-656.
- [Brock et al., 2008] Brock, G., Pihur, V., Datta, S., and Datta, S., (2008). clValid: An R package for cluster validation. *Journal of Statistical Software*, 25(4): 1–22.
- [Buqing et al., 2013] Buqing, C., Jianxun, L., Liu, X. F., Bing, L., Dong, Z., and Guosheng, K., (2013). Chc-tscm: A trustworthy service composition method based on an improved chc genetic algorithm. *China Communications*, 10(12): 77 – 91.

- [Canfora et al., 2005] Canfora, G., Penta, M. D., Esposito, R., and Villan, M. L., (2005). An approach for QoS-aware service composition based on genetic algorithms. In Proceedings of the Genetics and Evolutionary Computation Conference, pages 1069-1075.
- [Chang, 2012] Chang, G., (2012). Qos-based web service selection approach. Software Engineering and Knowledge Engineering, springer, AINSC, volume 115, pages 887-892.
- [Chen et al., 2009] Chen, Z., Wang, H., and Pan, P., (2009). An approach to optimal web service composition based on QoS and user preference. In Proceedings of the International Joint Conference on Artificial Intelligence, China, pages 96–101.
- [Chen et al., 2010] Chen, C. T., Pai, P. F., and Hung, W. Z., (2010). An Integrated Methodology using Linguistic PROMETHEE and Maximizing deviation method for Third-party Logistics Supplier Selection. International Journal of Computational Intelligence Systems, 3(4): 438-451.
- [Chen et al., 2015] Chen, F., Yuan, S., and Mu, B. (2015). User-QoS-Based Web Service Clustering for QoS Prediction. In Proceedings of the IEEE International Conference on Web Services (ICWS), pages 583–590. doi:10.1109/ICWS.2015.83.
- [Chifu et al., 2014] Chifu, V. R., Salomie, I., Pop, C. B., Niculici, A. N., and Suia, D. S., (2014). Exploring the selection of the optimal Web Service Composition through Ant Colony Optimization. International Journal of Computing and Informatics, 33(5): 1047-1064.
- [Cho et al., 2016] Cho, J. H., Ko, H. G., and Ko, I. Y., (2016). Adaptive Service selection according to the service density in multiple QoS aspects. IEEE Transactions on Services Computing, 9(6): 883-894.
- [Claro et al., 2005] Claro, D. B., Albers, P., and Hao, J. K., (2005). Selecting Web Services for Optimal Composition. In Proceedings of the International Conference on Web Services (ICWS'05), pages 32-45.
- [Cohen, 1995] Cohen, W. W. (1995). Fast effective rule induction, In Proceedings of the International Conference on Machine Learning, pages 115-123.
- [Crowley et al., 2016] Crowley, D. N., Curry, E., and Breslin, J. G., (2016). Citizen actuation for smart environments. IEEE Consumer Electronics Magazine, 5(3): 90-94.
- [Darbar and Samanta, 2015] Darbar, R., and Samanta, D., (2015). Surfacesense: Smartphone can recognize where it is kept. In Proceedings of the 7th International Conference on HCI, IndiaHCI, pages 39-46.

- [Dasgupta et al., 2016] Dasgupta, K., Gajjar, S., and Sarkar, M., (June 2016). FAMACROW: Fuzzy and Ant Colony Optimization Based Combined MAC, Routing, and Unequal Clustering Cross-Layer Protocol for Wireless Sensor Networks. *Applied Soft Computing*, 43: 235-247.
- [Ding et al., 2009] Ding, C., Sambamoorthy, P., and Tan, Y., (2009). QoS Browsing for Web Service Selection. In *Proceedings of the International Joint Conference Service Oriented Computing*, Springer, pages 285-300.
- [Ding et al., 2015a] Ding, Z., Liu, J., Sun, Y., Jiang, C., and Zhou, M., (2015). A transaction and QoS aware service selection approach based on genetic algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, 45(7): 1035-1046.
- [Ding et al., 2015b] Ding, Z., Sun, Y., Liu, J., Pan, M., and Liu, J., (2015). A genetic algorithm based approach to transactional and QoS aware service selection. *Enterprise Information Systems*, 11(3): 339–58,.
- [Eren et al., 2015] Eren, P. E., Mishra, D., and Mishra, A., (2015). A Software Development Process Model for Cloud by Combining Traditional Approaches. In *Proceedings of the On the Move to Meaningful Internet Systems*, pages 421-430.
- [Ernst et al., 2006] Ernst, M. D., Lencevicius, R., and Perkins, J. H. (2006). Detection of Web Service substitutability and composability. In *Proceedings of the International Workshop on Web Services Modeling and Testing*, pages 123-135.
- [Ezenwoke et al., 2017] Ezenwoke, A., Daramola, O., and Adigun, M. (2017) Towards a Visualization Framework for Service Selection in Cloud e-Marketplaces, *IEEE World Congress on Services*, pages 122–129. doi:10.1109/SERVICES.2017.31.
- [Fethallah, 2012] Fethallah, H., Chikh, M. A., and Mohammed, D. Y. (2012). Qos-aware service selection based on genetic algorithm. In *Proceedings of the 3rd International Conference on computer Science and its Applications*, pages 825-829.
- [Fister et al., 2013] Fister Jr., I., Yang, X. S., Fister, I., Brest, J., and Fister, D., (2013). A Brief Review of Nature-Inspired Algorithms for Optimization, *Elektrotehnikski vestnik*, 80 (3): 1-7.
- [Gajjar et al., 2016] Gajjar, S., Tandon, A., Shah, V., and Sarkar, M., (2016). Evolving Applications for Internet of Things. *International Journal of Computer Science And Computing (IJCSC)*, 7(2): 169-174.
- [Gajjar et al., 2017] Gajjar, S., Sarkar, M., and Dasgupta, K., (2017). Low Energy Fuzzy based Unequal Clustering Multihop Architecture for Wireless Sensor Networks. In *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, pages 1-18.

- [Ganesh et al., 2005] Ganesh, T.S., Somani, A. K., and Kher, S. (2005). Dynamic Crossover Management in hardware accelerated implementations of Genetic algorithms. In Proceedings of the International Conference on Intelligent Systems (ICIS2005), pages 1-6.
- [Gang and Chunli, 2015] Gang, D., and Chunli, C., (2015). A novel approach to the selection of representative skyline web services. In Proceedings of the IEEE International Conference on Computational Intelligence and Communication Networks, pages 1013-1017.
- [Gangadharan et al., 2016] Gangadharan, G. R., Uden, L. and Lutthuis, P. O. (2016). Sourcing Requirements and Designs for Software as a Service, International Journal of Systems and Service-Oriented Engineering (IJSSOE), 6(1): 1-16.
- [Garriga et al., 2015] Garriga, M., Flores, A., Cechich, A., and Zunino, A., (2015). Web services composition mechanisms: A review. IETE Technical Review, 32(5): 376–83.
- [Goh and Singh, 2015] Goh, K. L., and Singh, A. K. (2015). Comprehensive Literature Review on Machine Learning Structures for Web Spam Classification. In Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems, pages 434-441.
- [Guo et al., 2011] Guo, G., Yu, F., Chen, Z., and Xie, D., (2011). A Method for Semantic Web Service Selection Based on QoS Ontology. Journal of Computers, 6(2): 377-386.
- [Gupta and Bhanodia, 2012] Gupta, S., and Bhanodia, P., (2012). A Flexible and Dynamic Failure Recovery Mechanism for Composite Web Services Using Subset Replacement. International Journal of Science and Research, 3(12): 1886-1890.
- [Gupta et al., 2015] Gupta, I. K., Kumar, J., and Rai, P., (2015). Optimization to quality-of-service driven web service composition using modified genetic algorithm. In Proceedings of the IEEE International Conference on Computer, Communication and Control (IC4-2015), pages 1–6.
- [Hala and Mohamed, 2012] Hala, A. E., and Mohamed, N. H., (2012). Mapping potential landfill sites for North Sinai cities using spatial multicriteria evaluation. The Egyptian Journal of Remote Sensing and Space Sciences, Elsevier, 15(1): 125–133.
- [Hao et al., 2012] Hao, Y., Zhang, Y., and Cao, J., (2012). A novel QoS model and computation framework in web service selection, World Wide Web, 15(5), pages 663-684.
- [Helal and Gamble, 2014] Helal, A., and Gamble, H. (2014). Introducing replaceability into web service composition, IEEE Transactions on Services Computing, 7(2): 198–209.

- [Herssens et al., 2008] Herssens, C., Jureta, I. J., and Faulkner, S., (2008). Dealing with Quality Tradeoffs during Service Selection. In Proceedings of the IEEE International Conference on Autonomic Computing (ICAC '08), pages 77-86.
- [Hosmer and Lemeshow, 2000] Hosmer, D. W., and Lemeshow, S., (2000). Applied Logistic Regression. John Wiley & Sons, publications, USA, pages 1-43.
- [Hoyle, 2001] Hoyle, D. (2001). ISO 9000–Quality Systems Handbook. Oxford publications.
- [Hu et al., 2017] Hu, X., Feng, Z., Huang, K., and Chen, S., (2017). Supporting interoperability among Web services through efficient matching. In Proceedings of the IEEE 41st Annual Computer Software and Applications Conference, pages 557-566.
- [Huang, 2013] Huang, X., (2013). UsageQoS: Estimating the qos of web services through online user communities. *ACM Transaction on Web*, 8(1): 1:1–1:31.
- [Hwang et al., 2015] Hwang, S. Y., Hsu, C. C., and Lee, C. H. (2015). Service Selection for Web Services with Probabilistic QoS. *IEEE Transactions on Services Computing*, 8(3): 467-480.
- [Ismaili, 2012] Ismaili, F. (2012). Hybrid web service selection by combining semantic and keyword approaches. In Proceedings of the International Conference on Recent Progress in Data Engineering and Internet Technology (LNEE), pages 31–38. doi:10.1007/978-3-642-28798-5_5
- [Ivan et al., 2011] Ivan, I., Virza, M., and Yuen, H., (2011). Algebraic Algorithms for Matching, Project in MIT's 6.854 (Advanced Algorithms), pages 1-15.
- [Jaeger and Muhl, 2007] Jaeger, M. C., and Muhl, G., (2007). QoS-based selection of services: The implementation of genetic algorithm. *IEEE Communication in Distributed Systems (KiVS)*, pages 1–12.
- [Jatoth et al., 2017] Jatoth, C., Gangadharan, G. R., and Buyya, R., (2017). Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review. *IEEE Transactions on Services Computing*, 10(3): 475-492.
- [Jian et al., 2016] Jian, X., Zhu, Q., and Xia, Y., (2016). An interval-based fuzzy ranking approach for QoS uncertainty-aware service composition, *Optik, International Journal for Light and Electron Optics*, 127(4): 2102-2110.
- [Jian-hua et al., 2008] Jian-hua, L., Song-qiao, C., Yong-jun, L., and Gui-lin, L., (2008). Application of genetic algorithm to QoS-aware web services composition. In Proceedings of the 3rd International Conference on Industrial Electronics and Applications, pages 516–521.

- [Jie et al., 2012] Jie, L. Y. and Jian, C., (2012). Web Service Classification based on Automatic Semantic Annotation and Ensemble Learning. In Proceedings of the SWPDP, pages 2274-2279.
- [Jin et al., 2008] Jin, C., Wu, M., Jiang, T., and Ying, J., (2008). Combine automatic and manual process on web service selection and composition to support QoS. In Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design, pages 459-464.
- [KalisZewski and Podkopaev, 2016] KalisZewski, I., and Podkopaev, D., (July 2016). Simple additive weighting-A metamodel for multiple criteria decision analysis methods. International Journal of Expert Systems with Applications, 54: 155-161.
- [Kang et al., 2011] Kang, G., Liu, J., Tang, M., Liu, X., and Fletcher, K. K., (2011). Web Service Selection for Resolving Conflicting Service Requests. In Proceedings of the International Conference on Web Services, pages 387-394.
- [Karim et al., 2011] Karim, R., Ding, C., Chi, C.H., (2011). An Enhanced PROMETHEE Approach for QoS-Based Web Service Selection. In Proceedings of the IEEE International Conference on Services Computing, pages 536-543.
- [Khanam et al., 2013] Khanam, S. A., Oh, K. H., Seol, W. S., and Youn, H. Y., (2013). A Novel Semantic Web Service Discovery Scheme Using Bipartite Graph. In Proceedings of the IEEE International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing, pages 1070-1077.
- [Kher et al., 2006] Kher, S., Speck, P., and Somani, A. K., (2006). Distributed dynamic clustering algorithm in randomly deployed wireless sensor networks. Technical Report, Iowa State University.
- [Kher et al., 2009] Kher, S., Ganesh, T. S., Ramesh, P., and Somani, A. K., (2009). Greedy Dynamic Crossover Management in Hardware Accelerated Genetic Algorithm Implementations using FPGA. In Proceedings of the International Conference on Computer Modelling and Simulation, Cambridge University, Emmanuel College, Cambridge, pages 1-6.
- [Kittler et al., 1998] Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J., (1998). On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3): 226-239.
- [Kritikos and Plexousakis, 2009] Kritikos, K., and Plexousakis, D. (2009). Requirements for QoS-Based Web Service Description and Discovery, IEEE Transactions on Service Computing, 2(4): 320-337.

- [Kumar and Mastorakis, 2010] Kumar, S., and Mastorakis, N. E. (2010). Novel Models for Multi-Agent Negotiation based Semantic web service Composition, *WSEAS Transactions on Computers*, 9(4): 339-350.
- [Kumar and Mishra, 2008a] Kumar, S., and Mishra, R. B., (2008). Cognition based service selection in semantic web service composition, *International Journal of Computer Science, Infocomp*, 7(3): 35-41.
- [Kumar and Mishra, 2008b] Kumar, S., and Mishra, R. B., (2008). A Hybrid Model for Service Selection in Semantic Web Service Composition. *International Journal of Intelligent Information Technologies*, 4(4): 55-69.
- [Kumar and Mishra, 2008c] Kumar, S., and Mishra, R. B., (2008). Semantic web service composition. *IETE Technical review*, 25(3): 105-121.
- [Kumar, 2012] Kumar, S., (2012). Agent-based semantic Web Service composition. *Briefs in Electrical and Computer Engineering*, Springer publications.
- [Kumara et al., 2014] Kumara, B. T. G. S., Paik, I., Ohashi, H., Chen, W., and Koswatte, K.R.C., (2014). Context aware post-filtering for web service clustering. In *Proceedings of the IEEE International Conference on Services Computing (SCC-2014)*, pages 440–447. doi:10.1109/SCC.2014.65.
- [Kuncheva, 2004] Kuncheva, L. I., (2004). Combining pattern classifiers - Methods and algorithms. John Wiley & Sons publications, Inc, USA, pages 16-148.
- [Kundu et al., 2011] Kundu, D., Sarma, M., Samanta, D., and Mall, R., (2011). A lazy learning approach for building classification models. *International Journal of Intelligent Systems*, 26(8): 773-786.
- [Lakshmi and Dhas, 2015] Lakshmi, M. D. and Dhas, J. P. M., (2015). A Hybrid Approach for Discovery of OWL-S Services Based on Functional and Non-Functional Properties. *WSEAS Transactions on Computers*, 14: 62-71.
- [Lim et al., 2012] Lim, E., Thiran, P., Maamar, Z., Bentahar, J. (2012). On the Analysis of Satisfaction for Web Services Selection. In *Proceedings of the IEEE International Conference on Services Computing*, pages 122 – 129.
- [Lin et al., 2012] Lin, F. R., Hsieh, P. S., and Uden, L., (2012). Analyzing Sustainable New Service Development: An Activity Theory Perspective. In *Proceedings of the International Joint Conference on Service Sciences*, pages 200-205.

- [Lin et al., 2012] Lin, Y., Yang, Y., Li, L., Wang, J., Zhao, C., and Guo, W., (2012). Web service selection based on improved genetic algorithm. In Proceedings of CIP, CCIS, pages 564-574.
- [Liu and Weng, 2012] Liu, S. C., and Weng, S. S., (2012). Applying genetic algorithm to select web services based on workflow quality of service. *Journal of Electronic Commerce Research*, 13(2): 157-172.
- [Liu et al., 2010] Liu, H., Zhong, F., Ouyang, B., and Wu, J. (2010). An approach for QoS aware web service composition based on improved genetic algorithm. In Proceedings of the International Conference on Web Information Systems and Mining, pages 123–128.
- [Liu et al., 2011] Liu, J., He, K., Wang, J. and Ning, D. (2011). A Clustering Method for Web Service Discovery. In Proceedings of the IEEE International Conference on Services Computing, pages 729–730. doi:10.1109/SCC.2011.47.
- [Liu et al., 2013] Liu, Z. Z., Xue, X., Shen, J., and Li, W. R., (2013). Web service dynamic composition based on decomposition of global QoS constraints. *International Journal of advanced manufacturing technology*, 69(9): 2247-60.
- [Liu et al., 2016a] Liu, X., Agarwal, S., Ding, C., and Yu, Q., (2016). A LDA-SVM Active Learning Framework for Web Service Classification. In Proceedings of the International Conference on Web Services, pages 49-56.
- [Liu et al., 2016b] Liu, Z. Z., Chu, D. H., Jia, Z.-Pu., Shen, J.-Q., and Wang, L., (2016). Two-stage approach for reliable dynamic Web service composition. *International Journal of Knowledge-Based Systems*, 97(1): 123–143.
- [Liu et al., 2015b] Liu, Z. Z., Jia, Z. P., Xue, X., and An, J. Y., (2015). Reliable web service composition based on QoS dynamic prediction. *Journal of Soft Computing*, Springer, 19(5): 1409-1425.
- [Ludwig and Schoene, 2011] Ludwig, S. A., and Schoene, T., (2011). Web Service Selection using Particle Swarm Optimization and Genetic Algorithms. In Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing, pages 225-230.
- [Ludwig, 2011] Ludwig, S. A., (2011). Memetic algorithm for web service selection. In Proceedings of the 3rd workshop on Biologically inspired algorithms for distributed systems, pages 1–7.
- [Ma and Zhang, 2008] Ma, Y., and Zhang, C. W. (2008). Quick convergence of genetic algorithm for QoS-driven web service selection. *International Journal of Computer Networks*, Elsevier, 52(5): 1093–1104.

- [Ma et al., 2015] Ma, H., Wang, A., and Zhang, M., (2015). A Hybrid Approach using Greedy Search and Genetic Programming for QoS-aware Web Service Composition. *Transactions on Large-Scale Data and Knowledge-Centered Systems*, LNCS 8980, pages 180-205.
- [Maiti et al., 2014] Clustering Web Search Results to Identify Information Domain. In *Proceedings of the Emerging Trends in Computing and Communication*, pages 291-303.
- [Mardukhia et al., 2013] Mardukhia, F., NematBakhsha, N., Zamanifara, K., and Baratib, A., (2013). QoS decomposition for service composition using genetic algorithm. *Applied Soft Computing*, 13(7): 3409-3421.
- [Margaris et al., 2015] Margaris, D., Georgiadis, P., and Vassilakis, C., (2015). A Collaborative Filtering Algorithm with Clustering for Personalized Web Service Selection in Business Processes. In *Proceedings of the RCIS*, pages 169-180.
- [Masri and Mahmoud, 2007] Masri, A. and Mahmoud, Q. E., (2007). QoS-based Discovery and Ranking of Web Services. In *Proceedings of the IEEE 16th International Conference on Computer Communications and Networks (ICCCN '07)*, pages 529-534.
- [Masri and Mahmoud, 2008] Masri, A. E., and Mahmoud, Q. H., (2008). Towards quality driven web service discovery. *IT Professional*, IEEE, 10(3): 24-28.
- [Masri and Mahmoud, 2009] Masri, E. A. and Mahmoud, Q. H., (2009). Web Service Discovery and Clients Goal. *IEEE Computer*, 42(1), pages 104-107.
- [Michalewicz and Vadis, 2012] Michalewicz, Z. and Vadis, Q., (2012). Evolutionary Computation? On a Growing Gap between Theory and Practice. In *Proceeding of the IEEE WCCI 2012*, LNCS 7311, pages 98–121.
- [Mier et al., 2010] Mier, P. R., Mucientes, M., Lama, M., and Couto, M. I., (2010). Composition of web services through genetic programming. *Evolutionary Intelligence*, Springer, 3(3) 171–186.
- [Mishra and Mishra, 2015] Mishra, A., and Mishra, D., (2015). Scale Up Internet-Based Business Through Distributed Data Centers. In *Proceedings of the workshop On the Move to Meaningful Internet Systems: OMT 2015*, pages 381-390.
- [Mishra et al., 2013] Mishra, D., Mishra, A., C.-Palacios, R., and C.-Lumbreras, C., (2013). Global Software Development and Quality Management: A Systematic Review. In *Proceedings of the Demey Y.T., Panetto H. (eds) On the Move to Meaningful Internet Systems: OTM 2013 Workshops*. *Lecture Notes in Computer Science*, pages 81-86.

- [Mishra, 2015] Mishra, A., (2015). Computer and Software Engineering: Emerging Trends and Challenges. In Proceedings of the International Conference on Emerging research in Computer and Software Engineering, pages 1-7.
- [Mobedpour and Ding, 2013] Mobedpour, D., and Ding, C., (2013). User-centered design of a QoS-based web service selection system, *International Journal of Service Oriented Computing and Applications*, Springer, 7(2): 117–127.
- [Mohanty et al., 2010] Mohanty, R., Ravi, V., and Patra, M. R. (2010). Web-Services classification using intelligent techniques, *Expert systems with applications*, 37(7): 5484-5490.
- [Mohanty et al., 2012] Mohanty, R., Ravi, V., Patra, M. R., (2012). Classification of Web Services Using Bayesian Network. *International Journal of Software Engineering and Applications*, 5(4): 291-296.
- [Mucha and Sankowski, 2004] Mucha, M., and Sankowski, P., (2004). Maximum matchings via Gaussian elimination. In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 248–255.
- [Mustafa and Kumaraswamy, 2014] Mustafa, A. S., and Kumaraswamy, Y. S., (2014). Performance evaluation of Web-Services Classification. *Indian Journal of Science and technology*, 7(10): 1674-1681.
- [Mustafa and Kumaraswamy, 2015] Mustafa, A. S., and Swamy, Y. S. K., (2015). Web Service classification using Multi-Layer Perceptron Optimized with Tabu Search. In Proceedings of the IACC, pages 290-294.
- [Newcomer, 2002] Newcomer, E., (2002). Understanding Web Services - XML, WSDL, SOAP, and UDDI. volume 1. Addison-Wesley publications.
- [Nisa and Qamar, 2015] Nisa, R. and Qamar, U., (2015). A text mining based approach for web service classification. *International Journal of Information System and E-Business Management*, 13(4): 751–768. doi:10.1007/s10257-014-0252-5.
- [Oriol et al., 2014] Oriol, M., Marco, J., and Franch, X., (2014). Quality models for web services: A Systematic Mapping. *International Journal of Information and Software Technology*, Elsevier, 56(10): 1167-1182.
- [Oskooei and Daud, 2014] Oskooei, M. A., and Daud, A. M., (2014). Quality of Service (QoS) model for web service selection. In Proceedings of the IEEE International Conference of Communication and Control Technology (I4CT 2014), pages 266-270.

- [Ouadah et al., 2015] Ouadah, A., Benouaret, K., Hadjali, A., and Nader, F., (2015). Combining Skyline and Multi-Criteria Decision Methods to Enhance Web Services Selection. In Proceedings of the IEEE symposium on Programming and Systems, pages 124-131.
- [Own and Yahyaoui, 2015] Own, H. S., and Yahyaoui, H. (2015). Rough set based classification of real world Web services. *Information System Frontier*, 17(6): 1301–1311.
- [Palanikkumar and Kousalya, 2012] Palanikkumar, D., and Kousalya, G., (2012). An evolutionary algorithmic approach based optimal web service selection for composition with quality of service. *International Journal of Computer Applications*, 8(4): 573–78.
- [Paolucci et al., 2002] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K., (2002). Semantic matching of web service capabilities. In Proceedings of the International Semantic Web Conference, pages 333-347.
- [Parisi et al., 1999] Parisi, V., Yahia, H., Font, J., Herlin, I., and Garcia-Ladona, E., (1999). Image motion analysis using scale-space approximation and simulated annealing. In [Proceedings of the International Workshop-Conference on Artificial and Natural Neural Networks, pages 645-654.
- [Patro and Patra, 2015] Patro, V. M., and Patra, M. R. (2015). Classification of Web Services using Fuzzy Classifiers with feature selection and Weighted Average Accuracy, *Transactions on Networks and Communications*, 3(1): 108-116.
- [Paulraj et al., 2011] Paulraj, D., Swamynathan, S., (2011). Content Based Service Discovery in Semantic Web Services Using WordNet. In Proceedings of the International Conference on Advanced Computing, Networking and Security, pages 48-56.
- [Phartiyal et al., 2018] Phartiyal, G. S., Brodu, N., Singh, D., and Yahia, H., (2018). A mixed spectral and spatial Convolutional Neural Network for Land Cover Classification using SAR and Optical data. EGU General Assembly, Vienna.
- [Pop et al., 2011] Pop, F. C., Pallez, D., and Cremene, M., (2011). QoS-based service optimization using differential evolution. In Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO'11), ACM, pages 1891–1898.
- [Pro, 2016] (2016). Protege editor to design owl for web services. owlseeditor.semwebcentral.org/documents/tutorial.pdf.
- [Pukkasenung et al., 2010] Pukkasenung, P., Sophatsathit, P., and Lursinsap, C., (2010). An Efficient Semantic Web Service Discovery Using Hybrid Matching. In Proceedings of the 2nd International Conference on Knowledge and Smart Technologies (KST2010), pages 49-53.

- [Purohit and Kumar, 2016a] Purohit, L., and Kumar, S., (2016). Web Service Selection using Semantic Matching. In Proceedings of the ACM Conference on Advances in Communication Technology & Computing (AICTC '16), pages 16:1–16:5.
- [Purohit and Kumar, 2016b] Purohit, L., and Kumar, S., (2016). Exploring K-Means Clustering and skyline for Web Service Selection. In Proceedings of the International Conference on Industrial Information System (ICIIS), pages 603-607. doi: 10.1109/ICIINFS.2016.8263010.
- [Purohit and Kumar, 2018a] Purohit, L., and Kumar, S., (2018). A Classification based Web Service Selection approach. IEEE Transactions on Services Computing. Doi: 10.1109/TSC.2018.2805352
- [Purohit and Kumar, 2018b] Purohit, L., and Kumar, S. (2018). Web Services in the Internet of Things and Smart Cities. IEEE Consumer Electronics Magazine, doi:10.1109/MCE.2018.2880808
- [Purohit and Kumar, 2018c] Purohit, L., and Kumar, S., (2018). A Study on Application of Evolutionary Computing based algorithms for Web Service Selection. Submitted to International Journal of Artificial Intelligence Review, Pages 1-58.
- [Purohit and Kumar, 2018d] Purohit, L., and Kumar, S., (2018). CPSky: A Multi-Layer model for Web Service Selection. Submitted to IEEE Transactions on Services Computing.
- [Purohit and Kumar, 2018e] Purohit, L., and Kumar, S., (2018). Web Services Clustering in Smart city context. Submitted to IEEE Computer, Pages 1-12.
- [Purohit and Kumar, 2018f] Purohit, L., and Kumar, S., (2018). Towards an Approach for Replaceability based Web Service Selection. Submitted to IEEE International Conference on Web Services (ICWS), Pages 1-8.
- [Purohit and Kumar, 2018g] Purohit, L., and Kumar, S., (2018). Clustering based Approach for Web Service Selection using Skyline Computations. Submitted to IEEE International Conference on Service Computing Conference (SCC), 2019, Pages 1-8.
- [Purohit et al., 2015] Purohit, L., Kumar, S., and Khirsagar, D., (2015). Analyzing genetic algorithm for web service selection. In Proceedings of the NGCT'2015, pages 999-1003.
- [Qamar et al., 2015] Qamar, U., Niza, R., Bashir, S., and Khan, F. H., (2015). A Majority vote based classifier Ensemble for Web Service Classification. International Journal of Business Information Systems Engineering, 58(4): 249–259.
- [Quarteroni and Saleri, 2014] Quarteroni, A., and Saleri, F., (2014). Scientific Computing with MATLAB and Octave. second edition, Springer book.

- [R.-Zapata et al., 2011] R.-Zapata, I. S., Gordijn, J., Leenheer, P. D., and Akkermans, H., (2011). Dynamic Cluster-based Service Bundling: A Value-oriented Framework the meeting of the CEC.
- [R.-Zapata et al., 2015] R.-Zapata, I. S., Leenheer, P. D., and Gordijn, J., (2015). Value-Based Service Bundling: A Customer-Supplier Approach. In Proceedings of the IEEE International Enterprise Distributed Object Computing Conference Workshops, pages 237-246.
- [Ran, 2003] Ran, S., (2003). A Model for Web Services Discovery with QoS. ACM SIGCOM Exchange, pages 1–10.
- [Renzis et al., 2016] Renzis, A. D., Garriga, M., Flores, A., and Cechich, A., (2016). Case-based Reasoning for Web Service Discovery and Selection. Electronic Notes in Theoretical Computer Science, Elsevier, pages 89–112.
- [Rhimi et al., 2015] Rhimi, F., Yahia, S. B., and Ahmed, S. B., (2015). Enhancing Skyline Computation With Collaborative filtering Technique for QoS-Based Web Service Selection. In Proceedings of the IEEE 14th Symposium on Network Computing and Applications, pages 247-250.
- [Saleem et al., 2015] Saleem, M. S., Ding, C., Liu, X., and Chi, C. H., (2015). Personalized Decision-Strategy based Web Service Selection using a Learning-to-Rank Algorithm. IEEE Transactions on Services Computing, 8(5): 727-739.
- [Sankowski, 2009] Sankowski, P., (2009). Maximum weight bipartite matching in matrix multiplication time. journal of Theoretical Computer Science, 410(44): 4480-4488.
- [Sasikaladevi and Arockiam, 2012] Sasikaladevi, N., and Arockiam, L., (2012). Genetic approach for service selection problem in composite web service. International Journal of Computer Applications, 44(4): 22–29.
- [Savic, 2002] Savic, D. A., (2002). Single objective vs multi objective optimization for integrated decision support. In Proceedings of Integrated assessment and decision support Proceedings of first biennial Meeting of the International Environmental Modeling and Software Society, pages 7–12.
- [Saxena et al., 2017] Saxena, A., Prasad, M., Gupta, et al., (2017). A review of clustering techniques and developments, Neurocomputing, 267(6): 664-681.
- [Schapire, 2001] Schapire, R. E., (2001). Random Forests. Machine Learning, 45(1): 5-32.
- [Seghir and Khababa, 2018] Seghir, F., and Khababa, A., (2018). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. Journal of Intelligent Manufacturing, 29(8): 1773-1792.

- [Sem, 2016] (2016). The sematic owl-s repository with IOPE description of web services (TC Ver4.1), www.projects.semwebcentral.org/projects/owls-tc.
- [Seo et al., 2005] Seo, Y. J., Jeong, H.Y., Song, Y.J., (2005). Best web service selection based on the decision making between QoS criteria of service. In Proceedings of the International Conference on Embedded Software and Systems, pages 408-419.
- [Sharifara et al., 2014] Sharifara, P., Yari, A., and kashani, M. M. R., (2014). An evolutionary algorithmic based web service composition with quality of service. In Proceedings of the International Symposium on Telecommunications (IST), pages 61–65.
- [Shehu et al., 2014] Shehu, U., Epiphaniou, G., and Safdar, G. A., (2014). A Survey of QoS-aware Web Service Composition Techniques. *International Journal of Computer Applications*, 89(12): 10-17.
- [Sheskin, 2011] Sheskin, D. J., (2011). *Handbook of parametric and nonparametric statistical procedures*, 5th ed., Boca Raton: Chapman & Hall /CRC press.
- [Shirazi, H., 2018] Shirazi, H., (2018). A novel OWL based semantic similarity measure for semantic matching of web services. 1(1): pages 1-22. doi: 10.13140/RG.2.2.27196.33924
- [Shuang et al., 2009] Shuang, K., Yu, S., and Su, S., (2009). TTS-coded genetic algorithm for QoS driven web service selection. In Proceedings of the IEEE International Conference on Communications Technology and Applications, pages 885–890.
- [Silva et al., 2016] Silva, A. S. D., Ma, H., and Zhang, M., (2016). Genetic programming for QoS-aware web service composition and selection. *Soft Computing*, Springer, 20(10): 3851–3867.
- [Singh and Rajput, 2018] Singh, L. K., and Rajput, H., (2018). Dependability Analysis of Safety Critical Real-Time Systems by Using Petri Nets. *IEEE Transactions on Control Systems Technology*, 26(2): 415-426.
- [Singh et al., 2010] Singh, A. K., Goh, A., and Leng, K., (2010). Web Search Engines Function as Key to the World Wide Web. Report Borneo Post, Malaysia.
- [Singh et al., 2012] Singh, A. K., Ravi, K. P., and Leng, A. G. K., (2012). Solving hanging relevancy using genetic algorithm. In Proceedings of the International Conference on Uncertainty Reasoning and Knowledge Engineering, pages 9-12.
- [Singh et al., 2014] Singh, L. K., Vinod, G., and Tripathi, A. K., (2014). Design Verification of Instrumentation and Control Systems of Nuclear Power Plants. *IEEE Transactions on Nuclear Science*, 61(2): 921-930.

- [Singh et al., 2016] Singh, L. K., Rajput, H., Vinod, G., and Tripathi, A. K., (2016). Computing Transition Probability in Markov Chain for Early Prediction of Software Reliability, *International Journal of Quality and Reliability Engineering*, 32(3): 1253-1263.
- [Song et al., 2010] Song, F., Guo, Z., and Mei, D., (2010). Feature selection using principal component analysis. In *Proceedings of the IEEE International Conference on System Science, Engineering Design and Manufacturing Informatization*, pages 27-30.
- [Stephen and Yin, 2011] Stephen, S. Y., and Yin, Y. (2011). QoS-based service ranking and selection for service-based systems. In *Proceedings of the IEEE International Conference on Service Computing*, pages 56-63.
- [Su et al., 2007] Su, S., Zhang, C., and Chen, J., (2007). An improved genetic algorithm for web services selection. In *Proceedings of the International Conference on Distributed Applications and Interoperable Systems, LNCS 4531*, pages 284-295.
- [Tan et al., 2014] Tan, T. H., Chen, M., Sun, J., Liu, Y., and Dong, J. S. (2014). Automated runtime recovery for QoS-based service composition. In *Proceedings of 23rd International Conference on World Wide Web*, pages 563–73.
- [Tang and Ai, 2010] Tang, M., and Ai, L., (2010). A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1-8.
- [Tewari et al., 2012a] Tewari, V., Dagdee, N., and Tiwari, A. (2012). User Oriented Web Services Discovery based on QoS Parameters in Multiple Registries, *International Journal of Computer Applications* 46(24): 8-12.
- [Tewari et al., 2012b] Tewari, V., Dagdee, N., and Tiwar, A., (2012). Extended SOA to Enable Web Service Discovery on Non Functional Parameters. *International Journal of Electronics Communication and Computer Engineering*, 3(2): 97-100.
- [Tripathy et al., 2014] Tripathy, A. K., Patra, M. R., Khan, M. A., Fatima, H., and Swain, P., (2014). Dynamic Web Service Composition with QoS Clustering. In *Proceedings of the IEEE International Conference on Web Services*, pages 678-679.
- [Uden et al., 2016] Uden, L., Liberona, D., and Feldmann, B., (2016). Learning Technology for Education in Cloud - The Changing Face of Education. In *Proceedings of the Communications in Computer and Information Science, Springer*, pages 25-28.
- [Vaadaala et al., 2013] Vaadaala, V., Rao, R. R., and Rao V., (2013). Classification of Web Services using JForty Eight. *International journal of Electronics Communication and Computer Engineering*, 4(6): 181-184.

- [Wang and Hou, 2008] Wang, J., and Hou, Y., (2008). Optimal web service selection based on multi-objective genetic algorithm. In Proceedings of the IEEE International Symposium on Computational Intelligence and Design, pages 553–57.
- [Wang et al., 2007] Wang, H., Tong, P., and Thompson, P., (2007). QoS-based web services selection. In Proceedings of the IEEE International Conference on e-Business Engineering, pages 631–637.
- [Wang et al., 2010] Wang, H., Shi, Y., Zhou, X., and Zhou, Q., (2010). Web Service Classification using Support Vector Machine. In Proceedings of the 22nd IEEE International conference on Tools with Artificial Intelligence, pages 3-6.
- [Wang et al., 2013] Wang, L., Shen, J., Luo, J., and Dong, F., (2013). An improved genetic algorithm for cost-effective data-intensive service composition. In Proceedings of the IEEE International Conference on Semantics, Knowledge and Grids, pages 105–112.
- [Wang et al., 2015a] Wang, D., Yang, Y., and Mi, Z., (April 2015). A genetic-based approach to web service composition in geo-distributed cloud environment. *Computers and Electrical Engineering*, 43: 129-41.
- [Wang et al., 2015b] Wang, P., Chao, K. M., and Lo, C. C., (2015). Satisfaction-based Web service discovery and selection scheme utilizing vague sets theory. *Information System Frontier*, 17(4): 827–844.
- [Wang et al., 2016] Wang, Y., Song, Y., and Liang, M., (2016). A Skyline-based Efficient Web Service Selection Method Supporting Frequent Requests. In Proceedings of the IEEE International Conference on Computer Supported Cooperative Work in Design, pages 328-333.
- [Wang, 2009] Wang, P. (2009). QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception. *International Journal of Expert Systems with Applications*, 36(3): 4460-4466.
- [Wijesiriwardana et al., 2012] Wijesiriwardana, C., Ghezzi, G., Giger, E., Sawada, A., and Gall, H., (2012). Dependency Based Approach for Software Analysis Web Services Replacement. In Proceedings of the Asia-Pacific Software Engineering Conference, pages 294-299.
- [Witten et al., 2016] Witten, I. H., Frank, E., and Hall, M. A., (2016). *Data Mining: Practical Machine Learning Tools and Techniques*, Third Edition, Morgan Kaufmann Publication.
- [Wu and Chen, 2007] Wu, Z., and Chen, Y., (2007). The maximizing deviation method for group multiple attribute decision making under linguistic environment, *International Journal of Fuzzy Sets and Systems*, 158(14): 1608-1617.

- [Wu et al., 2014] Wu, J., Chen, L., Zheng, Z., Lyu, M. R., and Wu, Z., (2014). Clustering web services to facilitate service discovery. *International Journal of Knowledge and Information Systems*, 38(1): 207–229.
- [Wu et al., 2015] Wu, C., Qiu, W., Zheng, Z. Wang, X., and Yang, X., (2015). QoS Prediction of Web Services Based on Two-Phase K-Means Clustering. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 161–168. doi:10.1109/ICWS.2015.31.
- [Xia et al., 2011] Xia, Y., Chen, P., Bao, L. Wang, M., and Yang, J., (2011). A QoS-aware web service selection algorithm based on clustering. In *Proceedings of the International Conference on Web Services*, pages 428-435.
- [Xiang et al., 2015] Xiang, Z., Deng, S., and Gao, H., (2015). Service Selection Using Service Clusters. In *Proceedings of the IEEE International Conference on Services Computing*, pages 769-772.
- [Xiao et al., 2012] Xiao, L., Chang, C. K., Yang, H-I., et al., (2012). Automated Web Service Composition using Genetic Programming. In *Proceedings of the IEEE International Conference on Computer Software and Applications Workshops*, pages 7-12.
- [Yahia et al., 1998] Yahia, H. M., Berroir, J. P., and Mazars, G., (1998). Model-Based Segmentation of Cloud Structures In Satellite Image Sequences. In *Proceedings of the of the IEEE International Conference on Computer Vision*, pages 77-85.
- [Yan et al., 2015] Yan, J., Gao, H., and Mu, Y., (2015). Business Rule Driven Composite Service Optimization and Selection. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 49-56.
- [Yang and Zhou, 2015] Yang, J., and Zhou, X., (2015). Semi-automatic Web Service Classification using Machine Learning, *International Journal of u-and e-service*, 8(4): 339-348.
- [Yang et al., 2015] Yang, Y., Dong, F., and Luo, J., (2015). Computing Service Skycube for Web Service Selection. In *Proceedings of the IEEE 19th International Conference on Computer Supported Cooperative Work in Design*, pages 614-619.
- [Yao and Chen, 2009] Yao, Y., and Chen, H., (2009). Qos-aware service composition using NSGA-II. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS '09*, pages 358–63.
- [Yilmaz and Karagoz, 2014] Yilmaz, A. E., and Karagoz, P., (2014). Improved Genetic Algorithm based Approach for QoS Aware Web Service Composition. In *Proceedings of the IEEE International Conference on Web Services*, pages 463-470.

- [Yin et al., 2009] Yin, K., Zhou, B., Zhang, S., Xu, B., and Chen, Y., (2009). QoS-aware Services Replacement of Web Service Composition. In Proceedings of the IEEE International Conference on Information Technology and Computer Science, pages 271-274.
- [Yin et al., 2016] Yin, F., Lu, L., Chai, J., and Yang, Y., (2016). Combination Weighting Method Based on Maximizing Deviations and Normalized Constraint Condition. International Journal of Security and Its Applications, 10(2): 39-50.
- [Yu and Bouguettaya, 2012] Yu, Q., and Bouguettaya, A. (2012). Multi-attribute optimization in service selection. International Journal of World Wide Web, 15 (1): 1-31.
- [Yu and Bouguettaya, 2013] Yu, Q., and Bouguettaya, A., (2013). Efficient service skyline computation for composite service selection. IEEE Transactions on Knowledge and Data Engineering, 25(4): 776–789.
- [Yu and Gen, 2010] Yu, X., and Gen, M. (2010). Introduction to Evolutionary Algorithms. Springer-Verlag London publications.
- [Yu et al., 2015] Yu, Q., Chen, L., and Li, B., (Jan 2015). Ant colony optimization applied to web service compositions in Cloud Computing. Computers and Electrical Engineering, 41: 18-27.
- [Yuan et al., 2013] Yuan, Y., Zhang, X., Sun, W., Cao, Z., and Wang, H., (2013). Optimal web service composition based on context-awareness and genetic algorithm. In Proceedings of the International Conference on Information Science and Cloud Computing Companion, pages 660–67.
- [Zaharie et al., 2011] Zaharie, D., Perian, L., and Negru, V., (2011). A view inside the classification with Non-Nested Generalized Exemplars. In Proceedings of the IADIS'11, pages 19-26.
- [Zaki and Meira, 2014] Zaki, M. J., and Meira, W., (2014). Data Mining and Analysis, Fundamental Concepts and Algorithms. Second Edition, Cambridge University press.
- [Zhang and Lin, 2009] Zhang, C., and Lin, X., (2009). A genetic algorithm with improved convergence capability for QoS aware web service selection. In Proceedings of the IEEE International Conference on Management and Service Science, pages 1-4.
- [Zhang and Ma, 2009a] Zhang, C., and Ma, Y., (2009). Dynamic genetic algorithm for search in web service compositions based on global QoS Evaluation. In Proceedings of the 8th IEEE International Conference on Embedded Computing, China, pages 644–649.

- [Zhang and Ma, 2009b] Zhang, C., and Ma, Y., (2009). Genetic algorithm for QoS-aware web service selection based on chaotic sequences. In Proceedings of the IEEE International Conference on Network-Based Information Systems, pages 410–416.
- [Zhang and Ren, 2011] Zhang, Y. and Ren, M., (2011). Web service selection based on utility of weighted QoS attributes. In Proceedings of the International Conference on Web Information Systems and Mining, pages 417-25.
- [Zhang et al., 2006a] Zhang, C., Su, S., and Chen, J., (2006). A novel genetic algorithm for qos-aware web services selection. In Proceedings of the International Workshop on Data Engineering Issues in E-Commerce and Services, pages 224–235.
- [Zhang et al., 2006b] Zhang, C., Su, S., and Chen, J., (2006). Efficient population diversity handling genetic algorithm for QoS-aware web services selection. In Proceedings of the of ICCS, LNCS 3994, pages 104-111.
- [Zhang et al., 2007] Zhang, C., Su, S., and Chen, J., (2007). Diga: Population diversity handling genetic algorithm for qos aware web services selection. *Computer Communications*, 30(5): 1082-90.
- [Zhang et al., 2013] Zhang, X., Wang, Z., Lv, X., and Qi, R., (2013). A Clustering-based QoS Prediction Approach for web service selection. In Proceedings of the ICISCCC, pages 201-206.
- [Zheng et al., 2011] Zheng, Z., Ma, H., Lyu, M. R., and King, I., (2011). QoS-Aware Web Service Recommendation by Collaborative Filtering. *IEEE Transactions Services Computing*, 4(2): 140-152.
- [Zhang, 2011] Zhang, C., (2011). Adaptive genetic algorithm for QoS-aware service selection. In Proceedings of the International Conference on Advanced Information Networking and Applications, Singapore, pages 273–278.
- [Zhao and Li, 2014] Zhao, X., and Li, Z. X., (2014). QoS-aware web service selection with negative selection algorithm. *Knowledge Information System*, 40(2): 349-373.
- [Zhao et al., 2014a] Zhao, X., Wen, Z., and Li, X., (2014). Qos-aware web service selection with negative selection algorithm. *Knowledge and Information Systems*, 40(2): 349–373.
- [Zhao et al., 2014b] Zhao, Y., Zhao, Y., Lin, R., Zou, H., (2014). Mining Service Tags with Enriched Information from the Internet. In Proceedings of the IEEE World Congress on Services. Pages 265–270. doi:10.1109/SERVICES.2014.55.
- [Zhao et al., 2017] Zhao, H., Wen, J., Zhao, J. and Luo, F., (2017). A new model-based Web service clustering algorithm. In Proceedings of the IEEE Region 10 Annual International Conference on TENCON, pages 3468–3472. doi:10.1109/TENCON.2016.7848699.

[Zheng et al., 2012] Zheng, K., Xiong, H., Cui, Y., Chen, J., and Han, L. (2012). User clustering-based web service discovery. In Proceedings of the 6th International Conference on Internet Computing Science Engineering (ICICSE), pages 276–279. doi:10.1109/ICICSE.2012.40.

[Zhi-peng et al., 2009] Zhi-peng, G., Jian, C., Xue-song, Q., and Luo-ming, M., (2009). QoE/QoS driven simulated annealing-based genetic algorithm for web services selection. The Journal of China Universities of Posts and Telecommunications, Elsevier, pages 102–107.

[Zhu et al., 2010] Zhu, Z., Yuan, H., Song, J., Bi, J., and Liu, G., (2010). WS-SCAN: A Effective Approach for Web Services Clustering. In Proceedings of the International Conference on Computer Application and System Modeling (ICCASM 2010), pages 618-622.

