QoS-aware Service Composition using Computational Intelligence Techniques

A thesis submitted to University of Hyderabad in partial fulfillment

for the degree of

Doctor of Philosophy

by

Jatoth Chandrashekar

Reg. No. 12MCPC11

SCHOOL OF COMPUTER AND INFORMATION SCIENCES
UNIVERSITY OF HYDERABAD
HYDERABAD -500046
Telangana
India

November, 2017

CERTIFICATE

This is to certify that the thesis entitled **"QoS-aware Service Composition using Computational Intelligence Techniques"** submitted by **Jatoth Chandrashekar** bearing **Reg. No. 12MCPC11** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under my supervision and guidance at IDRBT, Hyderabad.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Parts of this thesis have been published online in the following publications:

1. IEEE Transactions on Services Computing 2017 (Chapter 2)
2. Proceedings of KES-IDT 2015, Springer (Chapter 1 and Chapter 3)
3. Soft Computing, Springer 2017 (Chapter 4)
4. Future Generation Computer Systems, Elsevier 2017(Chapter 5)

Further, the student has passed the following courses towards fulfillment of course-work requirement for Ph.D:

	Course Code	Name	Credits	Pass/Fail
1	BT701	Data Structures and Algorithms	4	Pass
2	BT702	Operating System and Programming	4	Pass
3	BT709	Advanced Software Engineering	4	Pass
4	BT718	Enterprises IT Architecture	4	Pass

Supervisor	Director	Dean
Dr. G. R. Gangadharan	Dr. A.S. Ramasastri	Prof. Arun Agrawal
IDRBT	IDRBT	School of Computer and
Hyderabad-500 057, India	Hyderabad-500 057, India	Information Sciences, UOH
		Hyderabad-500 046, India

DECLARATION

I, **Jatoth Chandrashekar**, hereby declare that this thesis entitled **"QoS-aware Service Composition using Computational Intelligence Techniques"** submitted by me under the guidance and supervision of **Dr. G. R. Gangadharan**, is a bonafide research work and is free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Date:

Signature of the Student

(Jatoth Chandrashekar)

Reg. No.: 12MCPC11

//Countersigned//

Signature of the Supervisor

(Dr. G. R. Gangadharan)

Dedicated To My Family & Teachers

Acknowledgements

First and above of all, I praise **God Almighty**. I am very much grateful to him for his incredible love towards me, the gift of everlasting life and the grace of amazing energy to work hard.

This thesis would not have been possible without help and support of the kind people around me, to only some of whom it is possible to give the particular mention here. Among them, first of all, I owe my greatest gratitude to my supervisor **Dr. G. R. Gangadharan**, Associate Professor, IDRBT, Hyderabad for his precious regular encouragement and timely support from the introductory level to the concluding level that enabled me to understand and implement the concepts learned. He guided me in each and every stage of my research.

Later, I would like to thank my parents **Sri. Hatchaiah** and **Smt. Suguna** for their best support and encouragement all times in pursuing my dreams. I would also like to thank my wife **Indira Jyothi** and my daughters **Kritika Sashi** and **Shresta** for their infallible love and encouragement have always been my strength. I would also like to thank my uncle **Sri. Sukko Naik** and my aunt **Smt. Kotamma** for their support.

It is my previlage to thank **Prof. Arun Agrawal**, Dean, School of Computer and Information Sciences (SCIS), UoH, Hyderabad for his academic support throughout research work. It is an honor for me to thank **Dr. A.S. Ramasastri**, Director, IDRBT for his suggestions throughout my research. I also thank **Mr. B. Sambamurthy**, Former Director, IDRBT for extending his cooperation at the preliminary stage of my research. I would like to extend my sincere thanks to **Prof. V. N. Sastry**, Professor, IDRBT and **Prof. V. Ravi**, Professor, IDRBT for their regular reviews and inputs as the members of my Doctoral Research Committee.

I would like to express my thanks to my seniors including Dr. Sridhar Naik B., Dr. Ilaiah K., Dr. Srinivas Naik N. for their invaluable support. I would like to express my special thanks my research colleagues Pradeep Kumar D. and Gutha Jaya Krishna for their technical support in my research work. It is always pleasure to express my sincere thanks to the my research colleagues including Ghanashyam B., Srinu Naik M., Shravan B., Gopal, Hiran, Kumar Ravi, Sriramulu, Sandhya, Anup, D. Tiwari, Manu, Dinesh Reddy V., Kamaruddin Sk., Uyyala Ravi, Vamshi Krishna, Srujana, Sitara, Siva Kumar, Bhaskar Bhattu, and Babu Rao for their daily interactions with me and kind support. I would like to express my thanks to my colleagues including Chaya Jadhav, Arivanantham (and his Mother), Wasudev Rahane, Swati Nikam for keeping me motivated and financial support throughout the duration of work.

Nothing would have been possible without the prayer support of two churches namely The Pentecostal Mission, Palvancha and Secunderabad headed by Pastors. I never forget the kind prayers of Sunday School children and express my sincere and heart-felt thanks to all.

.....*Chandrashekar Jatoth*

Abstract

Service orientation is based on the idea of composing business workflows by discovering and invoking the most suitable services and building new applications to satisfy a specific business requirement. Web services are one of the most active and widely adopted implementations of service-oriented computing. There are several web services available on the Internet. However, these services, as individual entities, often cannot satisfy a user's requirements to perform a complex task. In this situation, combining several candidate web services to create a composite web service that aims to fulfill a user's functional as well as non-functional requirements is a challenge. As candidate services exhibit different quality of service (QoS) values, it becomes a problem to compose only the exact candidate services that will yield a composite service with the best QoS. An individual candidate service in isolation may exhibit an optimal level of performance. However, the same service in a composition may or may not be effective in delivering the same level of performance. Furthermore, the rapid growth in the number of services having similar functionality with different QoS parameters expands the candidate services space, which leads to a combinatorial explosion problem and involves enormous computational time and cost. Hence, handling these services with reduced cost and computational time while satisfying the user's QoS constraints has become crucial in QoS-aware service composition. In this thesis, we propose QoS-aware web service composition with balancing of QoS parameters and connectivity constraints, overcoming the intrinsic defects by the following methods: (1) OFASC-AGEGA, a novel approach that evaluates service fitness and composition fitness using an Adaptive Genotypes Evolution-based Genetic Algorithm, and (2) OFASC-MIWO, an approach

that evaluates individual service fitness and composition fitness using a modified Invasive Weed Optimization Algorithm.

Today, various cloud service providers offer cloud services with comparable functionality but varying in price and performance levels. Often, there may be trade-offs among different functional and non-functional requirements fulfilled by various cloud providers. Hence, it is hard to select suitable cloud services and evaluate their relative performance and their ranking based on different quality of service (QoS) parameters. In this thesis, we propose an extended version of Data Envelopment Analysis (DEA) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) variants for cloud service selection. In these approaches, Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), and Fuzzy AHP are used to determine the weights of the criteria (or QoS parameters), and these methods are integrated with DEA, Super-efficiency DEA (SDEA), TOPSIS, Fuzzy TOPSIS, and Grey TOPSIS to evaluate the relative efficiency of cloud services and their ranks.

The explosion of social, transactional, and sensor data has strongly influenced the rapid development of the Internet of Services (IoS). The services and resources involved in processing big data have dramatically increased in both number and complexity. These services typically come from multiple domains and heterogeneous networks. Owing to the rapid growth of big data and services, computational intelligence algorithms require more iterations and more computation time to find the optimal or near-optimal solutions, leading to research in big service composition. In this thesis, we propose a novel QoS-aware big service composition using a MapReduce-based evolutionary algorithm with guided mutation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Service-oriented computing (SOC) is a computing paradigm with the aim of developing dynamic business processes and applications covering different computing platforms and organizations by gathering application components into a loosely coupled network of services [7, 8]. Service orientation is based on the idea of composing business workflows by discovering and invoking the most suitable services using quality of service (QoS) parameters and building the new applications to satisfy a particular business requirement. Web services are one of the most active and widely adopted implementations of SOC; they include a set of technologies and composable standards with well-defined interfaces [8]. The World Wide Web Consortium (W3C) defines a web service as "a software application identified by a uniform resource identifier, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts" [9]. The success of web services is due to the fact that their development is based on existing, ubiquitous infrastructures such as the Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), and Extensible Markup Language (XML) [10].

Figure 1.1 depicts the architectural workflow of a web service, consisting of three types of entities: service provider, service broker, and service consumer (requester). Web Service Description Language (WSDL); Universal Description, Discovery, and Integration (UDDI); and Simple Object Access Protocol (SOAP) are XML-based standards developed for service description, discovery, and invocation (or communication), respectively. A service broker acts as an intermediary between service requesters and providers. A UDDI-based service registry is a specialized instance of a service broker.

Under this configuration, the UDDI registry serves as a broker where service providers publish the definitions of the services they offer using Web Service Description Language (WSDL) and service requesters find information about the services available.

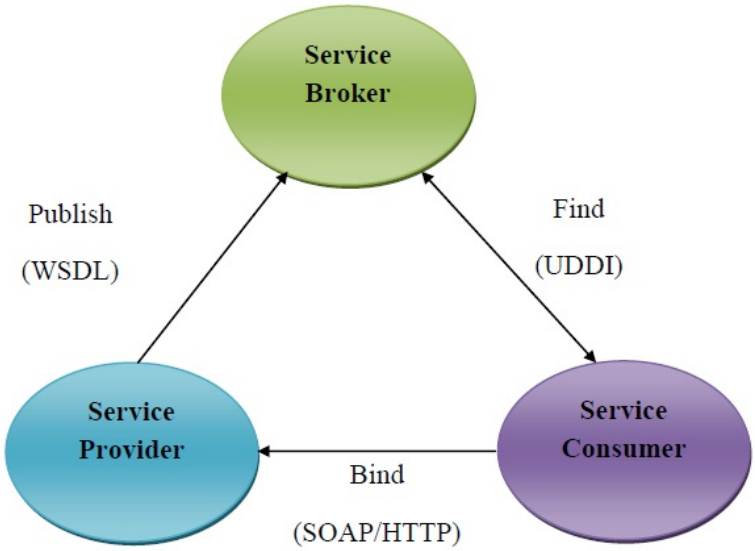


Figure 1.1: The architectural workflow of web services (Based on [1])

Nowadays, several web services available on the Internet [10]. They gained further popularity with the emergence of Cloud computing that provides an easy-to-use and setup environment for hosting applications as (web) services on a pay-as-you-go basis [11]. Cloud computing is a network-based model in which computers and other devices can accept various shared resources, software, and information on demand [12]. Without the ownership of technology infrastructure, a user can access technology-enabled services from the cloud via the Internet [11]. The architecture of cloud computing is depicted in detail in Fig. 1.2. The cloud offers several benefits to businesses by providing different services at reduced cost (the businesses pay only for what they use) [13]. There are, broadly, three service delivery models in cloud computing [13]:

- Software-as-a-Service (SaaS): A software distribution model in which a vendor / service provider hosts the applications, and these are made available to the consumers over the Internet.

- Platform-as-a-Service (PaaS): A paradigm under which the operating systems and associated services are delivered over the Internet without being downloaded/installed on host systems.
- Infrastructure-as-a-Service (IaaS): A model in which the equipment (including storage, hardware, servers, and networking components) for supporting various operations is outsourced.

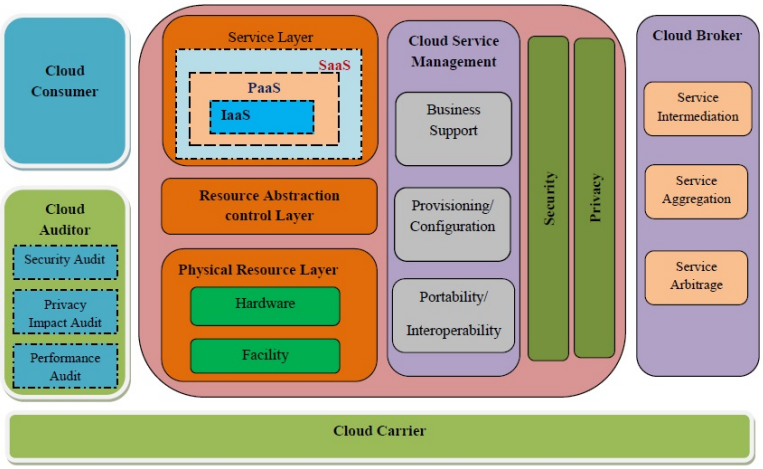


Figure 1.2: The detailed architecture of cloud computing (Based on [2])

Based on the concept of Everything as a Service (XaaS) or cloud computing architecture, including SaaS, IaaS, and PaaS layers, cloud computing provides the networked infrastructure with tremendous service capacity. It enables the virtualization and online consumption of massive services and resources in a cyber-physical world to serve the large customer base. The explosion of social, transactional, and sensor data has strongly influenced the rapid development of the Internet of Services (IoS) [14]. The services and resources involved in processing big data have dramatically increased in both number and complexity. These services typically come from multiple domains and heterogeneous networks. Big services comprise of interrelated services across virtual (e.g. cloud services) and physical domains (public transportation), and are rapidly evolving for analyzing and processing big data having the properties of customer focus, heterogeneity, massiveness, convergence, complexity, credibility, and value [3]. These features correspond to and enhance the Five Vs (volume, variety, velocity, veracity, and

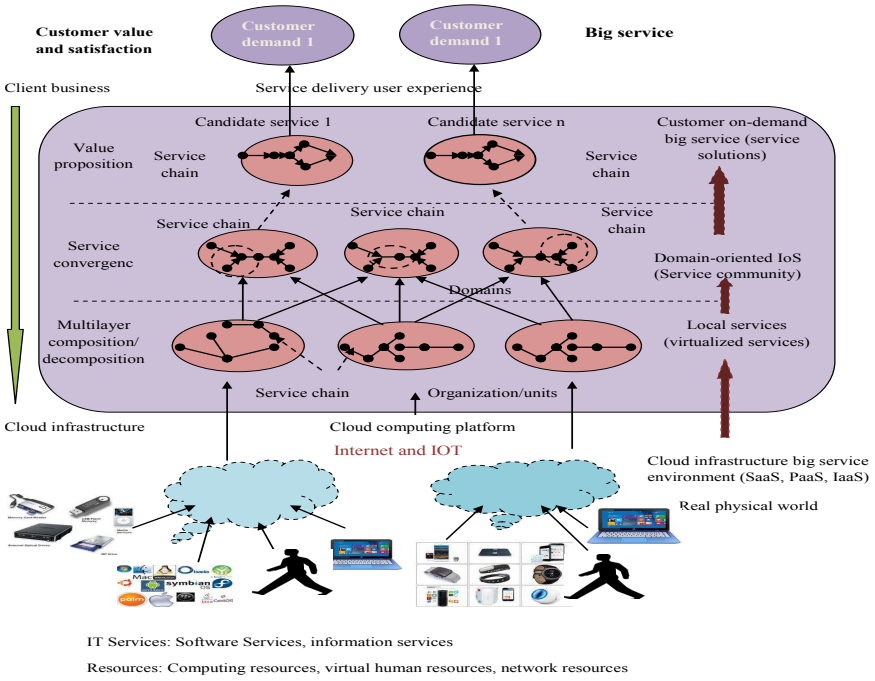


Figure 1.3: The reference architecture of Big Service (Based on [3])

value) of Big Data [15]. Figure 1.3 depicts the reference architecture of big services. A big service ecosystem (also known as an IoS) hypothesizes massive software services on the Internet [3]. These services take complex forms, come from various sources, and are networked and interoperate across multiple domains and heterogeneous networks.

1.1 QoS-aware Service Composition Primitives and Approaches

Service composition is a strategy of combining several services to create a composite service and that aims to fulfill the functional requirements as well as the non-functional requirements provided by a user [16, 17]. The output of the service composition process is a composite service whose Quality of Service (QoS) satisfies the user request and is a combination of the QoS values of its individual candidate services. For instance, consider that a user wants to plan her whole trip comprising flight booking, hotel booking, and payment transaction. Further, the user demands that the price of

the execution of this composite service should be minimized. It is impossible for a single web service to complete such a complex task as it includes several sub-tasks (flight booking, hotel booking, and payment transaction) and a QoS requirement, namely minimum execution cost, that must be satisfied before the overall request is successfully completed.

Generally, the service composition structure consists of four stages: planning, discovery, optimization and selection, and execution [18]. Figure 1.4 depicts the structure of a service composition and its corresponding components. Suppose a user requests a composite service; the initial stage is to identify the workflow of the abstract services. The second stage is a discovery process whose main aim is to search a set of candidate services from a repository for each abstract service and generate a candidate service model. The third stage is to select a set of optimal candidate services for creating a composite service. The fourth stage is concerned with execution, whereby the composite service is subsequently procured. As services are dynamic in nature and different service consumers may have different non-functional requirements (or QoS parameters) and functional requirements, we need to re-plan the workflow or re-select the candidate services if the composite service does not satisfy the customer requirements [18].

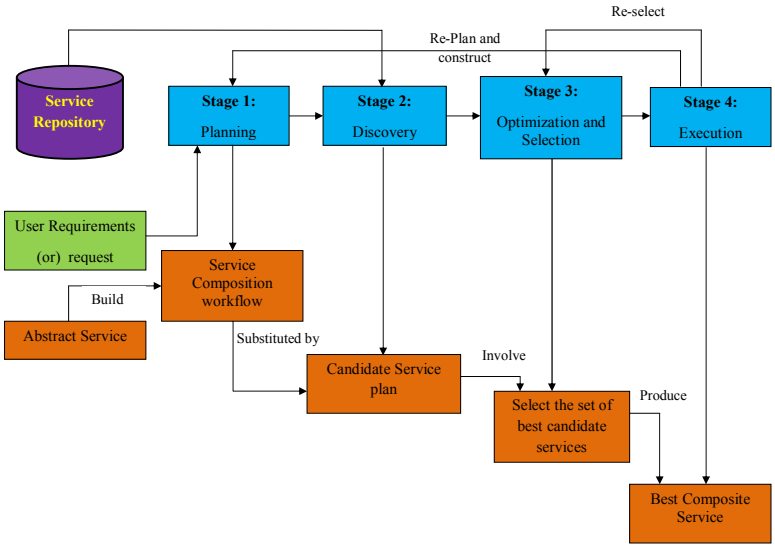


Figure 1.4: Service composition structure (Based on [4])

Let a web service composition task $CT = \{T_1, T_2, T_3, \ldots, T_n\}$ comprises n tasks (or abstract services) such that each task $T_i \in CT$ accommodates a set of services, which are the subset of available candidate services $CS = \{ST_1 = \{s_{1\,1}, s_{1\,2}, \ldots, s_{1\,i}\}, ST_2 = \{s_{2\,1}, s_{2\,2}, \ldots, s_{2\,j}\}, \ldots, ST_p = \{s_{p\,1}, s_{p\,2}, \ldots, s_{p\,m}\}\}$, where i, j, and m denote the number of candidate services in ST_1, ST_2, and ST_p, respectively. The services in a set $ST_i = \{s_{i\,1}, s_{i\,2}, \ldots, s_{i\,x}\}$ comprise the collection of the x services that can address the task T_i. The candidate services in the subset ST_i have similar functionalities but can differ in their QoS parameters. The QoS parameters of candidate services can be represented as $P = \{p_1, \ldots, p_q\}$, where q denotes the number of parameters. Further, each QoS parameter in P is associated with QoS constraints and preferences given by the user.

The QoS parameters of a service are bifurcated as positive parameters and negative parameters. A positive parameter uses higher values to represent higher user utility (i.e., availability and reliability); a negative parameter uses higher values to denote lower user utility (i.e., cost and response time) [19]. We need to calculate normalized values of the positive and negative QoS parameters as the range of their values is enormous. Table 1.1 reports the list of QoS parameters and their descriptions. Before evaluating the fitness function (or utility function) of a composite service, we determine the normalized value of each QoS parameter as follows [19, 20, 21]:

$$V_p(x) = \begin{cases} 1 & \text{if } p^{\max} = p^{\min} \\ \dfrac{p^{\max} - x}{p^{\max} - p^{\min}} & \text{if } p \text{ is negative} \\ \dfrac{x - p^{\min}}{p^{\max} - p^{\min}} & \text{if } p \text{ is positive} \end{cases} \tag{1.1}$$

where $p^{\min}$ and $p^{\max}$ are the minimum value and maximum value, respectively, of the QoS parameter p for each candidate service and x is the parameter value for a particular service.

The global QoS values are computed by aggregating the QoS parameter values of selected services for each task that is involved in the composition. The best composition is obtained by maximizing the fitness function values of the global QoS parameters

Table 1.1: List of QoS parameters and their description

S.No	QoS Parameter	Description
1	Cost	The amount needed to pay by user for requesting a service.
2	Throughput	The total number of web service invocations possible in a given amount of time (measured in invokes/sec).
3	Response time	Time taken in seconds to serve a request once the request is received (measured in milliseconds).
4	Availability	How frequently a service is available during the user request (measured in percentage).
5	Reliability	How frequent a service is available during the user request (measured in percentage).

based on the user's preferences. The user expresses her preferences as weights (w_i) for each QoS parameter.

Let A_p be the aggregation function of a QoS parameter p that computes the global values of p for a composite service. A fitness function, in general, evaluates the multi-dimensional QoS parameters of a composite service [19, 20, 21]. In this thesis, we use the Simple Additive Weight (SAW) method to evaluate the fitness of a composite service and its associated multi-dimensional QoS parameters. The fitness functions of the normalized parameters are assumed to be linear. Thus, the objective of the optimization is to find the value of X that maximizes the global fitness function, which can be stated as follows:

$$\begin{aligned} d(X) &= \sum_{i=1}^{m} \sum_{p \in Q} A_p\left(v_p(i)\right) \times w(p) \\ &\text{subject to constraints:} \quad A_p\left(v_p(i)\right) \leq A_p\left(v_p^c(i)\right); \\ &\text{having } \textstyle\sum_{p \in Q} w(p) = 1 \end{aligned} \tag{1.2}$$

where $A_p\left(v_p^c(i)\right)$ represents the global QoS constraints given by the user and $v_p(i)$ is evaluated by the aggregation functions of the global QoS parameters.

QoS aware service selection requires the selection of composition of services that satisfies all constraints while optimizing several QoS values at the same time. QoS aware service selection approaches proceed to the selection of services using local optimization and/ or global optimization. In local optimization, the process of selecting services is undertaken independently for each abstract service [22, 23, 24, 25, 26]. In

terms of computation time, this approach can be appropriate when the global QoS constraints are decomposed [27, 28, 29, 30, 31] into local constraints to overcome their drawbacks. However, the main drawbacks of this approach are (i) local selection relies on the greedy approach and (ii) the decomposition deals with the QoS parameters independently. Further, this approach does not satisfy global QoS constraints.

In global optimization, QoS-aware service composition problems can be solved at both local and global levels [5, 32, 33, 34]. When using the global optimization approach, the service selection problems are modeled as 0/1 knapsack problems [35, 36], multi-dimensional and/or multi-objective optimization problems [37, 38, 39, 40, 41], constraint optimization problems [42, 43], weighted acyclic graphs [44, 45], or mathematical programming problems [5, 25, 32, 46]. Global optimization approaches are generally categorized into three types: non-heuristic, heuristic, and meta-heuristic [16]. Detailed explanations of these algorithms are given in chapter 2.

Various composition approaches use the basic workflow patterns/structures. The workflow patterns are classified into four types: sequential, parallel, switching, and looping. Fig. 1.5 depicts the basic workflow patterns/ structures in a service composition workflow, where AS1, AS2,. . ., and AS8 are abstract services.

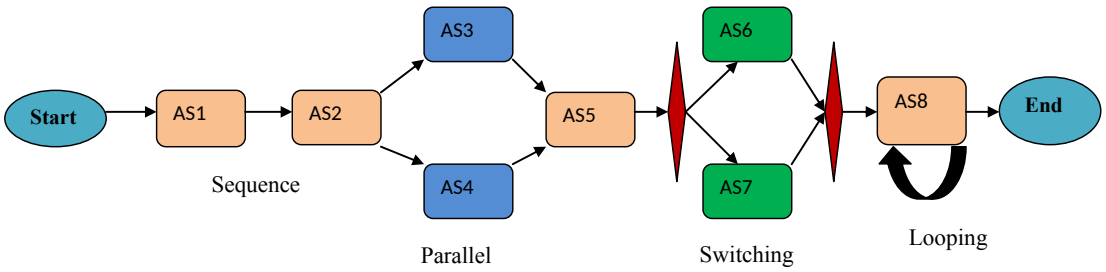


Figure 1.5: Basic composition patterns (Based on [5]) in a service composition workflow

The global QoS parameter is evaluated by applying the aggregation functions recursively to the building blocks that define the structure of the composition [33]. For example, if a user is interested in the global availability of a composite service and the services are executed in sequence, then the global availability is the sum of the availability of the individual services, but if the services are executed in parallel, it will be calculated as the product of the availability of the individual services.

Table 1.2 illustrates the aggregation functions of QoS parameters for the basic workflow patterns (sequential, parallel, switching, and looping), where P_i denotes the probability of branch i and k denotes the number of loops.

Table 1.2: Aggregation functions of QoS parameters

QoS parameter name	Sequential	Parallel	Switching	Looping
Cost	$\sum_{i=1}^{n} Cost\left(s_j^i\right)$	$\sum_{i=1}^{n} Cost\left(s_j^i\right)$	$k * \sum_{i=1}^{n} Cost\left(s_j^i\right)$	$k * \sum_{i=1}^{n} Cost\left(s_j^i\right)$
Throughput	$min_{i=1}^{n} Throughput\left(s_j^i\right)$	$min_{i=1}^{n} Throughput\left(s_j^i\right)$	$P_i * \prod_{i=1}^{n} Throughput\left(s_j^i\right)$	$k * \prod_{i=1}^{n} Throughput\left(s_j^i\right)$
Response time	$\sum_{i=1}^{n} Responsetime\left(s_j^i\right)$	$min_{i=1}^{n} Responsetime\left(s_j^i\right)$	$P_i * \sum_{i=1}^{n} Responsetime\left(s_j^i\right)$	$k * \sum_{i=1}^{n} Responsetime\left(s_j^i\right)$
Availability	$\prod_{i=1}^{n} Availability\left(s_j^i\right)$	$\prod_{i=1}^{n} Availability\left(s_j^i\right)$	$P_i * \prod_{i=1}^{n} Availability\left(s_j^i\right)$	$k * \prod_{i=1}^{n} Availability\left(s_j^i\right)$
Reliability	$\prod_{i=1}^{n} Reliability\left(s_j^i\right)$	$max_{i=1}^{n} Reliability\left(s_j^i\right)$	$P_i * \prod_{i=1}^{n} Reliability\left(s_j^i\right)$	$k * \prod_{i=1}^{n} Reliability\left(s_j^i\right)$

The objective of the composition is to select an optimal candidate service from among many services to resolve the task T_i, since the selection of the candidate service will influence the quality of the composition.

1.2 Computational Intelligence for Service Composition

Computational intelligence (CI) techniques are adaptive mechanisms to facilitate intelligent behavior in complex and real-world problems [47]. These techniques are able to study, model, and analyze complex phenomena so that they can solve complex and real-world problems for which there are no effective algorithms [47, 48]. The inspiration for CI is the human mind, and it is based on the fact that the human mind can store and process information that is pervasively imprecise, uncertain, and lacking in categoricity. The guiding principle of CI is to "exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness, and low solution cost" [49].

Generally, CI techniques are set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional or mathematical modelling may fail due to the complexity in mathematical modelling, uncertainties in the problems, and stochastic nature of the problem. CI techniques are combination of techniques such as fuzzy logic, neural network, learning theory, evolutionary

algorithms, and probabilistic methods. CI techniques are used for solving complex problems such as NP-hard for which there are no effective algorithms [47, 48].

QoS-aware service selection and composition are the pivotal problems in SOC. With the proliferation of seamless web services, it becomes challenging to find an optimal service for composition that satisfies a user's requirements. Further, the rapid growth in the number of services having similar functionality with different QoS parameters expands the candidate services space, which leads to a combinatorial explosion problem or NP-Hard problem [5, 33] and involves enormous computational time and cost to find the optimal composition. Hence, optimal service composition with reduced cost and computational time and satisfying the user QoS constraints has become crucial. CI techniques resolve this problem by producing near-optimal solution within less time. Based on literature survey, we classified the existing approaches as follows: non-heuristics, heuristics, and meta-heuristics [16, 19, 27, 33, 41, 48, 49]. Non-heuristic algorithms solve optimization problems significantly faster than exhaustive search [50]. Heuristic algorithms are the algorithms derived from experience with particular optimization problems. Usually, they find an optimal solution by a trial-and-error method in an adequate amount of time by taking full advantage of the particularities of the problem. The solutions by these algorithms may be better then an educated guess but may not be optimal [51]. Since non-heuristic algorithms take a massive amount of time to procure the optimal solution, heuristic algorithms are preferred because these algorithms can procure near-optimal solutions in an acceptable amount of time. Near-optimal solutions can be found through highly effective algorithms, often called Metaheuristics [52]. These search methods are highly recommended for getting the good solutions, that are optimal and may be sub-optimal in few cases, in polynomial time instead of exponential time which happens when we solve these problems using conventional methods. All the nature inspired metaheuristics algorithms strive to attain the global optimal solution by inspecting the most favorable locations in their domain search space, based on the natural mechanism that the species possess. The process of this inspection varies among various algorithms. Few algorithms may work well for certain problems, while not-so-good for other problems. This is due to the randomization or stochastic strategies that are used in solving those NP-Hard problems [47, 48].

As we could expect the presence of multiple local minima or local maxima in our service search space, metaheuristic algorithms forage and try to explore as many regions as possible to find the global solution. The efficiency of an algorithm is decided by the speed of convergence which is reaching for the global solution, an ability to escape local solutions and time taken for the execution of an algorithm. Thus, CI methods are preferred for QoS-aware service composition. The detailed description is presented in section 2.2.

1.3 Motivations, Objectives, and Contributions

QoS-aware service selection and composition are the pivotal problems in SOC. With the proliferation of seamless web, cloud, and Big services, it becomes challenging to find an optimal service for composition that satisfies a user's requirements. An individual candidate service in isolation may exhibit an optimal level of performance. However, the same service in a composition may or may not be effective in delivering the same level of performance [53]. Further, the rapid growth in the number of services with similar functionality but different QoS parameters expands the candidate services space, which leads to a combinatorial explosion problem [25, 33] and involves enormous computational time and cost. Hence, handling these services with reduced cost and computational time and satisfying the user QoS constraints has become crucial in QoS-aware service composition. The potential commercial benefits of optimal service composition are considerable, and this motivates the enormous research efforts being expended in this area [54, 55, 56].

To address this problem, in this thesis, we propose few approaches for web, cloud, and big services using CI techniques. The main aim of these approaches is to find the best composite service fulfilling the functional requirements and meeting the end-to-end QoS (non-functional) requirements. The objectives of this thesis are as follows:

- To gain insight from the state-of-the-art computational intelligence methods for selection and composition of web services, cloud services, and big services and to understand the research gaps that need to be addressed.
- To propose solutions for optimal web services selection and composition, balancing QoS factors and satisfying connectivity constraints.

- To propose solutions for assessing the relative performance of cloud services based on QoS factors and for selecting the best cloud service.
- To present a method for solving QoS-aware big service composition problems.

The contributions of this thesis are as follows:

- Solutions for optimal web service composition that balance QoS factors and satisfy QoS connectivity constraints for achieving search optimization and reducing computational complexity (see chapter 3) using

  (i) Optimal fitness-aware service composition using an Adaptive Genotypes Evolution-based Genetic Algorithm.

  (ii) Optimal fitness-aware service composition using Modified Invasive Weed Optimization technique.

- Solutions for evaluating the cloud services and their efficiencies using (see chapter 4)

  (i) Extended Data Envelopment Analysis (DEA) and Extended Super-efficiency DEA for evaluating the cloud services and their efficiencies with consideration of user preferences.

  (ii) Extended versions of the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), the Grey TOPSIS, and the Fuzzy TOPSIS for selection of the best cloud services.

- Solution for big service composition (see chapter 5) using MapReduce-Based Evolutionary Algorithm with Guided Mutation (MR-EA/G).

1.4 Thesis Organization

The thesis is structured as follows.

Chapter 2 describes a Systematic Literature Review (SLR) on web service composition using CI techniques.

Chapter 3 presents techniques for QoS-aware web service composition using Adaptive genotypes evolution based genetic algorithm and modified invasive weed optimization.

Chapter 4 presents two techniques for cloud service selection based on extended versions of DEA and SDEA and extended versions of TOPSIS and Grey TOPSIS and Fuzzy TOPSIS.

Chapter 5 presents a MapReduce-based evolutionary algorithm with guided mutation algorithm to solve QoS-aware Big service composition.

Chapter 6 presents the conclusions drawn and future research directions in the case of service composition using CI techniques.

Chapter 2

A Systematic Literature Review of QoS-aware Service Composition

A systematic literature review (SLR) is a structured process that identifies, taxonomically classifies, and systematically compares existing evidence of research suitable for a specific problem [6, 57]. In this chapter, we primarily focus on the state-of-the-art research on solving QoS-aware web service composition problems using computational intelligence techniques. The major contributions of this SLR include identifying the different objectives of CI-based QoS-aware service composition and classifying the different existing computational intelligence approaches. This SLR gives a systematic description for researchers in service-oriented computing and computational intelligence and helps to gain on research implications, solutions, and future directions.

2.1 Research Methodology

A research methodology is a process for the systematic and theoretical analysis of the existing approaches employed to address a specific problem. An SLR is a research methodology that, in turn, includes critical appraisal, evaluation, and the review of all available research articles and phenomena addressing a particular research query [6]. It reduces bias and follows precise and strictly sequential

steps to review the state of the art. We followed a five-step review process for conducting our systematic literature review, as described in [57, 58] (see Fig. 2.1).

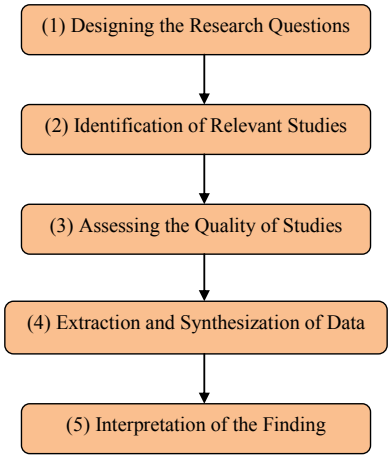


Figure 2.1: A process of conducting systematic literature review (Based on [6])

(1) ***Designing the Research Questions:*** In order to conduct a review, we designed the research questions and provide their motivations as shown in Table 2.1.

Table 2.1: Research questions and their motivations

Research Questions	Motivations
RQ1—What are the fundamental research inspirations for CI-based QoS-aware service composition?	To gain an understanding of CI-based QoS-aware service compositions that satisfy functional requirements and non-functional requirements.
RQ2—What are the QoS parameters generally used in CI-based QoS-aware service composition?	To identify the QoS parameters that are utilized as a part of CI-based QoS-aware service composition.
RQ3—What are the existing methods and techniques that support CI-based QoS-aware service composition?	To distinguish, analyze, and organize the existing methods and techniques that are used in CI-based QoS-aware service composition.
RQ4—What are the existing research issues and what are the upcoming areas in CI-based QoS-aware service composition?	To understand the research gaps that need to be addressed and to identify future directions in this field.

(2) ***Identification of Relevant studies:*** We referred to articles from popular digital databases such as Compendex, Springerlink, Association for Computing Machinery (ACM), IEEE Xplore, ScienceDirect, and Google Scholar. Non-peer-reviewed articles (such as white papers), abstracts, short papers (less than four pages long), editorials, and non-English manuscripts were excluded from the scope of this literature survey. Apart from this, we also did not consider a book, book chapter, paper extension, or dissertation if it had already been published in a journal or conference proceedings. For our search, we provided the following search string (shown in Table 2.2).

("Computational Intelligence")

AND

("Web service OR QoS-aware web service OR Web Service Composition (WSC) OR QoS-aware WSC")

AND

("Systematic Review OR Systematic Literature Review OR SLR OR Systematic Mapping OR Research synthesis")

Table 2.2: SLR search string

(3) ***Assessing the Quality of Studies:*** We refined our search terms following our research questions and motivations. For this survey, we extracted over 500 peer-reviewed research articles that were published during 2005 and 2016 in various conferences and journals via the following query:

("Web service OR Web service composition OR QoS-aware web service OR QoS-aware web service composition")

AND

("Heuristic search OR Meta-heuristic search OR Min-Max algorithm OR OA* search algorithm OR Tabu search OR Hill-climbing OR Constraint satisfaction OR Pruning algorithm OR Simulated annealing OR Genetic algorithm OR Genetic programming OR Cuckoo Search OR Bat algorithm OR A* search algorithm OR Ant colony optimization algorithm OR Harmony search OR Immune algorithm OR Particle swarm optimization OR Artificial Ant colony optimization algorithm OR Bee colony optimization OR ABC algorithm OR Win-Win Strategy OR firefly algorithm OR Grey wolf optimization OR bacterial colony optimization OR Gravitational search algorithm OR Glowworm optimization OR Ant lion algorithm")

The year 2005 was picked as we did not find no prior research article on QoS-aware service composition using CI techniques.

Initial selection: We extracted 568 articles that matched the research topic of CI-based QoS-aware web service composition across the digital databases. We explored the title, abstract, and key words of prospective primary studies and applied the above-mentioned criteria of inclusion/exclusion.

Final selection: For our final selection, we concentrated particularly on meta-heuristics approaches. So, we filtered the required articles collected in initial selection phase and finalized 90 articles for this review by using the following query.

(Title: ("WSC OR QoS-aware web service OR web service OR QoS-aware WCS OR WSC environment) OR Abstract: (WSC OR QoS-aware web service OR web service OR QoS-aware WSC OR WSC environment"))

OR

("key words: WSC OR keywords: QoS-aware web service OR keywords: QoS-aware WSC OR keywords: web service composition environment")

AND

("Title: (Meta-heuristic search OR OR Bee colony optimization OR Simulated annealing OR ABC algorithm OR Genetic programming OR Ant colony optimization algorithm OR Cuckoo search OR NSGA-II OR Harmony search OR Immune algorithm OR Bat algorithm OR bacterial colony optimization OR firefly algorithm OR Particle swarm optimization OR Artificial Ant colony optimization algorithm OR Grey wolf optimization OR Ant lion algorithm OR Gravitational search algorithm OR Glowworm optimization"))

OR

("key words: Meta-heuristic search OR key words: Ant colony optimization algorithm OR key words: Particle swarm optimization OR key words: Genetic programming OR key words: Artificial Ant colony optimization algorithm OR key words: Immune algorithm OR key words: firefly algorithm OR key words: Bee colony optimization key words: ABC algorithm OR key words: GRASP and Path-relinking algorithm OR key words: Tabu search OR key words: Simulated annealing OR key words: Cuckoo search OR key words: Harmony search OR key words: Grey wolf optimization OR key words: Bat algorithm OR key words: bacterial colony optimization OR key words: Gravitational search algorithm OR key words: Glowworm optimization OR key words: Ant lion algorithm OR key words: Fruit fly algorithm OR key words: NSGA-II")

The fourth step of this SLR *extraction and synthesization of data* is presented in Section 2.2 and fifth step *interpretation of findings* is discussed in Section 2.3.

2.2 Classification and Approaches in Web Service Composition

We classify QoS-aware web service composition into three classes: non-heuristic, heuristic, and meta-heuristic approaches. Figure 2.2 presents the classification of existing QoS-aware web service composition approaches.

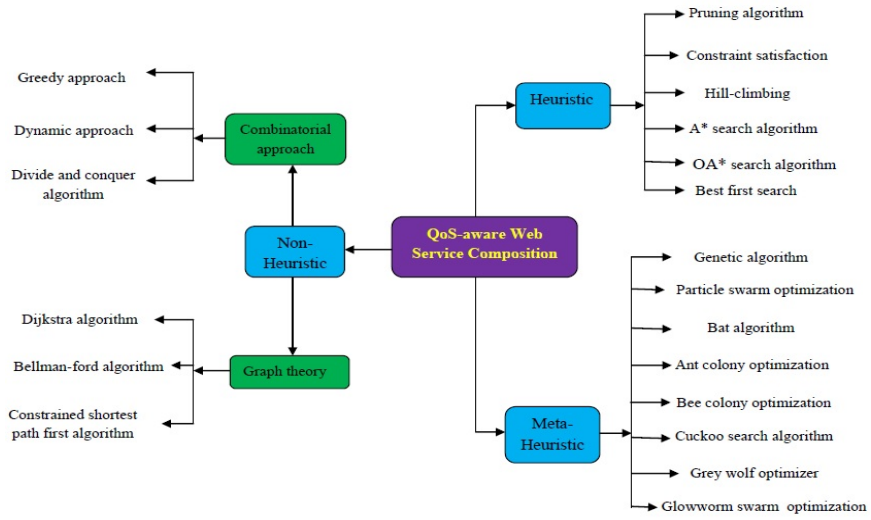


Figure 2.2: Classification of approaches based on QoS aware web service composition

2.2.1 Non-heuristic Algorithms

Non-heuristic algorithms solve optimization problems significantly faster than exhaustive search [50]. Zeng et al. [25, 26] used two optimization approaches for QoS-aware web service composition. They applied a local optimization approach for optimal web service selection for each task of a composite web service and a Linear Integer Programming (LIP) -based global optimization approach for optimal composition among all possible paths. Chattopadhyay et al. [59] proposed a new stochastic LIP approach for solving QoS-aware web service composition problems. In [60], a service selection scheme is proposed which optimizes the end to end aggregated QoS of all incoming flow of request by a simple LIP. Mohabey et al. [61] developed an integer programming model to solve QoS-aware web service composition. Ardagna et al. [5] proposed an adaptive service composition, maximizing end-to-end availability and maximizing QoS by choosing local maximization for each task during its implementation. Berbner et al. [62] developed H1 RELAX IP, H2 SWAP, and H3 ANNEAL approaches to find an optimal solution and improve the efficiency of QoS-aware

service composition. Alrifai et al. [63] addressed the search for the best decomposition of QoS constraints into native constraints using a hybrid methodology by applying Mixed Integer Programming to handle this problem. Gabrel et al. [64] discussed a model based on 0–1 linear programming for automatic transactional web service composition. Tao et al. [65] proposed an approach that maximizes the prenominal function and satisfies the global constraints by using a multi-dimensional and multi-choice knapsack model. Yu et al. [36] introduced a multi dimension multi choice 0-1 knapsack problem (MMKP) to solve QoS-ware web service selection problem. Yu et al. [66] proposed a model that considers multiple QoS constraints, and they used different workflows for different business processes. Jaeger et al. [67] evaluated different algorithms for performing the selection of candidates to optimize the overall QoS of a composition. Jaeger et al. [68] also presented an approach based on computational and resource values for finding the optimal solution for web service composition. Gao et al. [69] presented local-and-global-optimization-based service selection approaches. Huang et al. [70] developed a filtering algorithm for reducing the search space for computing the optimal QoS. Changlin et al. [71] developed an efficient divide-and-conquer algorithm for QoS-aware service composition based on a high-level conceptual approach. Liu et al. [72] discussed the convex hull approach and mathematical programming methods to find the optimal solution and employed multiple-criteria decision making to combine the multiple dimensional resources for local and global constraints in QoS-aware web service composition. Aiello et al. [73] developed a modified Breadth-First Search (BFS) algorithm to find the optimal service composition. Chen et al. [74] discussed Graphplan and graph search algorithms to meet the functional and non-functional requirements at the same time. In their method, the Graphplan algorithm is used to analyze graph reachability, and Dijkstra's algorithm is used to search the graph to solve the QoS-aware web service composition problem. In [75], the service composition problem is tackled through branch and bound for execution plan selection (BB4EPS). Gao et al. [76] proposed a weighted multistage graph model to find optimal service selection for service composition. Several have designed time-efficient algorithms for local and global constraints related to web service composition [56, 77, 78].

The list of non-heuristic algorithms with their specifications for solving QoS-aware web service composition is presented in Table 2.3.

Table 2.3: List of non-heuristic approaches and their specifications for QoS-aware web service composition

Reference	Optimization Mode		Support for QoS	Multi objective	Algorithm
	Local	Global	Constraints Specification	optimization	
Zeng et al. [25], Mohabey et al.[61], and Cardellini et al. [60]	✓	X	✓	✓	LIP
Zeng et al. [26]	X	✓	✓	✓	LIP
Chattopadhyay et al. [59]	✓	✓	✓	✓	Stochastic LIP
Ardagna et al. [5] and Alrifai et al. [63]	✓	X	✓	✓	MIP
Berbner et al. [62]	✓	X	✓	X	MIP heuristic
Aiello et al. [73]	✓	X	X	X	Breadth first search
Zhenqiu et al. [70] and Gao et al. [69]	X	✓	X	X	Mod. Dynamic Programming
Gao et al. [69]	X	✓	X	✓	Mod. Dynamic Programming
Yu et al. [66]	✓	X	✓	✓	MMKP & MCOP
Yu et al. [36]	X	✓	✓	✓	MMKP Algorithm
Jaeger et al. [68]	✓	X	✓	✓	Knapsack & Multi-Mode RCPSP
Jaeger et al.[67]	✓	X	✓	✓	Knapsack & Constraint Project scheduling problem (RCPSP)
Tao et al. [65]	✓	X	✓	X	MCKP & CSPP
Yang et al. [77]	X	✓	X	✓	MCOP Algorithm
Gabrel et al. [64]	✓	X	✓	✓	Dependency graph & 0-1 linear programming
Liu et al. [75]	✓	X	✓	✓	BB4EPS
Changlin et al. [71]	✓	X	✓	✓	Divide-and-Conquer
Yan et al. [78]	✓	X	✓	✓	Greedy Quick-hull
Liu et al. [72]	✓	X	✓	✓	Convex-hull
Chen et al. [74]	✓	X	✓	✓	Dijkstra Algorithm
Deng et al. [56]	✓	X	✓	✓	Correlation-aware service pruning method
Gao et al. [76]	✓	X	✓	✓	Weighted multistage graph

2.2.2 Heuristic Algorithms

Heuristic algorithms are the algorithms derived from experience with particular optimization problems. Usually, they find an optimal solution by a trial-and-error method in an adequate amount of time by taking full advantage of the particularities of the problem. The solutions by these algorithms may be better then an educated guess but may not be optimal [51]. Since non-heuristic algorithms take a massive amount of time to procure the optimal solution, heuristic algorithms are preferred because these algorithms can procure near-optimal solutions in an acceptable amount of time.

Luo et al. [79] introduced an improved heuristic algorithm for selection of service compositions with multiple constraints. Comes et al. [80] developed two

service selection algorithms to satisfy certain QoS requirements based on weight factors and priority factors. Luo et al. [81] presented a heuristic-enhanced cross-entropy algorithm for QoS-aware web service composition satisfying end-to-end QoS constraints. Pedro et al. [82] proposed a heuristic algorithm to solve the QoS-aware web service composition problem with the objective of minimizing search space and with realistic deadlines. Li et al. [83] proposed a global optimization selection (GoS) method to execute the local preprocessing service selection algorithm, and GoS is used to improve the runtime performance of the global service selection. Klein et al. [84] applied the hill-climbing approach and compared it with LIP to minimize the time complexity in finding a near-optimal solution. Qi et al. [85] proposed a local optimization method for finding the local candidate services and then combining them so that the optimal solution is obtained. Li et al. [86] presented a distributed heuristic approach to solving web service composition problems with a high approximation ratio, enabling adaptability and a near-optimal solution. Li et al. [87] introduced a win–win strategy approach by applying game theory in developing a mathematical model of service composition and a genetic algorithm for service composition with Pareto optimum. Li et al. [88] presented a reliable method for selection of the reliable services using a heuristic global optimization approach. Oh et al. [89] proposed a BF^*-based graph search algorithm to solve the QoS-aware web service composition problem. Niu et al. [90] proposed the heuristic graph search and the breadth heuristic uncertain composition approaches for solving uncertain web service composition problems to reduce the search space and for high efficiency in obtaining a service composition solution. Lin et al. [91] discussed a relaxable QoS-based service selection algorithm to attain the optimal solution. This method recommends prospective candidate services to users by relaxing the QoS constraints if no suitable web service can fulfill the user requirements exactly. Rodriguez-Mier [92] solved automatic web service composition through A* algorithm.

The list of heuristic approaches with their specifications for solving QoS aware web service composition is presented in Table 2.4.

Table 2.4: List of heuristic search algorithms and their specifications for QoS-aware web service composition (supporting multi-objective optimization and multiple QoS constraints)

	Optimization Mode		
Reference	**Global**	**Local**	**Algorithm**
Comes et al. [80], Luo et al. [81], Pedro et al. [82], Luo et al. [79]	✓	X	Heuristic search
Li et al. [83]	✓	✓	Heuristic algorithm
Li et al. [88]	✓	X	Global Heuristic algorithm
Klein et al. [84]	✓	X	Hill climbing
Qi et al. [85]	X	✓	Local optimization & enumeration method
Li et al. [87]	✓	X	Win-Win strategy algorithm
Lin et al. [91]	✓	X	Relaxable service selection algorithm
Li et al. [86]	✓	X	Distributed heuristic algorithm
Oh et al. [89]	✓	X	BF* algorithm
Rodriguez-mier et al. [92]	✓	X	A* algorithm
Niu et al. [90]	✓	X	Heuristic graph search

2.2.3 Meta-heuristic Algorithms

A meta-heuristic algorithm is a higher-level heuristic algorithm that is problem-independent and is applicable to a broad range of problems [51]. Some of the well known meta-heuristics are Simulated Annealing (SA), evolutionary algorithms (EAs), Particle Swarm Optimization (PSO) including Ant Colony Optimization (ACO), the Bat Algorithm (BA), the Firefly Algorithm (FA), the Genetic Algorithm (GA), Tabu Search (TS), and Harmony Search (HS). There are two types of key components in meta-heuristics: intensification and diversification. A balance between intensification and diversification is essential for an effective and efficient meta-heuristic algorithm [93]. This type of algorithm searches the entire solution space, and the search needs to be intensified around the neighborhood of the optimal or near-optimal solutions.

Canfora et al. [33] first applied GA for web service composition. Jaeger et al. [94] also applied GA for web service selection. Zhang et al. [95] developed a relation matrix encoding scheme based on improved GA to solve the QoS-aware

web service composition problem. Tang et al. [96] proposed a hybrid GA to solve the optimal constraint service selection problem. Chunming et al. [97] developed a tree-coded GA to solve the QoS-aware web service composition problem; the authors proved that the tree encoding scheme is more efficient than the relation matrix encoding scheme used in GA. Yu et al. [98] designed a tree-based GA to solve the QoS-aware WSC problem. Wang et al. [99] developed a chromosome coding approach for representing feasible solutions and designed genetic operators and their strategies to overcome the diversity of populations as well as the problem of getting trapped in local optima. Liu et al. [100] developed an improved GA algorithm that is based on ACO and the GA. This approach uses the learning advantages of both algorithms to select the initial population antibodies for better efficiency. Ma et al. [101] developed a quickly converging GA based on population diversity and a relation matrix for web service selection. Yilmaz et al. [102] proposed improved GA-based approaches, such as GA with SA and GA with Harmony Search, for QoS-aware web service composition. Li et al. [103] presented a multi-population GA to solve the QoS-aware web service composition problem. Xiangbing et al. [104] proposed Web Service Modeling Ontology (WSMO)–based web service composition using GA. In [105], the web service composition problem tackled through variable length chromosome GA. In this proposed method, variable length chromosome represents composite services plan in different paths and conducts gene crossover operation using service parameter matching. Wada et al. [40] proposed a novel approach E^3 multi-objective GA to address the web service composition. A number of authors have used the GAs [29, 40, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115] to reduce the search time. Although GA is widely used in web service composition, GA is more complicated than other approaches, and its computing time is very high.

Ma et al. [116] introduced a hybrid method that combines the use of Genetic Programming (GP) and random greedy search to solve the QoS-aware web service composition problem. In their proposed method, the greedy algorithm is used to produce the locally optimized valid initial population, and GP is used for mutation operators. Alexandre et al. [117] proposed a GP-based algorithm

for distributed QoS-aware web service composition in which multiple QoS constraints are considered. Rodriguez-mier et al. [118] developed an efficient GP method using a context-free grammar for generating the valid structures of the composite services and solved the composition of web services with the minimum execution path. Some researchers have proposed improved GPs to solve the QoS-aware web service composition problem [119, 120, 121, 122]. Bao et al. [123] proposed an orthogonal GA for QoS-aware web service composition. In their proposed method, the initial population is designed based on the orthogonal design used in the population generation process and on the crossover operator. Sharifara et al. [124] and Yao et al. [39] proposed Non-dominated Sorting Genetic Algorithm II (NSGA-II) to solve the QoS-aware web service composition problem. Li et al. [37] proposed a non-dominated GA to solve the QoS-aware web service composition problem. However, the said approaches still have some intrinsic defects, such as a low premature convergence rate and an increase of search space within local optima due to an exponential increase in Pareto front size, which in turn is due to the huge number of services in the service composition.

Another meta-heuristic approach for web service composition is Particle Swarm Optimization (PSO). Many researchers [125, 126] have adopted PSO for web service composition. Liu et al. [127] proposed a hybrid Genetic PSO to solve the QoS-aware web service composition problem. Liu et al. [128] introduced discrete PSO and Color Petri Nets (CPN) for solving web service composition problems. Wang et al. [129] developed a chaos-based PSO method supporting global constraints to solve the QoS-aware web service composition problem. Liao et al. [130] presented a niching PSO satisfying multiple global constraints and load balancing for web service composition. Liu et al. [131] developed a hybrid quantum PSO algorithm for solving the combinatorial optimization problem for web service composition. Wenbin et al. [125] proposed an improved PSO to solve the QoS-aware web service composition problem. They used a non-uniform mutation strategy for a global best particle to improve the population diversity, and adaptive weight adjustment and local best-first strategies to enhance convergence speed. Zhao et al. [132] designed an improved discrete Im-

mune optimization method based on PSO that combines Proportional Clone and Immune optimization algorithms for QoS-aware web service composition. Jiuxin et al. [133] proposed an efficient multi-objective service selection approach to solve service selection problem based on PSO. Jun et al. [134] introduced an environment-aware PSO algorithm to address service composition by simulating the bird flocking environment ware behaviour in seeking food process. Ludwig et al. [135] proposed a PSO approach to solve QoS-driven web service composition considering multiple workflow requests. Rezaie et al. [136] presented web service composition model based on a discrete multi-objective PSO. Some researchers have proposed hybrid PSO [137, 138, 139, 140, 141, 142] algorithms for finding a near-optimal solution for QoS-aware web service composition. However, PSO methods can often be trapped in local optima and often are not good at diversification.

Some researchers introduced Tabu search for finding the optimal QoS-aware web service composition [143, 144]. However, Tabu search has some limitations, such as the longer execution time, the high number of parameters that need to be evaluated, and the fact that the global optimum may not be obtained. In [145, 146, 147], the authors presented harmony search algorithms for finding a near-optimal solution using local and global constraints. Yan et al. [148] proposed a graph-based memetic algorithm to address QoS-aware web service composition.

Some researchers have adopted Immune algorithms [44, 149, 150, 151, 152] for finding near-optimal solutions for QoS-aware web service composition problems. Another efficient meta-heuristic technique for QoS-aware web service composition is ant colony optimization (ACO) [153, 154, 155]. For solving web service composition problems using ACO, artificial ants travel in a graph to search for optimal paths according to pheromone and fitness values, and the QoS constraints denote the weights of the edges of the graph. Li et al. [156] modeled QoS global optimization in web service selection using Multi-objective Chaos ACO and satisfying the user requirements. Yang et al. [157] proposed a hybrid method (ACO and GA) to solve the QoS-aware service composition. Mao et al. [158] analyzed the effect of different algorithms (PSO, GA, Estimation of Distribution Algorithm (EDA)) for web service composition and presented guidelines

of suitability of these algorithms for various web service selection and composition problems. Pop et al. [45] described a hybrid method (ACO and the Graph model) that was able to improve the accuracy and efficiency of web service composition. Wang et al. [159] proposed an adaptive ACO approach to solving web service composition based on variable trust degree and quality of service. Shanshan et al. [160] proposed an improved ACO algorithm adjusting transition probability and pheromone update strategy for QoS-aware web service composition. Wang et al. [161] proposed a culture max-min ant system for QoS aware web service composition. Fan et al. [162] developed a co-evolution approach (SPSO and SA) to address the web service selection problem. Rosenberg et al. [163] proposed a novel simulated annealing method, to solve the QoS-aware web service composition problem. However, although the robustness of this algorithm is high, it spends a long time in the initialization phase and stagnates to local optima.

Podili et al. [164] proposed a hybrid bat algorithm for QoS-aware web service selection. Zhou et al. [165] adopted a Differential Evolution (DE) algorithm for QoS-aware web service composition. Cremene et al. [41] presented a multi-objective generalized DE to solve the QoS-aware web service composition problem. However, in DE, for each iteration the new population is generated based on a mutation-based distribution of the solutions. If either the population is increased or the number of solutions computed for the mutation values is increased, the diversity of possible movements will also increase, promoting exploration of the search space. This algorithm reaches an unstable convergence in the last period; it is easy for it to drop into local optima.

Table 2.5: List of meta-heuristic algorithms for QoS-aware web service composition (supporting global optimization and multiple QoS constraints)

Approach	Algorithm
Canfora et al. [33, 106, 107], Ai et al. [108], Amiri et al. [109]	
Tang et al. [96], Zhang et al. [95, 114], Wang et al. [99], Mardukhi et al. [29]	
Liang et al. [110], Xiangbing et al. [104], Liu et al. [100], Ma et al. [101],	
Yilmaz et al. [102], Chen et al. [111], Tan et al. [112], and Ye et al. [113]	GA
Bao et al. [123]	Orthogonal GA
Wada et al. [40]	E^3-MOGA
Sharifara et al. [124], Yao et al. [39], and Li et al. [37]	NSGA-II
Jiang et al. [105]	Variable length chromosome GA
Li et al. [103]	Multi-population GA
Ai et al. [115]	GA and Hill climbing
Bahadori et al. [143] and Parejo et al. [144]	Hybrid GA and Tabu search
Chunming et al. [97]	Tree coded GA
Yu et al. [98], Wang et al. [119], Rodriguez-mier et al. [118], Alexandre et al. [117],	
Yang et al. [121], and Wang et al. [120]	GP
Yu et al. [122]	Hybrid GP
Ma et al. [116]	GP and Greedy search
Jafarpour et al. [145, 147] and Mohammed et al. [146]	Harmony search
Yan et al. [148]	Memetic algorithm
Jiuxin et al. [133], Jun et al. [134], Ludwig et al. [135],	
Amiri et al. [126], Xia et al. [137], and Zhang [140]	PSO
Laio et al. [138] and Rezaie et al. [136]	Multi-objective PSO
Yin et al. [142]	Hybrid Multi-objective discrete PSO
Liao et al. [139]	Accurate sub swarm PSO
Alexandre et al. [141]	Graph based PSO
Wenbin et al. [125]	Improved PSO
Liu et al. [127]	Hybrid GA and PSO
Liao et al. [130]	Niching PSO
Liu et al. [131]	HQPSO
Liu et al. [128]	Discrete PSO
Wang et al. [129]	Chaos PSO
Zhao et al. [132]	Improved discrete PSO

Table 2.6: List of meta-heuristic algorithms for QoS-aware web service composition (Continued from Table 2.5)

Approach	Algorithm
Li et al. [156], Zhang et al. [155], Shanshan et al. [160], Pop et al. [45], and Xia et al. [154]	ACO
Wang et al. [159]	Adaptive ACO
Wang et al. [161]	Culture minimax ant system (C-MMAS)
Yang et al. [157]	Hybrid ACO
Yunwu et al. [153]	Multi-objective chaos based ACO
Yan et al. [150], Xu et al. [149], Pop et al. [44], Zhao et al. [151], and Pop et al. [152]	Immune algorithm
Rosenberg et al. [163]	Simulated annealing
Podili et al. [164]	Hybrid bat algorithm
Zhou et al. [165]	DE
Cremene et al. [41]	Multi-objective DE
Wang et al. [55] and Wang et al. [166]	ABC
He at el. [167]	Improved ABC
Kousalya et al. [168] and Chifu et al. [169]	BCO
Fan et al. [162]	Co-evolution Algorithm
Parejo et al. [19]	GRASP and Path Relinking
Pop et al. [170]	Cuckoo-inspired search
Boussalia et al. [171]	Quantum inspired cuckoo search
De campos et al. [172] and Li et al. [38]	MOEA
Zhang et al. [173]	Improved Fruit Fly Optimization Algorithm
Mao et al. [158]	EDA

Chifu et al. [169] and Kousalya et al. [168] applied a Bee Colony Optimization (BCO) algorithms to solve the web service selection problem. Wang et al. [55] and Wang et al. [166] designed novel Artificial Bee Colony (ABC) algorithms to address QoS-aware web service selection. He et al. [167] applied an ABC algorithm to address QoS-aware service selection. Pop et al. [170] proposed an improved CS to solve the QoS-aware web service composition problem. Boussalia et al. [171] developed a novel quantum-inspired cuckoo search algorithm to address QoS-aware web service composition. De Campos et al. [172] and

Li et al. [38] proposed many objective evolutionary algorithms for web service composition. Parejo et al. [19] developed a hybrid of the Greedy Randomized Adaptive Search Procedure (GRASP) and the Path Relinking algorithm to solve the QoS-aware web service composition problem. The main disadvantage of GRASP is that it cannot integrate good solutions found previously into the current search iteration. As each iteration creates a completely different solution, there is no information added to the system when a good solution is found. The main disadvantage of path relinking is that it requires more execution time than other algorithms [174]. In [173], the service composition problem tackled through improved fruit fly optimization algorithm. The list of the said meta-heuristic approaches and their specifications is presented in Tables 2.5 and 2.6.

2.3 Analysis and Discussion of SLR Results

This section presents an analysis of state-of-the-art research of QoS-aware service composition addressing the research questions given in Table 2.1.

There are three major approaches in QoS-aware web service composition (see Fig. 2.3). It can be easily observed that 19% of articles focus on non-heuristic algorithms, 11% of articles focus on heuristic algorithms, and 70% of articles focus on meta-heuristic algorithms. As witnessed by the literature, we observe that meta-heuristic algorithms support large workflow sizes with global constraints and obtain near-optimal execution plans in less time. Thus, meta-heuristic methods appear as a preferred solution for QoS-aware service composition. Based on Fig. 2.3, we observe that most of the research on web service composition has been done on meta-heuristic approaches and that these meta-heuristic approaches are still receiving considerable attention from the research community.

Fig. 2.4 shows the percentage-wise distribution of selected research articles on each approach in the meta-heuristics category. From Fig. 2.4, it is clearly observed that GA (29%) has played a key role in solving the problem as it is adopted by many researchers. Further, PSO (20%), GP (9%), ACO (10%), the

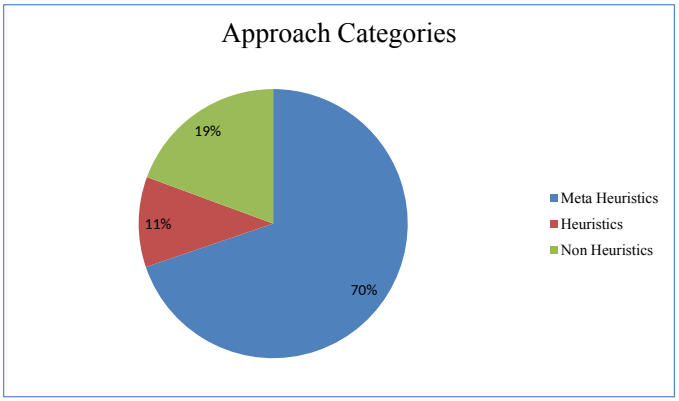


Figure 2.3: Importance of approaches in the literature

Immune algorithm (6%), MOEA (6%), and other approaches (6%) have been adopted by researchers. However, there is room for new algorithms and for improvement in existing algorithms.

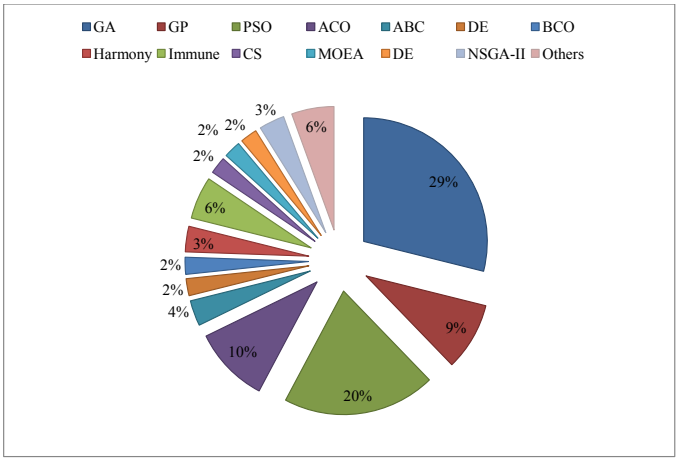


Figure 2.4: Importance of each approach and their percentage

Most of the researchers concentrated on the following QoS parameters in web service composition: availability, reputation, throughput, reliability, response time, cost, and security. The list of QoS parameters and their descriptions are shown in chapter 1 (see Table 1.1). The list of frequencies of QoS parameters and their percentages are shown in Fig.2.5, respectively.

Based on our SLR, we observe that most of the articles in CI-based QoS-aware

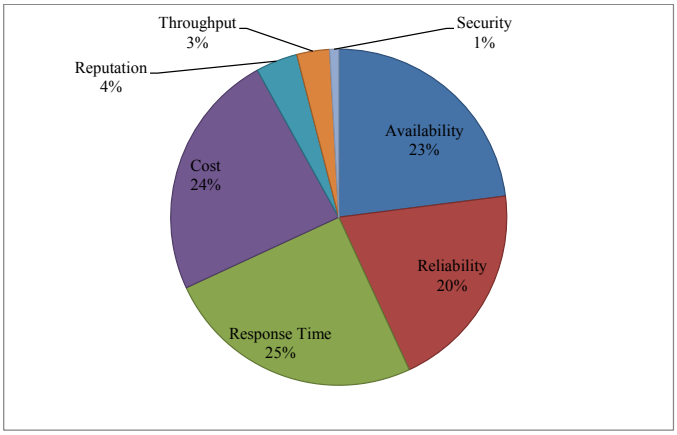


Figure 2.5: Percentage of QoS parameters in literature

web service composition are published in the proceedings of IEEE International conference on web services (ICWS), IEEE International conference on service-oriented computing (ICSOC), IEEE International conference on World Wide Web (WWW), IEEE Congress on Evolutionary Computation (CEC), IEEE International Conference on Services Computing (SCC), International Symposium on Telecommunications (IST), IEEE International Conference on Intelligent computer communication and processing (ICCP), and other major conferences. Among the 90 articles, 26 were published in major journals including *IEEE Transactions on Services Computing*, *Expert Systems with Applications*, *Applied Soft Computing*, *Future Generation Computer Systems*, *Service Oriented Computing and Applications*, *International Journal of Computational Intelligence Systems*, and *Evolutionary Intelligence*.

Threats to Completeness The main threat to the validity of this systematic literature review is in construction and evaluation of search string. Reviewers construct a search string in such a way that she could collect all relevant articles without spending much effort and time. For this SLR, we searched six digital databases and constructed the search string in such a way that it can include all the articles that are anyhow related to Computational intelligence and Web services. Further, we refined our initial selection by using the inclusion and exclusion criteria mentioned in Section 2. The search string for initial selection was

flexible enough to allow all the relevant articles, however, some articles might be missing due to linguistic barriers.

Threats to the Data Extraction We searched six digital databases and collected over 500 relevant articles in our initial selection. Thereafter, 90 articles were finally selected. We validated our search results by cross-checking individual journals and proceedings respectively.

2.4 Justification for the Present Work

This chapter's contribution has been to reveal the state-of-the-art research on web service composition using computational intelligence techniques. Based on the systematic literature review of the specified theme, we understand the scope and need for better QoS-aware web service composition algorithms. Based on this SLR, we classified the existing approaches into three classes: non-heuristics, heuristics, and meta-heuristics. From our observations, we conclude the following: (i) Web service composition problems are generally solved by meta-heuristic algorithms, and these algorithms are still receiving considerable attention from the research communities. Researchers continue to propose new algorithms and improvements to existing algorithms. (ii) Researchers do not consider the *balancing of QoS parameters* or the *connectivity constraints* between two compositions, and because of this, it is not possible to model the scalability of a service composition. (iii) These algorithms still have some intrinsic defects, including low premature convergence rate, low diversity, and increasing search space.

Nowadays, it becomes necessary and interesting to explore and analyze the seamless proliferation of cloud services and Big services, and the selection and composition of services in the cloud and Big services environment. This SLR does not aim to explore the state-of-the-art of selection and composition of cloud services and big services. As cloud services and big services are the popular trends in service-oriented computing, in this thesis, we address the selection and

composition of cloud services and big services. Several researchers have proposed various approaches for evaluating the cloud services and their ranking based on various QoS attributes. However, these methods do not consider relative performance, user preferences, or essential QoS attributes (See Section 4.4 for the related works on selection of cloud services).

We also notice that big service composition methods seem very impressive, motivating us to go forward in this direction of research. Computational intelligence algorithms may require more iterations and more computation time to find the optimal or near-optimal solutions for big service composition. Further, these methods have some intrinsic defects, including low premature convergence rate and an increase of search space within local optima (See Section 5.4 for the related works on big service composition).

Chapter 3

QoS-aware Web Service Composition

Service composition aims to fulfill the functional demands of the business rationale. The resulting composite service is characterized by QoS parameters, which help users assess the quality of the service composition. For example, cost and response time are two QoS parameters among many that can influence user satisfaction. Applications comprising multiple tasks of various natures require a composition strategy of web services based on QoS parameters and an execution priority based on the context of the composition [5]. As web services that are offered have different QoS levels for a specific task, a QoS-based ranking of these services helps the user in selecting services [175]. The user ranks the services according to the various QoS parameters such as service reliability over cost and cost minimization over other QoS parameters. A service can be ranked as the best service under a particular QoS parameter but may be ranked as the worst under another QoS parameter. The performance of a service can be ranked differently according to various QoS parameters. For instance, a service that is ranked best under a QoS parameter, reliability, for instance, can be given a low ranking under another QoS parameter, such as response time. Hence, the priorities of QoS parameters vary from one composition context to another. However, all QoS parameters should be equally considered (called the balancing of QoS parameters) in the selection of a service without neglecting the influence of a

primary QoS parameter in a service composition. Otherwise, the optimal service under a QoS parameter will fail if that service does not retain the optimality under another QoS parameter. However, most of the approaches in the literature [19, 33, 97, 102, 109, 123, 132, 176] consider either a single QoS parameter or two QoS parameters and do not consider the *balancing of QoS parameters* and/ or satisfying the *connectivity constraints* between two compositions, and because of this, it is not possible to model the scalability of the service composition. Furthermore, these approaches still have some intrinsic defects, including low premature convergence rate, low diversity, and increasing search space. These facts motivated us to develop optimal fitness-aware service composition strategies.

3.1 Illustrative Scenario

Let us consider a goods ordering process (see Fig. 3.1) to demonstrate the issues of QoS-aware service composition [53, 177]. This scenario demonstrates a composition of seven candidate web services with different providers, and each task has two services as possible solutions (see Table 3.1).

Table 3.1: Service providers, cost and execution time of the respective services for the illustrative scenario

Provider	Tasks	Services	Respective Costs (in cents)	Respective response time (in sec)
A	t_1, t_2	${s1}_A, {s2}_A$	1, 2	0.2, 0.2
B	t_1, t_2	${s1}_B, {s2}_B$	1.5, 5	0.1, 0.15
C	t_3, t_4	${s3}_C, {s4}_C$	1, 2	0.2, 0.2
D	t_3, t_4	${s3}_D, {s4}_D$	1, 5	0.4, 0.25
E	t_5	${s5}_E$	1	0.2
F	t_5	${s5}_F$	2	0.2
G	t_6	${s6}_G$	1	0.2
H	t_6	${s6}_H$	2	0.2
I	t_7	${s7}_I$	1.5	0.1
J	t_7	${s7}_J$	5	0.15

A customer initially registers the details of the goods and initiates the payment transaction through her credit card. Once the card is verified (task t_1), the pay-

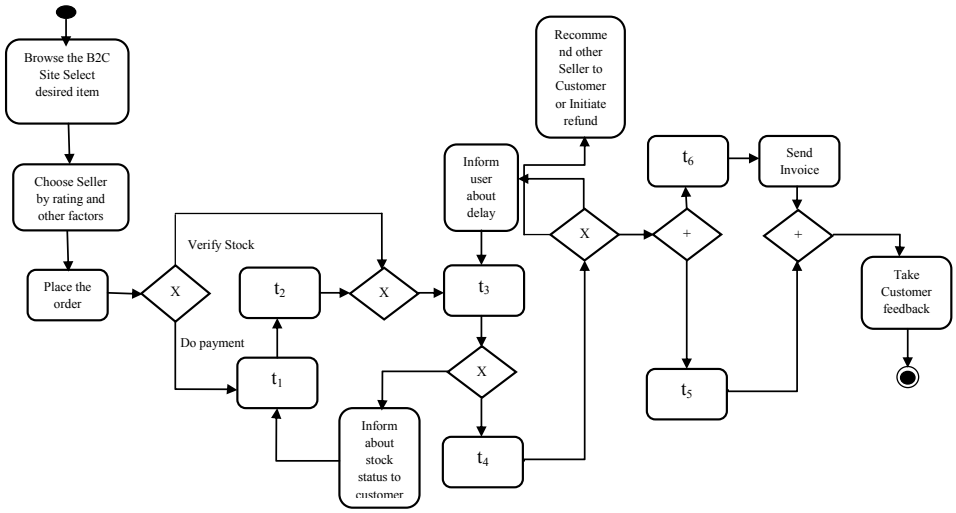


Figure 3.1: Exploring a composite goods ordering service

ment transaction (task t_2) is successfully completed, and the stock checking (task t_3) follows, reserving the registered goods for pickup (task t_4). If stock is not available, then the customer is notified of the delay in the delivery of the goods, and tasks t_3 and t_4 of the composition wait for a certain period and are then repeated. This wait-and-repeat continues until the maximum elapsed time given to the seller. If she fails to deliver within the given time interval, then the customer should be notified of this and may also receive a recommendation with another seller's details if any. If no other seller is ready to dispatch that item or the customer is no longer interested, then the amount of the payment should be refunded.

Task t_1 is accomplished by services s_{1_A} and s_{1_B}, and t_2 is accomplished by services s_{2_A} and s_{2_B}. s_{1_A} and s_{2_A} are owned by a provider A, and s_{1_B} and s_{2_B} by a provider B. A service composition to fulfill tasks t_1 and t_2 can be done using the respective services provided by A or B. If cost of the service is the constraint (user-provided) for the composition, then the choice of services provided by A is optimal, since these services are cheaper than the services provided by B. If response time is the composition constraint, then the selection of services provided by B is to be preferred over the services provided by A. If the constraint for the composition is a combination of both cost and response time, then the selection

of services for the composition is a more challenging task.

In the case of task t_4, the service should be invoked if the ordered good is in stock (verified by task t_3); otherwise, it waits until stock is available. The service that opted to accomplish task t_4 may wait and repeat based on the response of the service that opted for task t_3. Hence, the recommendation will be to choose services for t_3 and t_4 from the same provider.

Another composition context of the goods ordering process example depicted is the accomplishing of two tasks in parallel. Once goods are available in stock, tasks t_5 (pickup and delivery) and t_6 (sending a digitally signed invoice to the client via an email) can be performed in parallel. The failure of the service related to t_5 cannot be tolerated against successful completion of the service related to t_6. Upon the successful completion of the services related to t_5 and t_6, the service in the schedule to accomplish task t_7 (requesting feedback from the customer) is initiated. Further, other non-functional constraints, such as the maximum time allowed for process completion or the maximum time allowed for the completion of the composition, would also play a vital role.

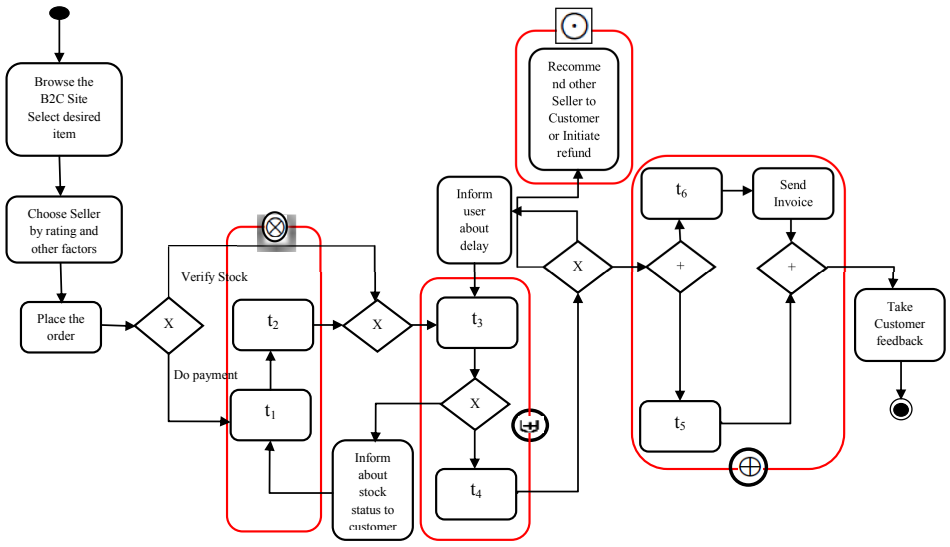


Figure 3.2: Exploring connectivity constraints in a composite goods ordering service

The illustrated scenario elucidates the following *connectivity constraints* between two compositions: (i) a service is dependent on another service (depen-

dency constraint), (ii) more than one service is to be executed in parallel (parallel constraint), (iii) the same task may be reinitiated by another service (rollbacking), and (iv) there may be a dependency upon more than one service to complete a task (sequence constraint). In the example depicted, tasks t_3 and t_4 represent the dependency constraint ($\uplus$ in Fig. 3.2) of the connection between relevant services, and tasks t_5 and t_6 reflect the parallelism constraint of the service composition ($\oplus$ in Fig. 3.1). On the failure of the seller to dispatch the selected item by the expected time, the process will rollback according to the customer's choice (rollback constraint, $\odot$ in Fig. 3.1). Tasks t_1 and t_2 reflect the sequence constraint of the connection between the respective services ($\otimes$ in Fig. 3.1). These constraints play a key role in real-time practice. If the respective service for t_3 observes that stock is not available, then the connected service for t_4 should wait until the service for t_3 confirms the availability of stock. Hence, the process between these two services is repetitive and dependent on both. In such cases, if both are from the same provider, or if they are not from the same provider but the providers at least have some official association, then the dependent constraint can be relaxed. If neither of these is the case, then the constraint remains intact, which degrades the QoS of the composition. A connectivity constraint refers to the connection between any two constraints that depend on each other. For example, t_1 and t_2 are performed together, and t_3 and t_4 have dependencies on each other. To complete t_1 and t_2, verification of t_3 and t_4 is necessary.

A service can be ranked as the best service under a particular QoS parameter but may be ranked as the worst under other QoS parameters. The performance of a service can be ranked differently according to various QoS parameters. For instance, a service that is ranked best under a QoS parameter, reliability, for instance, can be given a low ranking under another QoS parameter, such as response time. Hence, the priorities of QoS parameters vary from one composition context to another. However, all QoS parameters should be equally considered (called the *balancing of QoS parameters*) in the selection of a service without neglecting the influence of a primary QoS parameter in a service composition. Otherwise, the optimal service under a QoS parameter will fail if that service does not retain the optimality under another QoS parameter. Consider the illus-

trated scenario of a goods ordering process in which two different services that fulfill tasks t_1 and t_2 of the payment process (a composite service) can be selected from among possible services for each task. In selecting a service for task t_1 (considering the primary QoS parameter to be cost), the service $s_{1\,A}$ is chosen because it is lower in cost although it is poor in response time. For the same task t_1, if response time is considered the primary QoS parameter, then $s_{1\,B}$ is chosen, which has a lower response time although a higher cost. Hence, including any of these services as candidate services in the composition affects the performance of the overall composition, as these services exhibit poor performance under QoS parameters other than the primary QoS parameter considered. In order to resolve such QoS conflicts, a balancing of the QoS parameters for the services using a skewness-based approach is proposed, by which services are selected based on their ranking under the balanced QoS parameter.

3.2 Optimal Fitness Aware Web Service composition using an Adaptive Genotypes Evolution based Genetic Algorithm (OFASC-AGEGA)

3.2.1 Modelling QoS-aware Web Service Composition

A web service composition task $CT = \{T_1, T_2, T_3, \ldots, T_n\}$ consists of n tasks (or abstract services) such that each task $T_i \in CT$ accommodates a set of services, each of which is a subset of the available candidate services $CS = \{ST_1 = \{s_{1\,1}, s_{1\,2}, \ldots, s_{1\,i}\}, ST_2 = \{s_{2\,1}, s_{2\,2}, \ldots, s_{2\,j}\}, \ldots, ST_p = \{s_{p\,1}, s_{p\,2}, \ldots, s_{p\,m}\}\}$, where i, j, and m denote the number of candidate services in ST_1, ST_2, and ST_p, respectively.

The services in a set $ST_i = \{s_{i\,1}, s_{i\,2}, \ldots, s_{i\,x}\}$ constitute the collection of the x services that can address the task T_i. The candidate services in the subset ST_i have similar functionalities but can differ in their QoS parameters. Let a set of QoS parameters be $P = \{p_1, p_2, \ldots, p_{|q|}\}$, where $|q|$ denotes the number of QoS

parameters. Further, each QoS parameter in P is associated with each service in the given service set ST_i.

Let $E^{'} = \{E_1, E_2, \ldots, E_m \forall\ [E_x : t_y \rightarrow t_{y+1}]\}$ be the set of edges connecting two tasks in each possible composition sequence. Let $ST^{'}$ = $\{\{ST_i, ST_j, ST_k, \ldots\}, \{ST_x, ST_y, ST_z, \ldots\}, \ldots\}$ be the collection of sets of tasks such that the connections between each task set is affected by any of the connectivity constraints (dependency ($\uplus$), parallelism ($\bigoplus$), rollback ($\bigodot$), and sequence ($\bigotimes$)). Each task set group $ST^{'}$ is expecting associability with another task set. In our proposed method, associability is defined by the following conditions: (i) the services utilized for these tasks must be from the same service provider or (ii) the different providers have a service level agreement.

Adaptive Genotype Evolution using Genetic Algorithm Adaptive genotype evolution is a progressive evolution strategy [178] in which the evolution iterations are restricted to a minimal number. A progressive evolution strategy considers the new generation's offspring that have better fitness than any of their parents. The connectivity of QoS constraints and the balancing of QoS parameters for web service composition motivated us to define a genetic algorithm approach based on adaptive genotype evolution that discovers a set of best-fit QoS-aware service compositions from among all possible compositions. It finds the fitness of the resultant compositions of the GA evolution iterations as described in sections 3.2.1.1, 3.2.1.2, and 3.2.1.3. Further compositions are ranked according to their composition fitness value cfv, and the top n compositions are selected from the sorted list. These selected compositions are sorted in descending order of their associability fitness value afv and are used in the same sort to finalize services for further evolution or to recommend best compositions after completion of the evolution.

3.2.1.1 Evaluating discrete uniform rank distribution as service fitness

Let P be a set of QoS parameters $\{p_1, p_2, \ldots, p_q\}$ of each service in the given service set $ST_i = \{s_{i\,1}, \ldots, s_{i\,x}\}$. A QoS parameter P_{opt} is said to be the anchor

parameter for ranking the services. The QoS parameters of a service are bifurcated as positive parameters and negative parameters. A positive parameter uses higher values to represent higher utility (i.e., availability and reliability); a negative parameter users higher values to denote lower utility (i.e., cost and response time) [53]. Hence, we calculate the normalized values of positive and negative QoS parameters as the range of their values is enormous. Positive and negative parameter values are normalized according to the following equation:

$$m_k = \begin{cases} \dfrac{p_k - \min_k p_k}{\max_k p_k - \min_k p_k} & \text{for positive parameter} \\ \dfrac{\max_k p_k - p_k}{\max_k p_k - \min_k p_k} & \text{for negative parameter} \end{cases} \tag{3.1}$$

Further, the services related to a specific task are ranked in descending order of the values of these QoS parameters. Each service is ranked in different positions under different QoS parameters, which are used for measuring the discrete uniform rank distribution (DURD). For each service ST_i of task T, the discrete uniform distribution scope [179] of the QoS parameter ranks can be measured using the following three factors:

- Mean $\mu R(Ts_i)$ of QoS parameter ranks for the service.
- Variance $\upsilon QV(Ts_i)$ of correlations between discrete parts of the QoS parameter rank set for service s_i of task T. The number of discrete partitions of the rank set is q.
- Deviation $\sigma QV(Ts_i)$ of correlations between discrete parts of the QoS parameter rank set for service s_i of a task T.

Measurement of mean of QoS parameter ranks for a service Let a rank set for a service $[p_j \in ST_i \wedge ST_i \in CS]$ be $r(s_i) = [r(p_1), r(p_2), \cdots, r(p_n)]$. The service fitness value (sfv) of this service s_i is explored in the following step. The mean $\mu R(ts_i)$ of the ranks of all QoS parameters P is measured as follows:

$$\mu R(Ts_i) = \left[\frac{\sum_{i=1}^{|P|} r(p_j \in P)}{|P|} \right] \tag{3.2}$$

Measurement of correlations between discrete partition of the QoS parameter rank set We partition the rank set $r(s_i)$ into q partitions, which can be referred to as $Q1_{r(s_i)}, Q2_{r(s_i)}, Q3_{r(s_i)}, \ldots, Qq_{r(s_i)}$. Then, we find the correlation of each partition with all other partitions and the covariance of these resultant correlations as follows:

Algorithm 3.1 Correlations for each partition

```
1: procedure CORRELATION(Q_j)
2:     υ(Q_{s_i}) ← φ;                                   ▷ initializing a vector.
3:     for each k: 1, 2, ..., q do
4:         ρ(Q_k) ← φ ;                                  ▷ initializing a vector.
5:         for each j: 1, 2, ..., q do
6:             ρ(Q_k) ← cor(Qk_{r(s_i)}, Qj_{r(s_i)}) ;
7:         end for
8:         υ(Q_{s_i}) ← cov(ρQ_k) ; ▷ adding covariance of the values of vector ρ(Q_k) to
   υ(Q_{s_i}).
9:     end for
10: end procedure
```

In Algorithm 3.1, $\rho(Q_k)$ is a vector of size q, and each entry of this vector is the correlation between partition $Qk_{r(s_i)}$ and one of the partitions $Qj_{r(s_i)}$. $\upsilon(Q_{s_i})$ is a vector of size q, and each entry of this vector is the covariance of the vector $\rho(Q_k)$ entries (correlations observed between a part $Qk_{r(s_i)}$ and all parts of $r(s_i)$).

Further, we find the mean of all entries in the vector $\upsilon(Q_{Ts_i})$ for each service s_i of a specific task T as follows:

$$\upsilon QV(Ts_i) = \frac{\sum_{j=1}^{|\upsilon(Q_{Ts_i})|} \upsilon(Q_{Ts_i})_j}{|\upsilon(Q_{Ts_i})|} \tag{3.3}$$

where $\upsilon(Q_{Ts_i})$ is a vector of size q and each entry of this vector is the covariance of $\rho(Q_k)$. $\rho(Q_k)$ is a vector of correlations observed between a part $Qk_{rs(Ts_i)}$ and all parts of the $rs(Ts_i)$ entries.

Measurement of deviation of correlations between discrete partitions of the QoS parameters rank set We find the deviation $\sigma QV(Ts_i)$ of service s_i of a task T as follows:

$$\sigma QV(Ts_i) = \sqrt{\frac{\sum_{k=1}^{|\upsilon(Q_{Ts_i})|}((\mu(\upsilon(Q_{Ts_i})) - \upsilon(Q_{Ts_i}))_k)^2}{|\upsilon(Q_{Ts_i})|}} \tag{3.4}$$

where $\upsilon(Q_{Ts_i})$ is a vector of size q, each entry of which is the covariance of the vector $\rho(Q_k)$ entries (the correlations observed between a part $Qk_{rs_{Ts_i}}$ and all parts of rs_{Ts_i}), and $\mu QV(Ts_i)$ is the mean of the vector $\upsilon(Q_{Ts_i})$. The pseudocode representation of DURD is shown as Algorithm 3.2.

Algorithm 3.2 Discrete Uniform Rank Distribution (DURD)

procedure DURD(T_i)
Ranks of service: a set that contains ranks for each QoS parameter of a service.
Steps applied to all possible services to a task:
  for each task $(T_k, \forall k \in 0, 1, 2, \ldots, m)$ **do**
    for each service $(s_i, \forall i \in 0, 1, 2, \ldots, n)$ **do**
      $\mu R(s_i) \leftarrow \frac{\sum_{j=1}^{|r(s_i)|}(r(p_j) \in r(s_i))}{|r(s_i)|}$; ▷ Mean of Ranks
      $\mu R_{T_k} \leftarrow \mu R(s_i)$; ▷ Move $\mu R(s_i)$ as an entry to vector μR_{T_k}
      Partition the ranks set $r(T_k)$ into q sets ;
      for each partition $(Q_j \forall j \in 1, 2, 3, \ldots, q)$ **do**
        for each partition $(Q_l \forall l \in 1, 2, 3, \ldots, q)$ **do**
          $\upsilon(Q_{s_l}) \leftarrow Correlation(Q_j, Q_l)$; ▷ correlation of ranks
          $\rho_{s_l}(Q_l) \leftarrow \upsilon(Q_{s_l})$;
        end for
        $\upsilon(Q_{s_l}) \leftarrow variance(\rho_{s_l}(Q_l))$; ▷ Find variance $\rho_{s_l}(Q_l)$
      end for
      $\upsilon VQ(T_k) \leftarrow mean\ (\upsilon(Q_{s_l}))$; ▷ Find mean $\upsilon(Q_{T_k s_i})$
      $\sigma Q(t_k) \leftarrow deviation\ (\upsilon(Q_{s_j}))$; ▷ Find deviation $\upsilon(Q_{s_j})$
    end for
    Sort (μR_{T_k}); ▷ Sort the services Ascending Order
    for each service $s_i, \forall i \in 0, 1, 2, \ldots, n$ **do**
      if ($\mu R_{(}s_i) <$ Threshold) **then**
        $service\_variance.add(s_i)$;
      end if
    end for
    Sort ($service\_deviation$); ▷ Sort service deviation
    for each service $s_d \forall d \in 0, 1, 2, \ldots, m$ **do**
      if ($\sigma Q(s_d) <$ Threshold) **then**
        $select\_services.add(s_d)$;
      end if
    end for
    best_rank (select_ services); ▷ Select best services based on their average rank of QoS parameters.
  end for
end procedure

Service selection by DURD We apply the processes explored in this section (3.2.1.1) for all the available services for a task T. We sort all the services by their μR values in ascending order and discard the services with a μR value greater than the given threshold (generally the average of the μR values for all services). Then, we sort the remaining services in ascending order of their υQV values and discard services with an υQV value greater than the given threshold (generally the average of the υQV values for all services). Afterwards, the remaining services are sorted in ascending order of their σQV values, and ser-

vices with σQV values greater than the given threshold (generally the average of the σQV values for all services) will be discarded. The services surviving this DURD filtering process are those that exhibit the best QoS fitness. These services are then sorted in ascending order of their μR values.

3.2.1.2 Evaluation of associability fitness of the composition

Let a set of possible compositions C be $\{c_1, c_2, c_3, \ldots, c_n\}$. The connection associability score (cas) of composition c_i indicates the number of connections having associability (a connection formed between services of the same provider or providers that have a service level agreement) against the number of connections requiring associability (see section 3.1). Then the associability fitness value $afv(c_i)$ is measured as follows:

$$afv(c_i) = \frac{1}{1 + \mu(cas(c_i), AC)} \tag{3.5}$$

where $\mu(cas(c_i), AC)$ is the mean of $cas(c_i)$ and AC. AC is the associability count, representing the total number of edges between tasks that require associability in the target application of the service composition. $afv(c_i)$ is the associability fitness of composition c_i, which is measured by normalizing the standard deviation observed from $cas(c_i)$ and AC to $0 \leq afv(c_i) \leq 1$. To avoid a divide-by-zero error, the resultant standard deviation is increased by 1.

3.2.1.3 Evaluation of overall fitness of the composition

The overall composition fitness of a composition is measured by finding the service fitness values, the fitness score of the composition, and the composition fitness values and associability fitness value.

Finding service fitness The fitness values of the services involved in a composition is measured as follows:

If

$$\begin{cases} \mu R(T_j s_k) \leq \mu(\sum_{i=1}^{n} \mu R(T_j s_i)) \text{and} \\ \quad \mu QV(T_j s_k) \leq \mu(\sum_{i=1}^{n} \mu QV(T_j s_i)) \text{and} \\ \quad \sigma QV(T_j s_k) \leq \mu(\sum_{i=1}^{n} \sigma QV(T_j s_i)) \end{cases} \tag{3.6}$$

then $T_j s_k$ is fit.

Finding fitness score of the composition In a given composition c such that $c = T_i S_j \exists j \in [1, 2, 3, \ldots] \forall i \in [1, 2, 3, \ldots, |c|]$ (c being a composition of services related to all tasks in a sequence), if a service of the composition is fit, then the fitness score of the composition $fc(c)$ will be incremented by 1. Then we select all compositions having a fitness score less than a given threshold as input for assessing the discrete uniform service rank distribution.

Sorting by composition fitness and associability fitness values The resultant compositions are sorted in descending order of their composition fitness values cfv, and the one having the "maximum best service compositions ratio" ($mbscr$) is selected from the best compositions $mbsc$ among the resultant ordered compositions. Then, the compositions $mbsc$ are sorted in descending order of their associability fitness afv, and the one having the "maximum best associability compositions ratio" ($mbacr$) is selected from the ordered $mbsc$.

3.2.1.4 Evaluating discrete uniform service rank distribution as composition fitness

This process depicts the uniform distribution of the ranks of services assessed under DURD for each service related to the different QoS parameter considered. The process is similar to DURD. DURD considers the ranks assigned to the QoS parameters as the inputs to assess the service level fitness, whereas the discrete uniform service rank distribution (DUSRD) considers the ranks assigned to the services to assess the composition fitness.

In order to assess the DUSRD of a service S_i of task t, we measure the following three factors:

- Mean value $\mu R(c_i)$ of the ranks of the composition.
- Correlations for each partition in a composition.
- Deviation of correlations in each composition.

Measuring mean value of ranks of the composition Let the rank set of a composition $c = T_i s_j \exists j \in [1, 2, 3, \ldots, |T_i s|] \forall i \in [1, 2, 3, \ldots, |c|]$ (c being a composition of services related to all tasks in a sequence) be $rs(c_i) = [r(T_1 s_j \exists j \in [1, 2, 3, \ldots, |T_1 s|]), r(T_2 s_j \exists j \in [1, 2, 3, \ldots, |T_2 s|]), \ldots, r(T_m s_j \exists j \in [1, 2, 3, \ldots, |T_m s|])]$. Then, we find the mean of the ranks of all services (μR) of composition c_i, defined as follows:

$$\mu R(c_i) = \left(\frac{\sum_{k=1}^{|rs(c_i)|} r(T_k s_j \exists j \in [1,2,3,\ldots,|T_k s|])}{|rs(s_i)|} \right) \tag{3.7}$$

Measuring correlations for each partition in a composition We partition the rank set $rs(c_i)$ into q partitions, which can be referred to as $Q1_{rs(c_i)}, Q2_{rs(c_i)}, Q3_{rs(c_i)}, \ldots, Qq_{rs(c_i)}$. Then, we find the correlation of each part with all other parts and the covariance of these resultant correlations as given in Algorithm 3.3.

Algorithm 3.3 Correlation for each partition of composition (c_i)

```
1: procedure CORRELATION(C_j)
2:     υ(Q_{c_i}) ← φ;                                   ▷ initializing a vector
3:     for each k: 1, 2, ..., q do
4:         ρ(Q_{(c_i)_k}) ← φ;                           ▷ initializing a vector
5:         for each j: 1, 2, ..., q do
6:             ρ(Q_{(c_i)_k}) ← cor(Qk_{rs(s_i)}, Qj_{rs(s_i)}) ;
7:         end for
8:         υ(Q_{c_i}) ← cov(ρQ_k);        ▷ adding covariance of the values of vector ρ(Q_k) to υ(Q_{s_i})
9:     end for
10: end procedure
```

$\rho(Q_{(c_i)_k})$ is a vector of size q, and each entry of this vector is the correlation between partition $Qk_{rs(c_i)}$ and one of the partitions $Qj_{rs(c_i)}$ of $rs(c_i)$. $\upsilon(Q_{c_i})$ is a vector of size q, and each entry of this vector is the covariance of the vector $\rho(Q_{(c_i)k})$ entries (correlations observed between a part $Qk_{rs(c_i)}$ and all parts of $rs(c_i)$).

Further, we find the mean of all entries of the vector $\upsilon(Q_{c_i})$ for each composition c_i as follows:

$$\upsilon QV(c_i) = \frac{\sum_{j=1}^{|\upsilon(Q_{c_i})|} \upsilon(Q_{c_i})_j}{|\upsilon(Q_{c_i})|} \tag{3.8}$$

where $\upsilon(Q_{c_i})$ is a vector of size q and each entry of this vector is the covariance of $\rho(Q_{(c_i)_k})$. $\rho(Q_{(c_i)_k})$ is a vector of correlations observed between a part $Qk_{rs(c_i)}$ and all parts of the $rs(c_i)$ entries.

Measuring deviation of correlations in each composition We measure the deviation $\sigma QV_{(c_i)}$ of composition c_i as follows:

$$\sigma QV_{(c_i)} = \sqrt{\frac{\sum_{k=1}^{|\upsilon(Q_{c_i})|}(\mu(\upsilon(Q_{c_i})) - \upsilon(Q_{c_i})_k)^2}{|\upsilon(Q_{c_i})|}} \tag{3.9}$$

The pseudocode representation of DUSRD is shown as Algorithm 3.4.

Algorithm 3.4 Discrete Uniform Service Rank Distribution (DUSRD)

procedure DUSRD(C_i)
Ranks of service: a set that contains ranks for each service of a composition.
Steps applied to all possible tasks to a composition:
  $Generate\_composition(set\ of\ tasks)$; ▷ Generate possible composition
  for each $task(T_i)$ **do**
    $selected\_services \leftarrow DURD(s_i)$;
  end for
  for each composition $(c_i, \forall i \in 0, 1, 2, \ldots, m)$ **do**
    for each task $(T_j, \forall j \in 0, 1, 2, \ldots, n)$ **do**
      $\mu R(T_i) \leftarrow \frac{\sum_{j=1}^{|rs(c_i)|}(r(T_j)\exists r(r_j)\in rs(c_i))}{|rs(c_i)|}$; ▷ Mean of Ranks
      $\mu R_{c_i} \leftarrow \mu R(T_i)$; ▷ Move $\mu R(T_i)$ as an entry to vector μR_{c_i}
      Partition the ranks set $rs(c_i)$ into q sets;
      for each partition $(Q_i \forall i \in 1, 2, 3, \ldots, q)$ **do**
        for each partition $(Q_j \forall j \in 1, 2, 3, \ldots, q)$ **do**
          $\upsilon(Q_{T_i}) \leftarrow Correlation(Q_j, Q_l)$; ▷ Correlation of ranks
          $\rho_{T_i}(Q_i) \leftarrow \upsilon(Q_{T_i})$;
        end for
        $\upsilon(Q_{T_i}) \leftarrow variance(\rho_{T_i}(Q_l))$; ▷ Find variance $\rho_{T_i}(Q_l)$
      end for
      $\upsilon VQ(c_i) \leftarrow mean\ (\upsilon(Q_{T_i}))$; ▷ Find mean $\upsilon(Q_{T_k s_i})$
      $\sigma Q(c_i) \leftarrow deviation\ (\upsilon(Q_{T_i}))$; ▷ Find deviation $\upsilon(Q_{T_i})$
    end for

```
21:        υVQ(c_i) ← mean (υ(Q_{T_i}));                    ▷ Find mean υ(Q_{T_k s_i})
22:        σQ(c_i) ← deviation (υ(Q_{T_i}));                ▷ Find deviation υ(Q_{T_i})
23:    end for
24:    μR_C ← μR(c_i); υVQ(C) ← υVQ(c_i);
25:    σQ(C) ← σQ(c_i);
26:    Sort(μR(C));                                         ▷ Sort the services Ascending Order
27:    for each composition c_i, ∀i ∈ 0, 1, 2, ... |C| do
28:        if (μR_(c_i) < Threshold) then
29:            composition_variance.add(c_i);
30:        end if
31:    end for
32:    Sort (composition_variance);                         ▷ Sort composition variance
33:    for each composition c_i ∀i ∈ 0, 1, 2, ... |composition_variance| do
34:        if (υVQ(c_i) < Threshold) then
35:            composition_deviation.add(c_i);
36:        end if
37:    end for
38:    Sort(composition_deviation);                         ▷ Sort composition deviation
39:    for each composition s_d ∀d ∈ 0, 1, 2, ... composition_deviation do
40:        if (σQ(c_i) < Threshold) then
41:            select_compositions.add(c_i);
42:        end if
43:    end for
44:    best_rank (select_composition);
45: end procedure
```

Compositions selection by DUSRD We apply the processes explored in this section (3.2.1.4) for all compositions available as genotypes. We initially sort all compositions by their μR values in ascending order and discard the compositions with μR greater than the given threshold (generally the average of the μR values for all services). Then, we sort the remaining compositions by their υQV values in ascending order and discard compositions with an υQV value greater than the given threshold (generally the average for all compositions). Afterwards, the remaining compositions are sorted by their σQV values in ascending order, and compositions with σQV values greater than the given threshold (generally the average of the σQV values for all compositions) will be discarded. The compositions surviving this filtering process for assessing DUSRD are said to be the set of new genotypes for further GA evolution.

3.2.2 Optimal Fitness-Aware Service Composition using AGEGA

The inputs for AGEGA are as follows:

- A set of tasks $CT = \{T_1, T_2, T_3, \ldots, T_{|T|}\}$ involved in the given application
- A set of available candidate services $CS = \{ST_1 = \{s_{1\,1}, s_{1\,2}, \ldots, s_{1\,i}\}, ST_2 = \{s_{2\,1}, s_{2\,2}, \ldots, s_{2\,j}\}, \ldots, ST_p = \{s_{p\,1}, s_{p\,2}, \ldots, s_{p\,m}\}\}$, where i, j, and m denote the number of candidate services in ST_1, ST_2, and ST_p, respectively
- Set of all possible compositions: $C = \{c_1, c_2, c_3, \ldots, c_{|c|}\}$
- Associability count AC observed
- A set of task sets demanding associability $ST^{'} = \{\{ST_i, ST_j, ST_k, \ldots\}, \{ST_x, ST_y, ST_z, \ldots\}, \ldots\}$
- Maximum evolution iteration threshold, met

Following are the preprocessing steps for AGEGA:

- Perform the service level fitness calculation (see section 3.2.1.1).
- Perform the composition associability fitness calculation (see section 3.2.1.2).
- Perform the composition fitness calculation (see section 3.2.1.3).
- Sort the resultant compositions and select the best compositions by composition fitness value and composition associability fitness value (see section 3.2.1.4).

The proposed OFASC using Adaptive Genotypes Evolution-Based GA is as follows (see Algorithm 3.5 and Algorithm 3.6):

1. Find the common services from given any two compositions such that a common service must not be the crossover point if it currently has a desired associability in parent compositions and does not retain that associability in the resultant composition.

2. For all the common services found: (i) Divide the first and second compositions in the pair having a common service into two parts, denoted as lp_1 and rp_1 from the first composition and lp_2 and rp_2 from the second composition. Then, build the two new compositions by connecting lp_1 and rp_2, which forms the first one, and connecting lp_2 and rp_1, which forms the second. (ii) Consider any of the newly formed compositions to be best composition if its fitness is greater than the fitness of any of the parent compositions.
3. If any new compositions are formed in step 2, then add all of them to the composition set C.
4. Sort the composition set C and select the best compositions by composition and associability fitness values.
5. Continue the adaptive genotype evolution process on C up to the given maximum evolution iteration threshold met.
6. Verify the DUSRD (see section 3.2.1.4) of each composition among the resultant compositions.
7. Sort the resultant compositions and select the best compositions through fitness score, associability score, and DUSRD (see sections 3.2.1.3 and 3.2.1.4).

Algorithm 3.5 Pseudocode of OFASC-AGEGA

```
1: procedure CORRELATION(C_j)
2:     pc ← true;                                          ▷ Best composition process
3:     met ← 0;                                            ▷ Maximum evolution threshold
4:     while pc do
5:         nC ← φ;
6:         for each composition {c_i ∀c_i ∈ C} find crossover points do
7:             for each composition {c_j ∃(c_j ∈ C ∧ j ≠ i)} do
8:                 nC ← AGEGA(c_i, c_j);
9:             end for
10:        end for
11:        C̄ ← C;                                          ▷ Clone the C as C̄
12:        C̄ ← nC;                                         ▷ Adding new compositions to C
13:        DUSRD (C̄);                                      ▷ see section 3.2.1.4
14:        if C ≥ C̄ then
15:            pc ← false;                                 ▷ Equaling by definition
16:        end if
17:    end while
18: end procedure
```

Algorithm 3.6 Adaptive Genotypes Evolutions based Genetic Algorithm (AGEGA)

procedure AGEGA((c_i, c_j))
  $c_{i\,j} \leftarrow \phi$; ▷ a set of compositions formed from crossover of c_i, c_j
  for each service $s_k \forall s_k \in c_i$ **do**
    $cop \leftarrow \phi$; ▷ set of crossover points
    for each service $s_l \forall s_l \in c_j$ **do**
      if $((s_k \cong s_l) \& (a(e_{(s_{k-1} \rightarrow s_k)}) = 0 \& a(e_{(s_k \rightarrow s_{k+1})}) = 0) \& (a(e_{(s_{l-1} \rightarrow s_l)}) = 0 \& a(e_{(s_l \rightarrow s_{l+1})}) = 0))$ **then** ▷ $(a(e_{(s_{k-1} \rightarrow s_k)}) = 0 \& a(e_{(s_k \rightarrow s_{k+1})}) = 0)$ indicates that the s_k is not having associability with predecessor and successor services in composition ▷ $(a(e_{(s_{l-1} \rightarrow s_l)}) = 0 \& a(e_{(s_l \rightarrow s_{l+1})}) = 0)$ indicates that the s_l is not having associability with predecessor and successor services in composition
        $cop \leftarrow s_k$;
      end if
    end for
    for each $cp \forall cp \in cop$ **do**
      cross c_i at cross point cp and form $\overleftarrow{c_i}$ and $\overrightarrow{c_i}$;
      cross c_j at cross point cp and form $\overleftarrow{c_j}$ and $\overrightarrow{c_j}$;
      divide c_i in to two parts at cross point cp, and the left part label as $\overleftarrow{c_i}$ and right part as $\overrightarrow{c_i}$;
      divide c_j in to two parts at cross point cp, and the left part label as $\overleftarrow{c_j}$ and right part as $\overrightarrow{c_j}$;
      Form composition c_k as $c_k \leftarrow \phi; c_k \leftarrow \overleftarrow{c_i}; c_k \leftarrow cp; c_k \leftarrow \overrightarrow{c_j}$;
      $c_{i\,j} \leftarrow c_k$;
      Form composition r_k as $c_l \leftarrow \phi; c_l \leftarrow \overleftarrow{c_j}; c_l \leftarrow cp; c_l \leftarrow \overrightarrow{c_i}$;
      $c_{i\,j} \leftarrow c_l$;
    end for
    for each new composition $c \forall c \in c_{i\,j}$ **do**
      if $(fs(c) < fs(c_i) \& fs(c) < fs(c_j))$ **then**
        delete c form $c_{i\,j}$;
      end if
    end for
    Return $c_{i\,j}$;
  end for
end procedure

3.2.3 Performance Evaluation

To evaluate the scalability and effectiveness of the proposed approach, we conducted experiments on a personal computer with an Intel (R) core (TM) i5 2.60-GHz processor and 8 GB of memory, running Windows 8.1. There are some benchmark data sets for web services, [180] and [181], that are not suitable in the context of OFASC-AGEGA because these benchmark data sets describe only the values for response time and throughput. For our proposed approach, we used a synthetic data set comprising the major QoS parameters such as availability,

security, accessibility, cost, integrity, throughput, response time, and reliability. This synthetic data set comprises 2250 web services. We used the QoS data set described in [182] to assign the values to the services. The QoS data set has various QoS parameters and more than 2500 web services. From this data set, we randomly selected 2250 web services and their corresponding QoS parameter values. The missing QoS parameters and their corresponding values were randomly generated. Different combinations of tasks in the range of 70 (sparse) to 250 (dense) were used to conduct experiments in order to assess the performance. Each task comprises 7 to 16 services. We considered 450 different service composition scenarios. The number of tasks given for each service composition was between 7 and 25. We tested this data set using the Shapiro–Wilk test, and the results revealed that the given data set was normally distributed [183]. There are several researchers who have adopted synthetic data sets to validate their methods in service composition [132, 184, 185, 186, 187, 188]. The empirical analysis of our proposed approach was implemented in Java. Statistical measures such as discrete uniform rank distribution and discrete uniform service rank distribution of the discovered compositions were performed using the R language. In our evaluations, several parameters were fixed for all approaches, controlling the execution time and specifying the population size. We set the population size to 100. Each experiment was executed continuously 30 times, and the mean of each run was duly noted beacuse of the stochastic nature of algorithms. Table 3.2 presents the list of approaches (with name, abbreviations, and parameter settings) that we compared with our proposed approach.

Table 3.2: Compared approaches with parameter settings

Approach	Abbrevations	Parameter Setting
Genetic Algorithm [33]	GA	Crossover=0.8, mutation=0.001 and random selection approach is applied
Orthogonal Genetic Algorithm [123]	OGA	Crossover P_c=0.1, mutation P_m=0.02 and F = 4
Adaptive Genetic programming [98]	AGP	population size=30, g_{max}=500, k_c = 2.5, k_m = 2.0,Crossover P_c = 0.9, mutation P_m = 0.01
Genetic algorithm with simulated annealing [189] Genetic algorithm with Harmony Search	GASA, GAHS	Crossover=0.8, mutation=0.01, SA application = 0.03
Transactional Genetic Algorithm [190]	TGA	population = 50, Crossover=0.9, mutation=0.1, η (incentive factor = 0.5) and ρ (penality factor = 0.3)

We evaluated the performance of our proposed algorithm in the following ways:

- Evaluated the average fitness values by varying the number of abstract services.
- Evaluated the process completion time and computational complexity for a varying number of abstract services.

We compare our propose approach with other approaches such as GA [176], TGA [190], GASA [189], GAHS [189], AGP [98], and OGA [123]. Table 3.3 illustrates the average fitness values for different test cases with 10, 30, 50, 70, 90, and 100 abstract services.

Table 3.3: Average fitness values observed for our proposed approach and other compared approaches

		Mean of the fitness values						
Test Cases.	Number of Abstract Services	GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
65	10	0.327	0.3261	0.3333	0.323	0.3241	0.3875	**0.4138**
54		0.3166	0.3156	0.3159	0.3071	0.3229	0.3357	**0.4299**
49	30	0.2681	0.3083	0.3348	0.2369	0.2912	0.3132	**0.4108**
61		0.3297	0.2901	0.3451	0.2053	0.2874	0.3128	**0.4584**
65	50	0.2791	0.3049	0.3150	0.3083	0.3181	0.3351	**0.4330**
54		0.3138	0.3174	0.329	0.3138	0.3218	0.3315	**0.4603**
49	70	0.2873	0.2870	0.3343	0.3239	0.3189	0.3437	**0.4679**
61		0.3027	0.2907	0.3234	0.3358	0.3298	0.3473	**0.4897**
65	90	0.3071	0.3116	0.3435	0.3268	0.3014	0.4757	**0.4983**
54		0.327	0.3261	0.3333	0.323	0.3241	0.5075	**0.5138**
49	100	0.3235	0.4061	0.3781	0.3643	0.3591	0.5248	**0.5456**
65		0.327	0.4261	0.3833	0.3834	0.3739	0.5784	**0.6238**

Based on Table 3.3, we notice that our proposed approach gives reasonably better results than other approaches. The completion time of our proposed approach is presented in Fig. 3.3. Based on experiments, we observe that the proposed Adaptive Genotypes Evolution-Based Genetic Algorithm is scalable and robust compared to other approaches. Based on Fig. 3.3, we observe that our proposed approach takes less time to complete than other contemporary approaches. The completion times for TGA, GAHS, GASA, GA, AGP, and OGA to obtain the

best solutions were 317.559, 221.898, 210.244, 159.920, 95.795, 81.640 s, respectively, whereas the completion time for the OFASC-AGEGA evolution iterations was 48.086 s. The computational time taken by the adaptive genotype evolution strategy is noticeably minimal (see Fig. 3.4) compared to other approaches [98, 123, 176, 189, 190].

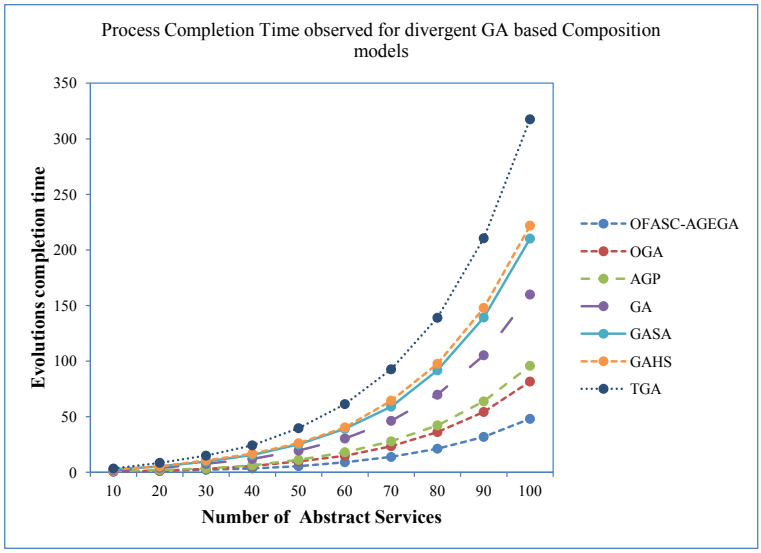


Figure 3.3: Process completion time observed for OFASC-AGEGA with other approaches.

The computational times for TGA, GAHS, GASA, GA, AGP, and OGA to obtain the best solutions were 31.755, 22.189, 21.024, 15.992, 9.579, and 8.164 s, respectively, whereas the computational time for OFASC-AGEGA was 4.808 s. In GA and TGA, with each iteration, a new population is generated, and the individual fitnesses are evaluated (based on fixed and predetermined crossover and unguided mutation). These processes cause GA and TGA to converge slowly and to become easily stuck in local maxima. In GAHS and GASA, with each iteration, a updated population is generated, and the individual fitnesses are assess (based on predetermined crossover and unguided mutation). It takes more time because of the higher number of parameter turnings and the single-point crossover. In the OGA algorithm, the orthogonal method is used to produce the new population, and the crossover strategy is employed on the two parents, thus causing the process to consume more time. In our proposed approach, we use the adaptive

genotype progressive evolution strategy to restrict the evolution iterations to a minimal number. The progressive evolution strategy considers those offspring in the new generation that have better fitness than any of their parents. We evaluate the service fitness and composition fitness by using DURD and DUSRD, which helps to prune the services and compositions having non-optimal solutions and selects only the best compositions, thereby gradually shrinking the valid search space. Hence, it consumes less time than other methods.

The evolution complexity obtained in our proposed model is linear (see Fig. 3.4). As observed from Fig. 3.3 and Fig. 3.4, our proposed approach is more effective, scalable, and robust than the other approaches.

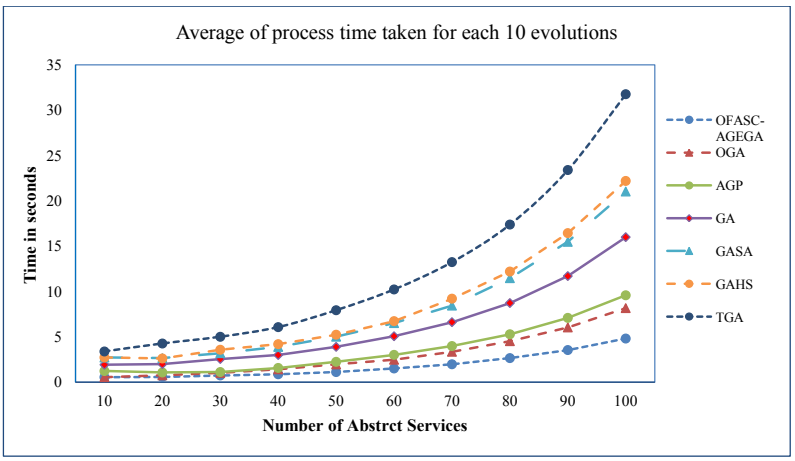


Figure 3.4: Computational complexity observed for OFASC-AGEGA with other approaches.

The time complexity of OFASC-AGEGA is $O(n)$, whereas the observed complexity of the other models is $O(n^2)$, and the observed complexity of the model in [98, 123, 189] is $O(n\log(n))$. The fitness distribution (DURD) of the services involved in the 10 best compositions recommended by OFASC-AGEGA is optimal, in contrast to other approaches (see Fig. 3.6).

The accuracy of the composition recommendations of OFASC-AGEGA is assessed using probability theory [191]. The success ratio of a composition is measured as the average success ratio of the services included in that composi-

tion. The assessment of the accuracy of a composition is defined as follows:

$$sup_S(S_j) = \frac{\sum_{i=1}^{|C_S|}(1 \exists S_j \in (c_i \forall c_i \in C_S))}{|C_S|} \tag{3.10}$$

where $sup_S(S_j)$ indicates the ratio (positive support) of service S_j in the set of compositions C_S that are labeled as S, c_i indicates the i^{th} composition of set C_S, and $|C_S|$ indicates the number of compositions labeled as S.

$$sup_F(S_j) = \frac{\sum_{i=1}^{|C_F|}(1 \exists S_j \in (c_i \forall c_i \in C_F))}{|C_F|} \tag{3.11}$$

where $sup_F(S_j)$ indicates the ratio (negative support) of service S_j in the set of compositions C_F that are labeled as F, c_i indicates the i^{th} composition of set C_F, and $|C_F|$ indicates the number of compositions labeled as F.

$$SR(s_j) = sup_S(s_j) - sup_F(s_j) \tag{3.12}$$

where $SR(s_j)$ indicates the success ratio of a service s_j in composition formation. Further, the success ratio of a composition c_i can be measured as follows:

$$SR(c_i) = \frac{\sum_{j=1}^{|c_i|}(SR(s_j) \exists s_j \in c_i)}{|c_i|} \tag{3.13}$$

where $SR(c_i)$ indicates the success ratio of composition c_i, which is the average of the success ratios of services involved in composition c_i. The compositions recommended by the OFASC have a lower root mean square error (RMSE) [191, 192] than the compositions recommended by [98, 123, 176, 189, 190] (see Fig. 3.5). The RMSE of a composition is measured as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(osr - SR(c_i))^2}{n}} \tag{3.14}$$

where osr indicates the optimal success ratio (which is 1 in this context) and n represents the n best compositions recommended.

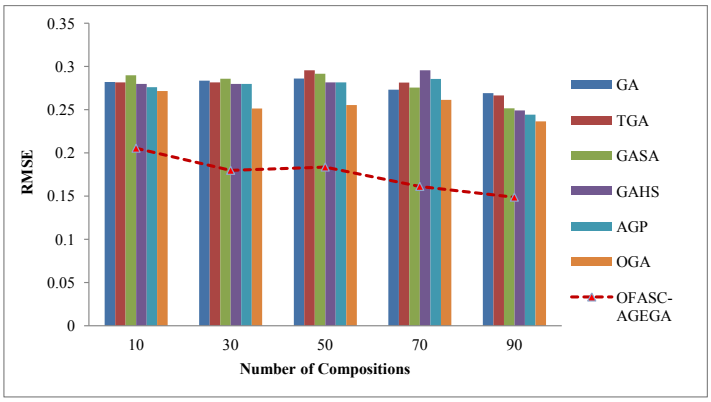


Figure 3.5: Average RMSE of the recommended compositions.

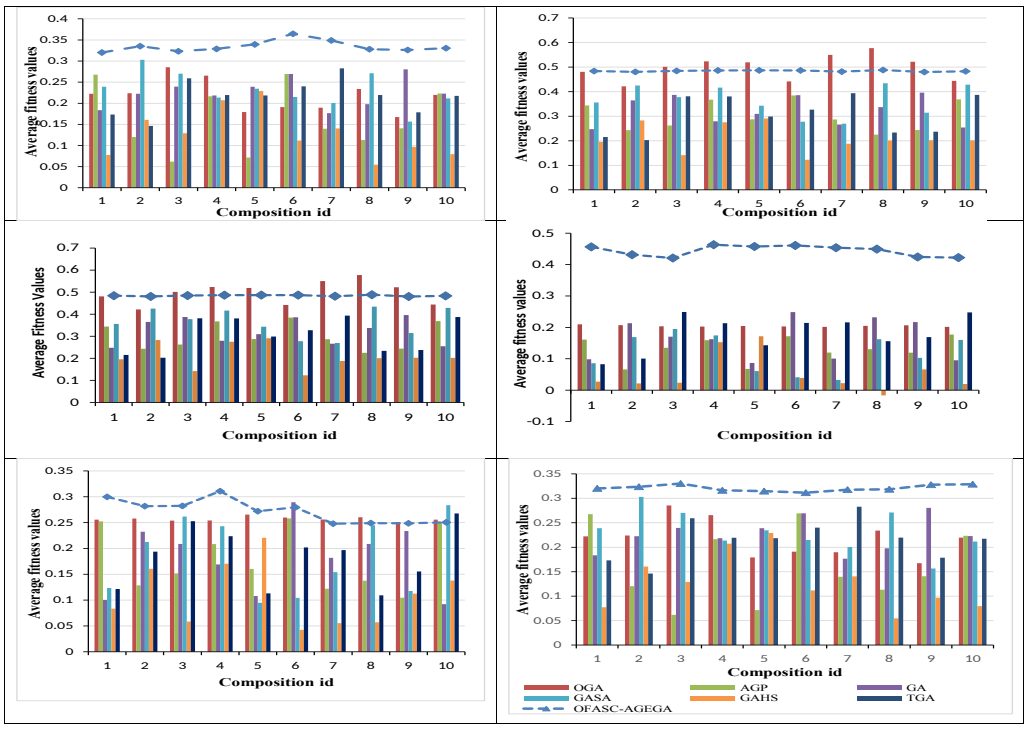


Figure 3.6: Fitness distribution (DURD) of services involved in 10 resultant compositions

We performed statistical tests (parametric and non-parametric) for our proposed OFASC-AGEGA and other contemporary algorithms to ascertain whether the results of the proposed algorithms are statistically significant. The T-test [193, 194] and the Wilcoxon signed-rank test [195] were applied to evaluate whether the

best mean values obtained for all algorithms have a significant difference with 58 degrees of freedom at a 1% level of significance. The results of the T-test and the Wilcoxon signed-rank test for all methods are presented in Tables 3.4 and 3.5, respectively, for 100 abstract services (with 100 candidate services). Based on Table 3.4, the values procured are statistically significant (all T-values are positive, and the P-values are near to zero). Based on Table 3.5, the values procured are statistically significant (all Z-values are obtained by positive ranks, and the P-values are near to zero). Therefore, based on the statistical tests conducted, our proposed method is more accurate than the other methods.

Table 3.4: T-Test results for OFASC-AGEGA and other compared approaches

Approaches		GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
GA		*	*	*	*	*	*	*
TGA		-0.3013	*	*	*	*	*	*
		(0.38303)						
GASA	**T-Value/**	0.11201	0.11201	*	*	*	*	*
	P-Value	(0.04559)	(0.06998)					
GAHS		-1.6894	-1.6894	-1.6894	*	*	*	*
		(0.05263)	(0.00547)	(0.052631)				
AGP		0.1807	0.1807	0.1807	0.04502	*	*	*
		(0.42913)	(0.42913)	(0.42913)	(0.4840)			
OGA		1.50505	1.69982	1.53129	2.77954	1.62635	*	*
		(0.07327)	(0.05163)	(0.06998)	(0.00547)	(0.059058)		
OFASC-AGEGA		4.5245	1.85282	4.429244	5.96342	4.83606	1.85282	*
		(0.00008)	(0.03869)	(0.00015)	(0.00001)	(0.000039)	(0.03869)	

Table 3.5: Wilcoxon signed-rank test results for OFASC-AGEGA and other compared approaches

Approaches		GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
GA		*	*	*	*	*	*	*
TGA		-1.1767 (0.119)	*	*	*	*	*	*
GASA	Z-Value/ P-Value	-0.7452 (0.22663)	-0.3922 (0.3482)	*	*	*	*	*
GAHS		-3.0594 (0.00111)	-3.0594 (0.00111)	-3.0594 (0.00111)	*	*	*	*
AGP		-0.6276 (0.26435)	-3.0594 (0.00111)	-0.2353 (0.40517)	-3.0594 (0.00111)	*	*	*
OGA		-2.8241 (0.0024)	-3.0594 (0.00111)	-2.7456 (0.00298)	-3.0594 (0.00111)	-2.8241 (0.00241)	*	*
OFASC-AGEGA		-2.9025 (0.00187)	-3.0594 (0.00111)	-2.9025 (0.00187)	-3.0594 (0.00111)	-2.9025 (0.00187)	-2.1181 (0.017)	*

3.3 Optimal Fitness-Aware Service Composition using Modified Invasive Weed Optimization (OFASC-MIWO)

3.3.1 Modeling QoS-Aware Web Service Composition

Let a web service composition task CT be a combination of n tasks $CT = \{t_1, \ldots, t_n\}$, let the services in a set $a_i = \{s_1, \ldots, s_{x_i}\}$ be the collection of the x_i services that can address task t_i, and let $A = \{a_1, \ldots, a_n\}$. The candidate services in a subset a_i have similar functionalities but can differ in their QoS parameters. Let a set of QoS parameters be $P = \{p_1, p_2, \ldots, p_{|q|}\}$, where $|q|$ denotes the number of QoS parameters. Further, each QoS parameter in P is associated with each service in the given service set A. Let $E^{'} = \{E_1, E_2, \ldots, E_m \forall\, [E_x : t_y \rightarrow t_{y+1}]\}$ be the set of edges connecting two tasks in each possible composition sequence. Let $ST^{'}$ = $\{\{ST_i, ST_j, ST_k, \ldots\}, \{ST_x, ST_y, ST_z, \ldots\}, \ldots\}$ be the collection of sets of tasks such that the connections between each task set

is affected by any of the connectivity constraints (dependency ($\biguplus$), parallelism ($\bigoplus$), rollback ($\bigodot$), and sequence ($\bigotimes$)). Each task set group $ST^{'}$ is expecting associability with another task set. In our proposed approach, associability is defined by the following conditions: (i) the services utilized for these tasks should be offered by the same service provider or (ii)the different providers have similar service level agreements.

3.3.1.1 Evaluation of local fitness of services

The QoS parameters are normalized in a manner similar to that described in section 3.2.1.1. Further, the services related to a specific task are ranked in descending order of the values of these normalized QoS parameters. Each service is ranked differently under different QoS parameters, which will be used to compute the local fitness. In the following, we will be using some statistics (mean, standard deviation, skewness), but we are only interested in their mathematical properties and are not concerned with their statistical interpretation. In particular, because in this context we are not dealing with sampling from a population, the population version of the formulas will be used, in contrast to the more frequently used formulas for estimation of statistics. Consider s_x as a service, where $s_x \in a_i, a_i \in A$. Let the ranks $r(s_x)$ of the service be $\Big[r(p_1), \ldots, r(p_{n_q})\Big]$. The mean rank $\mu(s_x)$ of all QoS parameters of service s_x is given by

$$\mu(s_x) = \frac{1}{n_q} \sum_{i=1}^{n_q} r(p_i) \tag{3.15}$$

The standard deviation $S(s_x)$ of the ranks of the QoS parameters assigned to service s_x is calculated as follows:

$$S(s_x) = \sqrt{\frac{\sum_{k=1}^{n_q} (r(p_k) - \mu(s_x))^2}{n_q}} \tag{3.16}$$

The skewness $\chi(s_x)$ [196] of the ranks of the QoS parameters distributed for service s_x is computed as follows:

$$\chi(s_x) = \frac{1}{n_q} \frac{\sum_{k=1}^{n_q} (r(p_k) - \mu(s_x))^3}{(S(s_x))^3} \tag{3.17}$$

$$\chi'(s_x) = |\chi(s_x)| \tag{3.18}$$

The skewness can be positive or negative. A skewness value of 0 occurs when the ranks of all QoS parameters are the same. As we are interested in positive skewness, we consider only the absolute value of $\chi(s_x)$. According to [197], the least variance between standard deviation, mean, and skewness indicates the distribution of ranks with the least deviation, the least skewness, and reasonably average ranks. The mean $\mu(\chi'(s_x), \mu(s_x), S(s_x))$ of the resultant skewness ($\chi(s_x)$), mean ($\mu(s_x)$), and standard deviation ($S(s_x)$) of service s_x is measured as follows:

$$\mu(\chi'(s_x), \mu(s_x), S(s_x)) = \frac{1}{3}\Big[\chi(s_x) + \mu(s_x) + S(s_x)\Big] \tag{3.19}$$

The variance of the resultant statistics of service s_x is measured as follows:

$$S^2(\chi'(s_x), \mu(s_x), S(s_x)) = \frac{1}{3}\left[\begin{array}{c}(\mu(\chi'(s_x), \mu(s_x), S(s_x)) - \chi'(s_x))^2 + \\ (\mu(\chi'(s_x), \mu(s_x), S(s_x)) - \mu(s_x))^2 + \\ (\mu(\chi'(s_x), \mu(s_x), S(s_x)) - S(s_x))^2\end{array}\right] \tag{3.20}$$

The local fitness of service s_x is measured as follows:

$$lfv(s_x) = [S^2(\chi'(s_x), \mu(s_x), S(s_x)) + 1]^{-1} \tag{3.21}$$

The resultant variance (S^2) is normalized to a value between 0 and 1, such that greater variance indicates less fitness. To prevent a divide-by-zero error, we add 1 to the variance. The pseudocode for local fitness of services is given as Algorithm 3.7.

Algorithm 3.7 Pseudocode for local fitness value of services

procedure LOCAL FITNESS VALUE(s_j)
Input: a set that contains ranks for each QoS parameter of a service
Output: best service or optimal service
Steps applied to all possible services to a task:
  for each task $(T_k, \forall k \in 0, 1, 2, \ldots, n)$ **do**
    for each service $(s_j, \forall j \in 0, 1, 2, \ldots, n)$ **do**
      $\mu(s_j) \ \frac{1}{n_q} \sum_{i=1}^{n_q} r(m_i)$; ▷ Mean of ranks of the QoS parameters
      $\sigma(s_j) \ \leftarrow \sqrt{\frac{\sum_{k=1}^{n_q} (r(p_k)-\mu(s_j))^2}{n_q}}$; ▷ Standard deviation of ranks of the QoS parameters
      $\chi(s_j) \ \leftarrow \frac{1}{n_q} \frac{\sum_{k=1}^{n_q} (r(p_k)-\mu(s_j))^3}{(S(s_j))^3}$; ▷ Skewness of ranks of the QoS parameters
      $\chi'(s_j) \ \leftarrow |\chi(s_j)|$;
      if $(\chi(s_j) \geq Positive)$ **then**
        $m_c = \chi(s_j)$;
      else
        $\chi'(s_j) \ \leftarrow |\chi(s_j)|$;
      end if
      $\mu(\chi'(s_j), \mu(s_j), S(s_j)) \ \leftarrow \frac{1}{3}\Big[\chi(s_x) + \mu(s_x) + S(s_x)\Big]$; ▷ fitness of the service s_j
      $S^2(\chi'(s_j), \mu(s_j), S(s_j)) \ \leftarrow \frac{1}{3}\left[\begin{array}{c}(\mu(\chi'(s_x),\mu(s_x),S(s_x))-\chi'(s_x))^2+\\(\mu(\chi'(s_x),\mu(s_x),S(s_x))-\mu(s_x))^2+\\(\mu(\chi'(s_x),\mu(s_x),S(s_x))-S(s_x))^2\end{array}\right]$;
      $lfv(s_j) \ \leftarrow [S^2(\chi'(s_x), \mu(s_x), S(s_x)) + 1]^{-1}$;
    end for
    Sort $(lfv(s_j))$; ▷ Sort the services in ascending order
  end for
  best_rank (select_ services); ▷ Select best services based on their average rank of QoS parameters.
end procedure

3.3.1.2 Evaluation of associability fitness of the composition

Let a set of possible compositions C be $\{c_1, c_2, c_3, \ldots, c_n\}$. The connection associability score (cas) of composition c_m indicates the number of connections having associability (a connection formed between services of the same provider or providers that have a service level agreement) against the number of connections requiring associability (see section 3.3.1). The associability fitness value $afv(c_m)$ is measured as follows:

$$afv(c_m) = \frac{1}{1 + \mu(cas(c_m), AC)} \tag{3.22}$$

where $\mu(cas(c_i), AC)$ is the mean of $cas(c_m)$ and AC. AC is the associability count, representing the total number of edges between tasks that require associability in the target application of the service composition. $afv(c_m)$ is the associability fitness of composition c_m, which is measured by normalizing the standard deviation observed from $cas(c_m)$ and AC to $0 \leq afv(c_m) \leq 1$. To avoid a divide-by-zero error, the resultant standard deviation is increased by 1.

3.3.1.3 Evaluation of fitness of the composition

The overall composition fitness of a service composition is evaluated as follows:

The mean of the fitness values of the services involved in a composition is measured as follows:

$$\mu(c_m) = \frac{1}{|c_m|}\sum_{k=1}^{|c_m|}\{sfv(s_k) \in c_k\} \tag{3.23}$$

sfv is also referred to as lfv.

The standard deviation of the fitness values of the services involved in a composition is measured as follows:

$$S(c_m) = \sqrt{\frac{1}{|c_m|}\left[\sum_{k=1}^{|c_m|}(\mu(c_m) - \{sfv(s_k) \in c_m\})^2\right]} \tag{3.24}$$

The skewness of the fitness values of the services involved in a composition is measured as follows:

$$\chi(c_m) = \frac{1}{|c_m|}\frac{\sum_{k=1}^{|c_m|}(\mu(c_m) - \{sfv(s_k) \in c_m\})^3}{(S(c_m))^3} \tag{3.25}$$

The absolute value of the skewness is considered, as follows:

$$\chi'(c_m) = |\chi(c_m)| \tag{3.26}$$

The variance of the standard deviation, mean, and skewness of the fitness values of the services involved in a composition is measured as follows:

$$S^2(\chi(c_m), \mu(c_m), S(c_m)) = \frac{1}{3}\left[\begin{array}{c}(\mu(\chi'(c_m), \mu(c_m), S(c_m)) - \chi'(c_m))^2 + \\ (\mu(\chi'(c_m), \mu(c_m), S(c_m)) - \mu(c_m))^2 + \\ (\mu(\chi'(c_m), \mu(c_m), S(c_m)) - S(c_m))^2\end{array}\right] \tag{3.27}$$

The composition fitness value $cfv(c_m)$ is measured as follows:

$$cfv(c_m) = \left[1 + S^2(\chi'(c_m), \mu(c_m), S(c_m))\right]^{-1} \tag{3.28}$$

In equation 3.28, S^2 is incremented by 1 to avoid a divide-by-zero error. The pseudocode for fitness of the composition is given as Algorithm 3.8.

Algorithm 3.8 Pseudocode for composition fitness of services

procedure COMPOSITION FITNESS VALUE(C_j)
Input: a set that contains ranks for each service of a composition
Output: Best Composition or optimal composition
Steps applied to all possible tasks to a composition:
  $Generate\_composition(set\ of\ tasks)$; ▷ Generate possible composition
  for each $task(T_i)$ **do**
    $selected\_services \leftarrow lfv(s_i)$;
  end for
  for each composition $(c_m, \forall i \in 0, 1, 2, \ldots, i)$ **do**
    $\mu(c_m) \leftarrow \frac{1}{|c_m|}\sum_{k=1}^{|c_m|}\{sfv(s_k) \in c_k\}$; ▷ Mean of fitness values of services
    $S(c_m) \leftarrow \sqrt{\frac{1}{|c_m|}\left[\sum_{k=1}^{|c_m|}(\mu(c_m) - \{sfv(s_k) \in c_m\})^2\right]}$; ▷ Standard deviation of fitness values of services
    $\chi(c_m) \leftarrow \frac{1}{|c_m|}\frac{\sum_{k=1}^{|c_m|}(\mu(c_m)-\{sfv(s_k)\in c_m\})^3}{(S(c_m))^3}$; ▷ Skewness of fitness values of services
    $\chi'(c_m) \leftarrow |\chi(c_m)|$;
    if $(\chi'(c_m) \geq Positive)$ **then**
      $m_c \leftarrow \chi(c_m)$;
    else
      $\chi'(c_m) \leftarrow |\chi(c_m)|$;
    end if
    $S^2(\chi'(c_m), \mu(c_m), S(c_m)) \leftarrow \frac{1}{3}\begin{bmatrix}(\mu(\chi'(c_m),\mu(c_m),S(c_m))-\chi'(c_m))^2+\\(\mu(\chi'(c_m),\mu(c_m),S(c_m))-\mu(c_m))^2+\\(\mu(\chi'(c_m),\mu(c_m),S(c_m))-S(c_m))^2\end{bmatrix}$;
    $cfv(c_m) \leftarrow \left[1 + S^2(\chi'(c_m), \mu(c_m), S(c_i))\right]^{-1}$;
  end for
  Sort $(cfv(c_m))$; ▷ Sort the compositions Ascending Order
  best_rank (select_ services); ▷ select best compositions based on their average rank of QoS parameters.
end procedure

3.3.1.4 Sorting composition and associability fitness values

The resultant compositions are sorted by their composition fitness value (cfv) in descending order, and the top n of the best compositions $mbsc$ are selected from the resultant ordered compositions based on their "maximum best service compositions ratio" (the $mbscr$). Then, $mbsc$ is sorted in descending order of associability fitness afv, and the "maximum best associability compositions ratio" (the $mbacr$) is identified from the ordered $mbsc$.

3.3.2 Optimal Fitness-Aware Service Composition using MIWO (OFASC-MIWO)

The architecture of OFASC using the Modified Invasive Weed Optimization algorithm is shown in Fig. 3.7. It consists of three phases: preprocessing, optimization, and test. The inputs for OFASC using modified IWO are as follows:

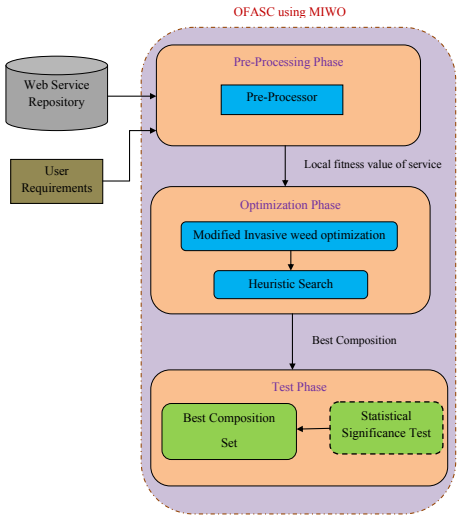


Figure 3.7: Architecture of OFASC using modified invasive weed optimization

- A set of n tasks $CT = \{t_1, t_2, t_3, \ldots, t_n\}$ involved in the given application
- $a_i = \{s_1, \ldots, s_{x_i}\}$, consisting of the collection of the x_i services that can address task $t_i \in CT$, and $A = \{a_1, \ldots, a_n\}$
- Set of all possible compositions: $C = \{c_1, c_2, c_3, \ldots, c_{|c|}\}$
- Associability count AC observed
- A set of task sets demanding associability $ST' = \{\{ST_i, ST_j, ST_k, \ldots\}, \{ST_x, ST_y, ST_z, \ldots\}, \ldots\}$
- Maximum evolution iteration threshold, met

1. **Preprocessing Phase**: This phase accepts the services from a web services repository and the user requirements as inputs. Services are ranked based on the descending order of normalized values. The local fitness values of each service are calculated using Algorithm 3.7.
2. **Optimization Phase**: In this phase, our proposed Modified Invasive Weed Optimization accepts the local fitness value of individual services and returns the best composite service that has the best global fitness value. Further, the best solution obtained by our Modified Invasive Weed Optimization (MIWO) algorithm is improved by a heuristic search. A detailed description of the optimization phase is as follows:

(a) **Modified Invasive Weed Optimization (MIWO):**

Invasive Weed Optimization (IWO) [198] is a meta-heuristic algorithm inspired by the colonization of invasive weeds. Feasible solutions are represented by weeds, and a set of weeds is considered as the population. A finite number of weeds are spread over the search environment. Every weed generates new seeds according to its fitness value. The weeds produced are randomly dispersed over search space by normally distributed random numbers having a mean value equal to zero. This procedure is continued until the limit on the maximum number of weeds is achieved, and only the weeds with better fitness can survive and generate seeds, while others are eliminated. This procedure is repeated until the maximum number of iterations is reached or the solution nearest to the optimal solution is found. In IWO, selection of the best solutions is based on competitive exclusion. The standard deviation of the fitness values of the weeds is calculated. Based on the standard deviation values, the inclusion of the weeds is determined. To avoid the overhead of standard deviation calculations and premature convergence, we propose a modified IWO algorithm as follows:

Step 1: Initial solution

The initial population is generated randomly (based on Algorithm 3.7) and dispersed over the D-dimensional search space with random positions. In our approach, the initial solutions represent the services.

Step 2: Reproduction

The number of seed solutions produced by a single weed depends on its own fitness along with the highest fitness and the lowest fitness in the population. The greater the fitness value, more seeds will be produced, according to the following formula:

$$weed_n = \frac{f - f_{min}}{f_{max} - f_{min}}(s_{max} - s_{min}) + s_{min} \tag{3.29}$$

where f_{min} and f_{max} represent the minimum and the maximum fitness of the population, respectively, f represents the fitness of the current weed, and s_{min} and s_{max} represent the minimum and maximum values, respectively, of seeds that can be produced.

Step 3: Spatial dispersion

The seeds generated in step 2 are dispersed spatially in the search area randomly with the distance between the seed and the parent plant distributed normally with zero mean and variable variance. This ensures that the seeds are distributed close to the parent plant. However, the standard deviation is updated to the final values from the previously defined initial values in the generation. A nonlinear version of the standard deviation is given by

$$\sigma_{curr} = \frac{(iter_{max} - iter)^{\nu}}{(iter_{max})^{\nu}}(\sigma_{init} - \sigma_{final}) + \sigma_{final} \tag{3.30}$$

where $iter_{max}$ is the maximum number of iterations, ν is the nonlinear modulation index, and σ_{curr} is the standard deviation at the current step.

Step 4: Final selection

The solutions are selected based on a threshold value that represents the weed fitness. The solutions that satisfy the threshold limit are selected, and the remaining solutions are eliminated. By this process, the solutions selected are confined to those having the best values, and the chance of premature convergence is reduced.

(b) **Heuristic search:**

The best solution obtained by the modified IWO algorithm is further improved by applying a heuristic search. In the heuristic search, the seeds having the best fitness are selected and added to the composition. The process is repeated until all seeds with the best fitness are added for all possible compositions in a set C. Next, we calculate the fitness of the compositions and measure the associability fitness of the compositions in set C (based on sections 3.3.1.3 and 3.3.1.2). Then, we sort the composition fitness values and associability fitness values in descending order (based on section 3.3.1.4) and evaluate the kurtosis [199] of the compositions as follows:

We find the mean of the cfv of all compositions in set C as follows:

$$\mu(cfv) = \frac{\sum_{m=1}^{|C|} cfv(c_m)}{|C|} \tag{3.31}$$

We find the standard deviation of the cfv distribution across the compositions in C:

$$\sigma(cfv) = \sqrt{\frac{\sum_{m=1}^{|C|} (cfv(c_m) - \mu(cfv))^2}{|C|}} \tag{3.32}$$

We find the fourth moment of the cfv distribution as follows:

$$m^4 = \frac{\sum_{m=1}^{|C|} (cfv(c_m) - \mu(qfs))^4}{|C|} \tag{3.33}$$

The kurtosis of the cfv distribution over the compositions in C is as follows:

$$K(C) = \frac{(m)^4}{S_{cfv}} \tag{3.34}$$

where S_{cfv} represents the standard deviation of the composition fitness value. We verify the kurtosis of the cfv distribution amidst the resultant compositions. If the distribution is leptokurtic, then we proceed with the evolution for the next maximum count given. If the distribution is mesokurtic or platykurtic, then we stop the evolution and return the best composition found so far based on the calculated fitness.

3. **Test Phase**: In this phase, the best fitness values of our proposed approach are tested for statistical significance. The pseudocode for the OFASC algorithm using modified IWO is given as Algorithm 3.9.

Algorithm 3.9 Pseudocode for OFASC using modified IWO algorithm

procedure MIWO(Composition)
  Input: Set the parameters for IWO and Services
  Output: Best Composition or optimal composition
  Generate initial weed solutions randomly based on Algorithm 3.7;
  while $Composition_{num} \leq populations$ **do**
    for each weed **do**
      Calculate $number\ of\ seeds$;
      $Weed_n = \frac{f-f_{min}}{f_{max}-f_{min}}(s_{max} - s_{min}) + s_{min}$;
      Upadte σ
      $\sigma_{curr} = \frac{(iter_{max}-iter)^{\nu}}{(iter_{max})^{\nu}}(\sigma_{init} - \sigma_{final}) + \sigma_{final}$;
      Generate seeds over the search space by $N(0, \sigma_{curr})$;
      for each seed **do**
        Calculate fitness by using equation 3.21 ;
        if seed $\leq$ threshold **then**
          select (best _seed);
        end if
      end for
      Sort(seeds);
      ac = seeds; ▷ seeds add to composition;
    end for
    $C_i = ac$;
    while ! best Composition **do**
      Calculate cfv of C_i based on Algorithm 3.8;
      Calculate associability fitness by using equation 3.22;
      kc = find the kurtosis of cfv distribution ; ▷ find the kurtosis of the fitness distribution beyond the evolutions
      if $kc = 3 \&\& > 1$ **then**
        $met = met * 0.5$; ▷ proceed evolutions for further met times
        $ec = 0$; ▷ mesokurtic
      else if $kc > 3$ **then**
        $ec = 0$ ▷ proceed evolutions for further met times ▷ leptokurtic
      else if $kc \leq 3$ **then**
        $best\_composition \leftarrow true$;
      end if
    end while ▷ platykurtic
    Sort(best_ coomposition); ▷ see section 3.4
  end while
end procedure

3.3.3 Performance Evaluation

Our proposed approach was implemented in Java on a personal computer with an Intel i7 processor consisting of 16 GB RAM and 1 TB memory. We used the QWS data set V2 [182]. We tested the QWS data set V2 using the Shapiro–Wilk test [183], and the results revealed that the data set is normally distributed. Ta-

ble 3.6 presents the list of approaches (with name, abbreviations, and parameter settings) that we compared with our proposed approach.

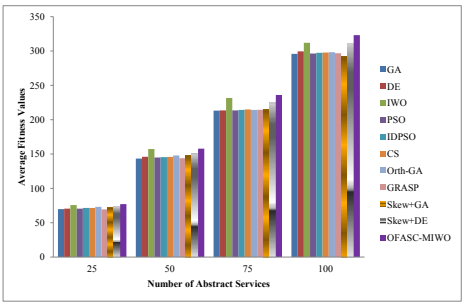
Table 3.6: Compared approaches with parameter settings

Approach	Abbrevation	Parameter Setting
Genetic Algorithm [33]	GA	Crossover=0.8, mutation=0.001 and random selection approach is applied
Differential Evolution [200]	DE	Scale factor is 0.4 and crossover probability is 0.7
Invasive Weed Optimization [198]	IWO	$\sigma_{initial}$=10, σ_{final}=0.5, S_{Max}=15, $S_{Min} = 1$, $P_{Max} = 15$, and n=4
Particle Swarm Optimization [201]	PSO	$\omega = 0.72984$, $\phi_1 = 1.4962$ and $\phi_2 = 1.4962$
Improved Discrete Immune Optimization [132]	IDPSO	Mutation/ hypermuation are 0.1 and 0.01
Cuckoo Search [202]	CS	$P_a = 0.25$ and $\alpha = 1$
Orthogonal Genetic Algorithm [123]	Orth-GA	Crossover P_c=0.1, mutation P_m=0.02 and F = 4
Greedy Randomized Adaptive Search [19]	GRASP	$\alpha = 0.25$, elite solution is 5, N_{paths}=2, and N_{steps}=50
Skewness based Genetic Algorithm	Skew + GA	Crossover=0.8, mutation=0.001 and random selection approach is applied
Skewness based Differential Evolution	Skew + DE	Scale factor is 0.4 and crossover probability is 0.7
Optimal Fitness Aware Service Composition using Modified IWO	OFASC	$\sigma_{initial}$=10, σ_{final}=0.5, S_{Max}=15,$S_{Min} = 1$, $P_{Max} = 15$, and n=4

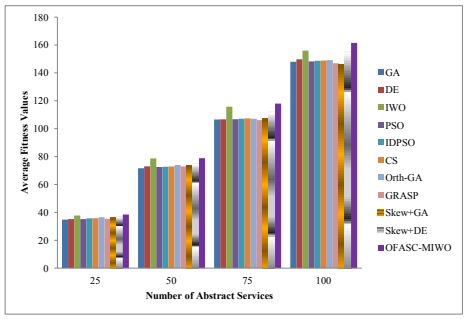
We evaluated the performance of our proposed algorithm in the following ways:

- Evaluated average fitness values by varying the numbers of abstract services and candidate services.
- Evaluated execution time by varying the number of abstract services.

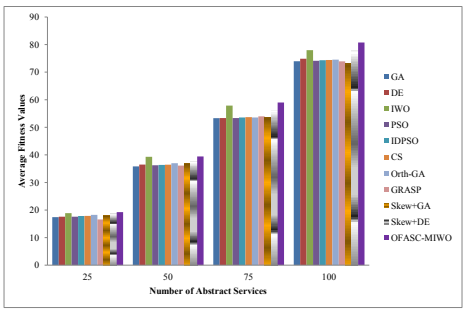
In Figs. 3.8a to 3.8d, we illustrate the results of our OFASC-MIWO algorithm, with 25, 50, 75, and 100 abstract services, and 100, 250, 500, and 1000 candidate services for each abstract service. As observed in Figs. 3.8a to 3.8d, the average fitness values increased as the number of abstract services increased. In our proposed method, this increase is exponential, whereas the increase is linear in other methods. Thus, our proposed method has performance superior to that of the other methods compared. We executed our proposed algorithm 30 times, and the average is noted because of the stochastic nature of algorithms.



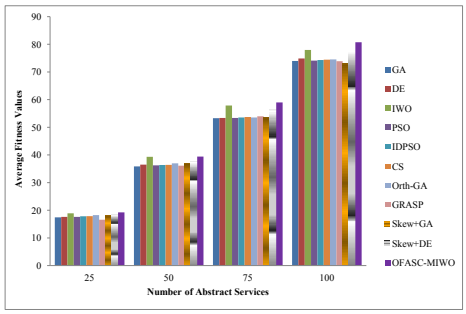
(a) Average Fitness Values for 100 candidate services



(b) Average Fitness Values for 250 candidate services



(c) Average Fitness Values for 500 candidate services



(d) Average Fitness Values for 1000 candidate services

Figure 3.8: Results of OFASC-MIWO (Average fitness values for different abstract and candidate services)

Figure 3.8d illustrates a scenario wherein a composite service comprises 25, 50, 75, and 100 abstract services and each abstract service has 1000 candidate services. From Fig. 3.8d, the best average fitness values obtained by our method for 100 abstract services was 645.90, whereas the best solutions obtained by GA, DE, IWO, PSO, IDPSO, CS, Orth-GA, GRASP, Skew+GA, and Skew+DE were 591.70, 598.87, 623.67, 592.74, 594.56, 595.36, 596.35, 591.56, 584.50, and 621.68, respectively.

Table 3.7 presents the execution times of different methods for 25, 50, 75, and 100 abstract services with respect to 1000 candidate services. From Table 3.7, the execution time for GA, DE, IWO, PSO, IDPSO, CS, Orth-GA, GRASP,

Skew+GA, and Skew+DE to evaluate the best solutions were 56.35, 30.56, 28.63, 30.12, 29.14, 38.56, 31.24, 37.12, 53.73, and 27.40 s, respectively, and the execution time for OFASC-MIWO was 26.33 s. From Table 3.7, we observe that our proposed method is more efficient than the other methods compared. Hence, our proposed method gives satisfactory results in less time and converges very quickly than the other methods.

Table 3.7: Execution time (in seconds) analysis for varying abstract services

No of Services	**GA**	**DE**	**IWO**	**PSO**	**IDPSO**	**CS**	**Ortho-GA**	**GRASP**	**Skew+GA**	**Skew+DE**	**OFASC-MIWO**
25	56.35	30.56	25.63	28.12	26.14	24.56	25.24	27.12	22.45	28.51	**12.93**
50	56.58	31.12	20.45	34.59	34.56	32.15	26.24	28.48	33	29.57	**18.13**
75	56.35	33.56	20.59	48.12	36.14	34.56	27.58	29.19	49.16	31.13	**23.03**
100	56.35	30.56	28.63	30.12	29.14	38.56	31.24	37.12	53.73	27.4	**26.33**

We performed statistical tests (parametric and non-parametric) for our proposed method and for other contemporary methods [193, 194, 195]. Tables 3.8, 3.9, 3.10, and 3.11 present the results of the T-test and the Wilcoxon signed-rank test for all algorithms. Based on Table 3.8 and Table 3.9, the results obtained are statistically significant (all T-values are positive, and the P-values are near to zero). Based on Table 3.10 and Table 3.11, the procured results are statistically significant (all Z-values are obtained by positive ranks, and the P-values are near to zero). Therefore, our proposed method is significantly more accurate than the other methods compared.

Table 3.8: Statistical analysis results for T-Test (T-Value/ P-Value)

No of Services	Approaches	GA	DE	IWO	PSO	IDPSO	CS	OGA	GRASP	Skew+GA	Skew+DE	OFASC-MIWO
25	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	69.09166 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	160.7873 (0)	145.6384 (0)	*	*	*	*	*	*	*	*	*
	PSO	73.77837 (0)	174.9666 (0)	146.507 (0)	*	*	*	*	*	*	*	*
	IDPSO	137.1648 (0)	110.3008 (0)	115.0562 (0)	122.999 (0)	*	*	*	*	*	*	*
	CS	113.8882 (0)	81.70712 (0)	111.0976 (0)	86.62438 (0)	1.919635 (0.065147)	*	*	*	*	*	*
	OGA	254.9185 (0)	269.7301 (0)	74.19993 (0)	298.3244 (0)	126.5231 (0)	99.389 (0)	*	*	*	*	*
	GRASP	182.6289 (0)	165.6198 (0)	88.13003 (0)	176.5646 (0)	67.03851 (0)	54.70652 (0)	41.19086 (0)	*	*	*	*
	Skew+GA	109.9041 (0)	82.43772 (0)	94.24563 (0)	85.00513 (0)	24.13003 (0)	20.63394 (0)	52.79554 (0)	22.02561 (0)	*	*	*
	Skew+DE	374.6015 (0)	439.665 (0)	36.95023 (0)	490.9214 (0)	251.3195 (0)	197.1351 (0)	120.1361 (0)	144.4416 (0)	125.5379 (0)	*	*
	OFASC-MIWO	**741.0905 (0)**	**181.2563 (0)**	**36.21212 (0)**	**164.6653 (0)**	**632.437 (0)**	**441.6841 (0)**	**454.9104 (0)**	**404.8808 (0)**	**284.3365 (0)**	**313.94 (0)**	*
50	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	95.55953 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	326.8445 (0)	339.5257 (0)	*	*	*	*	*	*	*	*	*
	PSO	56.51095 (0)	96.88349 (0)	355.2403 (0)	*	*	*	*	*	*	*	*
	IDPSO	70.80686 (0)	30.03905 (0)	326.3595 (0)	31.42531 (0)	*	*	*	*	*	*	*
	CS	78.96525 (0)	1.374147 (0.180299)	301.533 (0)	46.14436 (0)	17.7624 (0)	*	*	*	*	*	*
	OGA	166.3122 (0)	247.9359 (0)	272.3268 (0)	244.826 (0)	155.0781 (0)	106.6511 (0)	*	*	*	*	*
	GRASP	140.4159 (0)	130.7333 (0)	281.6071 (0)	167.889 (0)	107.593 (0)	72.05641 (0)	41.81392 (0)	*	*	*	*
	Skew+GA	114.3769 (0)	66.43753 (0)	217.2155 (0)	95.2941 (0)	73.4064 (0)	57.24629 (0)	5.3779 (9.88E-06)	13.37095 (0)	*	*	*
	Skew+DE	245.7506 (0)	380.5931 (0)	192.8975 (0)	367.2273 (0)	274.903 (0)	215.8088 (0)	179.6044 (0)	192.2375 (0)	87.56003 (0)	*	*
	OFASC-MIWO	**430.1029 (0)**	**564.0742 (0)**	**15.60743 (0)**	**560.357 (0)**	**492.0656 (0)**	**436.2212 (0)**	**442.9875 (0)**	**442.407 (0)**	**287.1479 (0)**	**308.8038 (0)**	*

Table 3.9: Statistical analysis results for T-Test (T-Value/ P-Value)

No of Services	Approaches	GA	DE	IWO	PSO	IDPSO	CS	OGA	GRASP	Skew+GA	Skew+DE	OFASC-MIWO
75	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	2.950872 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	248.8589 (0)	252.5819 (0)	*	*	*	*	*	*	*	*	*
	PSO	1.269516 (,0.21471)	4.724156 (5.9E-05)	279.0176 (0)	*	*	*	*	*	*	*	*
	IDPSO	14.85753 (0)	12.23742 (0)	247.3377 (0)	18.65519 (0)	*	*	*	*	*	*	*
	CS	21.21525 (0)	18.88339 (0)	233.198 (0)	25.66934 (0)	7.330608 (6E-08)	*	*	*	*	*	*
	OGA	18.24766 (0)	15.70392 (0)	248.6577 (0)	22.79212 (0)	3.301773 (0.002629)	4.328447 (0.000173)	*	*	*	*	*
	GRASP	26.5988 (0)	24.1995 (0)	266.8873 (0)	34.4098 (0)	10.4835 (0)	1.4728 (0.151947)	6.9333 (1.5E-07)	*	*	*	*
	Skew+GA	21.305 (0)	18.4001 (0)	305.2334 (0)	30.8355 (0)	2.0070 (0.054487)	7.9225 (1E-08)	2.5683 (0.0158)	14.83051 (0)	*	*	*
	Skew+DE	250.2066 (0)	261.1085 (0)	104.2113 (0)	336.203 (0)	258.2109 (0)	229.3898 (0)	264.9495 (0)	344.1856 (0)	750.9648 (0)	*	*
	OFASC-MIWO	**463.8631 (0)**	**486.9096 (0)**	**84.8764 (0)**	**615.3769 (0)**	**495.7862 (0)**	**450.5001 (0)**	**512.2861 (0)**	**666.7984 (0)**	**457.1056 (0)**	**576.9665 (0)**	*
100	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	41.6016 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	187.2811 (0)	155.8811 (0)	*	*	*	*	*	*	*	*	*
	PSO	9.8878 (0)	41.0090 (0)	225.0564 (0)	*	*	*	*	*	*	*	*
	IDPSO	22.2526 (0)	26.5534 (0)	206.4705 (0)	16.8159 (0)	*	*	*	*	*	*	*
	CS	27.9015 (0)	19.2664 (0)	194.3318 (0)	24.2806 (0)	7.8647 (0)	*	*	*	*	*	*
	OGA	33.6702 (0)	12.6926 (0)	186.3392 (0)	31.8467 (0)	15.4020 (0)	7.3715 (0)	*	*	*	*	*
	GRASP	6.1047 (0.00000138)	55.0041 (0)	279.1005 (0)	8.5075 (0)	30.4377 (0)	39.0370 (0)	48.4507 (0)	*	*	*	*
	Skew+GA	41.3623 (0)	94.8980 (0)	272.7903 (0)	67.4440 (0)	81.63027 (0)	86.6608 (0)	93.3102 (0)	79.9486 (0)	*	*	*
	Skew+DE	188.5550 (0)	155.8574 (0)	13.2633 (0)	236.2179 (0)	214.6742 (0)	200.3379 (0)	191.2971 (0)	312.4202 (0)	287.3142 (0)	*	*
	OFASC-MIWO	**332.7873 (0)**	**312.8212 (0)**	**151.2976 (0)**	**422.1707 (0)**	**396.9279 (0)**	**377.5900 (0)**	**367.3463 (0)**	**543.3601 (0)**	**463.1837 (0)**	**180.9734 (0)**	*

Table 3.10: Statistical analysis results for Wilcoxon signed-rank Test (Z-Value/ P-Value)

No of Services	Approaches	GA	DE	IWO	PSO	IDPSO	CS	OGA	GRASP	Skew+GA	Skew+DE	OFASC-MIWO
25	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-4.7821 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*	*
	PSO	-4.7821 (0)	-1.6352 (0.101)	-4.7821 (0)	*	*	*	*	*	*	*	*
	IDPSO	-4.7821 (0)	-4.7821(0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*
	CS	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-1.7586 (0.0784)	*	*	*	*	*	*
	OGA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*
	GRASP	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*
	Skew+GA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*
	Skew+DE	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	OFASC-MIWO	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	*
50	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-4.7821 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*	*
	PSO	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*
	IDPSO	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*
	CS	-4.7821 (0)	-1.6352 (0.101)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*
	OGA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*
	GRASP	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*
	Skew+GA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-3.3835 (0.00072)	-4.7821 (0)	*	*	*
	Skew+DE	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	OFASC-MIWO	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	*

Table 3.11: Statistical analysis results for Wilcoxon signed-rank Test (Z-Value/ P-Value)

No of Services	Approaches	GA	DE	IWO	PSO	IDPSO	CS	OGA	GRASP	Skew+GA	Skew+DE	OFASC-MIWO
75	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-2.4373 (0.01468)	*	*	*	*	*	*	*	*	*	*
	IWO	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*	*
	PSO	-1.0387 (0.29834)	-3.98 (6E-05)	-4.7821 (0)	*	*	*	*	*	*	*	*
	IDPSO	-4.7821 (0)	-4.7821(0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*
	CS	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.2474 (0)	*	*	*	*	*	*
	OGA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-2.3756 (0.01732)	-3.5686 (0.00036)		*	*	*	*
	GRASP	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-0.4834 (0.6312)	-3.9199 (8E-05)	*	*	*	*
	Skew+GA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-0.9359 (0.3472)	-4.7821 (0)	-1.1415 (0.2542)	-4.6999 (0)	*	*	*
	Skew+DE	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	OFASC-MIWO	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*
100	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-4.7821 (0)	*	*	*	*	*	*	*	*	*	*
	IWO	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*	*
	PSO	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*
	IDPSO	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*
	CS	-4.7821 (0)	-1.6352 (0.101)	-4.7821 (0)	-4.7821 (0)	-4.5765 (0)	*	*	*	*	*	*
	OGA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.597 (0)	*	*	*	*	*
	GRASP	-4.453 (0)	-4.7821 (0)	-4.7821 (0)	-4.2474 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*	*
	Skew+GA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*
	Skew+DE	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	OFASC-MIWO	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	*

3.3.3.1 Comparison of results of OFASC-MIWO and OFASC-AGEGA on a synthetic data set

We performed experiments for OFASC-MIWO on the same synthetic data set that we used for analyzing the performance of OFASC-AGEGA in section 3.3.3. We evaluated the performance of our proposed approach in the following ways:

- Evaluated the average fitness values by varying the number of compositions.
- Evaluated the process completion time and computational complexity by varying the number of compositions.

Table 3.12: Comparison of average fitness values

		Mean of the fitness values											
Test Cases.	Number of Abstract Services	GA	DE	PSO	IDPSO	AGP	OGA	GRASP	IWO	Skew+GA	Skew+DE	OFASC-AGEGA	OFASC-IWO
65	10	0.327	0.3364	0.3386	0.343	0.3441	0.3875	0.3907	0.3981	0.4181	0.5132	0.4138	**0.7632**
54		0.3166	0.3156	0.3159	0.3071	0.3229	0.3357	0.3587	0.3671	0.4299	0.5168	0.4299	**0.7816**
49	30	0.2681	0.3083	0.3348	0.2369	0.2912	0.3132	0.3638	0.3781	0.4108	0.5296	0.4108	**0.7912**
61		0.3297	0.2901	0.3451	0.2053	0.2874	0.3128	0.3459	0.3613	0.4584	0.5316	0.4584	**0.7886**
65	50	0.2791	0.3049	0.3150	0.3083	0.3181	0.3351	0.3781	0.3816	0.4330	0.5392	0.4330	**0.7916**
54		0.3138	0.3174	0.329	0.3138	0.3218	0.3315	0.3932	0.3998	0.4603	0.5106	0.4603	**0.8032**
49	70	0.2873	0.2870	0.3343	0.3239	0.3189	0.3437	0.3813	0.4218	0.4679	0.5232	0.4679	**0.8103**
61		0.3027	0.2907	0.3234	0.3358	0.3298	0.3473	0.3856	0.4313	0.4897	0.4901	0.4897	**0.8196**
65	90	0.3071	0.3116	0.3435	0.3268	0.3014	0.4757	0.4613	0.4698	0.4983	0.5416	0.4983	**0.8816**
54		0.327	0.3261	0.3333	0.323	0.3241	0.5075	0.4896	0.4918	0.5138	0.5686	0.5138	**0.8898**
49	100	0.3235	0.4061	0.3781	0.3643	0.3591	0.5248	0.4916	0.5213	0.5456	0.5816	0.5456	**0.9613**
65		0.327	0.4261	0.3833	0.3834	0.3739	0.5784	0.5313	0.5486	0.6238	0.6817	0.6238	**0.9802**

Table 3.12 illustrates the average fitness values of compositions for different test cases with 10, 30, 50, 70, 90, and 100 abstract services. Based on Table 3.12, we notice that our proposed OFASC-MIWO approach produces reasonably better results than other approaches, including OFASC-AGEGA. Figure 3.9 shows the completion times for our proposed approaches and other compared approaches for 100 compositions. Based on the experiments, we notice that our proposed approaches are scalable and robust compared to other contemporary approaches. The evolution completion times obtained by GA, DE, PSO, IDPSO, AGP, OGA, GRASP, IWO, Skew+GA, Skew+DE, and OFASC-AGEGA were 31.7559, 22.1898, 21.0244, 15.992, 9.5796, 8.9864, 8.664, 8.264,

7.98264, 6.98264, and 4.8086 s, respectively, whereas the completion time for OFASC-MIWO was 3.1086 s. Hence, our proposed OFASC-MIWO method gives satisfactory results in less time and converges very quickly than the other methods.

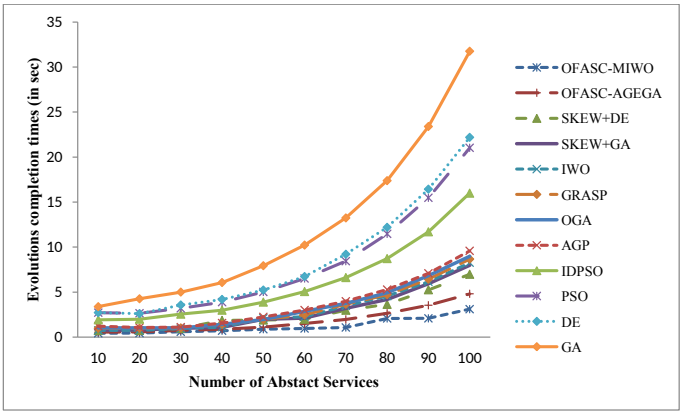


Figure 3.9: Completion time observed for OFASC-MIWO and other algorithms.

Figure 3.10 illustrates the computational complexity observed for evolution iterations under QoS-aware service composition approaches compared with our proposed approach. From Fig. 3.10, we observe that our proposed OFASC-MIWO approach takes much less computational time than other approaches for 100 compositions. The computational times for GA, DE, PSO, IDPSO, AGP, OGA, GRASP, IWO, Skew+GA, Skew+DE, and OFASC-AGEGA to obtain the best solutions were 317.559, 221.898, 210.244, 159.920, 95.796, 81.640, 81.234, 79.831, 77.345, 75.315, and 48.086 s, respectively, and the computational time for OFASC-MIWO was 40.356 s. In GA, AGP, and Skew+GA, with each iteration, a new population is generated, and the individual fitnesses are assessed (based on fixed and predetermined crossover and unguided mutation). These processes cause GA and AGP to converge slowly and to become easily stuck in local maxima. It takes more time because of the higher number of parameter turnings and the single-point crossover. In DE and Skew+DE, the new population is generated based on a mutation-based distribution of the solutions. If either the population is increased or the number of solutions computed for the mutation values is increased, the diversity of possible movements will also increase, promoting exploration of the search space. This algorithm reaches an

unstable convergence in the last period; it is easy for it to drop into local optima. Similarly, in the OGA algorithm, the orthogonal method is used to produce the new population. In this algorithm, the crossover strategy is employed on the two parents and the orthogonal method is applied to produce the new offspring. This whole process consumes more time. In PSO and IDPSO, with each iteration, the new population is generated, and the particle fitness is assessed (based on best visited position and best visited velocity of all the particles) and changes with each iteration. Therefore, the probability of being trapped in local optima is increased. This change decreases the convergence rate and increases the search space, which in turn consumes more time compared with our proposed algorithm.

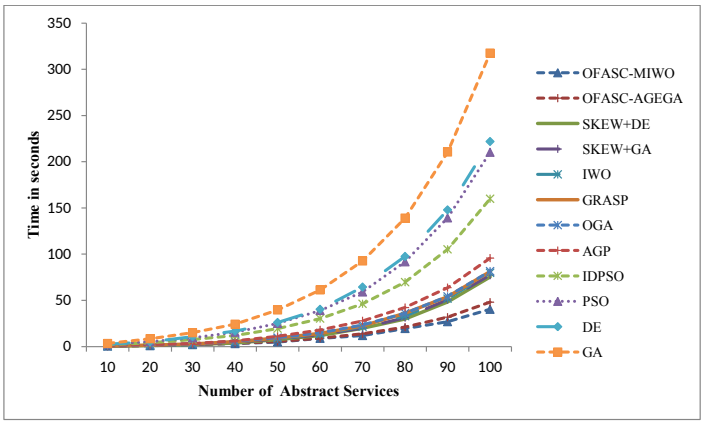


Figure 3.10: Computational complexity observed for OFASC-MIWO and other algorithms.

In our proposed OFASC-AGEGA approach, we use an adaptive genotype progressive evolution strategy to restrict the evolution iterations to a minimal number. The progressive evolution strategy considers those offspring in the new generation that have better fitness than any of their parents. We evaluate the service fitness and composition fitness by using DURD and DUSRD, which helps to prune the services and compositions having non-optimal solutions and selects only the best compositions, thereby gradually shrinking the valid search space. However, because of DURD and DUSRD, it still may have some intrinsic defects, including low premature convergence rate and an increase of search space within local optima due to an exponential increase in Pareto front size, which in

turn is due to the high number of candidate services in the composition. In our proposed OFASC-MIWO method, we evaluate the service fitness and composition fitness using a skewness-based approach and modified IWO, which helps to prune the services and compositions having non-optimal solutions and selects only the best compositions, thereby gradually shrinking the valid search space. Hence, it consumes less time than other methods.

The computational complexity of our proposed OFASC-MIWO method is linear. As observed from Figs. 3.9 and 3.10, our proposed approach is more effective, scalable, and robust than the other approaches. The time complexity of OFASC-AGEGA and OFASC-MIWO is $O(n)$, the observed complexity of Skew+GA, Skew+DE, OGA, AGP, and IWO is $O(n \log n)$, and that of the other models is $O(n^2)$.

Table 3.13: Statistical analysis results for T-Test (T-Value/ P-Value) of OFASC-MIWO and other approaches

No of compositions	Approaches	GA	DE	IWO	PSO	IDPSO	OGA	GRASP	Skew+GA	Skew+DE	OFASC-AGEGA	OFASC-MIWO
100	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-3.3013 (0.3803)	*	*	*	*	*	*	*	*	*	*
	IWO	0.1807 (0.42913)	0.1807 (0.42913)	*	*	*	*	*	*	*	*	*
	PSO	0.11201 (0.04559)	0.11201 (0.04559)	-0.2186 (0.4181)	*	*	*	*	*	*	*	*
	IDPSO	-1.6894 (0.05263)	-1.6894 (0.0005499)	-1.6894 (0.0526)	-1.8132 (0.00519)	*	*	*	*	*	*	*
	OGA	1.50505 (0.07327)	1.69982 (0.05163)	1.53129 (0.06998)	2.77954 (0.00547)	1.62635 (0.059058)	*	*	*	*	*	*
	GRASP	0.1807 (0.42913)	0.1807 (0.42913)	0.4502 (0.42973)	0.2813 (0.38138)	1.8613 (0.005313)	0.1876 (0.4138)	*	*	*	*	*
	Skew+GA	4.86321 (0.0001)	4.8635 (0.00039)	3.86321 (0.00015)	3.86321 (0.00015)	2.77956 (0.000547)	1.53129 (0.06998)	1.1807 (0.05412)	*	*	*	*
	Skew+DE	5.2151 (0.0001)	5.2156 (0.0003)	3.8121 (0.00039)	4.00513 (0.00015)	2.87613 (0.005321)	1.96321 (0.003268)	2.77954 (0.00547)	1.86321 (0.0001)	*	*	*
	OFASC-AGEGA	4.5425 (0.00008)	1.85282 (0.03869)	1.96321 (0.03268)	4.42924 (0.00015)	5.96342 (0.00001)	4.83606 (0.000039)	4.83601 (0.000039)	1.86321 (0.00001)	2.77954 (0.00546)	*	*
	OFASC-MIWO	**6.9632 (0.00001)**	**6.83606 (0.00039)**	**5.92924 (0.00011)**	**5.8131 (0.00001)**	**4.93606 (0.00039)**	**4.81381 (0.00008)**	**4.61381 (0.00008)**	**3.86321 (0.00015)**	**3.86321 (0.00015)**	**2.77954 (0.00547)**	*

We performed statistical tests (parametric and non-parametric) for our proposed approach and other contemporary approaches and used the T-test [193, 194, 195] and Wilcoxon signed-rank test [195] to evaluate whether the best mean values obtained by all approaches have a significant difference with 58 degrees of free-

dom at a 1% level of significance. Table 3.13 and Table 3.14 illustrate the statistical results of the T-test and Wilcoxon signed-rank test. Based on Table 3.13, the results obtained are statistically significant (all T-values are positive, and the P-values are less than zero). Based on Table 3.14, we observe that the results procured are statistically significant (all T-values are positive, and the P-values are less than zero). By performing the T-test and the Wilcoxon signed-rank test at a 1% level of significance, we observe that our proposed approach performs significantly better than other approaches.

Table 3.14: Statistical analysis results for Wilcoxon signed-rank Test (Z-Value/ P-Value) of OFASC-MIWO and other approaches

No of compositions	Approaches	GA	DE	IWO	PSO	IDPSO	OGA	GRASP	Skew+GA	Skew+DE	OFASC-AGEGA	OFASC-MIWO
100	**GA**	*	*	*	*	*	*	*	*	*	*	*
	DE	-2.4373 (0.01468)	*	*	*	*	*	*	*	*	*	*
	IWO	-4.7821 (0)	-4.7821 (0)	*	*	*	*	*	*	*	*	*
	PSO	-1.0387 (0.29834)	3.98 (6E-05)	-4.7821 (0)	*	*	*	*	*	*	*	*
	IDPSO	-3.0594 (0.00111)	-3.0594 (0.00111)	-3.0594 (0.00111)	-3.0594 (0.00111)	*	*	*	*	*	*	*
	OGA	-2.8241 (0.0024)	-3.0594 (0.00111)	-2.7456 (0.00298)	-3.0594 (0.00111)	-2.8241 (0.00241)	*	*	*	*	*	*
	GRASP	-3.0594 (0.00111)	-2.7456 (0.00298)	-2.1181 (0.017)	-3.5686 (0.00036)	-2.7456 (0.00298)	-2.1181 (0.017)	*	*	*	*	*
	Skew+GA	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-3.0594 (0.00111)	-3.0594 (0.00111)	*	*	*	*
	Skew+DE	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-3.0594 (0.00111)	-3.0594 (0.00111)	*	*	*	*
	OFASC-AGEGA	-2.9025 (0.00187)	-3.0594 (0.00111)	-2.9025 (0.00187)	-3.0594 (0.00111)	-2.9025 (0.00187)	-2.1181 (0.0017)	-3.5686 (0)	-3.5686 (0)	-4.7821 (0)	*	*
	OFASC-MIWO	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.6999 (0)**	*

3.4 Summary

In this chapter, we proposed two QoS-aware web service composition approaches, including Optimal Fitness-Aware Service Composition using an Adaptive Genotypes Evolution-Based Genetic Algorithm and Optimal Fitness-Aware Service Composition using Modified Invasive Weed Optimization techniques.

These approaches assess the fitness of candidate services as well as composition fitness with consideration of the balancing of QoS parameters and connectivity constraints for addressing QoS-aware service composition. The empirical study demonstrated that the proposed approach is scalable, robust, and optimal, especially in comparison with existing approaches. Based on the experimental results, we observe that our proposed method performs better than all other methods compared, according to the statistical significant test carried out at a 1% level of significance.

Chapter 4

QoS-aware Cloud Service Selection

Nowadays, various cloud service providers offer cloud services with comparable functionality but varying in price and performance levels. Often, there may be trade-offs among different non-functional and functional requirements fulfilled by various cloud providers. Hence, it is hard to select suitable cloud services and evaluate their relative performance and their ranking based on different quality of service (QoS) attributes. Several researchers have proposed various approaches [203, 204, 205, 206, 207, 208] for evaluating cloud services and their ranking based on various QoS attributes such as price/hour, virtual core, security, accountability, assurance, compute units, memory, and disk sizes. However, these approaches do not consider user preferences, relative performance, or essential QoS attributes such as I/O operational consistency, disk storage performance, processing performance, and memory performance. Hence, we propose two approaches, (i) Extended Data Envelopment Analysis (EDEA) and Extended Super-efficiency Data Envelopment Analysis (ESDEA) and (ii) Extended Technique for Order Preference by Similarity to Ideal Solution (ETOPSIS), Extended Grey TOPSIS (EGTOPSIS), and Extended Fuzzy TOPSIS (EFTOPSIS) for cloud service selection.

4.1 Illustrative Scenario

XYZ Bank is a medium sized public sector bank in India planning to implement innovative customer-centric services to improve its customer trustworthiness and efficiency. The bank offers various services to customers, such as core banking services, payment services, and HR-related services. In order to provide these services with high efficiency and low maintenance cost, the bank plans to use cloud services, considering the following reasons:

- Cost reduction: The maintenance cost of dedicated hardware, software, and support personnel in XYZ Bank will be greatly reduced by using cloud services.
- Improvement in flexibility and scalability: Cloud services will enable XYZ Bank to respond faster by dynamically scaling requirements up and down.
- Increase in efficiency: Integrating new technologies and applications is much easier in the cloud, which will result in a higher efficiency of XYZ Bank.
- Better customer relationships: Combined with big data analytics, the cloud enables XYZ Bank to make better decisions on behalf of its customers, and services can be more customized.

For implementing various services, XYZ Bank looks for an infrastructure-as-a-service (IaaS) cloud service provider to fulfill its requirements. In today's market, there are many IaaS providers, such as Amazon, HP, Microsoft, and Rackspace. Each service provider offers several services varying in their QoS attributes and with varying numbers of virtual cores and amounts of memory. In order to select a service from among those available, the bank considers some of the QoS attributes of the services and analyzes their performance based on the QoS attributes.

XYZ Bank considers five QoS attributes: price, processing performance, I/O operational consistency, disk storage performance, and memory performance (See Table 4.1 for the definitions of these QoS attributes). Table 4.2 lists sample values of these attributes for four available services (A_1, A_2, A_3, A_4).

Table 4.1: List of QoS attributes and their description

S.No	QoS attributes	Description
1	Price per hour	The cost of a virtual machine per hour
2	Processsing Performance	The number of jobs that a computer can execute in a given amount of time (the processing and orchestration of all applications as integer and floating point operations per second).
3	I/O Operational Consistency	The average time required for disk I/O operations to remain consistent (measured in I/O operations per second).
4	Disk Storage Performance	The number of operations performed on a disk in a certain amount of time (measured in I/O operations per second).
5	Memory Performance	The relationship between speed and latency.

Table 4.2: The sample QoS attribute values for XYZ bank

Service	Virtual core	Cost	Processing performance	I/O consistency	Disk performance	Memory performance
A_1	2	55	75	80	90	85
A_2	4	60	60	85	80	90
A_3	4	80	80	65	85	75
A_4	2	90	60	90	70	75

Now, the bank needs to choose one service from among the available services. The selection of service will be carried out by comparing all QoS attributes and analyzing the performance of each service. The service having the best performance score will be selected.

In general, any bank will tend to select a cloud service that has faster computing and memory operations and lower maintenance cost. The process of selecting a service provider is based on the bank's requirements. Generally, XYZ Bank chooses services with higher scores. It can change the preferences by changing the weights (ranging from t_i to t_j) assigned to the QoS attributes according to its requirements. For example, let us consider three cases in which XYZ Bank changes its priorities (or weights). In case 1, XYZ Bank gives higher priority to CPU performance than to the other attributes by assigning weights such as CPU performance, 0.4; disk I/O consistency, 0.1; memory performance, 0.2; cost,

0.3; disk performance, 0.1. In this case, the service with high CPU performance, i.e., A_3, is selected. In case 2, XYZ Bank prefers a service with low cost and changes the weights accordingly: CPU performance, 0.1; disk I/O consistency, 0.1; memory performance, 0.1; cost, 0.6; disk performance, 0.1. In this case, the high priority is cost, and the cloud service with low cost is selected, i.e., cloud service A_1. In case 3, XYZ Bank prefers efficient disk operations. It assigns the weights as CPU performance, 0.1; disk performance, 0.5; disk I/O consistency, 0.3; cost, 0.1; and memory performance, 0.1, giving more importance to high disk performance. In this case, A_1 is selected. This type of selection process can be complicated as these changes can result in large variations given a set of cloud services with varying QoS attributes. In order to overcome such problems, our proposed methods can provide help in selecting a cloud service to fulfill user's requirements.

4.2 Evaluating the Efficiency of Cloud Services using EDEA and ESDEA

In this section, we present our proposed methods, Extended Data Envelopment Analysis (EDEA) and Extended Super-efficiency Data Envelopment Analysis (ESDEA). EDEA is a modified DEA model (see section A.3.1) in which we evaluate the preferred efficiency by assigning weights to the output parameters obtained using Analytical Hierarchical Process (AHP)/ Analytical Network Process (ANP). In our model, each decision-making unit (DMU) represents a cloud service and the cost of each cloud service (price/hour) as input, and the performance of the QoS attributes (I/O operational consistency, disk storage performance, processing performance, and memory performance) are obtained as output parameters.

The conventional DEA model locates cloud services that lie on the efficient frontier. This indicates that these cloud services perform at a relatively high level. However, there is still a chance of improvement for these cloud services if the slack variable is nonzero. This model gives the projected value for improving

the performance of cloud services that do not lie on the efficient frontier. To give preference to the QoS attributes for measuring the performance of the cloud services, we adopted the input-oriented model and modified it to evaluate the preferred efficiency by assigning weights to the output parameters obtained using AHP/ANP.

The conventional DEA's discrimination ability is reduced if more DMUs are found to be efficient units because the sum of the number of inputs and outputs is large compared with the total number of DMUs in the sample [209, 210, 211]. However, conventional DEA differentiates between efficient and inefficient DMUs; assigning a unique rank to them is not possible in conventional DEA. When DMUs are ranked using the conventional DEA model, various DMUs achieve an efficiency score of 1. In such cases, the ranking of efficient DMUs is a crucial challenge faced by the decision maker. To resolve this problem, in our extended DEA, the weights are assigned to the input variables by using AHP/ANP.

4.2.1 Extended Data Envelopment Analysis (EDEA)

Data Envelopment Analysis (DEA) is the optimization method of a mathematical programming model to generalize the single-input/single-output technical efficiency measure to the multiple-input/multiple-output case by constructing a relative efficiency score as the ratio of a single virtual output to a single virtual input. Thus, DEA has become a popular tool in operational research for measuring technical efficiency. The "envelopment" in DEA stems from the way observations are enveloped in order to find the "Pareto frontier" that is used to assess the relative performance of all peer entities [212, 213].

In the conventional DEA model, decision-making units (DMUs) represent the operations or processes that convert multiple inputs to multiple outputs. The DEA model maximizes the efficiency of a DMU subject to constraints relative to the best DMUs. The efficiency of all DMUs is, then, less than or equal to 1. Through performance evaluation, the model determines the "efficiency frontier" that represents the relative best practices. Inefficient strategies can be improved

with the suggested directions [214]. The basic assumption is that if a DMU inefficient, then a virtual DMU obtained via a combination of other efficient DMUs can either result in greater output for the same number of inputs or use fewer inputs to produce the same number of outputs, or some combination of both. There are two approaches to determining an efficiency frontier: input-oriented and output-oriented [214]. A detailed explanation of these approaches is presented in the Appendix A.3.1. We modify the input-oriented method to assess the preferred efficiency by assigning weights to the output parameters obtained by using AHP/ANP. The description of the EDEA model is given as follows.

Input Oriented EDEA: Suppose there are N DMUs, of which DMUo is the one under evaluation. Let $x_{i\,j}$ be the value of the i^{th} input (of P inputs) used by DMU_i, and let $y_{i\,k}$ denote the value of the k^{th} output (of Q outputs) produced by DMU_i. The objective function of our proposed method is given by equation 4.1: 4.4.

$$\theta^* = \min_{\theta,\mathbf{w}} \theta \tag{4.1}$$

subject to

$$\sum_{j=1}^{N} w_j x_{ij} \leq \quad \theta x_{io} \qquad i = 1, \ldots, P \tag{4.2}$$

$$\sum_{j=1}^{N} w_j y'_{kj} \geq \quad y'_{ko} \qquad k = 1, \ldots, Q \tag{4.3}$$

$$w_j \geq \quad 0 \qquad j = 1, \ldots, N \tag{4.4}$$

where y'_{ko} is the modified k^{th} output of DMUo. $\mathbf{w}$ is a weight vector; $\mathbf{w}$ denotes the contribution of the j^{th} DMU to the virtual DMU. If $\theta^* = 1$, then DMUo is on the "efficient frontier," and the current input level cannot be reduced. If $\theta^* < 1$, then DMUo is inefficient. In this case, we can increase efficiency by reducing input while maintaining output at the same level. Y_{ko} is the modified k^{th} output of DMUo. The pseudocode for the Extended DEA model integrated with AHP/ANP is shown as Algorithm 4.1.

Algorithm 4.1 Pseudo-code representation of EDEA with AHP/ANP

Input: M:set of cloud services, N: QoS attributes.
Output: Best M.
while $M \neq NULL$ **do**
  for each M **do**
    Initialize as a DMU;
  end for
  for each DMU **do**
    Evaluate relative performance by using modified input-oriented DEA;
    Determine preferred efficiency using AHP/ANP;
    Evaluate technical efficiency score by using eq. 4.1;
    if ($\theta^* = 1$) **then**
      DMUo is on the efficient frontier;
    else
      DMUo is inefficient;
    end if
  end for
  if more than one DMU on the efficient frontier **then**
    Compute efficiency score by using eq. A.16;
  else
    repeat for loop;
  end if
  Rank the DMU based on efficiency score (θ^*);
end while

4.2.2 Extended Super-efficiency Data Envelopment Analysis (ESDEA)

DEA measures the relative efficiency of a set of similar units comprising all units (including itself), whereas Super-efficiency DEA excludes each unit from its reference set when calculating the relative efficiency. This property causes Super-efficiency DEA to obtain an efficiency score that exceeds 1. There are two approaches to determining an efficiency frontier: input-oriented and output-oriented [214]. A detailed explanation of these approaches is presented in the Appendix A.3.1. The description of the ESDEA model is given as follows.

We extend the Super-efficiency DEA method that finds the preferred efficiency of cloud services by assigning weights to the output parameters obtained using AHP/ANP (similar to EDEA). The modified objective function is as follows:

Input-oriented SDEA: The linear program for the input-oriented model used for Super-efficiency DEA is the same as the one in equations 4.1–4.4, with the difference that DMUo is excluded from the summations in equations 4.6 and 4.7, which become

$$\theta^* = \min_{\theta, \mathbf{w}} \ \theta \tag{4.5}$$

subject to

$$\sum_{j=1, j\neq E}^{N} w_j x_{i\,j} \leq \theta x_{i\,o} \qquad i = 1, 2, \ldots, P \tag{4.6}$$

$$\sum_{j=1, j\neq E}^{N} w_j y_{k\,j}^T \geq y_{k\,o}^T \qquad k = 1, 2, \ldots, Q; \quad w_j \geq 0, \qquad j = 1, 2, \ldots, N \tag{4.7}$$

These notations have the same meaning as in EDEA. The pseudocode for the Extended SDEA model integrated with AHP/ANP is shown as Algorithm 4.2.

Algorithm 4.2 Pseudo-code representation of ESDEA with AHP/ANP

Input: M:set of cloud services, N: QoS attributes.
Output: Best M.

```
while M ≠ NULL do
    for (each M) do
        Initialize as a DMU;
    end for
    for (each DMU) do
        Evaluate relative performance by using modified input-oriented SDEA;
        Determine preferred efficiency using AHP/ANP;
        Evaluate technical efficiency score by using eq. 4.5 ;
        if (θ* = 1) then
            DMUo is on the efficient frontier;
        else
            DMUo is inefficient;
        end if
    end for
    if (more than one DMU on the efficient frontier) then
        Compute efficiency score by using eq. A.16;
    else
        repeat for loop;
    end if
    Rank the DMU based on efficiency score (θ*);
end while
```

4.2.3 Data Collection Methodology and Data Set Description

We considered seven real-world infrastructure-as-a-service (IaaS) cloud service providers: Amazon, Azure, CenturyLink, City Cloud, Google, HP, and Rackspace (not in any order). These cloud service providers offer two or three cloud services that vary in the number of virtual cores. We classified these cloud services into three categories: *Large* (two virtual cores), *Extra-large* (four virtual cores), and *2x-extra-large* (eight virtual cores). An overview of the data set used in our experiments is presented in Table 4.3. The cloud service providers are coded as $C_1, C_2, \ldots, C_7$ (in no particular order), and the services provided by each are further denoted as S_1, S_2, S_3, and so on. For each service, the specified data for price/hour (dollars), number of virtual cores, and memory (GB) were collected from the respective cloud service providers, and the values for I/O operational consistency, disk storage performance, processing performance, and memory performance were obtained from cloudharmony.com. We performed a consistency check while collecting data to identify data that were logically inconsistent or out of range. Inconsistent data were corrected if possible; otherwise, we excluded the service provider from the analysis. Our data set was limited, having certain QoS attributes for seven cloud service providers. However, collecting this real-world data on cloud service providers was extremely challenging. Our data set is open to use by researchers for the purpose of further research.

4.2.4 Performance Evaluation

Our experimental evaluation is based on the input-oriented DEA model, which addresses the question, "By how much can the input parameter (price/hour) be proportionally decreased without changing the output parameters (performance benchmark values)?" We used the cost of each cloud service (price/hour) offered by the providers as the input parameter for our analysis. Our proposed algorithm provides processing performance, disk storage performance, I/O operational consistency, and memory performance as output parameters.

Table 4.3: The collected cloud service dataset

Providers	Service	Price/Hr(Cents)	Virtual Core	Memory	Processing performance	I/O operational consistency	Disk storage performance	Memory Performance
C_1	C_1S_1	28	4	15	25.86	92.89	110.33	129.03
	C_1S_2	56	8	30	48.23	53.28	67.22	131.79
C_2	C_2S_1	14	2	7.5	13.89	114.44	97.38	144.86
	C_2S_2	28	4	15	23.66	119.63	100.55	131.81
	C_2S_3	56	8	30	51.7	77.46	73.44	125.59
C_3	C_3S_1	16	4	4	7.21	70.29	125.48	54.28
	C_3S_2	32	8	8	15.33	57.11	111.18	55.68
C_4	C_4S_1	18	2	3.5	8.83	67.87	83.73	52.27
	C_4S_2	36	4	7	16.07	67.97	78.49	61.8
	C_4S_3	72	8	14	28.4	78.72	70.91	27.33
C_5	C_5S_1	12	2	4	16.41	23.43	40.23	80.67
	C_5S_2	45	4	15	32.4	29.07	42.47	90.83
	C_5S_3	90	8	30	52.82	35.35	55.07	83.92
C_6	C_6S_1	8	2	4	17.34	43.02	141.23	51.71
	C_6S_2	16	4	8	37.05	36.15	102.74	132.87
	C_6S_3	32	8	16	71.11	39.66	99.15	135.88
C_7	C_7S_1	10.132	2	4	23.43	89.31	173.49	89.84
	C_7S_2	20.8624	4	8	42.05	59.63	174.5	97.16
	C_7S_3	34.6528	8	16	75.89	64.64	174.12	100.14

However, any number of parameters can be involved for inputs and outputs. Our process obtains the weights from AHP [215] and ANP [216] and assesses the preferred efficiencies of the given cloud services in terms of the said output parameters. We evaluate EDEA and ESDEA methods that find the preferred efficiency of cloud services by assigning wights to the QoS attributes obtained by using AHP. We used AHP to compute the weight w_j for each criterion of the QoS attributes of cloud service. The relative importance of the QoS attributes is measured using a scale of 1 to 9 as shown in Table 4.4. The weights of QoS attributes are presented in Table 4.5. The pairwise comparison was made with the help of domain experts.

Table 4.6 presents the relative significance of the QoS attributes. For instance, the relative significance of memory performance is four times that of disk storage performance, and the relative significance of processing performance is four times that of disk storage performance, and their proportions are in the lower triangular matrix; i.e., disk storage performance is 0.25 (1/4) times as important

as processing performance.

Table 4.4: Scales for comparison matrix of criteria

Intensity of Importance	Definition
1	Equal Importance
3	Moderate Importance of one over another
5	Essential or Strong Importance
7	Very Strong Importance
9	Extreme Importance
2,4,6,8	Intermediate value between the two adjacent judgments

Table 4.5: Weights for QoS attributes

QoS Attributes	**Weights**
Processing performance	48.66 %
I/O operational consistency	8.48 %
Disk storage performance	12.13 %
Memory performance	30.73 %

After calculating weights, we check the consistency of these weights using a parameter known as the *consistency ratio*, which tells us about inconsistencies in the matrix. We use *consistency ratio* to check whether the weights we have calculated are acceptable. The results are acceptable and consistent if and only if *consistency ratio* is less than or equal to 0.1. These weights are used to evaluate the preferred values of the performance attributes, which is done by multiplying these fractions by the sum of the normalized output parameters. Normalization is carried out to remove the units of different attributes so that all the outputs can be added to evaluate the desired precise value.

Table 4.6: QoS attributes and their relative significance

QoS Attribute	Processing performance	I/O operational consistency	Disk storage performance	Memory performance
Processing performance	1	5	4	2
I/O operational consistency	0.2	1	0.5	0.33
Disk storage performance	0.25	2	1	0.25
Memory performance	0.5	3	4	1

The results of EDEA and ESDEA integrated with AHP in comparison with DEA and SDEA are shown in Table 4.7 and Table 4.8. The results for DEA/SDEA indicate the efficiency score of DMUs with equal priorities for all QoS attributes, and the results for EDEA/ESDEA indicate the efficiency score of DMUs with priority weights obtained using AHP. From Table 4.7, cloud service C_7S_1 provides better performance on the preferred QoS attributes at a reasonable service charge irrespective of the number of virtual cores. C_7S_1 is the best cloud service among the cloud services having two virtual cores. Similarly, if a user wishes for

a better service with four virtual cores, then cloud service C_6S_2 may be selected from among the other cloud services. Likewise, cloud service C_7S_3 is the best cloud service among the cloud services having eight virtual cores.

Table 4.7: Relative efficiency scores of cloud services using DEA and EDEA with AHP

Cloud Services	Efficiency Scores		Ranks
	DEA	EDEA	
C_1S_1	0.49	0.36	10
C_1S_2	0.37	0.17	16
C_2S_1	1.00	0.71	3
C_2S_2	0.50	0.38	8
C_2S_3	0.40	0.18	14
C_3S_1	0.50	0.41	6
C_3S_2	0.21	0.20	13
C_4S_1	0.43	0.32	12
C_4S_2	0.21	0.18	15
C_4S_3	0.17	0.09	18
C_5S_1	0.72	0.38	7
C_5S_2	0.31	0.13	17
C_5S_3	0.25	0.09	19
C_6S_1	1.00	0.83	2
C_6S_2	1.00	0.55	4
C_6S_3	0.96	0.35	11
C_7S_1	1.00	1.00	1
C_7S_2	0.87	0.51	5
C_7S_3	0.95	0.38	9

Table 4.8: Relative efficiency scores of cloud services using SDEA and ESDEA with AHP

Cloud Services	Efficiency Scores		Ranks
	SDEA	ESDEA	
C_1S_1	0.49	0.36	10
C_1S_2	0.37	0.17	16
C_2S_1	1.17	0.71	3
C_2S_2	0.50	0.38	8
C_2S_3	0.40	0.18	15
C_3S_1	0.50	0.41	6
C_3S_2	0.21	0.20	13
C_4S_1	0.43	0.32	12
C_4S_2	0.21	0.18	14
C_4S_3	0.17	0.09	18
C_5S_1	0.72	0.38	9
C_5S_2	0.31	0.13	17
C_5S_3	0.25	0.09	19
C_6S_1	1.03	0.83	2
C_6S_2	1.00	0.55	4
C_6S_3	0.96	0.35	11
C_7S_1	1.36	1.20	1
C_7S_2	0.87	0.51	5
C_7S_3	0.95	0.38	7

Based on Table 4.8, we notice that cloud service C_7S_1 provides better performance on the preferred QoS attributes at a reasonable service charge irrespective of the number of virtual cores. Cloud service C_7S_1 is the best cloud service among the cloud services having two virtual cores. If a user wishes for a better service with four virtual cores, then cloud service C_6S_2 may be selected from among the other cloud services. Similarly, cloud service C_7S_3 is the best cloud service among the cloud services having eight virtual cores. Based on Table 4.7 and Table 4.8, we observe that C_7S_1, C_6S_2, and C_7S_3 are efficient for users

looking for higher numbers of virtual cores (two, four, and eight, respectively). It can be seen that cloud service provider C_7 performs well relative to the other services.

We evaluate EDEA and ESDEA methods that find the preferred efficiency of cloud services by assigning wights to the QoS attributes obtained by using ANP. The relative preferences of the cloud service selection criteria were assessed by ANP Super Decisions 1.6.0 software. The relative importance of the QoS attributes is measured using a scale of 1 to 9 as shown in Table 4.4. The weight of each criterion are described in Table 4.9. The weights of each QoS attribute are presented in Table 4.10. The pairwise comparison was made with the help of domain experts. We use *consistency ratio* to check whether the weights we have calculated are acceptable. The results are acceptable and consistent if and only if *consistency ratio* is less than or equal to 0.1.

Table 4.9: Relative importance among QoS attributes

	Processing performance	I/O operational consistency	Disk storage performance	Memory performance
Processing performance	1	0.25	0.33	0.50
I/O operational consistency	4	1	0.20	0.14
Disk storage performance	3	5	1	0.25
Memory performance	2	7	4	1

Table 4.10: Weights for QoS attributes

QoS Attributes	Weights
Processing performance	27.69 %
I/O operational consistency	10.64 %
Disk storage performance	19.80 %
Memory performance	41.88%

The results of EDEA and ESDEA integrated with ANP in comparison with DEA and SDEA are presented in Table 4.11 and Table 4.12. The results for DEA/SDEA indicate the efficiency score of DMUs with equal priorities for all QoS attributes, and the results for EDEA/ESDEA indicate the efficiency score of DMUs with priority weights obtained using ANP. From Table 4.11, cloud service

C_1S_1 provides better performance on the preferred QoS attributes at a reasonable service charge irrespective of the number of virtual cores. C_6S_1 is the best cloud service among the cloud services having two virtual cores. Similarly, if a user wishes for a better service with four virtual cores, then cloud service C_1S_1 may be selected from among the cloud services. Likewise, cloud service C_7S_3 is the best cloud service among the cloud services having eight virtual cores.

Table 4.11: Relative efficiency scores of cloud services using DEA and EDEA with ANP

Cloud Services	Efficiency Scores		Ranks
	DEA	EDEA	
C_1S_1	0.562018	1	1
C_1S_2	0.682403	0.167805	17
C_2S_1	1	0.571429	5
C_2S_2	1	0.285714	11
C_2S_3	0.48329	0.177399	16
C_3S_1	0.578499	0.285714	12
C_3S_2	0.27028	0.25	13
C_4S_1	0.508029	0.4444	7
C_4S_2	0.254143	0.2222	14
C_4S_3	0.171651	0.1111	19
C_5S_1	0.801606	0.66667	4
C_5S_2	0.311036	0.177778	15
C_5S_3	0.26009	0.112308	18
C_6S_1	1	1	2
C_6S_2	1	0.5	6
C_6S_3	1	0.424601	8
C_7S_1	1	0.789578	3
C_7S_2	1	0.404566	10
C_7S_3	1	0.41838	9

Table 4.12: Relative efficiency scores of cloud services using SDEA and ESDEA with ANP

Cloud Services	Efficiency Scores		Ranks
	SDEA	ESDEA	
C_1S_1	0.562018	0.193416	10
C_1S_2	0.682403	0.109328	14
C_2S_1	1	0.41877	1
C_2S_2	1	0.194916	9
C_2S_3	0.48329	0.110431	13
C_3S_1	0.578499	0.144963	11
C_3S_2	0.27028	0.076561	16
C_4S_1	0.508029	0.119405	12
C_4S_2	0.254143	0.074824	17
C_4S_3	0.171651	0.047161	19
C_5S_1	0.801606	0.282703	5
C_5S_2	0.311036	0.093262	15
C_5S_3	0.26009	0.070167	18
C_6S_1	1	0.294915	4
C_6S_2	1	0.36619	3
C_6S_3	1	0.265679	6
C_7S_1	1	0.386664	2
C_7S_2	1	0.240992	8
C_7S_3	1	0.261832	7

Based on Table 4.12, we notice that cloud service C_2S_1 provides better performance on the preferred QoS attributes at a reasonable service charge irrespective of the number of virtual cores. From Table 4.12, cloud service C_2S_1 is the best cloud service among the cloud services having two virtual cores. If a user wishes for a better service with four virtual cores, then cloud service C_6S_2 may be selected from among the cloud services. Similarly, cloud service C_6S_3 is the best

cloud service among the cloud services having eight virtual cores. Based on Table 4.11 and Table 4.12, we observe that C_2S_1, C_6S_2, and C_6S_3 are efficient for users looking for higher numbers of virtual cores (two, four, and eight, respectively). It can be seen that cloud service provider C_2 performs well relative to the other services.

4.2.4.1 A comparative analysis of EDEA and ESDEA

In this section, we describe the details of the analysis performed on our proposed approaches. In this comparative analysis, we consider the following factors:

- Sensitivity to the weights assigned to output parameters
- Effectiveness under change in the services
- Handling of uncertainty

Sensitivity analysis The sensitivity analysis determines the robustness of our proposed approaches. We tested the effect of a change in priorities and final weights on the proposed approaches. For each QoS attribute, we experimented with varying priorities and their corresponding final weights. We moderately adjusted the weights of the QoS attributes one at a time and observed the impact of the changes in weights on the final decisions. In this way, the performance of each QoS attribute and its results were analyzed and applied to EDEA and ESDEA. The sensitivity analysis for processing performance, I/O operational consistency, disk storage performance, and memory performance attributes against 19 cloud services of EDEA are shown in Fig. 4.1, Fig. 4.2, Fig. 4.3, and Fig. 4.4. In Fig. 4.1, Fig. 4.2, Fig. 4.3, and Fig. 4.4, the x-axis represents cloud services, and we projected the efficiency scores of the DMUs on the y-axis.

We analyzed the efficiency scores of the cloud services by varying the weights of the output parameters (processing performance, I/O operational consistency, disk storage performance, and memory performance) one at a time and reported the efficiency scores in descending order for each attribute. We noted the top ten cloud services that gave the best efficiency scores

for different weights in all of the cases. From Fig. 4.1, we notice that $C_7S_1, C_6S_2, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_5S_1, C_2S_1, C_1S_1$, and C_2S_3 give the best efficiency scores for different weights of processing performance attribute. Similarly, in Fig. 4.2, $C_6S_2, C_7S_1, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_5S_1, C_2S_1, C_2S_3$, and C_1S_1 give the best efficiency scores for different weights of I/O operational consistency attribute. Likwise, in Fig. 4.3, we notice that $C_7S_1, C_6S_2, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_5S_1, C_2S_1, C_2S_2$, and C_2S_3 give the best efficiency scores for different weights of disk storage performance attribute. The cloud services $C_7S_1, C_6S_2, C_7S_3, C_6S_3, C_6S_1, C_7S_2, C_5S_1, C_2S_1, C_2S_2$, and C_2S_3 show the best efficiency scores for the different weights of memory performance attribute as shown in Fig. 4.4.

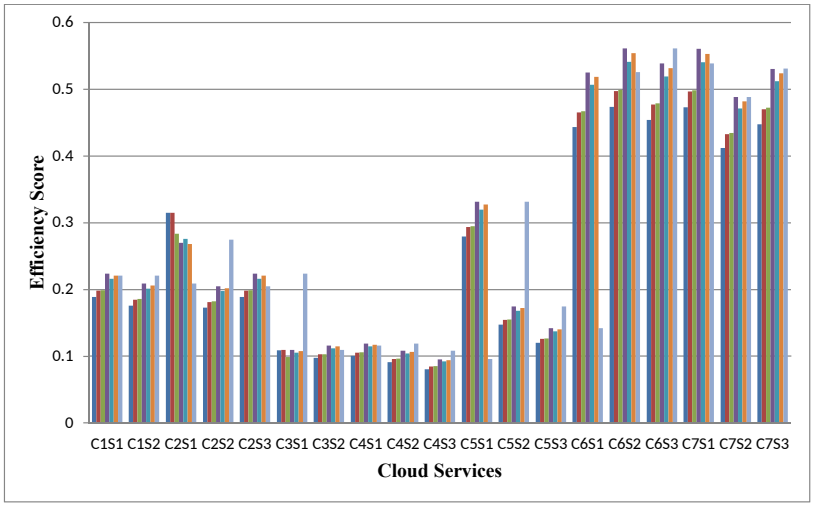


Figure 4.1: Sensitivity analysis of EDEA for processing performance

In our experiments, we observed that the priority of a DMU is proportional to the weight of the corresponding attribute and that these changes have no significant effect on the DMUs' ranking and efficiency. In addition, the final decision does not change in most of the cases. By these experiments, we found that cloud service C_7 is the best cloud service provider in all cases and also has a high number of virtual cores (i.e., eight).

The sensitivity analysis for processing performance, I/O operational consistency, disk storage performance, and memory performance attributes against 19 DMUs

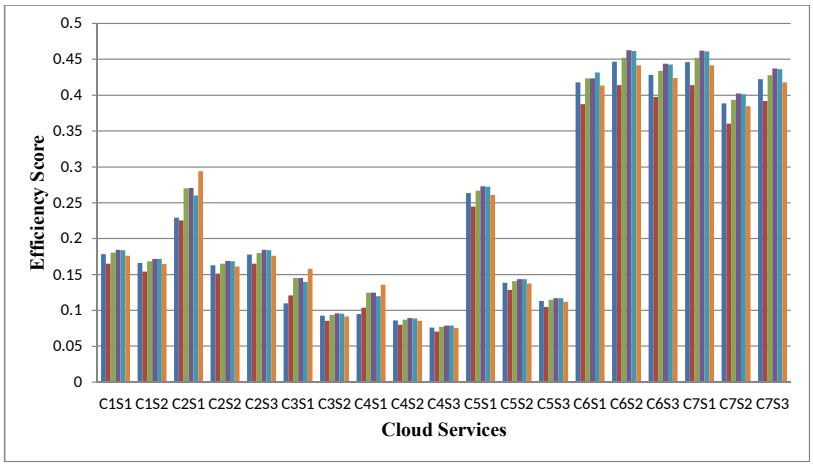


Figure 4.2: Sensitivity analysis of EDEA for I/O operational consistency

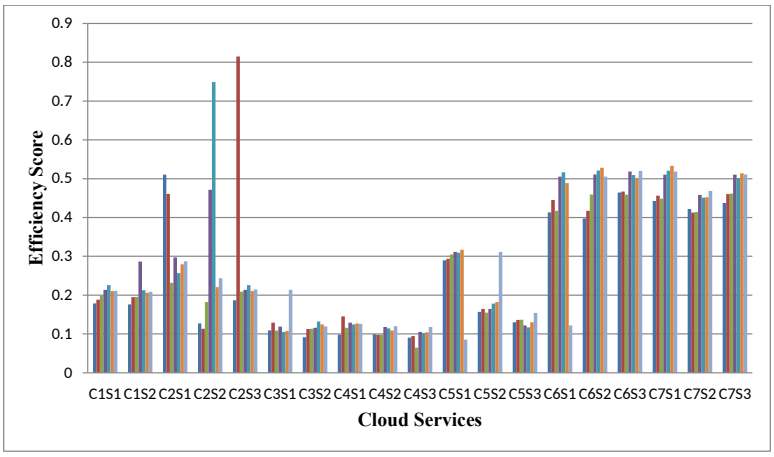


Figure 4.3: Sensitivity analysis of EDEA for disk storage performance

of the ESDEA are shown in Fig. 4.5, Fig. 4.6, Fig. 4.7, and Fig. 4.8. In Fig. 4.5, Fig. 4.6, Fig. 4.7, and Fig. 4.8, the x-axis represents the weights of the cloud services, and the y-axis represents the efficiency scores of the DMUs.

We analyzed the efficiency scores of the cloud services by varying the weights of the output parameters (processing performance, I/O operational consistency, disk storage performance, and memory performance) one at a time and reported the efficiency scores in descending order for each attribute. We noted the top ranking ten cloud services that gave the best efficiency scores for different weights in all of the cases. From Fig. 4.5,

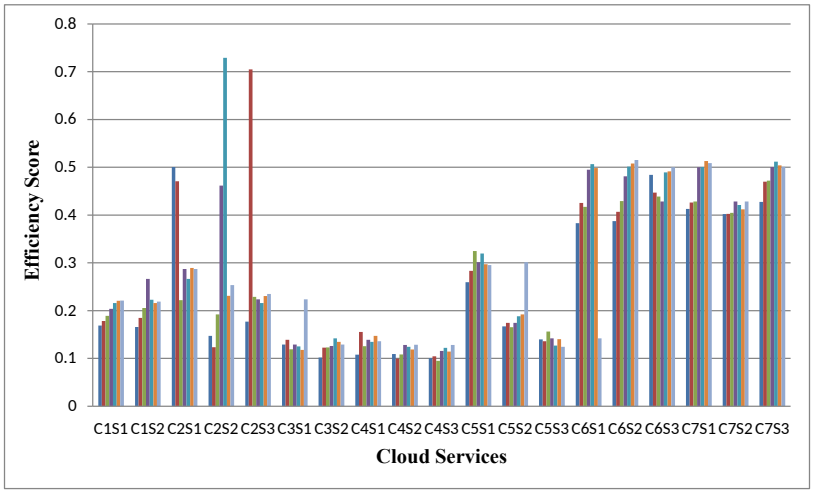


Figure 4.4: Sensitivity analysis of EDEA for memory performance

we observe that $C_6S_2, C_7S_1, C_7S_3, C_6S_3, C_6S_1, C_7S_2, C_2S_1, C_5S_1, C_1S_1$, and C_2S_3 services give the best efficiency scores for different weights of processing performance attribute. Likewise, in Fig. 4.6, $C_7S_1, C_6S_2, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_2S_1, C_5S_1, C_2S_3$, and C_1S_1 services give the best efficiency scores for different weights of I/O operational consistency attribute. The cloud services $C_7S_1, C_6S_2, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_2S_1, C_5S_1, C_2S_3$, and C_1S_1 show the best efficiency scores for the different weights of disk storage performance attribute as shown in Fig. 4.7. Similarly, from Fig. 4.8, we notice that $C_7S_1, C_6S_2, C_6S_3, C_7S_3, C_6S_1, C_7S_2, C_2S_1, C_5S_1, C_5S_2$, and C_2S_3 give the best efficiency scores for different weights of memory performance.

In our experiments, we observed that the priority of a DMU is proportional to the weight of the corresponding attribute and that these changes have no significant effect on the DMUs' ranking and efficiency. In addition, the final decision does not change in most of the cases. By these experiments, we found that cloud service provider C_6 is the best cloud services provider in all cases and also has a high number of virtual cores (i.e., eight).

Adequacy under change in services For evaluating the efficiencies of different cloud services, we may require some inclusion and exclusion criteria on

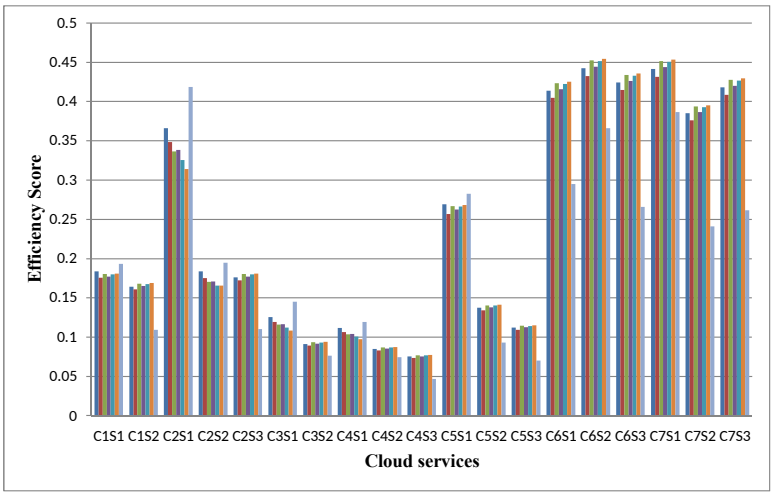


Figure 4.5: Sensitivity analysis of ESDEA for processing performance

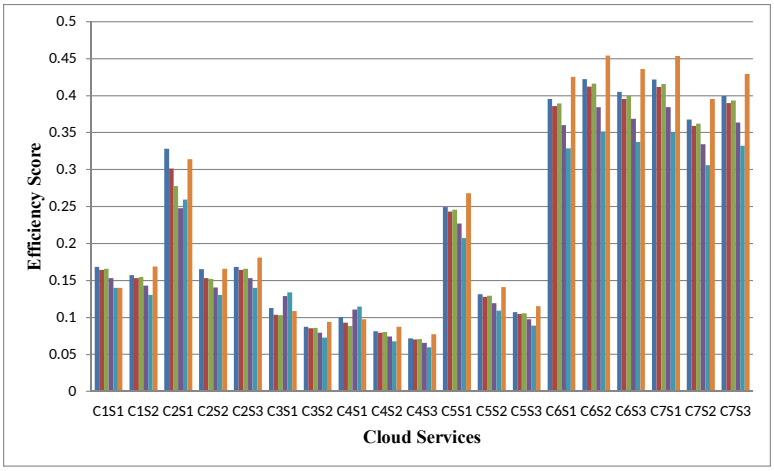


Figure 4.6: Sensitivity analysis of ESDEA for I/O operational consistency

cloud services. Cloud service selection on the basis of efficiency implies a consistent order of priorities for cloud service. To rank the 19 DMUs, EDEA and ESDEA were integrated with AHP and ANP. Initially, we assumed equal weights for all QoS attributes. The resultant rankings of all cloud services under all the proposed approaches are as follows:

- $C_6S_3, C_7S_3, C_6S_2, C_7S_2, C_2S_1, C_6S_1, C_7S_1, C_5S_1, C_2S_3, C_1S_2, C_5S_3, C_3S_2, C_4S_3, C_2S_2, C_3S_1, C_1S_1, C_5S_2, C_4S_2$, and C_4S_1 (using EDEA with AHP)

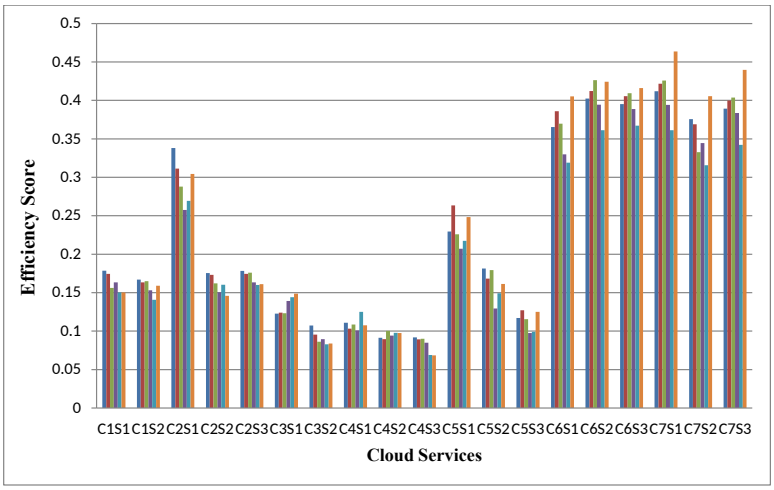


Figure 4.7: Sensitivity analysis of ESDEA for disk storage performance

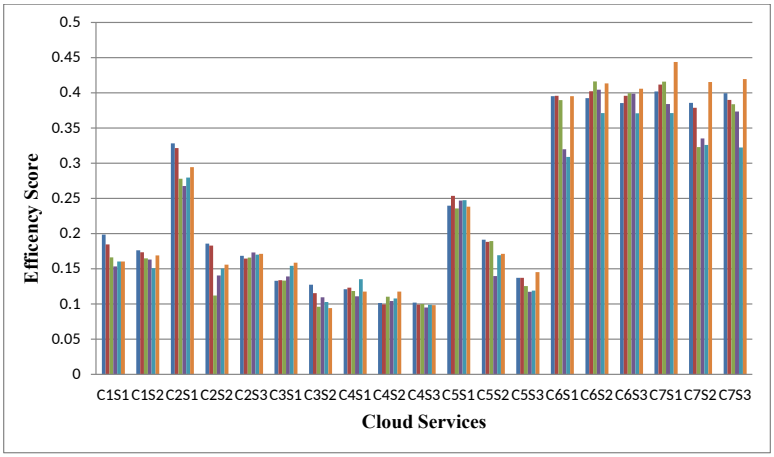


Figure 4.8: Sensitivity analysis of ESDEA for memory Performance

- $C_7S_3, C_6S_3, C_7S_2, C_6S_2, C_2S_2, C_7S_1, C_6S_1, C_2S_1, C_5S_1, C_1S_2, C_3S_1, C_1S_1, C_2S_3, C_4S_1, C_3S_2, C_5S_2, C_5S_3, C_4S_2$, and C_4S_3 (using EDEA with ANP)
- $C_7S_3, C_6S_3, C_3S_2, C_2S_3, C_1S_2, C_4S_3, C_5S_3, C_6S_2, C_7S_2, C_3S_1, C_2S_2, C_1S_1, C_4S_2, C_5S_2, C_7S_1, C_6S_1, C_2S_1, C_5S_1$, and C_4S_1 (using ESDEA with AHP)
- $C_6S_3, C_7S_3, C_2S_3, C_1S_2, C_3S_2, C_5S_3, C_4S_3, C_6S_2, C_7S_2, C_2S_2, C_1S_1, C_3S_1, C_5S_2, C_4S_2, C_2S_1, C_7S_1, C_6S_1, C_5S_1$, and C_4S_1 (using ESDEA with ANP)

The ranks were assigned based on the descending order of the number of virtual cores (i.e., from eight to two). To test EDEA and ESDEA, we added another

cloud service with equal priority weights to the existing cloud services. In most of the cases, we observed that the addition of a new cloud service with equal priority weights had no significant effect on the efficiency of the cloud services. We observed the same effect with all other proposed models. The order of priority remained the same in all test cases and had the same ranking as in the equal-priority case.

Handling of uncertainty In our proposed methods, we employ ANP to deal with the inherent lack of clarity in the data on the efficiencies of cloud services. The ANP model was selected as there is ambiguity in the judgment of quantitative variables and uncertainty in human decision making in the pairwise comparison. In EDEA and ESDEA, we employ pairwise comparison using comparative linguistic variables (described in section 4.2.4).

4.3 Cloud Service Selection using TOPSIS Variants

This section presents our proposed methods Extended Technique for Order Preference by Similarity to Ideal Solution (ETOPSIS), Extended Fuzzy Technique for Order Preference by Similarity to Ideal Solution (EFTOPSIS), and Extended Grey Technique for Order Preference by Similarity to Ideal Solution (EGTOPSIS) for selecting the best cloud service.

4.3.1 ETOPSIS Using AHP

The TOPSIS method is a Multi-criteria Decision-Making (MCDM) technique that sorts the alternatives for a decision problem in order of their distance from the positive and negative ideal solutions [217]. The best solution is the one that has a minimum distance from the positive ideal solution and a maximum distance from the negative ideal solution. The positive ideal solution is the one that maximizes the benefits and minimizes the costs, whereas the negative ideal solution is the one that minimizes the benefits and maximizes the costs. A detailed description of the TOPSIS method is presented in A.3.7.

We establish a set of appropriate evaluation alternatives and criteria with respect to the QoS factors of cloud services. Then, we choose appropriate linguistic variables and linguistic ratings for cloud services with respect to QoS factors, which are determined by experts. Subsequently, we evaluate the criterion weights by using the AHP method and evaluate the alternatives with respect to the criteria (as described in section 4.3.4). We construct a decision matrix of the cloud services based on the linguistic scores and compute the weighted decision matrix (based on equation 4.8). The *decision matrix D* is constructed for the alternatives based on the criteria with each row representing an alternative and each column representing a criterion. Each element is represented by x_{ij}, where i represents the i^{th} alternative, $i = 1, 2, \ldots, m$; and j represents the j^{th} criterion, $j = 1, 2, \ldots, n$:

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \ldots & C_n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \ldots & x_{mn} \end{pmatrix} \end{array} \tag{4.8}$$

where A_i $(A_1, A_2, A_3, \ldots, A_m)$ represents the m alternatives and C_j $(C_1, C_2, C_3, \ldots, C_n)$ represents the n criteria.

After this process, we calculate the positive ideal solution (A^*) and the negative ideal solution (A^-) of the cloud services and calculate the positive and negative ideal solution for each cloud service (based on equations 4.11 and 4.12). Afterwards, we calculate the separation of each criterion for each alternative from the positive ideal solution (S_i^*) and that from the negative ideal solution (S_i^-) (according to equations 4.13 and 4.14). Finally, we determine the relative closeness of each cloud service and give the rating according to the order of preference for cloud services (based on equation 4.15). The pseudocode for ETOPSIS using AHP is presented as Algorithm 4.3.

Algorithm 4.3 Pseudocode representation of ETOPSIS with AHP

Input: Alternatives ($a\_i\ j$), N: QoS attributes.
Output: Best M.

while $M \neq NULL$ **do**
  Construct the Decision matrix D; ▷ Based on equation 4.8
  for each $x_{i\ j}$ in D **do**

$$r_{i\ j} = \frac{x_{i\ j}}{\sqrt{\sum_{i=1}^{n} x_{i\ j}^2}} \tag{4.9}$$

▷ Normalized decision matrix D

$$v_{i\ j} = w_j \times r_{i\ j} \tag{4.10}$$

▷ Compute the weighted normalized decision matrix ($v_{i\ j}$), the weights w_j are calculated by using AHP.

$$A^* = \{v_1^*, v_2^*, \ldots, v_n^*\} \tag{4.11}$$

where $v_j^* = \{max(v_{i\ j}) \quad if j \in J; min(v_{i\ j}) \quad if j \in J'\}$

$$A^- = \{v_1^-, v_2^-, \ldots, v_n^-\} \tag{4.12}$$

    if $j \in J$ **then**
      $A^* = max(v_{i\ j}); A^- = min(v_{i\ j})$
    else if $j \in J$ **then**
      $A^* = min(v_{i\ j}); A^- = max(v_{i\ j})$
    end if
  end for
  for $J \in j \in C_j$ **do**

$$S_i^* = \left(\sum_{j=1}^{n} (v_{i\ j} - v_j^*)^2\right)^{\frac{1}{2}} \tag{4.13}$$

$$S_i^- = \left(\sum_{j=1}^{n} (v_{i\ j} - v_j^-)^2\right)^{\frac{1}{2}} \tag{4.14}$$

where $i = 1, 2, 3, \ldots, m$ ▷ Calculate the separation from the positive ideal solution and negative ideal solution

  end for
  for each $x_{i\ j}$ in D **do**

$$C_i^* = \frac{S_i^-}{(S_i^* + S_i^-)} \tag{4.15}$$

  end for
  Rank the alternatives based on C_i^*;
end while

4.3.2 EFTOPSIS Using Fuzzy AHP

Fuzzy TOPSIS (FTOPSIS) is a combination of fuzzy theory and TOPSIS, where a decision maker assigns a precise performance rating to an alternative for the attribute under consideration. The advantage of using a fuzzy approach is that the relative importance of attributes is assigned using fuzzy numbers instead of precise numbers [218, 219]. A detailed description of the FTOPSIS method is

presented in section A.3.8.

In the Extended FTOPSIS model, we establish the proper appraisal alternatives and criteria with respect to QoS attributes of cloud services. We choose the proper linguistic variables and linguistic ratings for cloud services with respect to the QoS attributes (as determined by experts). Subsequently, we evaluate the criterion weights by using the Fuzzy AHP method and evaluate the alternatives with respect to the criteria (as described in section 4.3.4). We construct a fuzzy decision matrix for cloud services based on linguistic scores, normalize the values for each cloud service, and calculate the weighted fuzzy decision matrix (based on equations 4.16 to 4.19). We calculate the fuzzy positive ideal solution ($\tilde{A}^*$) and the fuzzy negative ideal solution ($\tilde{A}^-$) of the cloud services by using equations 4.20 and 4.21. After this process, we determine the separation measures for each alternative from the positive ideal solution (S^*) and the negative ideal solution (S^-) (using equations 4.22 and 4.23). Afterwards, we calculate the relative closeness of each cloud service and give the score according to the order of preference for cloud services (based on equation 4.25). The pseudocode for EFTOPSIS is presented as Algorithm 4.4.

4.3.3 EGTOPSIS Using AHP

Grey TOPSIS (GTOPSIS) is a combination of grey theory and the TOPSIS method [220, 221]. A detailed description of GTOPSIS is presented in section A.3.10.

Algorithm 4.4 Pseudocode representation of EFTOPSIS with FAHP

Input: Alternatives ($a\_i\ j$), N: QoS attributes.

Output: Best M.

while $M \neq NULL$ **do**

Construct the Decision matrix D; ▷ Based on equation 4.8

for each $x_{i\ j}$ in D **do**

$$R = [\tilde{r}_{i\ j}]_{m*n} \tag{4.16}$$

$$\tilde{r}_{i\ j} = (\frac{p_{i\ j}}{r^*_j}, \frac{q_{i\ j}}{r^*_j}, \frac{r_{i\ j}}{r^*_j}), \quad where \quad r^*_j = max \ \ (r_{i\ j}) \tag{4.17}$$

$$\tilde{r}_{i\ j} = (\frac{p^-_i}{r^*_{i\ j}}, \frac{p^-_i}{q^*_{i\ j}}, \frac{p^-_i}{p^*_{i\ j}}), \quad where \quad p^-_i = max \ \ (p_{i\ j}) \tag{4.18}$$

▷ normalized fuzzy decision matrix (fdm) is denoted by R where its elements are denoted by $\tilde{r}_{i\ j}$ (i being from 1 to m and j from 1 to n).

for each $x_{i\ j}$ in D **do**

$$\tilde{v}_{i\ j} = \tilde{r}_{i\ j} \times \tilde{w}_j \tag{4.19}$$

▷ constructed weighted normalized fuzzy decision matrix $\tilde{V}$. Weights are obtained by Fuzzy AHP method [222](Described in Section A.3.9).

$$\tilde{A}^* = \{\tilde{v}^*_1, \tilde{v}^*_2, \ldots, \tilde{v}^*_n\}, \quad (Benefit \ \ criteria) \tag{4.20}$$

where $\tilde{v}^*_j = \{max(\tilde{v}_{i\ j}), (i = 1, 2, \ldots, m), (j = 1, 2, 3, \ldots, n)\}$

$$\tilde{A}^- = \{\tilde{v}^-_1, \tilde{v}^-_2, \ldots, \tilde{v}^-_n\}, \quad (Cost \ \ criteria) \tag{4.21}$$

where

$$\tilde{v}^-_j = \{min(\tilde{v}_{i\ j}), \ldots, (i = 1, 2, \ldots, m), (j = 1, 2, 3, \ldots, n)\}$$

if $j \in J$ **then**

$\tilde{A}^* = max(\tilde{v}_{i\ j}); \tilde{A}^- = min(\tilde{v}_{i\ j})$

else if $j \in J$ **then**

$\tilde{A}^* = min(\tilde{v}_{i\ j}); \tilde{A}^- = max(\tilde{v}_{i\ j})$

end if

end for

▷ Determine FPIS, FNIS, we used FPIS and FNIS as $\tilde{v}^*_j = (1, 1, 1)$ and $\tilde{v}^-_j = (0, 0, 0)$ [223].

for $J \in j \in C_j$ **do**

$$S^*_i = \sum_{j=1}^{n} d_s(\tilde{v}_{i\ j}, \tilde{v}^*_j) \tag{4.22}$$

$$S^-_i = \sum_{j=1}^{n} d_s(\tilde{v}_{i\ j}, \tilde{v}^-_j) \tag{4.23}$$

where $i = 1, 2, 3, \ldots, m$ and the distance between two fuzzy number are calculated using equations 4.24.

$$d(\tilde{x}, \tilde{y}) = \sqrt{\frac{1}{3} \times [(p_x - p_y)^2 + (q_x - q_y)^2 + (r_x - r_y)^2]} \tag{4.24}$$

where (p_x, p_y, p_z) and (q_x, q_y, q_z) are two TFNs.

for each $x_{i\,j}$ in D **do**

$$CC_i^* = \frac{S_i^-}{S_i^* + S_i^-} \tag{4.25}$$

end for ▷ Compute the closeness coefficient (CC_i)

end for

end for

Rank the alternatives according to the value of CC_i^*; ▷ Greater value of CC_i^* ensures higher rank

end while

To determine the preferable cloud services, we initialize the suitable alternatives and criteria with respect to QoS attributes of cloud services. Subsequently, we select the suitable linguistic variables and ratings for cloud services (as determined by experts). Afterwards, we calculate the criterion weights by using AHP and determine the alternatives with respect to the criteria for cloud services. We establish a *grey decision matrix D* for cloud services (see equation 4.26) based on the criteria with each row representing an alternative and each column representing a criterion.

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \dots & C_n \end{array} \\ \begin{pmatrix} \otimes x_{11} & \otimes x_{12} & \otimes x_{13} & \dots & \otimes x_{1n} \\ \otimes x_{21} & \otimes x_{22} & \otimes x_{23} & \dots & \otimes x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \otimes x_{m1} & \otimes x_{m2} & \otimes x_{m3} & \dots & \otimes x_{mn} \end{pmatrix} \end{array} \tag{4.26}$$

where $\otimes x_{ij}$ denotes the evaluation of grey numbers of the i-th alternative with respect to the i-th criteria. $A_i\,(A_1, A_2, \dots, A_m)$ represents the m alternatives and $C_j(C_1, C_2, \dots, C_n)$ represents the n criteria.

After this process, we calculate the normalized grey decision matrix and the weighted grey decision matrix of cloud services (based on equations 4.27 and 4.28). Afterwards, we compute the positive ideal alternative (A^*) and the negative ideal alternative (A^-) for the cloud services and quantify the A^* and A^* of each cloud service by using equations 4.29 and 4.30. Following that, we determine the separation measure of the positive ideal alternative (d^*) and that of the negative ideal alternative (d^-) according to equations 4.31 and 4.32. The pseudocode for EGTOPSIS using AHP is shown as Algorithm 4.5.

Algorithm 4.5 Pseudocode representation of EGTOPSIS with AHP

Input: Alternatives ($a\_i\ j$), N: QoS attributes.

Output: Best M.

while $M! = NULL$ **do**

Construct the Grey Decision matrix D; ▷ Based on equation 4.26

for each $x_{i\ j}$ in D **do**

$$\otimes r_{i\ j} = \left(\frac{\underline{x}_{i\ j}}{max(\overline{x}_{i\ j})}; \frac{\overline{x}_{i\ j}}{max(\underline{x}_{i\ j})} \right) \tag{4.27}$$

$$\otimes r_{i\ j} = \left(1 - \frac{\underline{x}_{i\ j}}{max(\overline{x}_{i\ j})}; 1 - \frac{\overline{x}_{i\ j}}{max(\underline{x}_{i\ j})} \right) \tag{4.28}$$

where $\underline{x}_{i\ j}$ represents the lower limit value and $\overline{x}_{i\ j}$ represents the upper limit value. ▷ Compute the normalized grey decision matrix ($\otimes r_{i\ j}$).

$$A^* = \{(max(\overline{r}_{i\ j})|j \in J), (min(\underline{r}_{i\ j})|j \in J')\} \quad (benefit \quad criteria) \tag{4.29}$$

and

$$A^- = \{(min(\underline{r}_{i\ j})|j \in J), (max(\overline{r}_{i\ j})|j \in J')\} \quad (cost \quad criteria) \tag{4.30}$$

if $j \in J$ **then**

$A^* = max(\underline{r}_{i\ j}); A^- = min(\underline{r}_{i\ j})$

else if $j \in J$ **then**

$A^* = min(\underline{r}_{i\ j}); A^- = max(\underline{r}_{i\ j})$

end if

end for

for $J \in j \in C_j$ **do**

$$d_i^* = \sqrt{\frac{1}{2} \sum_{j=1}^{n} w_i \left[|r_j^* - \underline{r}_{i\ j}|^2 + |r_j^* - \overline{r}_{i\ j}|^2 \right]} \tag{4.31}$$

$$d_i^- = \sqrt{\frac{1}{2} \sum_{j=1}^{n} w_i \left[|r_j^- - \underline{r}_{i\ j}|^2 + |r_j^- - \overline{r}_{i\ j}|^2 \right]} \tag{4.32}$$

end for ▷ Compute the separation measure of positive (d^+) and negative ideal (d^-) alternatives, where w_i represents each criteria weights obtained by AHP method.

for each $x_{i\ j}$ in D **do**

$$C_i^* = \frac{d_i^-}{d_i^* + d_i^-} \tag{4.33}$$

end for ▷ Calculate the relative closeness (C_i^*) to the positive ideal alternatives.

Rank the alternatives based on C_i^****;*** ▷ The larger indexed value is considered as the better alternative.

end while

4.3.4 Performance Evaluation

We used the same data set described in section 4.2.3. The cloud services ($C_1S_1, \ldots, C_7S_3$) were encoded as alternatives A_1 to A_{19} (in the same order).

The QoS attributes including price, processing performance, I/O operational consistency, disk storage performance, and memory performance of each cloud service were encoded as criteria Cr_1 to Cr_5. The evaluation of the potential cloud services on each criterion was made based on linguistic judgments given by decision makers who are a group of researchers from the cloud provider company. Figure 4.9 depicts the hierarchy structure of the cloud service selection. This hierarchy structure consists of three levels: the first, an objective or goal for the problem; the second, criteria (Cr_1 through Cr_5 in Fig. 4.9); and the third, the alternatives (A_1 through A_{19}).

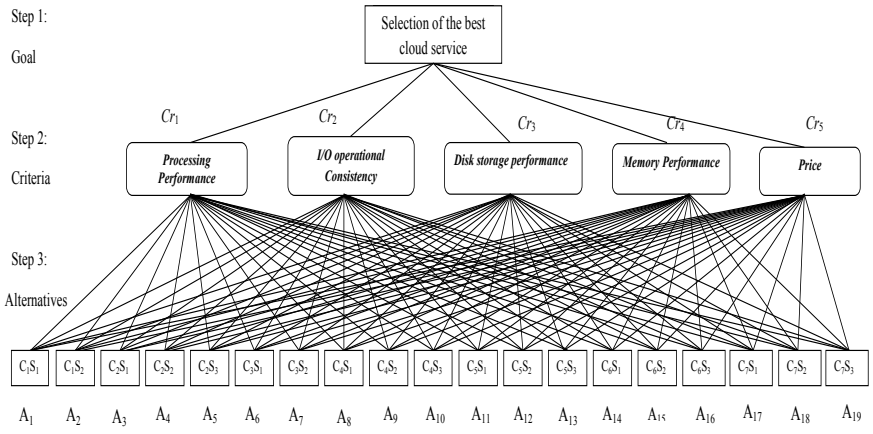


Figure 4.9: The hierarchy structure of the cloud service selection

Cloud service selection using ETOPSIS with AHP We used AHP to compute the weight w_j for each criterion of the QoS attributes of cloud service. The weights of each attribute are presented in Table 4.13. To check the consistency of the calculated weights, we obtained the *consistency ratio* as 0.0511134. This ratio expresses how inconsistent the matrix is, and the result is acceptable if *consistency ratio* is less than or equal to 0.1. Thus, our matrix is consistent, and the weights are valid. The decision matrix for ETOPSIS analysis is shown in Table 4.14. Each element of the matrix was normalized, and the normalized decision matrix is shown in Table 4.15. The results and ranking of the alternatives are summarized in Table 4.16. Based on Table 4.16, we observe that A_{15} is the best alternative, followed by $A_{18} > A_{17} > A_{16} > A_{14} > A_{19} > A_{11} > A_3 > A_1 > A_6 > A_4 > A_8 > A_7 > A_9 > A_{12} > A_5 > A_2 > A_{10} > A_{13}$.

Table 4.13: Priority weights of criteria

Criteria	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5	Weight vector
Cr_1	1	2	3	2	3	0.345230
Cr_2	0.5	1	5	4	2	0.289980
Cr_3	0.33	0.20	1	0.50	0.33	0.068046
Cr_4	0.5	0.25	2	1	0.25	0.103810
Cr_5	0.33	0.50	3	4	1	0.192931

Table 4.16: ETOPSIS analysis and ranking for cloud services

No	$v_{i\,1}$	$v_{i\,2}$	$v_{i\,3}$	$v_{i\,4}$	$v_{i\,5}$	S^*	S^-	C_i^*	Rank
A_1	0.0136	0.0795	0.0175	0.0257	0.0977	0.1938	0.3743	0.6589	9
A_2	0.0782	0.1482	0.0101	0.0157	0.0999	0.2937	0.2478	0.4577	17
A_3	0.0013	0.0426	0.0216	0.0228	0.1098	0.1945	0.4525	0.6996	8
A_4	0.0136	0.0727	0.0226	0.0235	0.0999	0.1992	0.3739	0.6524	11
A_5	0.0782	0.1588	0.0147	0.0172	0.0952	0.2907	0.2522	0.4645	16
A_6	0.0022	0.0222	0.0133	0.0293	0.0412	0.2270	0.4321	0.6555	10
A_7	0.0196	0.0471	0.0108	0.0260	0.0422	0.2430	0.3401	0.5832	13
A_8	0.0034	0.0272	0.01280	0.0196	0.0396	0.2265	0.4201	0.6498	12
A_9	0.0267	0.0494	0.0129	0.0184	0.0468	0.2548	0.3170	0.5545	14
A_{10}	0.1390	0.0873	0.0149	0.0166	0.0207	0.4109	0.1240	0.2319	18
A_{11}	0.00055	0.05039	0.0044	0.0094	0.0611	0.1938	0.4570	0.7022	7
A_{12}	0.0465	0.0995	0.0055	0.0099	0.0688	0.2592	0.2775	0.51702	15
A_{13}	0.2282	0.1622	0.0066	0.0129	0.0636	0.4861	0.1466	0.2316	19
A_{14}	0	0.054	0.00812	0.03294	0.0392	0.1939	0.4797	0.7122	5
A_{15}	0.002171	0.11378	0.00682	0.0240	0.1007	0.1304	0.4481	0.7746	1
A_{16}	0.0196	0.2184	0.0075	0.0232	0.1030	0.1426	0.3996	0.7370	4
A_{17}	0.00016	0.0720	0.0168	0.0405	0.0681	0.1669	0.4715	0.73849	3
A_{18}	0.0057	0.1292	0.0113	0.0407	0.0736	0.1336	0.4213	0.7593	2
A_{19}	0.0242	0.2331	0.0122	0.0407	0.0759	0.1593	0.3906	0.7104	6
v_j^*	0.0466	0.23304	0.02256	0.0407	0.1098				
v_j^-	0.5243	0.0222	0.0045	0.0094	0.0207				

Cloud service selection using EFTOPSIS with Fuzzy AHP We used the Fuzzy Analytical Hierarchy Process (FAHP) [222] to determine the weight w_j for the criteria of the QoS attributes of cloud services. The linguistic variables

Table 4.14: Decision matrix for ETOPSIS

No	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5
A_1	28	25.86	92.89	110.33	129.03
A_2	56	48.23	53.28	67.22	131.79
A_3	14	13.89	114.44	97.38	144.86
A_4	28	23.66	119.63	100.55	131.81
A_5	56	51.7	77.46	73.44	125.59
A_6	16	7.21	70.29	125.48	54.28
A_7	32	15.33	57.11	111.18	55.68
A_8	18	8.83	67.87	83.73	52.27
A_9	36	16.07	67.97	78.49	61.80
A_{10}	72	28.4	78.72	70.91	27.33
A_{11}	12	16.41	23.43	40.23	80.67
A_{12}	45	32.4	29.07	42.47	90.83
A_{13}	90	52.82	35.35	55.07	83.92
A_{14}	8	17.34	43.02	141.23	51.71
A_{15}	16	37.05	36.15	102.74	132.87
A_{16}	32	71.11	39.66	99.15	135.88
A_{17}	10.132	23.43	89.31	173.49	89.84
A_{18}	20.8624	42.05	59.63	174.5	97.16
A_{19}	34.6528	75.89	64.64	174.12	100.14

Table 4.15: Normalized decision matrix for ETOPSIS

No	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5
A_1	0.1632	0.1581	0.3070	0.2323	0.2964
A_2	0.3263	0.2949	0.1761	0.1416	0.3027
A_3	0.0816	0.0850	0.3782	0.2051	0.3327
A_4	0.1632	0.1447	0.3954	0.2117	0.3027
A_5	0.3263	0.3161	0.2560	0.1547	0.2892
A_6	0.0933	0.0441	0.2321	0.2642	0.1247
A_7	0.1865	0.0938	0.1888	0.2341	0.1279
A_8	0.1049	0.0540	0.2243	0.1763	0.1201
A_9	0.2098	0.0983	0.2246	0.1653	0.1420
A_{10}	0.4195	0.1736	0.2602	0.1493	0.0628
A_{11}	0.0701	0.1004	0.0775	0.0847	0.1853
A_{12}	0.2622	0.1981	0.0961	0.0895	0.2086
A_{13}	0.5243	0.3229	0.1169	0.1160	0.1928
A_{14}	0.0467	0.1060	0.1422	0.2974	0.1188
A_{15}	0.0933	0.2265	0.1195	0.2163	0.3052
A_{16}	0.1865	0.4347	0.1311	0.2088	0.3121
A_{17}	0.0591	0.1433	0.2952	0.3653	0.2064
A_{18}	0.12165	0.2571	0.1971	0.3674	0.2231
A_{19}	0.2019	0.4639	0.2136	0.3665	0.2299

used for ranking the alternatives by pairwise comparison are presented in Table 4.17. The linguistic variables used for evaluating the importance of the criterion weights by pairwise comparison are presented in Table 4.18. The pairwise comparison was made with the help of domain experts. The pairwise comparison with respect to the criteria is presented in Table 4.19. The criterion weights with respect to the cloud services selection are presented in Table 4.20.

Table 4.17: Linguistic variables for cloud services

Linguistic variable	Fuzzy number
Very Low	(0,0.05,0.15)
Low	(0.1,0.2,0.3)
Fairly Low	(0.2,0.35,0.5)
Fair	(0.3,0.5,0.7)
Fairly Good	(0.5,0.65,0.8)
Good	(0.7,0.8,0.9)
very Good	(0.85,0.95,1)

Table 4.18: Linguistic variables for criteria weights

Code	linguistic variables	Fuzzy Numbers
1	Just equal	(1,1,1)
2	Equally important	(0.5,1,1.5)
3	Weakly more important	(1,1.5,2)
4	Strongly more important	(1.5,2,2.5)
5	Very strongly more important	(2,2.5,3)
6	Absolutely more important	(2.5,3,3.5)

Table 4.19: Mean of pairwise comparisons with respect to criteria

Criteria	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5
Cr_1	(1,1,1)	(1.625,2.125,2.625)	(1.125,1.625,2.125)	(1.125,1.625,2.125)	(1.125,1.625,2.125)
Cr_2	(0.383,0.475,0.625)	(1,1,1)	(0.5,1,1.5)	(1.125,1.625,2.125)	(0.75,1.25,1.75)
Cr_3	(0.475,0.625,0.917)	(0.667,1,2)	(1,1,1)	(0.875,1.375,1.875)	(1.25,1.625,2)
Cr_4	(0.492,0.667,1.083)	(0.475,0.625,0.917)	(0.542,0.75,1.25)	(1,1,1)	(1,1.5,2)
Cr_5	(0.475,0.625,0.917)	(0.6,0.875,1.667)	(0.575,0.667,0.833)	(0.5,0.667,1)	(1,1,1)

Table 4.20: Priority weights of criteria

Criteria	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5	d'	w
Cr_1	1	1	1	1	1	1	0.306
Cr_2	0.642	1	0.957	1	1	0.642	0.197
Cr_3	0.71	1	1	1	1	0.71	0.217
Cr_4	0.521	0.871	0.824	1	1	0.521	0.160
Cr_5	0.39	0.741	0.689	0.865	1	0.39	0.119
Sum						3.263	1

Table 4.21 represents the fuzzy decision matrix (FDM) [224]. The normalized

FDM and the weighted normalized FDM are presented in Table 4.22 and Table 4.23, respectively. The fuzzy positive ideal solution (FPIS, $\tilde{A}^*$) and fuzzy negative ideal solution (FNIS, $\tilde{A}^-$) are determined as follows:

$\tilde{A}^* = [(1,1,1)]$

$\tilde{A}^- = [(0,0,0)]$

The distance between positive ideal solution S_i^* and the negative ideal solution S_i^- of each alternative rating from S^* and S^- was calculated and is shown in Table 4.24. The EFTOPSIS analysis and ranking for cloud services is also shown in Table 4.24. This calculation gives a ranking to all alternatives based on their performance. Based on Table 4.24, we observe that alternative A_{11} is the best cloud service, followed by alternatives $A_{18} > A_{12} > A_{16} > A_4 > A_3 > A_2 > A_{10} > A_{15} > A_5 > A_8 > A_{17} > A_6 > A_{14} > A_7 > A_{13} > A_{19} > A_9 > A_1$.

Table 4.21: Fuzzy decision matrix for EFTOPSIS

No	Negative	Positive	Positive	Positive	Positive
A_1	(0.35,0.5,0.65)	(0.225,0.388,0.55)	(0.275,0.425,0.588)	(0.3,0.463,0.625)	(0.4,0.538,0.675)
A_2	(0.4,0.575,0.75)	(0.525,0.65,0.775)	(0.5,0.65,0.8)	(0.425,0.538,0.65)	(0.388,0.538,0.675)
A_3	(0.35,0.538,0.725)	(0.45,0.575,0.7)	(0.375,0.538,0.7)	(0.588,0.725,0.85)	(0.4,0.538,0.675)
A_4	(0.3,0.425,0.55)	(0.45,0.613,0.775)	(0.25,0.425,0.6)	(0.475,0.613,0.75)	(0.425,0.575,0.725)
A_5	(0.45,0.613,0.775)	(0.575,0.688,0.8)	(0.6,0.725,0.85)	(0.3,0.425,0.55)	(0.413,0.538,0.663)
A_6	(0.35,0.5,0.65)	(0.35,0.5,0.65)	(0.25,0.425,0.6)	(0.425,0.575,0.725)	(0.613,0.725,0.825)
A_7	(0.475,0.613,0.75)	(0.313,0.425,0.525)	(0.35,0.5,0.65)	(0.538,0.688,0.825)	(0.65,0.763,0.875)
A_8	(0.4,0.538,0.675)	(0.3,0.463,0.625)	(0.35,0.5,0.65)	(0.538,0.688,0.825)	(0.55,0.688,0.825)
A_9	(0.375,0.538,0.7)	(0.325,0.5,0.675)	(0.35,0.5,0.65)	(0.225,0.388,0.55)	(0.513,0.65,0.775)
A_10	(0.563,0.688,0.8)	(0.538,0.688,0.825)	(0.6,0.725,0.825)	(0.325,0.5,0.675)	(0.475,0.613,0.75)
A_11	(0.3,0.425,0.563)	(0.45,0.613,0.775)	(0.55,0.688,0.825)	(0.6,0.725,0.85)	(0.375,0.5,0.638)
A_12	(0.275,0.425,0.588)	(0.4,0.538,0.675)	(0.45,0.575,0.7)	(0.375,0.538,0.7)	(0.463,0.613,0.75)
A_13	(0.375,0.538,0.7)	(0.2,0.35,0.5)	(0.425,0.575,0.725)	(0.4,0.538,0.675)	(0.5,0.65,0.8)
A_{14}	(0.4,0.575,0.75)	(0.5,0.65,0.8)	(0.463,0.575,0.688)	(0.25,0.388,0.525)	(0.45,0.613,0.775)
A_{15}	(0.4,0.538,0.675)	(0.6,0.725,0.85)	(0.375,0.538,0.7)	(0.275,0.463,0.65)	(0.475,0.613,0.75)
A_{16}	(0.4,0.575,0.75)	(0.4,0.538,0.675)	(0.65,0.763,0.875)	(0.325,0.463,0.6)	(0.625,0.763,0.875)
A_{17}	(0.438,0.575,0.7)	(0.5,0.65,0.8)	(0.375,0.538,0.7)	(0.375,0.5,0.638)	(0.463,0.575,0.675)
A_{18}	(0.25,0.388,0.525)	(0.388,0.538,0.675)	(0.5,0.65,0.8)	(0.45,0.575,0.7)	(0.3,0.463,0.625)
A_{19}	(0.425,0.575,0.725)	(0.3,0.463,0.625)	(0.413,0.538,0.663)	(0.35,0.5,0.65)	(0.475,0.613,0.75)

Table 4.22: Normalized fuzzy decision matrix for EFTOPSIS

No	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5
A_1	(0.385,0.5,0.714)	(0.265,0.456,0.647)	(0.314,0.486,0.671)	(0.353,0.544,0.735)	(0.457,0.614,0.771)
A_2	(0.333,0.435,0.625)	(0.618,0.765,0.912)	(0.571,0.743,0.914)	(0.5,0.632,0.765)	(0.443,0.614,0.771)
A_3	(0.345,0.465,0.714)	(0.529,0.676,0.824)	(0.429,0.614,0.8)	(0.691,0.853,1)	(0.457,0.614,0.771)
A_4	(0.455,0.588,0.833)	(0.529,0.721,0.912)	(0.286,0.486,0.686)	(0.559,0.721,0.882)	(0.486,0.657,0.829)
A_5	(0.323,0.408,0.556)	(0.676,0.809,0.941)	(0.686,0.829,0.971)	(0.353,0.5,0.647)	(0.471,0.614,0.757)
A_6	(0.385,0.5,0.714)	(0.412,0.588,0.765)	(0.286,0.486,0.686)	(0.5,0.676,0.853)	(0.7,0.829,0.943)
A_7	(0.333,0.408,0.526)	(0.368,0.5,0.618)	(0.4,0.571,0.743)	(0.632,0.809,0.971)	(0.743,0.871,1)
A_8	(0.37,0.465,0.625)	(0.353,0.544,0.735)	(0.4,0.571,0.743)	(0.632,0.809,0.971)	(0.629,0.786,0.943)
A_9	(0.357,0.465,0.667)	(0.382,0.588,0.794)	(0.4,0.571,0.743)	(0.265,0.456,0.647)	(0.586,0.743,0.886)
A_{10}	(0.313,0.364,0.444)	(0.632,0.809,0.971)	(0.686,0.829,0.943)	(0.382,0.588,0.794)	(0.543,0.7,0.857)
A_{11}	(0.444,0.588,0.833)	(0.529,0.721,0.912)	(0.629,0.786,0.943)	(0.706,0.853,1)	(0.429,0.571,0.729)
A_{12}	(0.426,0.588,0.909)	(0.471,0.632,0.794)	(0.514,0.657,0.8)	(0.441,0.632,0.824)	(0.529,0.7,0.857)
A_{13}	(0.357,0.465,0.667)	(0.235,0.412,0.588)	(0.486,0.657,0.829)	(0.471,0.632,0.794)	(0.571,0.743,0.914)
A_{14}	(0.333,0.435,0.625)	(0.588,0.765,0.941)	(0.529,0.657,0.786)	(0.294,0.456,0.618)	(0.514,0.7,0.886)
A_{15}	(0.37,0.465,0.625)	(0.706,0.853,1)	(0.429,0.614,0.8)	(0.324,0.544,0.765)	(0.543,0.7,0.857)
A_{16}	(0.333,0.435,0.625)	(0.471,0.632,0.794)	(0.743,0.871,1)	(0.382,0.544,0.706)	(0.714,0.871,1)
A_{17}	(0.357,0.435,0.571)	(0.588,0.765,0.941)	(0.429,0.614,0.8)	(0.441,0.588,0.75)	(0.529,0.657,0.771)
A_{18}	(0.476,0.645,1)	(0.456,0.632,0.794)	(0.571,0.743,0.914)	(0.529,0.676,0.824)	(0.343,0.529,0.714)
A_{19}	(0.345,0.435,0.588)	(0.353,0.544,0.735)	(0.471,0.614,0.757)	(0.412,0.588,0.765)	(0.543,0.7,0.857)
Weights	0.306	0.197	0.217	0.16	0.119

Table 4.23: Weighted normalized FDM for EFTOPSIS

No	Cr_1	Cr_2	Cr_3	Cr_4	Cr_5
A_1	(0.118,0.153,0.219)	(0.052,0.09,0.127)	(0.068,0.105,0.146)	(0.056,0.087,0.118)	(0.054,0.073,0.092)
A_2	(0.102,0.133,0.191)	(0.122,0.151,0.18)	(0.124,0.161,0.198)	(0.08,0.101,0.122)	(0.053,0.073,0.092)
A_3	(0.106,0.142,0.219)	(0.104,0.133,0.162)	(0.093,0.133,0.174)	(0.111,0.136,0.16)	(0.054,0.073,0.092)
A_4	(0.139,0.18,0.255)	(0.104,0.142,0.18)	(0.062,0.105,0.149)	(0.089,0.115,0.141)	(0.058,0.078,0.099)
A_5	(0.099,0.125,0.17)	(0.133,0.159,0.185)	(0.149,0.18,0.211)	(0.056,0.08,0.104)	(0.056,0.073,0.09)
A_6	(0.118,0.153,0.219)	(0.081,0.116,0.151)	(0.062,0.105,0.149)	(0.08,0.108,0.136)	(0.083,0.099,0.112)
A_7	(0.102,0.125,0.161)	(0.072,0.099,0.122)	(0.087,0.124,0.161)	(0.101,0.129,0.155)	(0.088,0.104,0.119)
A_8	(0.113,0.142,0.191)	(0.07,0.107,0.145)	(0.087,0.124,0.161)	(0.101,0.129,0.155)	(0.075,0.094,0.112)
A_9	(0.109,0.142,0.204)	(0.075,0.116,0.156)	(0.087,0.124,0.161)	(0.042,0.073,0.104)	(0.07,0.088,0.105)
A_{10}	(0.096,0.111,0.136)	(0.125,0.159,0.191)	(0.149,0.18,0.205)	(0.061,0.094,0.127)	(0.065,0.083,0.102)
A_{11}	(0.136,0.18,0.255)	(0.104,0.142,0.18)	(0.136,0.171,0.205)	(0.113,0.136,0.16)	(0.051,0.068,0.087)
A_{12}	(0.13,0.18,0.278)	(0.093,0.125,0.156)	(0.112,0.143,0.174)	(0.071,0.101,0.132)	(0.063,0.083,0.102)
A_{13}	(0.109,0.142,0.204)	(0.046,0.081,0.116)	(0.105,0.143,0.18)	(0.075,0.101,0.127)	(0.068,0.088,0.109)
A_{14}	(0.102,0.133,0.191)	(0.116,0.151,0.185)	(0.115,0.143,0.171)	(0.047,0.073,0.099)	(0.061,0.083,0.105)
A_{15}	(0.113,0.142,0.191)	(0.139,0.168,0.197)	(0.093,0.133,0.174)	(0.052,0.087,0.122)	(0.065,0.083,0.102)
A_{16}	(0.102,0.133,0.191)	(0.093,0.125,0.156)	(0.161,0.189,0.217)	(0.061,0.087,0.113)	(0.085,0.104,0.119)
A_{17}	(0.109,0.133,0.175)	(0.116,0.151,0.185)	(0.093,0.133,0.174)	(0.071,0.094,0.12)	(0.063,0.078,0.092)
A_{18}	(0.146,0.197,0.306)	(0.09,0.125,0.156)	(0.124,0.161,0.198)	(0.085,0.108,0.132)	(0.041,0.063,0.085)
A_{19}	(0.106,0.133,0.18)	(0.07,0.107,0.145)	(0.102,0.133,0.164)	(0.066,0.094,0.122)	(0.065,0.083,0.102)

Cloud service selection using EGTOPSIS with AHP We used AHP to determine the weight w_j as mentioned in section 4.3.4. The pairwise comparison was made by domain experts. We obtained the *consistency ratio* as 0.0511134. As *consistency ratio* is less than or equal to 0.1, our model is consistent, and the weights are valid.

Table 4.24: EFTOPSIS analysis and ranking for cloud services

Alternatives	S_i^*	S_i^-	CC_i^*	Rank
A_1	4.483	0.54	0.107	19
A_2	4.374	0.64	0.128	7
A_3	4.372	0.647	0.129	6
A_4	4.371	0.651	0.13	5
A_5	4.378	0.633	0.126	10
A_6	4.412	0.608	0.121	13
A_7	4.418	0.594	0.118	15
A_8	4.4	0.617	0.123	11
A_9	4.45	0.571	0.114	18
A_{10}	4.374	0.638	0.127	8
A_{11}	4.295	0.722	0.144	1
A_{12}	4.356	0.666	0.133	3
A_{13}	4.437	0.582	0.116	16
A_{14}	4.41	0.606	0.121	14
A_{15}	4.381	0.636	0.127	9
A_{16}	4.356	0.658	0.131	4
A_{17}	4.406	0.608	0.121	12
A_{18}	4.332	0.692	0.138	2
A_{19}	4.444	0.572	0.114	17

The scale of the criterion rating used for evaluating the ranking of the alternative linguistic variables is represented in Table 4.25. Table 4.26 describes the criterion rating with respect to the alternatives and the grey decision matrix (GDM). The normalized extended GDM (EGDM) for cloud services is presented in Table 4.27. We determined the positive ideal solution (A^*) and negative ideal solution (A^-) as shown in Table 4.28. To obtain A^* and A^-, we computed the upper limit Q_1 of the criteria to be 1 and the lower limit to be 0.19. Then, we calculated the separation measure of the positive ideal solution (d^+) and that of the negative

ideal solution (d^-) as presented in Table 4.28. Subsequently, we determined the relative closeness (C_i^*) as presented in Table 4.28.

Table 4.25: Scale of criteria rating

Scale					Grey number
C_1	C_2	C_3	C_4	C_5	
Very Height (VH)	Very Poor (VP)				[1, 2]
Height (H)	Poor (P)				[2, 3]
Medium Height (MH)	Medium Poor (MP)				[3, 4]
Medium (M)	Fair (F)				[4, 5]
Medium Low (ML)	Medium Good (MG)				[5, 6]
Low (L)	Good (G)				[6, 7]
Very Low (VL)	Very Good (VG)				[7, 8]

In EFTOPSIS, the fuzzy approach cannot handle incomplete data and information, though it is adequate for dealing with uncertain and imprecise data. In ETOPSIS and EFTOPSIS, there is no concept of rank reversal, which can help in updating the ranks when a non-optimal service is entered into the system [223], whereas our proposed EGTOPSIS model can deal with both the fuzziness situation and incomplete information and also incorporates the rank reversal concept. From Table 4.28, we observe that alternative A_{12} is the best cloud service, followed by alternatives $A_9 > A_8 > A_{14} > A_7 > A_{16} > A_{19} > A_{11} > A_{13} > A_6 > A_2 > A_{17} > A_{15} > A_3 > A_5 > A_{18} > A_1 > A_{10} > A_4$.

4.3.4.1 A comparative analysis of ETOPSIS, EFTOPSIS, and EGTOPSIS

In this section, we provide a comparative analysis of the proposed approaches, considering the following factors:

- Sensitivity to the weights assigned to output parameters
- Effectiveness under change in the alternatives
- Adequacy to support group decision making
- Handling of uncertainty

Table 4.26: Attribute rating for cloud services

Q_i	A_i	D_1	D_2	D_3	D_4	$\otimes G_{ij}$	Q_i	D_1	D_2	D_3	D_4	$\otimes G_{ij}$
Q_1	A_1	VP	P	VP	P	[1.41, 2.45]	Q_2	M	MH	MH	M	[3.46, 4.47]
	A_2	F	M	F	MG	[4.23, 5.23]		M	ML	L	L	[5.18, 6.19]
	A_3	P	P	P	P	[2.00, 3.00]		L	VL	VL	VL	[6.74, 7.74]
	A_4	F	MG	F	F	[4.23, 5.23]		H	H	H	VH	[1.68, 2.71]
	A_5	P	MP	P	P	[2.21, 3.22]		VL	VL	L	ML	[6.19, 7.20]
	A_6	F	MG	G	F	[4.68, 5.69]		ML	ML	L	H	[4.16, 5.24]
	A_7	G	G	G	MG	[5.73, 6.74]		L	M	ML	L	[5.18, 6.19]
	A_8	VG	VG	G	VG	[6.74, 7.74]		ML	ML	L	VL	[5.69, 6.70]
	A_9	G	G	G	G	[6.00, 7.00]		L	L	L	L	[6.00, 7.00]
	A_{10}	P	P	VP	MP	[1.86, 2.91]		VH	H	H	H	[1.68, 2.71]
	A_{11}	G	G	MG	G	[5.73, 6.74]		ML	L	ML	ML	[5.23, 6.24]
	A_{12}	VG	G	VG	G	[6.48, 7.48]		VL	VL	VL	L	[6.74, 7.74]
	A_{13}	MG	G	G	MG	[5.48, 6.48]		VL	L	VL	L	[6.48, 7.48]
	A_{14}	G	G	VG	VG	[6.48, 7.48]		ML	ML	ML	L	[5.23, 6.24]
	A_{15}	F	P	MP	MP	[2.91, 3.94]		L	L	L	VL	[6.24, 7.24]
	A_{16}	G	G	MG	G	[5.73, 6.74]		ML	ML	L	H	[4.16, 5.24]
	A_{17}	F	MG	G	F	[4.68, 5.69]		M	MH	MH	M	[3.46, 4.47]
	A_{18}	MG	G	G	MG	[5.48, 6.48]		H	H	H	VH	[1.68, 2.71]
	A_{19}	MG	G	G	VG	[6.48, 7.48]		VL	L	L	L	[6.24, 7.24]
Q_3	A_1	MG	G	G	G	[5.73, 6.74]	Q_4	VP	P	VP	P	[1.41, 2.45]
	A_2	MG	F	G	F	[4.68, 5.69]		F	M	F	MG	[4.23, 5.23]
	A_3	P	P	P	VP	[1.68, 2.71]		P	P	P	P	[2.00, 3.00]
	A_4	MP	MP	P	P	[2.45, 3.46]		F	MG	F	F	[4.23, 5.23]
	A_5	MP	P	P	VP	[1.86, 2.91]		P	MP	P	P	[2.21, 3.22]
	A_6	G	G	G	G	[6.00, 7.00]		F	MG	G	F	[4.68, 5.69]
	A_7	G	VG	MG	VG	[6.19, 7.20]		G	G	G	MG	[5.73, 6.74]
	A_8	G	G	VG	MG	[5.69, 6.96]		VG	VG	G	VG	[6.74, 7.74]
	A_9	G	G	VG	VG	[6.48, 7.48]		G	G	G	G	[6.00, 7.00]
	A_{10}	G	G	G	MG	[5.73, 6.74]		P	P	VP	MP	[1.86, 2.91]
	A_{11}	F	MP	P	F	[3.13, 4.16]		G	G	MG	G	[5.73, 6.74]
	A_{12}	G	MG	MG	VG	[5.69, 6.70]		VG	G	VG	G	[6.48, 7.48]
	A_{13}	P	P	P	MP	[2.21, 3.22]		MG	G	G	MG	[5.48, 6.48]
	A_{14}	VG	VG	VG	VG	[7.00, 8.00]		G	G	VG	VG	[6.48, 7.48]
	A_{15}	P	P	MP	MP	[2.45, 3.46]		F	P	MP	MP	[2.91, 3.94]
	A_{16}	VG	G	MG	MG	[5.69, 6.70]		G	G	MG	G	[5.73, 6.74]
	A_{17}	G	G	G	G	[6.00, 7.00]		F	MG	G	F	[4.68, 5.69]
	A_{18}	F	MP	P	F	[3.13, 4.16]		MG	G	G	MG	[5.48, 6.48]
	A_{19}	MP	P	P	VP	[1.86, 2.91]		MG	G	G	VG	[6.48, 7.48]

Sensitivity analysis The sensitivity analysis determines the robustness of our proposed approaches. We tested the effect of a change in priorities and final weights on the proposed approaches. For each QoS attribute, we experimented with varying priorities and their corresponding final weights. We moderately adjusted the weights of the QoS attributes one at a time and observed the impact

Table 4.27: Normalized EGDM for cloud services

	Q_1		Q_2		Q_3		Q_4		Q_5	
A_1	0.43	0.56	0.19	0.32	0.72	0.85	0.19	0.32	0.72	0.85
A_2	0.21	0.34	0.55	0.68	0.72	0.72	0.55	0.68	0.59	0.72
A_3	0	0.13	0.26	0.39	0.21	0.34	0.26	0.39	0.21	0.34
A_4	0.65	0.79	0.55	0.68	0.31	0.44	0.55	0.68	0.31	0.44
A_5	0.07	0.21	0.29	0.42	0.24	0.37	0.29	0.42	0.24	0.37
A_6	0.33	0.47	0.6	0.74	0.75	0.88	0.61	0.74	0.75	0.88
A_7	0.21	0.34	0.74	0.88	0.78	0.9	0.75	0.88	0.78	0.90
A_8	0.14	0.27	0.87	1	0.72	0.87	0.88	1	0.72	0.87
A_9	0.1	0.23	0.77	0.91	0.81	0.94	0.78	0.91	0.81	0.94
A_{10}	0.65	0.79	0.24	0.38	0.72	0.85	0.25	0.38	0.72	0.85
A_{11}	0.2	0.33	0.74	0.88	0.4	0.52	0.75	0.88	0.4	0.52
A_{12}	0	0.13	0.83	0.97	0.72	0.84	0.84	0.97	0.72	0.84
A_{13}	0.04	0.17	0.7	0.84	0.28	0.41	0.71	0.84	0.28	0.41
A_{14}	0.2	0.33	0.83	0.97	0.88	1	0.84	0.97	0.88	1
A_{15}	0.07	0.2	0.37	0.51	0.31	0.44	0.38	0.51	0.31	0.44
A_{16}	0.33	0.47	0.74	0.88	0.72	0.84	0.75	0.88	0.72	0.84
A_{17}	0.43	0.56	0.6	0.74	0.75	0.88	0.61	0.74	0.75	0.88
A_{18}	0.65	0.79	0.7	0.84	0.4	0.52	0.71	0.84	0.4	0.52
A_{19}	0.07	0.2	0.83	0.97	0.24	0.37	0.84	0.97	0.24	0.37

Table 4.28: EGTOPSIS analysis and ranking for cloud services

	d^+	d^-	A^+	A^-	C_i^+	**Rank**
A_1	0.563162	0.349619	1	0.19	0.383026	17
A_2	0.340162	0.473053	0	0.79	0.581707	11
A_3	0.562498	0.447627	1	0.21	0.44314	14
A_4	0.5896	0.287524	1	0.19	0.327803	19
A_5	0.544004	0.410383	1	0.21	0.429996	15
A_6	0.33499	0.492483			0.595165	10
A_7	0.227103	0.591541			0.722586	5
A_8	0.178727	0.655258			0.785695	3
A_9	0.166872	0.649084			0.795489	2
A_{10}	0.619909	0.311969			0.334774	18
A_{11}	0.343847	0.517125			0.60063	8
A_{12}	0.148246	0.686024			0.822305	1
A_{13}	0.372908	0.554014			0.597693	9
A_{14}	0.184493	0.659408			0.78138	4
A_{15}	0.48123	0.434997			0.47477	13
A_{16}	0.29645	0.53956			0.645399	6
A_{17}	0.377354	0.467993			0.55361	12
A_{18}	0.534066	0.390268			0.422215	16
A_{19}	0.373112	0.595637			0.614852	7

of the changes in weights on the final decisions. In this way, the performance of each QoS attribute and its results were analyzed and applied to the extended versions of TOPSIS, Fuzzy TOPSIS, and Grey TOPSIS. The sensitivity analysis for processing performance, I/O operational consistency, disk storage performance, and memory performance attributes against 19 cloud services of ETOPSIS are depicted in Fig. 4.10, Fig. 4.12, Fig. 4.11, and Fig. 4.13 respectively.

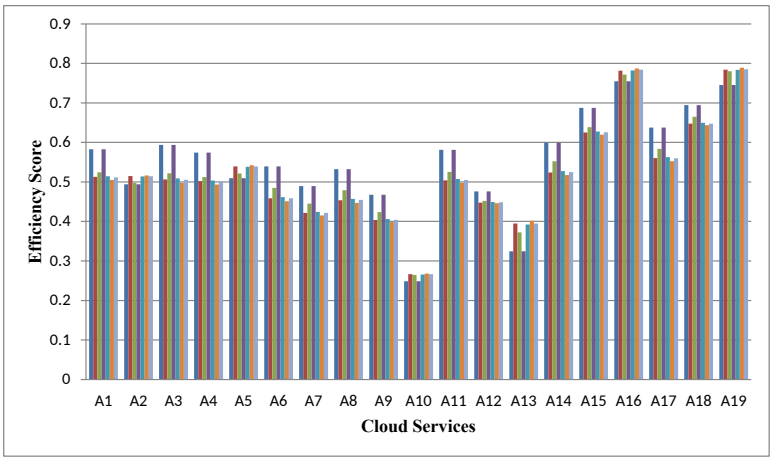


Figure 4.10: Sensitivity analysis of ETOPSIS for processing performance

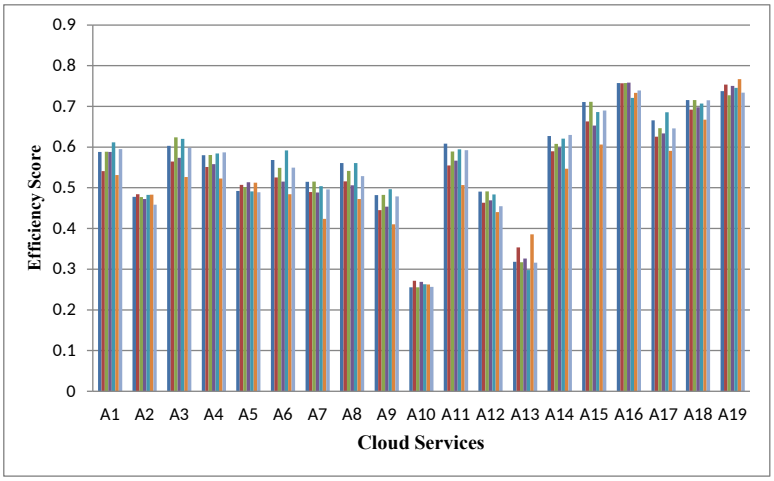


Figure 4.11: Sensitivity analysis of ETOPSIS for I/O operational consistency

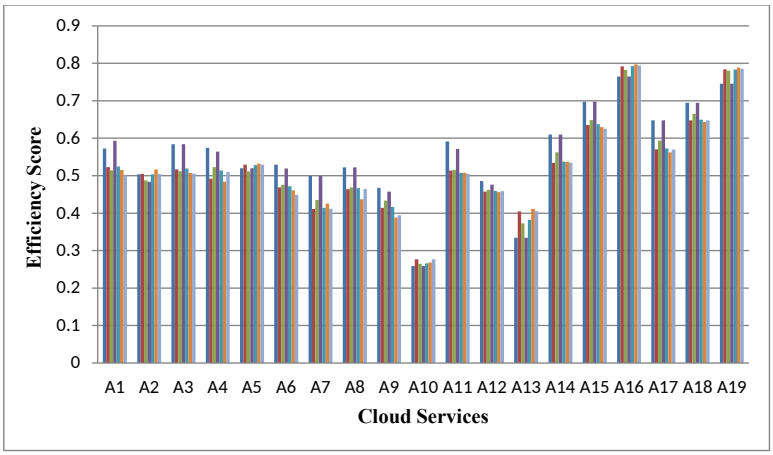


Figure 4.12: Sensitivity analysis of ETOPSIS for disk storage performance

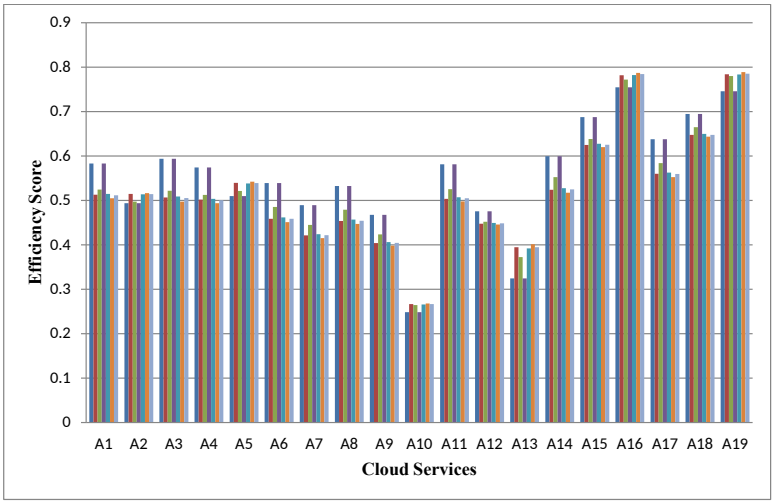


Figure 4.13: Sensitivity analysis of ETOPSIS for memory Performance

In Fig. 4.10, Fig. 4.11, Fig. 4.12, and Fig. 4.13, the cloud services or alternatives and efficiency scores of the alternatives are presented on the x-axis and y-axis, respectively. We analyzed the efficiency scores of the alternatives by varying the weights of the processing performance, I/O operational consistency, disk storage performance, and memory performance one at a time and reported the efficiency scores in descending order for each attribute. We noted the top ten alternatives that gave the best efficiency scores for different weights in all of the cases. From Fig. 4.10, we notice that al-

ternatives $A_{19}, A_{16}, A_{18}, A_{15}, A_{17}, A_{14}, A_5, A_{11}, A_3,$ and A_1 give the best efficiency scores for different weights of processing performance attribute. From Fig. 4.11, the alternatives $A_{16}, A_{19}, A_{18}, A_{15}, A_{17}, A_{14}, A_{11}, A_1, A_3,$ and A_4 give the best efficiency scores for different weights of I/O operational consistency attribute. Likewise, from Fig. 4.12, we observe that alternatives $A_{16}, A_{19}, A_{15}, A_{18}, A_{17}, A_{14}, A_1, A_5, A_3,$ and A_4 give the best efficiency scores for different weights of disk storage performance attribute. The alternatives $A_{16}, A_{19}, A_{18}, A_{15}, A_{17}, A_{14}, A_{11}, A_1, A_3,$ and A_4 show the best efficiency scores for the different weights of memory performance attribute as shown in Fig. 4.13.

In our experiments, we observed that the priority of a alternative is proportional to the weight of the corresponding attribute and that these changes have no significant effect on the alternatives ranking and efficiency. In addition, the final decision does not change in most of the cases. After all of these experiments, we found that alternative A_{16} is the best cloud service in all cases and also has a high number of virtual cores (i.e., eight).

The sensitivity analyses for processing performance, I/O operational consistency, disk storage performance, and memory performance attributes against 19 cloud services of EFTOPSIS are depicted in Fig. 4.14, Fig. 4.15, Fig. 4.16, and Fig. 4.17 respectively.

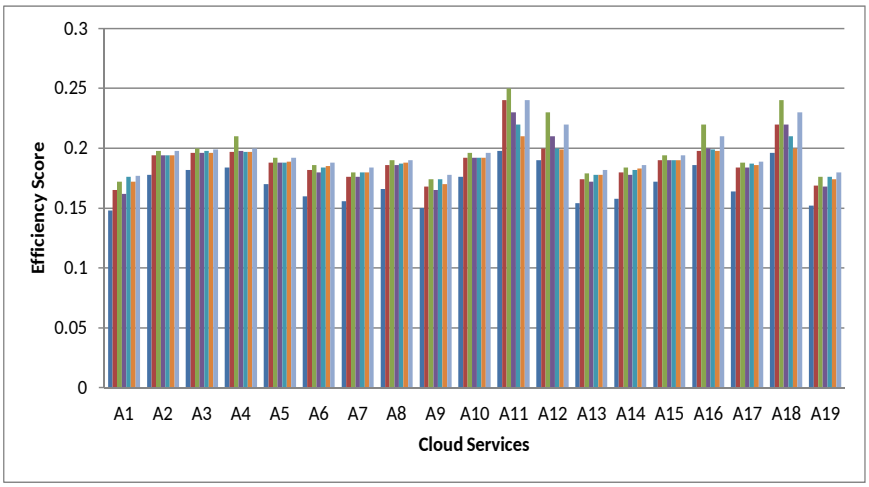


Figure 4.14: Sensitivity analysis of EFTOPSIS for processing performance

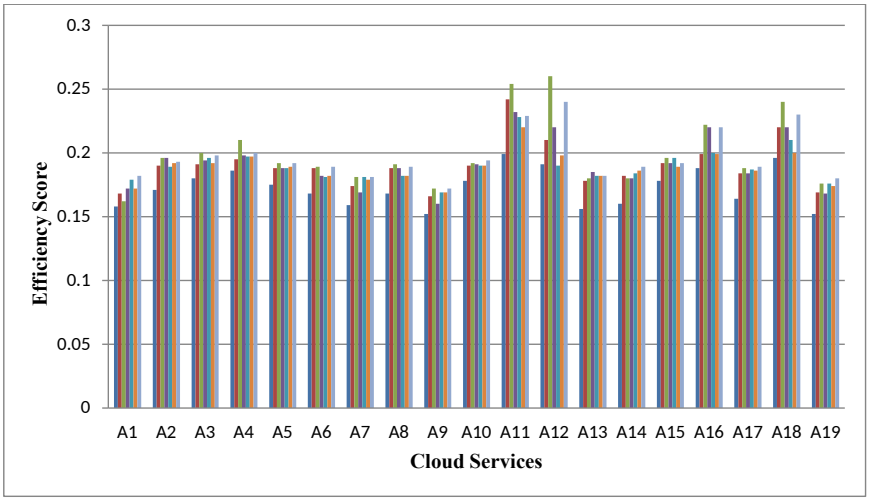


Figure 4.15: Sensitivity analysis of EFTOPSIS for I/O operational consistency

We analyzed the efficiency scores of the alternatives by varying the weights of the processing performance, I/O operational consistency, disk storage performance, and memory performance one at a time and reported the efficiency scores in descending order for each attribute. We noted the top ten alternatives that gave the best efficiency scores for different weights in all of the cases. The alternatives $A_{11}, A_{18}, A_{12}, A_{16}, A_4, A_3, A_2, A_{10}, A5$, and A_{15} show the efficiency scores for the different weights of processing performance attribute as shown in Fig. 4.14. From Fig. 4.15, we observe that alternatives $A_{11}, A_{12}, A_{18}, A_{16}, A_4, A_3, A_2, A_{15}, A_{10}$, and A_5 give the best efficiency scores for different weights of I/O operational consistency attribute. Similarly, in Fig. 4.16, the alternatives $A_{11}, A_{18}, A_{12}, A_{16}, A_4, A_3, A_2, A_{10}, A_5$, and A_{15} give the best efficiency scores for different weights of disk storage performance attribute. Likewise, in Fig. 4.17, we notice that alternatives $A_{11}, A_{12}, A_{18}, A_{16}, A_4, A_3, A_2, A_{15}, A_5$, and A_6 give the best efficiency scores for different weights of memory performance attribute.

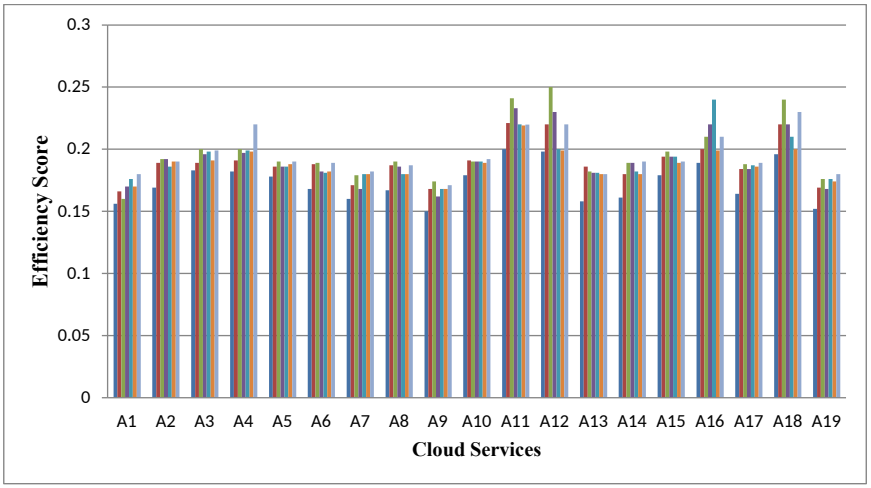


Figure 4.16: Sensitivity analysis of EFTOPSIS for disk storage performance

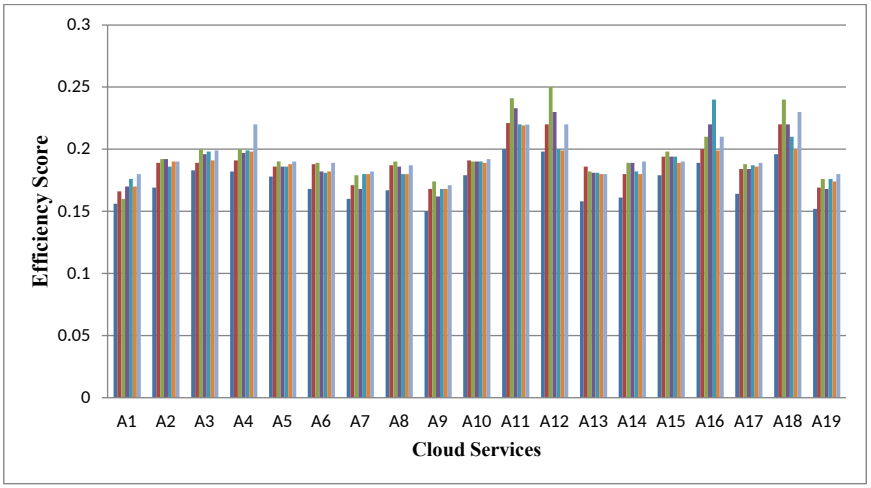


Figure 4.17: Sensitivity analysis of EFTOPSIS for memory Performance

Based on these experiments, we notice that there are no significant changes in the efficiency scores of the alternatives or their ranking if varying weights are given. In addition, the final decision does not change in most of the cases. After all of these experiments, we found that alternative A_{11} is the best cloud service in all cases and also has a high number of virtual cores (i.e., eight).

Fig. 4.18, Fig. 4.19, Fig. 4.20, and Fig. 4.21 illustrate the sensitivity analyses for processing performance, I/O operational consistency, disk storage performance,

and memory performance attributes against 19 cloud services of EGTOPSIS method. In Fig. 4.18, Fig. 4.19, Fig. 4.20, and Fig. 4.21, the x-axis represents cloud services, and we projected the efficiency scores of the alternatives on y-axis.

From Fig. 4.18, we perceive that alternatives $A_{12}, A_9, A_8, A_{14}, A_7, A_{16}, A_{19}, A_{11}, A_{13}$, and A_6 (i.e., descending order) give the efficiency scores for the various weights of processing performance attribute. Likewise, in Fig. 4.19, the alternatives $A_{12}, A_8, A_9, A_{14}, A_7, A_{16}, A_{19}, A_{11}, A_{13}$, and A_6 (i.e., descending order) gave the best efficiency scores for different weights of I/O operational consistency attribute. From Fig. 4.20, we notice that alternatives $A_{12}, A_9, A_8, A_{14}, A_7, A_{16}, A_{19}, A_{11}, A_{13}$, and A_6 (i.e., descending order) give the best efficiency scores for different weights of disk storage performance attribute. Similarly, in Fig. 4.21, we notice that alternatives $A_{12}, A_8, A_9, A_{14}, A_7, A_{16}, A_{19}, A_{11}, A_{13}$, and A_6 (i.e., descending order) give the best efficiency scores for different weights of memory performance attribute.

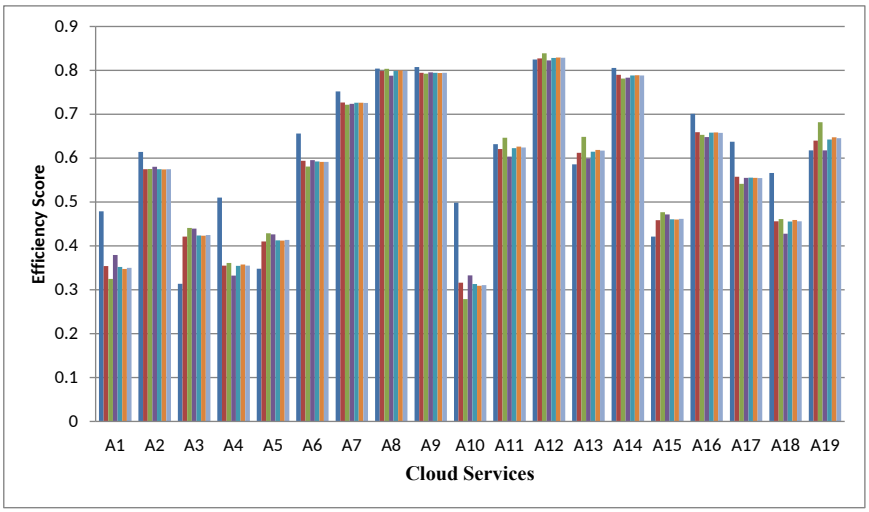


Figure 4.18: Sensitivity analysis of EGTOPSIS for processing performance

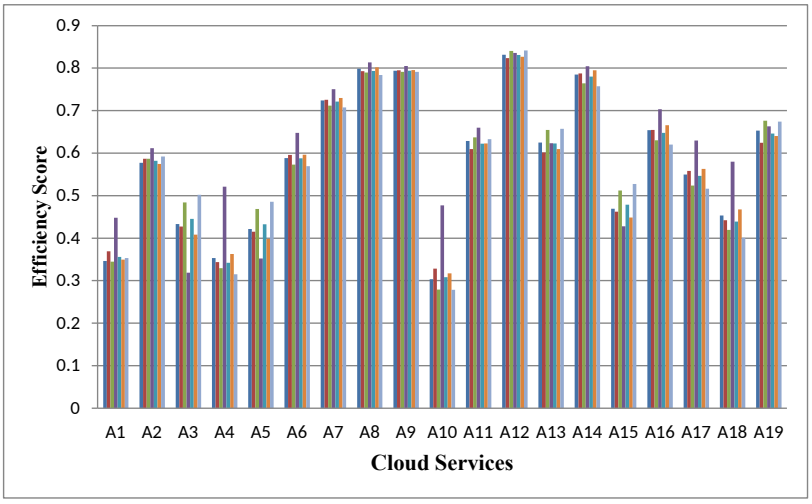


Figure 4.19: Sensitivity analysis of EGTOPSIS for I/O operational consistency

Based on our experiments, we observed that the priority of a alternative is proportional to the weight of the corresponding attribute and that these changes have no significant effect on the alternatives ranking and efficiency. In addition, the final decision does not change in most of the cases. After all of these experiments, we found that alternative A_{12} is the best cloud service in all cases and also has a high number of virtual cores (i.e., eight).

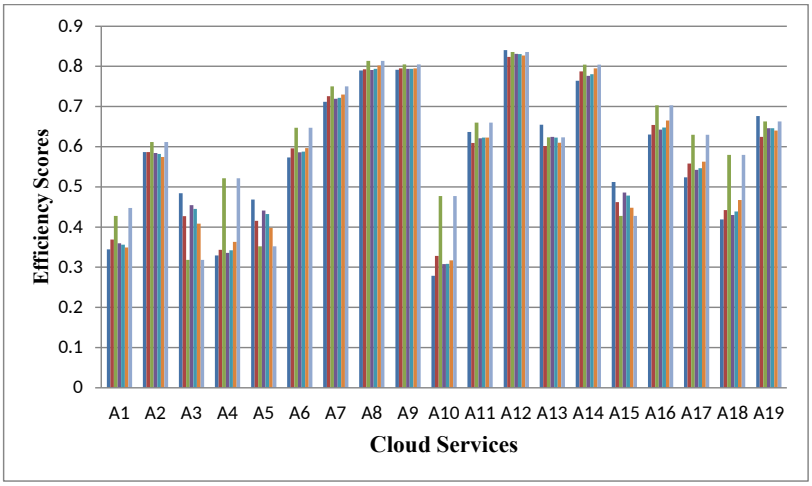


Figure 4.20: Sensitivity analysis of EGTOPSIS for disk storage performance

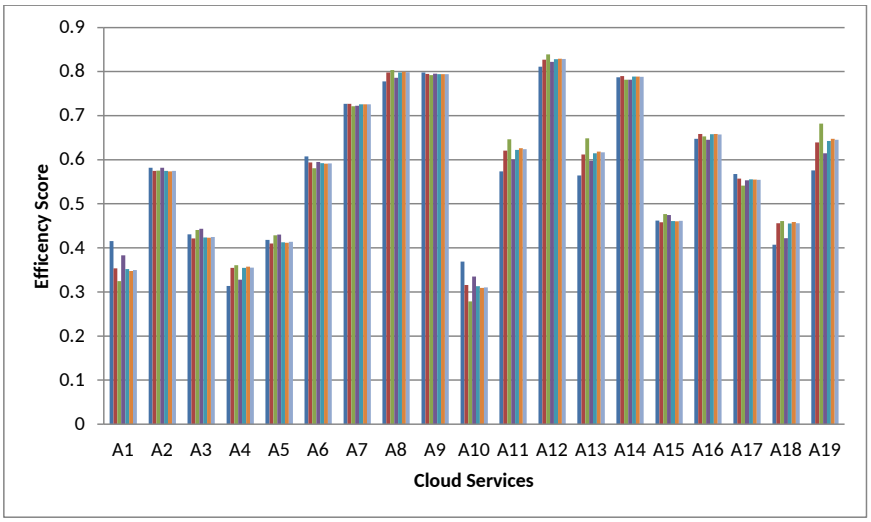


Figure 4.21: Sensitivity analysis of EGTOPSIS for memory Performance

Adequacy under change in alternatives An analysis of adequacy under a change in the alternatives was performed on our proposed methods similar to that described in section 4.2.4.1. In this case, the cloud service selection method produces a consistent order of priorities for the alternatives. In the extended versions of the TOPSIS, Fuzzy TOPSIS, and Grey TOPSIS application cases, with 19 alternatives and equal weights for all criteria, the rankings were ($A_{15}, A_{17}, A_{18}, A_{19}, A_{16}, A_{14}, A_{11}, A_3, A_7, A_1, A_6, A_8, A_4, A_5, A_9, A_{12}, A_{13}, A_{10}, A_2$), ($A_{11}, A_{18}, A_{12}, A_{16}, A_3, A_4, A_2, A_{10}, A_{15}, A_5, A_{13}, A_{17}, A_8, A_6, A_{14}, A_7, A_{19}, A_9, A_1$), and ($A_{12}, A_9, A_8, A_{14}, A_7, A_{16}, A_{19}, A_{11}, A_{13}, A_6, A_2, A_{17}, A_{15}, A_3, A_5, A_{18}, A_1, A_{10}, A_4$), respectively.

To test the extended versions of TOPSIS and Grey TOPSIS, we added an additional alternative with equal priority weights to the existing alternatives. In most of the test cases, the results do not show any significant changes in the ranking of alternatives. Similarly, the same sequence of tests was applied to the EFTOPSIS method. The order of priority remains the same in all cases tested, and the results do not show any changes in the ranking of the alternatives.

Adequacy to support group decision making The extended versions of the Fuzzy TOPSIS and Grey TOPSIS methods allow aggregation of the judgments of multiple decision makers. The quantity of data needed by Extended Fuzzy TOPSIS is greater than for the extended versions of the TOPSIS and Grey TOPSIS methods. An increase in the number of decision makers will accordingly cause a large increase in the computational complexity of EFTOPSIS compared to the extended versions of TOPSIS and Grey TOPSIS. Because of the influence of computational complexity, the extended versions of TOPSIS and Grey TOPSIS are preferable to the extended version of Fuzzy TOPSIS.

Handling of uncertainty In the EFTOPSIS and EGTOPSIS methods, we utilize fuzzy set theory to deal with the intrinsic lack of clarity in the data for selecting cloud services. In these methods, the fuzzy number structure is the main resource for quantifying vagueness. Owing to the uncertainty of judgments of quantitative variables, triangular membership function parameters are selected. To evaluate the alternatives, the decision makers used linguistic terms for the different decision criteria. In EFTOPSIS and EGTOPSIS, we employed a pairwise comparison using comparative linguistic variables. Tables 4.4 and 4.25 describe the judging of weights of the criteria and the alternatives.

4.4 Related Work

As there is a growth of popularity in Cloud Computing, it has been applied in scientific computing and web services selection based on attributes including security, assurance, accountability, performance, and cost. Tran et al. [225] developed an AHP-based ranking algorithm for web service selection, considering different QoS attributes of web services using a QoS ontology for obtaining various QoS information and constraints (tendency, mandatory, weighting, relationship, grouping, etc.) of web services. Garg et al. [204] developed a framework that could rank cloud services using AHP depending on the QoS requirements. Godse et al. [226] proposed a method to rank various business functionalities of

SaaS services using AHP, considering functionality, architecture, usability, vendor reputation, and cost. Although AHP is an efficient method for making decisions, it does not consider the uncertainty of decisions in determining pairwise comparison. In this context, fuzzy AHP is introduced to overcome this difficulty, allowing decision-makers to use fuzzy ranking in place of exact ranking. Kwon et al. [206] proposed a decision method to choose a cloud service by considering various QoS attributes using fuzzy AHP. Pernici et al. [227] presented a novel approach using hierarchical fuzzy inference systems for selecting adaptation strategies in service-oriented systems. Shivakumar et al. [203] adopted a fuzzy multi-attribute decision-making method for ranking cloud service. In their proposed method, QoS parameters are considered as fuzzy sets and used fuzzy 'and' operator to model the final decision as the intersection of the underlying fuzzy sets. Bedi et al. [205] proposed a model for cloud service selection based on a cooperative model of society and handling uncertainty through fuzzy inference system. But, in fuzzy AHP, the decision maker has to give membership value of alternatives which might be within an interval. However, it is often difficult for the expert to precisely quantify her selected number within the interval $[0, 1]$. Silas et al. [228] presented a methodology for selecting middle-ware services in cloud computing environment using ELECTRE based on an MCDM model. The attributes considered in the process of service selection are scalability, flexibility, time, service-cost, trust, and capability. Li et al. [208] proposed a framework namely CloudCmp that compared various cloud services. CloudCmp measures various services offered by a cloud along metrics that directly reflect their impact on the performance of customer applications. Yan et al. [229] developed a systematic framework on top of a hybrid cloud platform for business that could automatically recommend and select cloud service as per business requirements, company policies and standards, and its specifications. Esposito et al. [207] proposed a novel method, in which, they applied the fuzzy set theory to set the preferences of the users and Dempster-Shafer theory and game theoretic method to select the services. Zheng et al. [230] presented a QoS ranking prediction framework to select cloud service by taking benefits of the past cloud service usage experiences of the user. Xu et al. [231] adopted a non-parametric

DEA method for evaluating cloud services based on the values of price/hour, virtual core, compute units, memory, and disk.

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [232] is an MCDM method used to determine the best alternative, which is defined as the one having the shortest distance from the PIA (Positive Ideal Alternative) and the longest distance from the NIA (Negative Ideal Alternative). Fuzzy TOPSIS and Grey TOPSIS are the combinations of fuzzy set theory and Grey theory to TOPSIS respectively, used to choose the best alternative in a fuzzy environment. In fuzzy TOPSIS method, there is no concept of rank reversal that helps in updating new ranks when the non-optimal service is entered into the system [223]. A fuzzy TOPSIS method proposed for selecting web services when a group of users has different opinions during evaluation [233]. Kabir et al. [218] evaluated the major factors for quality of web service from the viewpoint of users and developed a systematic multiple attribute evaluation model using TOPSIS and Fuzzy TOPSIS. Saripalli et al. [234] described a fuzzy TOPSIS method for selecting cloud services. They used the wide-band Delphi method for evaluating the weights of criteria and a simple additive weight method for ranking the cloud services.

4.5 Summary

In this chapter, we presented two different approaches for evaluating cloud services based on DEA and TOPSIS. We proposed EDEA and ESDEA methods integrated with AHP/ANP for evaluating realtive performance of the cloud services based on QoS attributes specified by a user. We proposed ETOPSIS, EFTOPSIS, and EGTOPSIS methods to evaluate the ranks of the alternatives (or cloud services) using AHP/ANP/FAHP to determine the weigts of criteria (or QoS attributes). We conducted comprehensive comparative analysis of our proposed approaches considering various factors.

Chapter 5

QoS-aware Big Service Composition

This chapter describes QoS-aware big service composition using the MapReduce-Based Evolutionary Algorithm with Guided Mutation. The rapid growth in big services from various domains, such as the Internet of Things, cloud computing, and social networking, makes it hard to select and compose from these services while satisfying the requirements of customers as it involves enormous computational time and cost. Hence, handling these services while satisfying user constraints with reduced cost and time is becoming a crucial factor in big service composition in a big data environment. Some researchers have proposed new approaches [54, 235] for solving the QoS-aware big service selection problem. However, these methods have some intrinsic defects, including low premature convergence rate and an increase of search space within local optima. Further, because of the rapid growth of big data and services, compu tational intelligence algorithms require more iterations and more computation time to find the optimal or near-optimal solutions. This fact motivated us to develop a novel MapReduce-Based Evolutionary Algorithm with Guided Mutation (MR-EA/G) for big service composition.

5.1 MapReduce for Big Service Composition

The MapReduce (MR) model is a popular parallel processing framework for data-intensive applications and is used in the distributed processing of large data sets [236]. The MR model consists of two main functions: Map() function and Reduce() function. The paradigm behind MR is to split a very large quantity of data into smaller groups or into equal splits that can run in parallel. Each split is designated with a Map() function.

The Map() function takes as input key/value pairs and applies a map operator on each key/value pair to construct a set of intermediate key/value pairs. The Reduce() function takes the intermediate key/value pairs as input and produces the final key/value pairs. The reducer gathers together all the intermediate key/value pairs and sorts them so they are associated with the same key within the list. Conceptually, the Map() and Reduce() functions are given as follows:

Map: $< In\ k_1, In\ v_1 > \rightarrow list < Out\ k_1, Out\ v_1 >$
Reduce: $< Out\ k_1, list(Out\ v_1) > \rightarrow list < Out\ k_2, Out\ v_2 >$

The input data files for MapReduce are conventionally stored in a distributed file system. The Map() function accepts $< In\ k_1, In\ v_1 >$ as an input series and provides $list < Out\ k_1, Out\ v_1 >$ as output series, where the key represents the offset of the file and the value represents the values of the distinct lines in the file. The output of the Map() function series is $list < Out\ k_1, Out\ v_1 >$ pairs where the key represents the services and the value represents the QoS parameters of the candidate service.

The Map() function is presented to shuffle(), which sorts the values by key and obtains the values for various keys. It provides a series of tuples in the form of $< Out\ k_1, Out\ v_1, Out\ v_2, \ldots >$. In our approach, Map() takes a list of candidate services and associated QoS parameters to create the $< key, value >$ pairs. For each service, a list of QoS parameters is sent to the reducer. Finally, Reduce() identifies the best candidate service among the available services (see Fig. 5.1).

The Reduce() function $< Out\ k_1, list(Out\ v_1) >$ accepts the pairs as an input of tuples in the form $< Out\ k_1, Out\ v_1, Out\ v_2, \ldots >$ from the Map function()

and provides the output series as $< Out\ k_2, Out\ v_2 >$ pairs. The output of key/value pairs is stored in a distributed file system. The simplified execution flow of MapReduce-based big service composition is shown in Fig. 5.1.

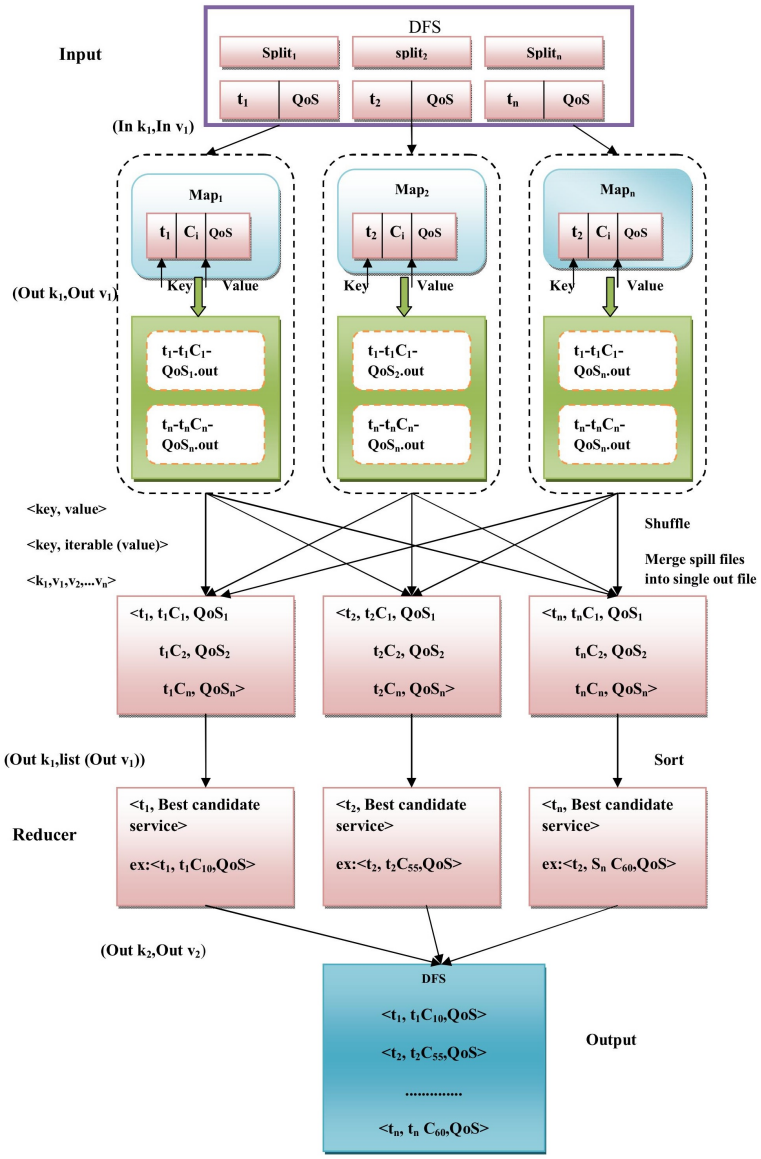


Figure 5.1: MapReduce Execution Flow for Big service composition

Service composition aims to select a set of services, one from each abstract ser-

vice class, that maximize the overall utility, while satisfying requirements of the users. But the chosen candidate services from each abstract service class with highest utility value may not guarantee a correct solution that satisfies the end-to-end constraints. Hence, the composition consists of a combination of selected candidate services from each abstract service class. Still, not all candidate services are potential candidates for the solution. Thus, skyline operation is used to identify the potential candidates from each abstract service class for the composition. Later, the non-potential candidates are pruned to reduce the search space [237, 238, 239]. Let CS_1 and CS_2 be two candidate services in a service composition having QoS parameter vectors $QoS(CS_1) = (qos_1, qos_2, \ldots, qos_n)$ and $QoS(CS_2) = (qos_1, qos_2, \ldots, qos_n)$, respectively. Pareto-based dominance is described as follows [237, 239, 240]:

The candidate service CS_1 said to dominate CS_2 if and only if the following two conditions are true:

- $QoS(CS_1) \geq QoS(CS_2)$ if CS_1 is strictly better than or equal to CS_2 for all QoS parameters.
- $QoS(CS_1) > QoS(CS_2)$ if CS_1 is strictly better than CS_2 for at least two QoS parameters.

The contrast between two QoS parameter values relies upon the classification of the QoS parameters, i.e., negative (response time and cost) and positive (availability and reliability), as discussed in chapter 1.

In a Big service environment, the process of skyline service selection becomes a time consuming process because of increase in services and their QoS parameters. Thus, MapReduce paradigm is used for parallel skyline service selection to improve the efficiency [241, 242]. The skyline service selection strategy consists of three phases [240] (see Fig. 5.2): map, compute local skyline, and reduce. In the map phase, the data points are split into multiple blocks by the master server based on the QoS parameters. In the local skyline computation phase, local skylines are generated from the subdivided data blocks by each slave server. In the reduce phase, the global skyline fabricated by the local skyline is produced by a slave server. This entire process is applied to all the services being evaluated.

The quality of the services selected depends on the computation of local skylines and the execution of the concatenation process. Therefore, the effectiveness of the MR-skyline process and the QoS depend on the most proficient method to investigate the conveyed parallelism to speed up the map stage. The efficiency of the mapping relies upon data space partitioning, and data points are split into separate districts.

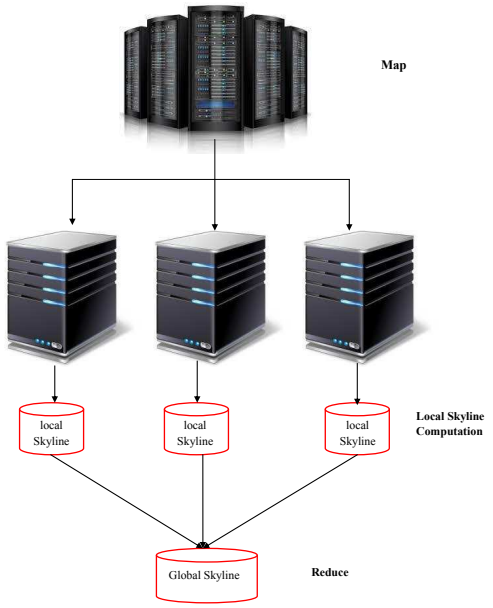


Figure 5.2: MR-Skyline for service selection to optimize QoS

The primary objectives are to accomplish load balancing, to avoid rehashed computations, and to fit new services into the local memory when old services are dropped and new services are uploaded dynamically. As the number of candidate services increases the estimated cost of expensive skyline services, in the middle phase (local skyline computation), the local skyline process reduces the unnecessary services and delivers the others to the reduce phase. Finally, all local skylines are combined, and thus a global skyline is produced during the reduce phase.

There are three types of MR-skyline methods: MR-grid, MR-angular, and MR-block [240]. Among these three methods, block elimination is used effectively for reducing the search space of data points. Hence, we adopted the block elimi-

nation method to process our skyline operation to reduce the redundant services in a service composition.

5.2 MapReduce-Based Modified EA/G for Big Service Composition

5.2.1 Modeling QoS-Aware Big Service Composition

Let a big service composition task BT be a set of n tasks (or abstract services) $\{T_1, T_2, T_3, \ldots, T_n\}$ such that each task $T_i \in BT$ accommodates a set of services, each of which is a subset of the available services $BS = \{ST_1 = \{s_{1\,1}, s_{1\,2}, \ldots, s_{1\,i}\}, ST_2 = \{s_{2\,1}, s_{2\,2}, \ldots, s_{2\,j}\}, \ldots, ST_p = \{s_{p\,1}, s_{p\,2}, \ldots, s_{p\,m}\}\}$, where i, j, and m denote the number of candidate services in ST_1, ST_2, and ST_p, respectively. The services in a set $ST_i = \{s_{i\,1}, s_{i\,2}, \ldots, s_{i\,x}\}$ comprise a collection of x services that can address task T_i. The candidate services in the subset ST_i have similar functionalities but can differ in their QoS parameters. Let a set of QoS parameters $P = \{p_1, p_2, \ldots, p_{|m|}\}$ be associated with each service in the given service set ST_i. Further, each QoS parameter in P is associated with constraints and preferences given by the users. The QoS parameters and their normalizations are described in chapter 1.

As mentioned in chapter 1, the goal in a big service composition problem is to maximize the aggregate values of QoS parameters by considering the execution paths and user preferences. In other words, the goal is to find the value of X that maximizes a global utility function. We use the Simple Additive Weight model as our utility function to evaluate the fitness of a composite big service stated by equation 5.1.

$$d(X) = \sum\nolimits_{p \in P} P_p(Agg_q(X)) \times W_q \tag{5.1}$$

$$\text{having } \sum\nolimits_{p \in Q} W_p = 1$$

The aggregate functions of all QoS parameters are mentioned in chapter 2. We assumed the tasks to be sequential in the composition plan while transforming other structures into a sequential model using [19, 25, 243]. For example, the response time (rt) and price (pr) of big service composition may be considered. Then, an overall utility function for all execution paths is computed by equation 5.2.

$$Agg_p(X) = \frac{p_{pr}^{max} - Pr}{p_{pr}^{max} - p_{pr}^{min}} \times W_{pr} + \frac{p_{rt}^{max} - x}{p_{rt}^{max} - p_{rt}^{min}} \times w_{rt} \tag{5.2}$$

p_{pr}^{max} and p_{pr}^{min} are the maximum and minimum values, respectively, of the price parameter over all possible paths. p_{rt}^{max} and p_{rt}^{min} are the maximum and minimum values, respectively, of the response time parameter over all possible paths. The variables w_{pr} and w_{rt} represent the weights of price and response time, respectively, over all possible paths. These weights or preferences are obtained by the Analytical Hierarchical Process (AHP) [244]. For instance, w_{pr} = 0.2 and w_{rt} = 0.1 means that the price is twice as important as the response time. The scale that we use, ranging from 1 to 9, is presented in Table 4.4. The pairwise comparison was made with the help of domain experts. The weights of all QoS parameters are given in Table 5.1. After calculating the weights, we check the consistency of these weights using a parameter known as the *consistency ratio*, that tells us about inconsistencies in the matrix. The results are acceptable and consistent if and only if *consistency ratio* is less than or equal to 0.1.

Table 5.1: Priority weights of criteria

	Availability	Reliability	Response time	Cost	Throughput	Weights vector
Availability	1.00	0.25	3.00	0.50	2.00	18.94%
Reliability	4.00	1.00	0.50	0.33	0.50	17.05%
Response time	0.33	2.00	1.00	0.25	2.00	15.81%
Cost	2.00	3.00	4.00	1.00	0.50	27.83%
Throughput	0.50	2.00	0.50	2.00	1.00	20.37%

5.2.2 Modified Evolutionary Algorithm with Guided Mutation (EA/G)

The Evolutionary Algorithm with Guided Mutation (EA/G) [245] is a combination of the Estimation of Distribution Algorithm (EDA) and the Genetic Algorithm (GA). This algorithm makes use of both global and local information on solutions found so far to produce offspring. Global information is used to build a probability model of promising solutions. An offspring is produced by sampling from this model, in combination with partial reuse of parent solutions. The EA/G process is as follows:

We generate an initial solution randomly. From this solution, we produce offspring based on a probability guidance model in such a way that the newly generated solution retains similarity to the parent solution. To achieve this, we retain a certain percentage (user-specified) of the elements of the parents in the offspring generation process. Thus, we are able to control the similarity between an offspring and its parent, and the resultant solution can fall within or close to a promising area as characterized by the probability model. The problem with this approach is the random generation of the initial solution, which may decrease the convergence rate of the EA/G algorithm. To overcome this issue, we propose a modified EA/G in which we filter out redundant services among the available services to reduce the search space. Further, we generate the initial solution on this reduced search space using a novel skyline operator.

Solution encoding The objective is to find a service composition with the top k candidate services from the abstract services. The chromosome encoding model is shown in Fig. 5.3. We designed the chromosome to represent all the abstract services of the composition, such as $task_1, \ldots, task_n$, shown in Fig. 5.3. Each cell of the chromosome in Fig. 5.3 corresponds to an abstract service, and the value in each cell represents a concrete service chosen from the set C_i of the candidate services for the corresponding abstract service. For example, $task_2$ selects candidate service CS_3 to be included in the composition. Similarly, $task_4$ selects candidate service CS_5, $task_6$ selects candidate service CS_3, and so on.

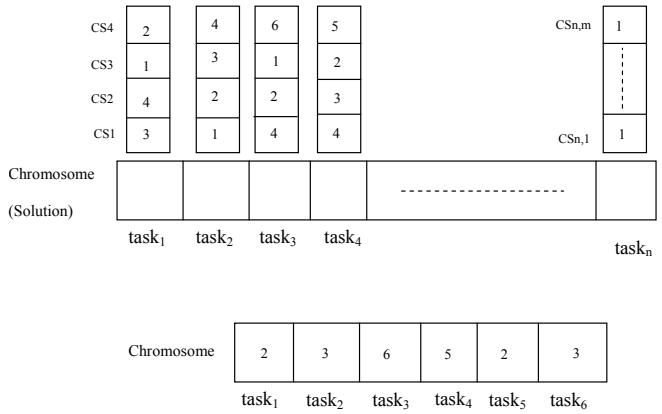


Figure 5.3: Chromosome encoding model

Initial solution using skyline operator Let V be a set of candidate services (C_i) for each abstract service t_i, where $i = 1, 2, 3, \ldots, m$. Initially, we populate all the candidate services to a set $\Re$. From $\Re$, we select some candidate services for the initial population based on the fitness value (calculation of the fitness function is given in section 5.2.1). The candidate service C_i having the maximum added to the initial population, which is denoted by τ. From $\Re$, the candidate services that have fitness values nearest to candidate service C_i in τ are selected for further processing and are added to ζ and removed from $\Re$. ζ is added to χ, which represents the shortlisted services. For the services in χ, we check whether there are candidate services from τ with the nearest fitness values. From among these selected candidate services, we find the candidate service having the nearest fitness value, add it to τ, and delete it from χ. This process is repeated until all the services in $\Re$ are covered. The final τ list becomes the initial solution for the given input.

Initial probabilities In our proposed modified EA/G approach, we choose a univariate marginal distribution for the estimation of the distribution of solutions in the search space. We form a probability vector $PV = (pv_1, \ldots, pv_R)$ to characterize the solution distribution, where R indicates the number of candidate services considered (i.e., $|V|$). PV is initialized with the N_c initial solutions. Once the initial solutions are generated, we count the number of solutions containing

the candidate service s_i, and each pv_i is set to the frequency of occurrence of s_i, i.e., the count divided by N_c. We make use of this vector PV in the guided mutation process where it is further updated at each new generation.

Algorithm 5.1 Initial population generation using skyline operator

Input: V:set of candidate services
Output: dominating set of candidate services.
$\Re$: set of candidate services in each abstract service.

1: **for** each population **do**
2: $\Re \leftarrow V$; ▷ Candidate services in V assigned to $\Re$;
3: $\chi \leftarrow \phi, \zeta \leftarrow \phi, d_t \leftarrow \phi$; ▷ τ: set of services selected for initial population; ζ: neighbor services of τ; d_t: Possible dominant set
4: $V \leftarrow random(\Re)$;
5: Insert V into τ; ▷ Candidate services with max fitness values are added to τ
6: Delete V from $\Re$; ▷ Remove services from $\Re$
7: **while** $\Re \neq \phi$ **do**
8: Compute neighbor services (ζ) for V;
9: **for** each service j in $\Re$ **do**
10: **if** $V_{fitness}$ - $j_{fitness} < c1$ **then**
11: $\zeta \leftarrow j$;
12: **end if** ▷ The candiadte services have fitness values nearest to τ are selected and added to ζ
13: **end for**
14: $\chi \leftarrow \zeta$; ▷ Short listed services added to χ
15: Select neighbor services which has at least one service with nearest fitness as neighbor ;
16: **for** each service k in χ **do** ▷ Check whether the candidate services from τ with nearest fitness values in χ
17: **for** each service j in $\Re$ **do**
18: **if** $k_{fitness} - j_{fitness} < c1$ **then**
19: $\zeta \leftarrow j$;
20: **end if**
21: **end for**
22: **if** neighbor is present **then**
23: $d_t \leftarrow k$;
24: **end if**
25: **end for**
26: χ.remove(service);
27: $\chi \leftarrow \zeta$;
28: Select a service from χ with fitness value nearest to τ;
29: $\tau \leftarrow service$; ▷ Set of service selected for initial population
30: **end while**
31: **end for**

Probability update At each generation, a parent population is formed by se lecting the best $N_c/4$ solutions from the current population set, where N_c is the sequence number of active solution. The parent population is used to update PV. The pseudocode for the probability updating is given as Algorithm 5.2, where $\lambda \in [0, 1]$ is the learning rate that governs the parent population contribution.

With high values of λ, the updated PV reflects more of the characteristics of the parent population, and vice versa. The change in probability for a candidate service depends on the frequency of occurrence of that service in the parent population.

Algorithm 5.2 Probability Update

Input: initial PV, population, parent population.
Output: updated PV.
1: **for** each $s_i \in$ population **do**
2: $n \leftarrow$ number of solutions containing s_i in the parent population;
3: $pv'_i = (\lambda * (4n/N_c)) + (1 - \lambda) * pv_i$;
4: **end for**
5: $PV \leftarrow PV'$;

Guided mutation The guided mutation operator combines the global statistical information in PV with the generated solutions to generate new solutions. Algorithm 5.3 explains the process of generating a new solution, wherein $\beta \in [0, 1]$ and d_t is the new solution generated, whose services are either sam pled from the model given by PV or retained as those of the best solution in the parent. This process is controlled by a parameter β, which specifies the contributions of PV and the parent population. As the value of β decreases, the parent population's contribution to the new solution increases whereas that of PV decreases, and vice versa. The generated solution d_t may not be valid. Hence, it needs to be checked for feasibility and, if required, a repair operator is applied to it.

Algorithm 5.3 Guided Mutation

Input: population, PV, and best solution.
Output: Generated solution d_t.
1: initialize β ;
2: **for** each service s_i in population **do**
3: generate a random value $r \in [0, 1]$;
4: **if** $r < \beta$ **then**
5: **if** $p_{v_i} <$ threshold_probability **then**
6: $d_t \leftarrow d_t \cup s_i$;
7: **else if** $s_i \in$best solution **then**
8: $d_t \leftarrow d_t \cup s_i$;
9: **end if**
10: **end if**
11: **end for**

Repair operator The repair operator is applied only on infeasible solutions. In this repair operator, all the services from $\Re$ that are not present in the χ list are selected, and the $\Re$ list is reinitialized with those selected services. A service from $\Re$ is selected randomly, and its nearest neighborhood is calculated. The repair operator selects a service from this set and adds it to the infeasible solution set. Then, the selected service is deleted from $\Re$. This process is repeated until the set becomes feasible, i.e., until $\Re$ becomes null. The selection of a service is based on a probability threshold value or is random. This helps to maintain the diversity of the population. The pseudocode for the repair operator is shown in Algorithm 5.4.

Algorithm 5.4 Repair Operator

Input: $\Re$, population, $Prob$.
Output: update d_t.
while $\Re \neq \phi$ **do** ▷ check for feasible solution in $\Re$
  $v \leftarrow$ random service from population;
  if v not in $\Re$ **then**
    if $V\_$ probability $<$ threshold_ probability **then**
      $d_t \leftarrow V$;
      delete V from population;
    end if
  end if
end while
return $\Re$; ▷ $\Re$ is an infeasible solution on which repair operator is applied

Our proposed modified EA/G (see Algorithm 5.5) uses the conventional EA/G with a novel skyline operator to generate solutions at a faster convergence rate by removing the redundant services. Further, N_c initial solutions are generated. The probability for each candidate service is computed using these generated solutions. Later, the probability of each candidate service is updated using the best $N_c/4$ parent populations. The composition is generated using the modified EA/G based on both parent solutions and PV. Further, the composition generated is tested for feasibility. If the composition generated is not feasible, then the repair operator is applied to make it feasible. This process is repeated N times or until the optimal composition with the best fitness is obtained. The pseudocode for the modified EA/G algorithm is presented in Algorithm 5.5.

Algorithm 5.5 Modified Evolutionary Algorithm with Guided mutation

Input: population, P.
Output: best service composition.
1: $iter \leftarrow 0$;
2: **do**
3: $iter \leftarrow iter + 1$;
4: Generate N_c initial solutions for g by using algorithm 5.1;
5: Compute initial probabilities for all the services in generation;
6: Generate the parent population;
7: Update P based on parent population by using algorithm 5.2;
8: **do**
9: Apply Guided Mutation by using algorithm 5.3;
10: **while** (services $\neq$ end)
11: Verify feasibility of the generated composition;
12: **if** composition is not feasible **then**
13: Apply repair operator on obtained solution using algorithm 5.4;
14: **end if**
15: **while** $iter \leq N$
16: Reinitialize generation g;

5.2.3 MapReduce-Based Modified EA/G

Our proposed MapReduce-Based Modified EA/G algorithm to address the challenge of big service composition with scalability and robustness consists of three phases: initialization, MapReduce, and repair (as illustrated in Fig. 5.4).

1. Initialization Phase In order to reduce search space of the candidate services and filter for the optimal candidate services for each abstract service, we use the skyline operator (described in section 5.1) in the MapReduce framework. These candidate services are given as the initial population. From the given population, the initial solution is generated by evaluating the fitness of the services.

Initially, the service with the maximum fitness value is selected and added to the solution. Then, we proceed further by selecting those services having fitness values nearest to those of the prior service. We adopted the block elimination method [240] (described in section 5.1) to process our skyline method to reduce the redundant services in service compositions. The service files are stored as a $< key, value >$ pair structure in a distributed file system, where key is the service ID and $value$ is the service information. This file is used as input to the

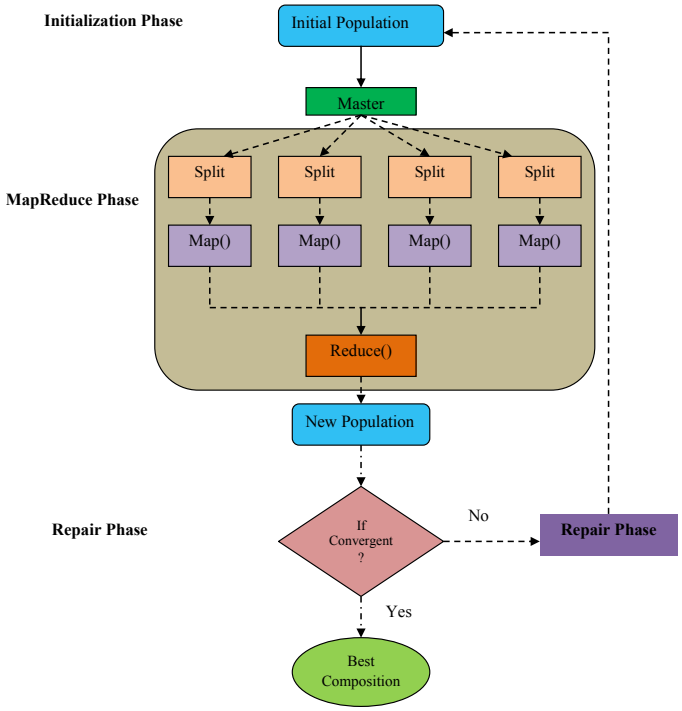


Figure 5.4: Flowchart of MR-EA/G

MapReduce job in the MapReduce phase. The structure of the $< key, value >$ pair used in MR-EA/G is shown in Fig. 5.5. The components of the services are separated by semicolons.

I: S_n	S_nC_n: Candidate service; C_n: QoS attribute; Fitness(n)
Key	**Value**

Figure 5.5: Representation of subpopulation

2. MapReduce Phase The iterative process of MapReduce jobs, where each MapReduce job represents one iteration in the MR-EA/G algorithm, is as follows. The result of each MapReduce job is an updated population or a swarm. This updated information is used as input to the next MapReduce phase.

Map function:

The master splits the data or input into n splits and stores them in the distributed file system; p is the number of map tasks. Hence, the swarm is divided into q populations. Then, each map task is updated with its subpopulation. These subpopulations are stored as $< key, value >$ pairs, where key is the abstract service (s_n) and $value$ is a candidate service with its QoS attributes. The output of the different tasks with their key is sent to the Reduce() function as its input. The Map() function is given as Algorithm 5.6.

Algorithm 5.6 Map Function

Map($Key : offset, value :$ $set\ of\ can.services\ with\ QoS,\ context : output$)
1: $String\ line = split\ (value)$;
2: $key : outkey$;
3: $Value : outvalue$;
4: $outkey.set\ (abstract\ services)$;
5: $outvalue.set\ (candidate\ services,\ QoS)$;
6: $output.write\ (outkey,\ outvalue)$;
7: end

Reduce function:

The Reduce() function of MR-EA/G is shown as Algorithm 5.7. First, the reducer takes all information about one abstract service at a time using parallel batch processing. Then, the reducer combines all information of the iterable subpopulations. All these subpopulations are stored in a list. The entire list is passed to the MR-EA/G algorithm. The MR-EA/G algorithm outputs (Out key, dominant population) or the dominant population for each service and the best optimal service composition with scalability and robustness.

3. Repair Phase The solution generated from the MapReduce phase is further checked for feasibility. A solution is called infeasible if it has low diversity and low evolution. Such infeasible solutions are passed to the repair operator to enhance their diversity and quality. If a solution is feasible, it is passed directly to the next MapReduce job or else it is given to the repair phase. The resultant solution is given to the next MapReduce job, replacing the previous population in the distributed file system. Otherwise, there is no need to consider the repair phase. The new population emitted from the last MapReduce job in the distributed file is sent to the MapReduce job.

Algorithm 5.7 Reduce Function

```
    Reduce(Key : outkey, value : iterable< outvalue >, context : output)
1:  for each population do
2:      < list > init_population, < list > dominant_population, < list > near_population < list >;
3:      function INITIAL SOLUTION(< initial_population >)                    ▷ See Algorithm 5.1
          return init_population;
4:          near_population;
5:      end function
6:      function Initial_probability(< initial_population >)                ▷ See Algorithm 5.2
          return probability of initial_population;
7:          near_population;
8:      end function
9:      function Update_probability(< initial_population >)
          return update_probability;
10:     end function
11:     function Parent_population(< initial_population >, number of count)
          return Parent_population;
12:     end function
13:     function Guided_mutation(< Parent_population >, probability)        ▷ See Algorithm 5.3
14:         < list > best_population;
          return dominent_population;
15:     end function
16:     function FD(location)                                               ▷ See Algorithm 5.4
          return fd;
17:     end function
18:     function RANDOM(null)
          return random;
19:     end function
20: end for
21: output.write(outkey, dominant_population);
```

5.3 Performance Evaluation

Experiments were implemented using Java on a Hadoop cluster consisting of 12 nodes provided with Ubuntu 14.10 (OS), 16 GB RAM, an Intel i7 processor, and 1 TB of memory space. One node was set up as a master node, and the remaining 11 nodes were set up as slave nodes. For the purpose of the experiments, we used a synthetic data set that consisted of 100 abstract services, and for each abstract service, 10,000 candidate services were considered. Each candidate service had the following QoS parameters: cost, response time, availability, throughput, and reliability. There are some benchmark data sets of web services [182, 246] that

are not suitable in the context of our proposed approach because of their limited size. There is no big service data set available for experiments. Hence, we needed a synthetic data set for our experiments. The data set used in our experiments was synthetically generated (similar to that used in [5, 19, 247]). Using the Shapiro–Wilk test [183], we tested our synthetic data set to ascertain whether it is normally distributed; the test results revealed that the data set is normally distributed.

We compared our proposed MR-EA/G with other MapReduce-based optimization algorithms: MR-GA [248], MR-GA2 [249], MR-PSO [201], MR-IDPSO [235], and MR-ABC [186]. We have implemented MapReduce based versions of MR-GA2 [249], MR-PSO [201], and MR-ABC [186] for our experimental purposes. In our experiments, a few parameters were fixed for all algorithms, controlling the execution time and specifying the size of the population. We set the population size to 30, each experiment was executed continuously for 30 runs, and the mean of each run was duly noted. The weights of the QoS criteria were set as w_1 = 18.94, w_2 = 17.05, w_3 = 15.81, w_4 = 27.83, and w_5 = 20.37 based on user preferences. These QoS values were obtained using the AHP method. Table 5.2 shows the parameter values for our experiments.

Table 5.2: Compared approaches with parameter settings

Approach	Abbrevations	Parameter Setting
MapReduce-based Genetic Algorithm [248]	MR-GA	Crossover=0.5, mutation=0.001 and random selection approach is applied
MapReduce-clonal selection based Genetic Algorithm [249]	MR-GA2	Mutation / hypermutation are 0.1 and 0.01, Crossover=0.5, mutation=0.001
MapReduce-based Particle Swarm Optimization [201]	MR-PSO	Inertia weight varies from 0.9 to 0.7 linearly, c_1, and c_2=2
MapReduce-based Improved Discrete Particle Swarm Optimization [235]	MR-IDPSO	Inertia weight varies from 0.9 to 0.7, c_1, and c_2=2, Mutation/ hyper mutation are 0.1 and 0.01
MapReduce-based Artificial Bee Colony [186]	MR-ABC	
MapReduce-based EA/G	MR-EA/G	N_c=16, β=0.035, λ= 0.34, probability threshold = 0.55, and fitness threshold= 0.29

We conducted several experiments, each experiment comprising a user composition request with M (i.e., 25, 50, 75, or 100) abstract services and each abstract service having N (i.e., 500, 1000, 5000, or 10,000) candidate services. Varying these M and N values, we observed the results. Table 5.3 shows the average fitness values of the algorithms compared for 500, 1000, 5000, and 10,000 candidate services for varying numbers of abstract services.

Table 5.3: Comparison of the best average fitness values (MR-EA/G and other approaches)

Candidate Services	Abstract Services	Algorithms					
		MR-GA	**MR-ABC**	**MR-PSO**	**MR-GA2**	**MR-IDPSO**	**MR-EA/G**
500	25	12.8919	14.3613	14.9602	15.1814	16.8919	**27.5216**
	50	14.1342	15.9795	15.1432	16.1832	18.4342	**28.8618**
	75	15.9436	16.2816	16.1596	18.1946	19.9436	**36.5476**
	100	16.9492	17.3242	17.2489	19.9614	20.9492	**44.5816**
1000	25	14.2193	15.8748	15.9832	16.9635	17.9189	**28.9299**
	50	15.2242	17.3855	16.2216	18.8132	19.9216	**29.5474**
	75	16.3848	17.5616	17.3214	19.8319	20.9342	**39.5472**
	100	17.2912	18.2624	18.1534	20.9242	21.9198	**53.9747**
5000	25	16.2318	17.1392	16.8318	17.4896	19.3853	**29.6768**
	50	16.7816	18.1176	17.7816	19.9213	21.3214	**31.4238**
	75	17.8396	18.4164	18.8396	21.3816	23.3512	**42.378**
	100	18.7932	19.2154	19.7932	22.4796	25.3864	**62.3543**
10000	25	16.5241	17.8186	17.9853	18.1341	22.8186	**30.0284**
	50	17.3243	18.4143	18.8148	22.2832	24.2117	**32.3368**
	75	17.6541	19	19.9018	23.7871	26.2742	**44.3822**
	100	19.3241	20.8705	21.7689	24.4896	29.3672	**65.1824**

Based on Table 5.3, we observe from the best average fitness values obtained by MR-EA/G that it is more effective than the other algorithms compared. Fig. 5.6 illustrates the average fitness values of the composite service for varying M and N values. From Fig. 5.6, we observe that the average fitness values increase with the increase in the number of abstract services but do not affect the solution quality. In our proposed algorithm, this increase is exponential, whereas the increase is linear in other algorithms. The proposed algorithm uses a probability method to generate offspring, which in turn ensures the distribution of promising solutions in each generation. Instead of using the position values directly, our proposed algorithm updates the probability values for each generation based on the global statistical information and uses the guided mutation. Thus, our proposed algorithm performs better than the other algorithms. We executed our algorithm 30 times, and the average is noted because of the stochastic nature of algorithms.

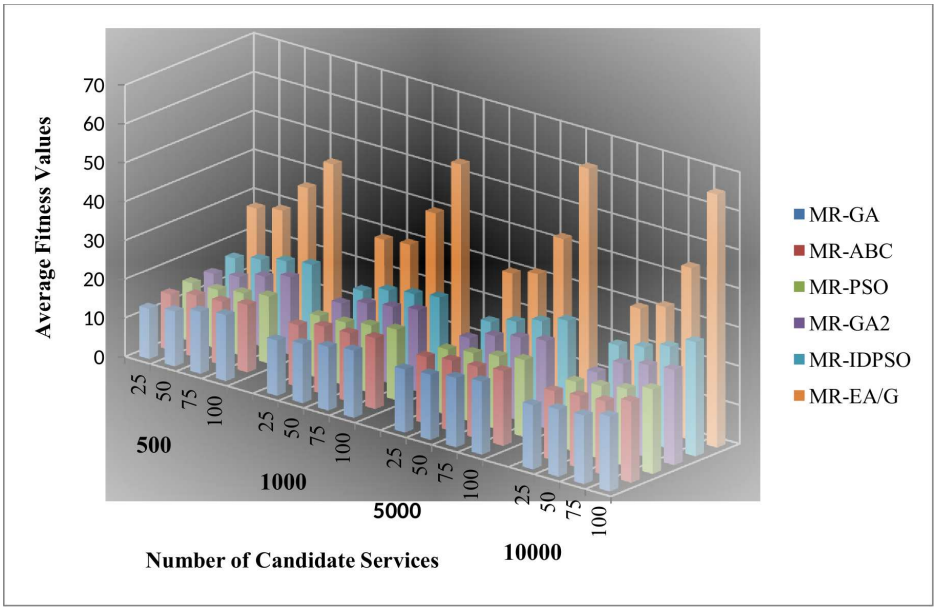


Figure 5.6: Average fitness values for 500, 1000, 5000, 10000 candidate services (MR-EA/G versus other algorithms)

Figure 5.7 depicts the evaluation of fitness values by increasing the number of iterations for 100 abstract services, where each abstract service has 10,000 candidate services. We observe that the average fitness value increases slightly with an increase in the number of iterations for all approaches. However, our proposed MR-EA/G converges and gives near-optimal solutions in less time than the other algorithms (MR-GA, MR-ABC, MR-PSO, MR-GA2, and MR-IDPSO).

Table 5.4 shows the execution time of our proposed algorithm and other compared algorithms for 100 abstract services. The execution times for MR-GA, MR-ABC, MR-PSO, MR-GA2, and MR-IDPSO to obtain the best solutions were 62.58, 115.59, 61.08, 57.86, and 61.95 s, respectively, whereas the execution time for MR-EA/G to obtain the best solution was 30.23 s.

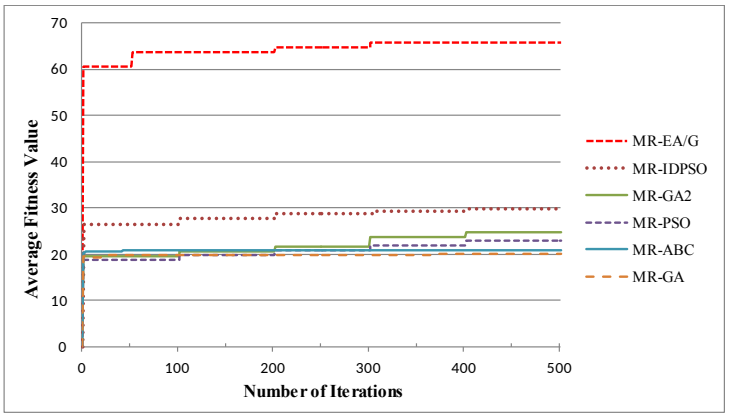


Figure 5.7: Average fitness values for varying the number of iterations (MR-EA/G versus other approaches)

Our proposed algorithm converges very quickly and gives satisfactory results in a lesser time than the other algorithms. In MR-PSO and MR-IDPSO, with each iteration, the new population is generated, and the particle fitness is evaluated (based on best visited position and best visited velocity of all the particles) and changes with each iteration. Therefore, the probability of being trapped in local optima is increased. This change decreases the convergence rate and increases the search space, which in turn consumes more time compared with our proposed algorithm. Similarly for MR-GA, with each iteration a new population is generated, and the individual fitnesses are evaluated (based on fixed and predetermined crossover and unguided mutation). These processes cause it to be very slow to converge and to become easily stuck in local maxima. The whole process consumes more time, decreases the convergence rate, and increases the search space. In MR-ABC, with each iteration, the initial population is generated. This population is repeated for the cycles of the search processes of the employed, onlooker, and scout bees, respectively. However, it lacks a centralized processor to guide it toward good solutions, and the time to convergence remains uncertain. Likewise, in MR-GA2, the antibody repertoire is randomly generated, and then low-affinity antibodies are replaced by new random antibodies during the mutation process in each iteration. This is also a time-consuming process.

In our proposed approach, the MR-skyline operator was developed to abandon-some candidate services with non-optimal solutions, thereby gradually shrinking

Table 5.4: Execution time (sec) analysis for varying candidate services (MR-EA/G) versus other approaches

	Candidate services			
Algorithms	500	1000	5000	10000
MR-GA [248]	63.22	57.32	61.27	62.58
MR-ABC [186]	120.2	120.54	80.86	115.59
MR-PSO [201]	48.76	52.19	40.89	61.08
MR-GA2 [249]	56.52	55.12	58.36	57.86
MR-IDPSO [235]	51.08	55.38	59.36	61.95
MR-EA/G	**30.18**	**33.09**	**30.19**	**30.23**

the valid search space. We used the guided mutation method, in which an offspring is produced by combining both statistical information about the search space and local information about the parent solution. Hence, our proposed ap proach reduces the search space and consumes less execution time. The time complexity of MR-EA/G, MR-GA, MR-PSO, MR-IDPSO, MR-GA2, and MR-ABC are $O(n)$, $O(n^2)$, $O(n^2)$, $O(n \cdot \log n)$, $O(n \cdot \log n)$, and $O(n^2)$ respectively.

Table 5.5: T-test statistical analysis results (MR-EA/G)

Abstract Services	Approaches		MR-GA	MR-IDPSO	MR-PSO	MR-ABC	MR-GA2	MR-EA/G
25	MR-GA	T-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		249.219 (0)	*	*	*	*	*
	MR-PSO		21.3605 (0)	83.8613 (0)	*	*	*	*
	MR-ABC		30.4911 (0)	114.6498(0)	0.5161(0.6076)	*	*	*
	MR-GA2		68.7899 (0)	129.1001 (0)	9.6099 (0)	13.7649 (0)	*	*
	MR-EA/G		**422.5523(0)**	**188.5225 (0)**	**197.536 (0)**	**257.54 (0)**	**297.6201 (0)**	*
50	MR-GA	T-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		171.9023 (0)	*	*	*	*	*
	MR-PSO		28.7490 (0)	107.4319 (0)	*	*	*	*
	MR-ABC		32.1787 (0)	123.105 (0)	2.2617 (0.2747)	*	*	*
	MR-GA2		249.3636 (0)	55.5187 (0)	94.9382 (0)	126.0214 (0)	*	*
	MR-EA/G		**560.1384 (0)**	**184.9778 (0)**	**310.2354 (0)**	**376.5437 (0)**	**467.4155 (0)**	*
75	MR-GA	T-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		247.276 (0)	*	*	*	*	*
	MR-PSO		41.6562 (0)	119.4937 (0)	*	*	*	*
	MR-ABC		38.88 (0)	146.9068 (0)	7.8567 (0)	*	*	*
	MR-GA2		132.9658 (0)	56.0843 (0)	61.4957 (0)	77.1474 (0)	*	*
	MR-EA/G		**373.925 (0)**	**232.5213 (0)**	**293.0033 (0)**	**316.9459 (0)**	**256.1271 (0)**	*
100	MR-GA	T-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		348.7109 (0)	*	*	*	*	*
	MR-PSO		57.5519 (0)	187.2576 (0)	*	*	*	*
	MR-ABC		28.2106 (0)	180.2931 (0)	16.6644 (0)	*	*	*
	MR-GA2		226.3003 (0)	177.5601 (0)	74.6589 (0)	82.7483 (0)	*	*
	MR-EA/G		**319.683 (0)**	**246.8641 (0)**	**297.448 (0)**	**299.4491 (0)**	**284.5778 (0)**	*

Table 5.6: Wilcoxon signed rank-test results (MR-EA/G)

Abstract Services	Approaches		MR-GA	MR-IDPSO	MR-PSO	MR-ABC	MR-GA2	MR-EA/G
25	MR-GA	Z-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		-4.7821 (0)	*	*	*	*	*
	MR-PSO		-4.7821 (0)	-4.7821 (0)	*	*	*	*
	MR-ABC		-4.7821 (0)	-4.7821 (0)	-0.7919 (0.4295)	*	*	*
	MR-GA2		-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	MR-EA/G		**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	*
50	MR-GA	Z-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		-4.7821 (0)	*	*	*	*	*
	MR-PSO		-4.7821 (0)	-4.7821 (0)	*	*	*	*
	MR-ABC		-4.7821 (0)	-4.7821 (0)	-0.977 (0.327)	*	*	*
	MR-GA2		-4.7821 (0)	-4.7821 (0)	-0.4628 (0.64552)	-4.7821 (0)	*	*
	MR-EA/G		**-4.7821(0)**	**-4.7821 (0)**	**-4.7821 0)**	**-4.7821 (0)**	**-4.7821 (0)**	*
75	MR-GA	Z-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		-4.7821 (0)	*	*	*	*	*
	MR-PSO		-4.7821 (0)	-4.7821 (0)	*	*	*	*
	MR-ABC		-2.9104 (0.0036)	-4.7821 (0)	-1.43 (0.1527)	*	*	*
	MR-GA2		-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-1.6352 (0.101)	*	*
	MR-EA/G		**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	**-4.7821 (0)**	*
100	MR-GA	Z-Value / P-Value	*	*	*	*	*	*
	MR-IDPSO		-4.7821 (0)	*	*	*	*	*
	MR-PSO		-4.7821 (0)	-4.7821 (0)	*	*	*	*
	MR-ABC		-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*	*
	MR-GA2		-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	-4.7821 (0)	*	*
	MR-EA/G		**-4.7821(0)**	**-4.7821 (0)**	**-4.7821 0)**	**-4.7821 (0)**	**-4.7821 (0)**	*

We performed statistical analyses (parametric and non-parametric) for our proposed approach and other contemporary approaches and used the T-test [193, 194] and Wilcoxon signed-rank test [195]to evaluate whether the best mean values obtained by all approaches have a significant difference with 58 degrees of freedom at a 1% level of significance. The statistical results of the T-test and Wilcoxon signed-rank test are presented in Table 5.5 and Table 5.6. Based on Table 5.5, the results obtained are statistically significant (all T-values are positive, and the P-values are less than zero). Based on Table 5.6, we observe that the results obtained are statistically significant (all T-values are positive, and the

P-values are less than zero). By performing the T-test and the Wilcoxon signed-rank test at a 1% level of significance, we observe that our proposed approach performs significantly more accurate than other approaches.

5.4 Related Work

In the era of Big Data, Big service composition [3] is a new challenge in the field of service oriented computing. Very few works exists in the field of Big service composition. Zhang et al. [235] designed an MapReduce based on Improved Discrete PSO (MR-IDPSO) for QoS-aware web service composition. In their proposed method, a skyline operator is used for filtering the non-optimal candidate services, a premature trimming operator is used to overcome a premature convergence of particles, and the MapReduce framework is employed to improve the search efficiency. Chen at al. [241] proposed an MR-based skyline operator to prune the candidate services and find the optimal solution for QoS-aware skyline service selection. In their proposed approach, an angle-based data space partitioning method is used in MapReduce-based skyline service selection and a Paper-Tape Model is used to handle the dynamic nature of service environment.

5.5 Summary

In this chapter, we proposed a novel QoS-aware Big service composition based on modified evolutionary algorithm with guided mutation (EA/G) under MapReduce environment. Based on the results of experiments, our proposed MR-EA/G gave significantly better results compared to other similar algorithms. Based on the experiments and statistical analysis, we infer that our proposed approach outperforms against several other approaches.

Chapter 6

Conclusions and Future Directions

Service Composition is a strategy of combining several services to create a composite service that aims to fulfill the functional requirements as well as non-functional (QoS) requirements. The thesis mainly focused in proposing various novel computational intelligence techniques for QoS-aware Web, Cloud, and Big service composition.

Chapter 3 proposed solutions for optimal web service composition *balancing QoS parameters* and satisfying *connectivity constraints* for achieving search optimization and reducing computational complexity. In the first approach, we developed a novel OFASC-AGEGA to evaluate the service fitness and composition fitness using Discrete Uniform Rank Distribution and Discrete Uniform Service Rank Distribution. Then, we used an Adaptive Genotypes Evolution based Genetic Algorithm (AGEGA) to improve the convergence rate and to reduce computational complexity. In the second approach, we developed the OFASC-MIWO to evaluate the local fitness strategy for individual services and global fitness evaluation strategy for composite services by considering *balancing QoS parameters* and *connectivity constraints* to obtain minimal search and minimum time complexity. Based on the experimental results, we found that our proposed approaches are scalable, robust, and optimal.

In Chapter 4, we proposed various cloud service selection approaches including extended Data Envelopment Analysis (DEA), extended Super efficiency DEA,

extended versions of Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), Fuzzy TOPSIS, and Grey TOPSIS. In these approaches, Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), and Fuzzy AHP are used to determine the weights of criteria (or QoS parameters), and these methods are integrated with DEA, SDEA, TOPSIS, Fuzzy TOPSIS, and Grey TOPSIS to evaluate the relative efficiency of cloud services and their ranks. The results revealed that modified SDEA and modified Grey TOPSIS could outperform the other approaches.

In Chapter 5, we proposed a MapReduce-based evolutionary algorithm with guided mutation (MR-EA/G) algorithm to solve QoS-aware Big service composition. In this approach, a MR-based skyline operator accomplished the QoS-based service pre-selection process and a novel EA/G with MapReduce framework is presented to improve the searching efficiency of service selection and composition.

6.1 Future Directions

The following research directions are suggested for future:

- **Exploring other Computational Intelligence techniques for services composition:** The rapid growth in the number of services with similar functionality but different QoS parameters expand the candidate services space, which leads to a combinatorial explosion problem and involves enormous computational time and cost. To handle these services with reduced cost and computational time and satisfying the user QoS constraints new algorithms can be proposed based on CI techniques. Some of the CI techniques such as natural evolution strategies [250], Glowworm swarm optimization [251], Grey wolf optimizer [252], Jaya algorithm [253], Election Algorithm [254], and Galaxy-based Search Algorithm [255] can be explored for composing web services, cloud services and big services. The composition of different services from different service providers becomes complex in a real multiple cloud environment, and requires a massive

amount of data interchange among all service participants. CI techniques can be extended to service composition in multiple cloud environments and cyber-physical-social systems.

- **Exploring various computational intelligence techniques for Cognitive Services:** Our proposed approaches are applied in service composition domains. However, these algorithms are not restricted to this domains. Cognitive services are a collection of APIs that helps to build powerful intelligence into applications to enable natural and contextual interactions that augment users' experiences via the power of machine-learned [256]. Researches are in progress for developing Deep learning architectures for cognitive services [257, 258], and formulating cognitive algorithms for managing QoS of services in the cloud and fog computing environments [259].
- **Exploring new approaches for service selection and composition in different parallel data processing platforms:** Big service composition deals with huge volume of Big data and services that generated from virtual and physical domains. Our proposed approach in Chapter 5 is built on MapReduce frame-work for big service composition. New frameworks are evolving for parallel processing in a distributed environment. We would like to develop new approaches on various data parallel data processing platforms like Spark for enhancing the efficiency of service composition.

List of Publications

[1] Chandrashekar Jatoth, G.R. Gangadharan, and Rajkumar Buyya, "Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review", *IEEE Transactions on Services Computing, Vol. 10, No. 3, pp. 475-492, IEEE, 2017.*

[2] Chandrashekar Jatoth, G.R. Gangadharan, and Ugo Fiore, "Evaluating the efficiency of cloud services using modified data envelopment analysis and modified super-efficiency data envelopment analysis", *Soft Computing, Vol. 21, No. 23, pp 7221–7234, Springer, 2017.*

[3] Chandrashekar Jatoth, G.R. Gangadharan, Ugo Fiore, and Rajkumar Buyya, "QoS-aware Big Service composition using MapReduce based Evolutionary Algorithm with Guided Mutation", *Future Generation Computer Systems, Elsevier, 2017, doi:10.1016/j.future.2017.07.042.*

[4] Chandrashekar Jatoth, G.R. Gangadharan, "Fitness Metrics for QoS-Aware Web Service Composition Using Metaheuristics", *In Proceedings of the* 7^{th} *KES International Conference on Intelligent Decision Technologies (KES-IDT)*, Sorrento, Italy, pages 267–277, Springer, 2015.

[5] Chandrashekar Jatoth and G.R. Gangadharan, "QoS-aware Web Service Composition Using Quantum Inspired Particle Swarm Optimization", *In Proceedings of the* 7^{th} *KES International Conference on Intelligent Decision Technologies (KES-IDT)*, Sorrento, Italy, pages 255–265, Springer, 2015.

[6] Chandrashekar Jatoth, G.R. Gangadharan, and Rajkumar Buyya, "Optimal Fitness Aware Service Composition using an Adaptive Genotypes Evolution

based Genetic Algorithm", *IEEE Transactions on Systems, Man, and Cybernetics: Systems, IEEE* (Revised Version Submitted).

[7] Chandrashekar Jatoth, G.R. Gangadharan, and Ugo Fiore, "Optimal Fitness Aware Service Composition using Modified Invasive Weed Optimization", *Soft Computing, Springer* (Under Review).

[8] Chandrashekar Jatoth, G.R. Gangadharan, Ugo Fiore, and Rajkumar Buyya, "Decision Models for Selection of Cloud Services using Extended Versions of TOPSIS, Fuzzy TOPSIS, and Grey TOPSIS", *Computers and Electrical Engineering, Elsevier* (Under Review).

References

[1] MUNINDAR P SINGH AND MICHAEL N HUHNS. *Service-oriented computing: semantics, processes, agents*. John Wiley & Sons, 2006.

[2] FANG LIU, JIN TONG, JIAN MAO, ROBERT BOHN, JOHN MESSINA, LEE BADGER, AND DAWN LEAF. **NIST cloud computing reference architecture**. *NIST special publication*, **500**:292–320, 2011.

[3] XIAOFEI XU, QUAN Z SHENG, LIANG-JIE ZHANG, YUSHUN FAN, AND SCHAHRAM DUSTDAR. **From Big Data to Big Service**. *Computer*, **48**(7):80–83, 2015.

[4] DANIELA BARREIRO CLARO, PATRICK ALBERS, AND JIN-KAO HAO. *Web Services Composition*, pages 195–225. Springer, 2006.

[5] DANILO ARDAGNA AND BARBARA PERNICI. **Adaptive Service Composition in Flexible Processes**. *IEEE Transactions on Services Computing*, **33**(6):369–384, June 2007.

[6] PEARL BRERETON, BARBARA A KITCHENHAM, DAVID BUDGEN, MARK TURNER, AND MOHAMED KHALIL. **Lessons from applying the systematic literature review process within the software engineering domain**. *Journal of Systems & Software*, **80**(4):571–583, April 2007.

[7] MICHAEL PAPAZOGLOU. *Web services: principles and technology*. Pearson Education, 2008.

[8] MICHAEL P PAPAZOGLOU, PAOLO TRAVERSO, SCHAHRAM DUSTDAR, AND FRANK LEYMANN. **Service-oriented computing: State of the art and research challenges**. *Computer*, **40**(11), 2007.

[9] GUSTAVO ALONSO, FABIO CASATI, HARUMI KUNO, AND VIJAY MACHIRAJU. *Web services*. Springer, 2004.

[10] SCHAHRAM DUSTDAR AND MICHAEL P. PAPAZOGLOU. **Services and Service Composition - An Introduction**. *it - Information Technology*, **50**(2):86–92, 2008.

[11] RAJKUMAR BUYYA, CHEE SHIN YEO, AND SRIKUMAR VENUGOPAL. **Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities**. In *10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13. IEEE, 2008.

[12] RAJKUMAR BUYYA, CHEE SHIN YEO, SRIKUMAR VENUGOPAL, JAMES BROBERG, AND IVONA BRANDIC. **Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility**. *Future Generation Computer Systems*, **25**(6):599 – 616, 2009.

[13] RAJKUMAR BUYYA, JAMES BROBERG, AND ANDRZEJ M GOSCINSKI. *Cloud Computing: Principles and Paradigms*. John Wiley, USA, 2010.

[14] XIAOFEI XU, GIANMARIO MOTTA, XIANZHI WANG, ZHIYING TU, AND HANCHUAN XU. **A New Paradigm of Software Service Engineering in the Era of Big Data and Big Service**. *arXiv preprint arXiv:1608.08342*, 2016.

[15] STEPHEN KAISLER, FRANK ARMOUR, J ALBERTO ESPINOSA, AND WILLIAM MONEY. **Big data: Issues and challenges moving forward**. In *Proceedings of the 46th Hawaii international conference on System sciences (HICSS)*, pages 995–1004. IEEE, 2013.

[16] JATOTH CHANDRASHEKAR, G.R. GANGADHARAN, AND BUYYA RAJKUMAR. **Computational Intelligence based QoS-aware Web Service Composition: A Systematic Literature Review**. *IEEE Transactions on Services Computing*, **10**(3):475–492, 2017.

[17] QUAN Z. SHENG, XIAOQIANG QIAO, ATHANASIOS V. VASILAKOS, CLAUDIA SZABO, SCOTT BOURNE, AND XIAOFEI XU. **Web services composition: A decade's overview**. *Information Sciences*, **280**:218 – 238, 2014.

[18] ATHMAN BOUGUETTAYA, MUNINDAR SINGH, MICHAEL HUHNS, QUAN Z. SHENG, HAI DONG, QI YU, AZADEH GHARI NEIAT, SAJIB MISTRY, BOUALEM BENATALLAH, BRAHIM MEDJAHED, MOURAD OUZZANI, FABIO CASATI, XUMIN LIU, HONGBING WANG, DIMITRIOS GEORGAKOPOULOS, LIANG CHEN, SURYA NEPAL, ZAKI MALIK, ABDELKARIM ERRADI, YAN WANG, BRIAN BLAKE, SCHAHRAM DUSTDAR, FRANK LEYMANN, AND MICHAEL PAPAZOGLOU. **A Service Computing Manifesto: The Next 10 Years**. *Commun. ACM*, **60**(4):64–72, mar 2017.

[19] JOSE ANTONIO PAREJO, SERGIO SEGURA, PABLO FERNANDEZ, AND ANTONIO RUIZ-CORTES. **QoS-aware web services composition using GRASP with Path Relinking**. *Expert Systems with Applications*, **41**(9):4211 – 4223, 2014.

[20] WANCHUN DOU, XUYUN ZHANG, JIANXUN LIU, AND JINJUN CHEN. **HireSome-II: Towards privacy-aware cross-cloud service composition for big data applications**. *IEEE Transactions on Parallel and Distributed Systems*, **26**(2):455–466, 2015.

[21] JATOTH CHANDRASHEKAR AND G. R. GANGADHARAN. **QoS-aware Web Service Composition Using Quantum Inspired Particle Swarm Optimization**. In *Proceedings of the 7th International KES Conference on Intelligent Decision Technologies*, pages 255–265. Springer, 2015.

[22] BOUALEM BENATALLAH, MARLON DUMAS, QUAN Z SHENG, AND ANNE HH NGU. **Declarative composition and peer-to-peer provisioning of dynamic web services**. In *Proceedings of the 18th International Conference on Data Engineering*, pages 297–308. IEEE, 2002.

[23] FEI LI, FANGCHUN YANG, KAI SHUANG, AND SEN SU. **Q-peer: A decentralized qos registry architecture for web services**. In *Proceedings of the International Conference on Service-Oriented Computing*, pages 145–156. Springer, 2007.

[24] CHINTAN PATEL, KAUSTUBH SUPEKAR, AND YUGYUNG LEE. **A QoS oriented framework for adaptive management of web service based workflows**. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 826–835. Springer, 2003.

[25] LIANGZHAO ZENG, BOUALEM BENATALLAH, ANNE HH NGU, MARLON DUMAS, JAYANT KALAGNANAM, AND HENRY CHANG. **QoS-aware middleware for Web services composition**. *IEEE Transactions on Services Computing*, **30**(5):311–327, May 2004.

[26] LIANGZHAO ZENG, BOUALEM BENATALLAH, MARLON DUMAS, JAYANT KALAGNANAM, AND QUAN Z. SHENG. **Quality Driven Web Services Composition**. In *Proceedings of the 12th International conference on World Wide Web*, pages 411–421, 2003.

[27] MOHAMMAD ALRIFAI, THOMAS RISSE, AND WOLFGANG NEJDL. **A hybrid approach for efficient Web service composition with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, **6**(2):7, 2012.

[28] MOHAMMAD ALRIFAI AND THOMAS RISSE. **Combining global optimization with local selection for efficient QoS-aware service composition**. In *Proceedings of the 18th International conference on World Wide Web*, pages 881–890. ACM, 2009.

[29] FARHAD MARDUKHI, NASER NEMATBAKHSH, KAMRAN ZAMANIFAR, AND ASGHAR BARATI. **QoS decomposition for service composition using genetic algorithm**. *Applied Soft Computing*, **13**(7):3409–3421, 2013.

[30] SHERRY X SUN AND JING ZHAO. **A decomposition-based approach for service composition with global QoS guarantees**. *Information Sciences*, **199**:138–153, 2012.

[31] JUN JIN, YU ZHANG, YUANDA CAO, AND RUITAO ZHOU. **An enhanced QoS decomposition approach for efficient service composition**. In *Proceedings of the 5th International Conference on Computer Science and Education (ICCSE)*, pages 1680–1684. IEEE, 2010.

[32] DANILO ARDAGNA AND BARBARA PERNICI. **Global and Local QoS Guarantee in Web Service Selection**. In *Proceedings of the International Conference on Business Process Management*, pages 32–46. Springer, 2006.

[33] GERARDO CANFORA, MASSIMILIANO DI PENTA, RAFFAELE ESPOSITO, AND MARIA LUISA VILLANI. **An Approach for QoS-aware Service Composition Based on Genetic Algorithms**. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1069–1075, 2005.

[34] MICHAEL C JAEGER AND GERO MUHL. **QoS-based selection of services: The implementation of a genetic algorithm**. In *Proceedings of the 2007 Conference on ITG-GI Communication in Distributed Systems (KiVS)*, pages 1–12, 2007.

[35] MICHAEL C JAEGER, GERO MÜHL, AND SEBASTIAN GOLZE. **Qos-aware composition of web services: An evaluation of selection algorithms**. In *Proceedings of the OTM Confederated International Conferences on CoopIS, DOA, and ODBASE*, pages 646–661. Springer, 2005.

[36] TAO YU, YUE ZHANG, AND KWEI-JAY LIN. **Efficient algorithms for Web services selection with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, **1**(1):1–26, 2007.

[37] LI LI, PENGYI YANG, LING OU, ZILI ZHANG, AND PENG CHENG. **Genetic algorithm-based multi-objective optimisation for QoS-aware web services composition**. In *Proceedings of the International Conference on Knowledge Science, Engineering and Management*, pages 549–554. Springer, 2010.

[38] LI LI, PENG CHENG, LING OU, AND ZILI ZHANG. **Applying multi-objective evolutionary algorithms to QoS-aware web service composition**. In *Proceedings of the International Conference on Advanced Data Mining and Applications*, pages 270–281. Springer, 2010.

[39] YUJIE YAO AND HAOPENG CHEN. **QoS-aware Service Composition Using NSGA-II1**. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 358–363. ACM, 2009.

[40] HIROSHI WADA, PASKORN CHAMPRASERT, JUNICHI SUZUKI, AND KATSUYA OBA. **Multiobjective Optimization of SLA-Aware Service Composition**. In *Proceedings of the IEEE Congress on Services*, pages 368–375, July 2008.

[41] MARCEL CREMENE, MIHAI SUCIU, DENIS PALLEZ, AND DUMITRU DUMITRESCU. **Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition**. *Applied Soft Computing*, **39**:124–139, 2016.

[42] TAO YU AND KWEI-JAY LIN. **Service selection algorithms for composing complex services with multiple qos constraints**. In *Proceedings of the 3rd International conference on Service-Oriented Computing*, pages 130–143. Springer, 2005.

[43] TAO YU AND KWEI-JAY LIN. **Service selection algorithms for Web services with end-to-end QoS constraints**. *Information systems and e-business management*, **3**(2):103–126, 2005.

[44] CRISTINA BIANCA POP, VIORICA ROZINA CHIFU, IOAN SALOMIE, AND MIHAELA DINSOREANU. **Optimal Web service composition method based on an enhanced planning graph and using an immune-inspired algorithm**. In *Proceedings of the IEEE 5th Intl. Conf. on Intelligent Computer Communication & Processing*, pages 291–298, Aug 2009.

[45] CRISTINA BIANCA POP, VIORICA ROZINA CHIFU, IOAN SALOMIE, MIHAELA DINSOREANU, TUDOR DAVID, AND VLAD ACRETOAIE. **Ant-Inspired Technique for Automatic Web Service Composition and Selection**. In *Proceedings of the 12th Intl. Sym. on Symbolic & Numeric Algorithms for Scientific Computing*, pages 449–455, Sept 2010.

[46] ANGUS FM HUANG, CI-WEI LAN, AND STEPHEN JH YANG. **An optimal QoS-based Web service selection scheme**. *Information Sciences*, **179**(19):3309–3322, 2009.

[47] AMIT KONAR. *Computational intelligence: principles, techniques and applications*. Springer, 2006.

[48] ANDRIES P ENGELBRECHT. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[49] RUDOLF KRUSE, CHRISTIAN BORGELT, FRANK KLAWONN, CHRISTIAN MOEWES, MATTHIAS STEINBRECHER, AND PASCAL HELD. *Computational intelligence: a methodological introduction*. Springer Science & Business Media, 2013.

[50] GERHARD J. WOEGINGER. **Exact Algorithms for NP-Hard Problems: A Survey**. In *Proceedings of the 5th International Workshop*, pages 185–207. Springer Berlin Heidelberg, 2003.

[51] XIN-SHE YANG. **Harmony Search as a Metaheuristic Algorithm**. In *Music-Inspired Harmony Search Algorithm*, **191**, pages 1–14. Springer, 2009.

[52] CHRISTIAN BLUM AND ANDREA ROLI. **Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison**. *ACM Comput. Surv.*, **35**(3):268–308, sep 2003.

[53] JATOTH CHANDRASHEKAR AND G. R. GANGADHARAN. **Fitness Metrics for QoS-aware Web Service Composition using Metaheuristics**. In *Proceedings of the 7th International KES Conference on Intelligent Decision Technologies*, pages 267–277. Springer, 2015.

[54] LING HUANG, QINGLIN ZHAO, YAN LI, SHANGGUANG WANG, LEI SUN, AND WU CHOU. **Reliable and efficient big service selection**. *Information Systems Frontiers*, 2017. doi : 10.1007/s10796-017-9767-x.

[55] XIANZHI WANG, XIAOFEI XU, QUAN Z SHENG, ZHONGJIE WANG, AND LINA YAO. **Novel Artificial Bee Colony Algorithms for QoS-Aware Service Selection**. *IEEE Transactions on Services Computing*, pages 1–1, 2016. doi : 10.1109/TSC.2016.2612663.

[56] SHUIGUANG DENG, HONGYUE WU, DANING HU, AND J LEON ZHAO. **Service selection for composition with QoS correlations**. *IEEE Transactions on Services Computing*, **9**(2):291–303, 2016.

[57] BARBARA KITCHENHAM, O. PEARL BRERETON, DAVID BUDGEN, MARK TURNER, JOHN BAILEY, AND STEPHEN LINKMAN. **Systematic literature reviews in software engineering – A systematic literature review**. *Information and Software Technology*, **51**(1):7 – 15, 2009.

[58] POOYAN JAMSHIDI, AAKASH AHMAD, AND CLAUS PAHL. **Cloud Migration Research: A Systematic Review**. *IEEE Transactions on Cloud Computing*, **1**(2):142–157, July 2013.

[59] S. CHATTOPADHYAY AND A. BANERJEE. **QSCAS: QoS Aware Web Service Composition Algorithms with Stochastic Parameters**. In *Proceedings of the IEEE International Conference on Web Services*, pages 388–395, 2016. doi : 10.1109/ICWS.2016.57.

[60] VALERIA CARDELLINI, EMILIANO CASALICCHIO, VINCENZO GRASSI, AND FRANCESCO LO PRESTI. **Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes**. In *Proceedings of the IEEE International Conference on Web Services*, pages 743–750, July 2007.

[61] MEGHA MOHABEY, YADATI NARAHARI, SUDEEP MALLICK, P SURESH, AND SV SUBRAHMANYA. **A Combinatorial Procurement Auction for QoS-Aware Web Services Composition**. In *Proceedings of the IEEE International Conference on Automation Science & Engineering*, pages 716–721, Sept 2007.

[62] RAINER BERBNER, MICHAEL SPAHN, NICOLAS REPP, OLIVER HECKMANN, AND RALF STEINMETZ. **Heuristics for QoS-aware Web Service Composition**. In *Proceedings of the International Conference on Web Service*, pages 72–82, Sept 2006.

[63] MOHAMMAD ALRIFAI, THOMAS RISSE, AND WOLFGANG NEJDL. **A Hybrid Approach for Efficient Web Service Composition with End-to-end QoS Constraints**. *ACM Transactions on the Web*, **6**(2):1–31, 2012.

[64] VIRGINIE GABREL, MAUDE MANOUVRIER, AND CECILE MURAT. **Optimal and Automatic Transactional Web Service Composition with Dependency Graph and 0-1 Linear Programming**. In *Proceedings of the 12th International Conference on Service-Oriented Computing*, pages 108–122. Springer, 2014.

[65] TAO YU AND KWEI JAY LIN. **Service selection algorithms for Web services with end-to-end QoS constraints**. In *Proceedings of the IEEE Intl. Conf. on e-Commerce Tech.*, pages 129–136, July 2004.

[66] TAO YU AND KWEIJAY LIN. **Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints**. In *Proceedings of the International Conference on Service-Oriented Computing*, pages 130–143, 2005.

[67] MICHAEL C JAEGER, GERO MUHL, AND SEBASTIAN GOLZE. **QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms**. In *Proceedings of the International Conference on CoopIS, DOA, and ODBASE on the Move to Meaningful Internet Systems*, pages 646–661, 2005.

[68] MICHAEL C JAEGER, GERO MUHL, AND SEBASTIAN GOLZE. **QoS-aware composition of Web services: a look at selection algorithms**. In *Proceedings of the IEEE International Conference on Web Services*, pages 807–808, July 2005.

[69] YAN GAO, JUN NA, BIN ZHANG, LEI YANG, AND QIANG GONG. **Optimal Web Services Selection Using Dynamic Programming**. In *Proceedings of the 11th IEEE Symposium on Computers & Communications*, pages 365–370, June 2006.

[70] ZHENQIU HUANG, WEI JIANG, SONGLIN HU, AND ZHIYONG LIU. **Effective Pruning Algorithm for QoS-Aware Service Composition**. In *Proceedings of the IEEE Conference on Commerce & Enterprise Computing*, pages 519–522, July 2009.

[71] CHANGLIN WAN, C. ULLRICH, LIMIN CHEN, RUI HUANG, JIEWEN LUO, AND ZHONGZHI SHI. **On Solving QoS-Aware Service Selection Problem with Service Composition**. In *Proceedings of the 7th International Conference on Grid & Cooperative Computing (GCC)*, pages 467–474, Oct 2008.

[72] DONGMEI LIU, ZHIQING SHAO, CAIZHU YU, AND GUISHENG FAN. **A Heuristic QoS-Aware Service Selection Approach to Web Service Composition**. In *Proceedings of the 8th IEEE/ACIS Intl. Conf. on Computer and Information Science*, pages 1184–1189, June 2009.

[73] MARCO AIELLO, ELIE EL KHOURY, ALEXANDER LAZOVIK, AND PATRICK RATELBAND. **Optimal QoS-Aware Web Service Composition**. In *Proceedings of the 7th IEEE Intl. Conf. on E-Commerce Tech.*, pages 491–494, July 2009.

[74] MIN CHEN AND YUHONG YAN. **QoS-aware Service Composition over Graphplan through Graph Reachability**. In *Proceedings of the 2014 IEEE International Conference on Services Computing*, pages 544–551. IEEE, 2014.

[75] MIN LIU, MINGRUI WANG, WEIMING SHEN, NAN LUO, AND JUNWEI YAN. **A Quality of Service (QoS)-aware Execution Plan Selection Approach for a Service Composition Process**. *Future Generation Computer Systems*, **28**(7):1080–1089, July 2012.

[76] BIN ZHANG GAO, YAN, JUN NA, LEI YANG, AND DAI. **Optimal Selection of Web Services for Composition Based on Interface-Matching and Weighted Multistage Graph**. In *Proceedings of the 6th International Conference on Parallel & Distributed Computing, Applications and Technologies*, pages 336–338, 2005.

[77] YANG LI, JINPENG HUAI, TING DENG, HAILONG SUN, HUIPENG GUO, AND ZONGXIA DU. **QoS-aware Service Composition in Service Overlay Networks**. In *Proceedings of the IEEE International Conference on Web Services*, pages 703–710, July 2007.

[78] YAN YANG, SHENGQUN TANG, YOUWEI XU, WENTAO ZHANG, AND LINA FANG. **An Approach to QoS-aware Service Selection in Dynamic Web Service Composition**. In *Proceedings of the 3rd International Conference on Networking & Services*, pages 18–23, June 2007.

[79] YUAN-SHENG LUO, YONG QI, LIN-FENG SHEN, DI HOU, CHANYACHATCHAWAN SAPA, AND YING CHEN. **An Improved Heuristic for QoS-Aware Service Composition Framework**. In *Proceedings of the*

10th IEEE Intl. Conf. on High Performance Computing & Communications, pages 360–367, Sept 2008.

[80] DIANA COMES, HARUN BARAKI, ROLAND REICHLE, MICHAEL ZAPF, AND KURT GEIHS. **Heuristic approaches for qos-based service selection**. In *Proceedings of the 8th International Conference on Service-Oriented Computing*, pages 441–455. Springer, 2010.

[81] YUANSHENG LUO, YONG QI, DI HOU, LIN FENG SHEN, YING CHEN, AND XIAO ZHONG. **A novel heuristic algorithm for QoS-aware end-to-end service composition**. *Computer Communications*, **34**(9):1137–1144, June 2011.

[82] PEDRO FELIPE DO PRADO, LUIS HIDEO VASCONCELOS NAKAMURA, JULIO CEZAR ESTRELLA, MARCOS JOSE SANTANA, AND REGINA HELENA CARLUCCI SANTANA. **A performance evaluation study for QoS-aware web services composition using heuristic algorithms**. In *Proceedings of the 7th Intl. Conf. on Digital Society*, pages 53–58, 2013.

[83] MU LI, DANFENG ZHU, TING DENG, HAILONG SUN, HUIPENG GUO, AND XUDONG LIU. **GOS: a global optimal selection strategies for QoS-aware web services composition**. *Service Oriented Computing and Applications*, **7**(3):181–197, 2013.

[84] ADRIAN KLEIN, FUYUKI ISHIKAWA, AND SHINICHI HONIDEN. **Efficient Heuristic Approach with Improved Time Complexity for Qos-Aware Service Composition**. In *Proceedings of the IEEE International Conference on Web Services*, pages 436–443, July 2011.

[85] LIANYONG QI, YING TANG, WANCHUN DOU, AND JINJUN CHEN. **Combining Local Optimization and Enumeration for QoS-aware Web Service Composition**. In *Proceedings of the IEEE International Conference on Web Services*, pages 34–41, July 2010.

[86] JING LI, YONGWANG ZHAO, MIN LIU, HAILONG SUN, AND DIANFU MA. **An adaptive heuristic approach for distributed QoS-based service composition**. In *Proceedings of the IEEE Sym. on Computers & Communications*, pages 687–694, June 2010.

[87] YINGQIU LI AND TAO WEN. **An Approach of QoS-Guaranteed Web Service Composition Based on a Win-Win Strategy**. In *Proceedings of the IEEE 19th International Conference on Web Services*, pages 628–630, June 2012.

[88] JUN LI, XIAOLIN ZHENG, SONGTAO CHEN, WILLIAMWEI SONG, AND DEREN CHEN. **An efficient and reliable approach for quality-of-service-aware service composition**. *Information Sciences*, **269**:238 – 254, June 2014.

[89] SEOG-CHAN OH, BYUNG-WON ON, ERIC J LARSON, AND DONGWON LEE. **BF*: Web services discovery and composition as graph search problem**. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce & e-Service*, pages 784–786, 2005.

[90] SEN NIU, GUOBING ZOU, YANGLAN GAN, ZHIMIN ZHOU, AND BOFENG ZHANG. **UCLAO* and BHUC: Two Novel Planning Algorithms for Uncertain Web Service Composition**. In *Proceedings of the IEEE International Conference on Services Computing*, pages 531–538. IEEE, 2016.

[91] CHIA-FENG LIN, RUEY-KAI SHEU, YUE-SHAN CHANG, AND SHYAN-MING YUAN. **A relaxable service selection algorithm for QoS-based web service composition**. *Information and Software Technology*, **53**(12):1370–1381, 2011.

[92] PABLO RODRIGUEZ-MIER, MANUEL MUCIENTES, AND MANUEL LAMA. **Automatic web service composition with a heuristic-based search algorithm**. In *Proceedings of the IEEE International Conference on Web Services*, pages 81–88. IEEE, 2011.

[93] FRED GLOVER AND MANUEL LAGUNA. *Tabu Search*. Kluwer Academic Publishers, 1997.

[94] MICHAEL C. JAEGER AND GERO MUEHL. **QoS-based Selection of Services: The Implementation of a Genetic Algorithm**. In *Proceedings of the 2007 ITG-GI Conf. Communication in Distributed Systems*, pages 1–12, Feb 2007.

[95] SEN SU, CHENGWEN ZHANG, AND JUNLIANG CHEN. **An Improved Genetic Algorithm for Web Services Selection**. In *Proceedings of the Distributed Applications & Interoperable Systems*, **4531**, pages 284–295. Springer, 2007.

[96] MAOLIN TANG AND LIFENG AI. **A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[97] CHUNMING GAO, MEILING CAI, AND HUOWANG CHEN. **QoS-aware Service Composition Based on Tree-Coded Genetic Algorithm**. In *Proceedings of the 31st Annual Intl. Conf. Computer Software & Applications (COMPSAC)*, pages 361–367, July 2007.

[98] YANG YU, HUI MA, AND MENGJIE ZHANG. **An adaptive genetic programming approach to QoS-aware web services composition**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1740–1747, June 2013.

[99] JUNLI WANG AND YUBING HOU. **Optimal Web Service Selection based on Multi-Objective Genetic Algorithm**. In *Proceedings of the Intl. Symposium on Computational Intelligence & Design*, pages 553–556. IEEE, Oct 2008.

[100] HUAN LIU, FARONG ZHONG, BANG OUYANG, AND JIAJIE WU. **An Approach for QoS-Aware Web Service Composition Based on Improved Genetic Algorithm**. In *Proceedings of the International*

Conference on Web Information Systems & Mining, pages 123–128, Oct 2010.

[101] YUE MA AND CHENGWEN ZHANG. **Quick convergence of genetic algorithm for QoS-driven web service selection**. *Computer Networks*, **52**(5):1093 – 1104, 2008.

[102] A ERDINC YILMAZ AND PINAR KARAGOZ. **Improved Genetic Algorithm Based Approach for QoS Aware Web Service Composition**. In *Proceedings of the IEEE International Conference on Web Services*, pages 463–470. IEEE, June 2014.

[103] QIAN LI, RUNLIANG DOU, FUZAN CHEN, AND GUOFANG NAN. **A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of things**. *International Journal of Computational Intelligence Systems*, **7**:26–34, 2014.

[104] ZHOU XIANGBING, MA HONGJIANG, AND MIAO FANG. **An Optimal Approach to the QoS-Based WSMO Web Service Composition Using Genetic Algorithm**. In *Proceedings of the International Conference on Service-Oriented Computing*, pages 127–139. Springer, 2013.

[105] HONGHONG JIANG, XIAOHU YANG, KETING YIN, SHUAI ZHANG, AND JERRY A CRISTOFORO. **Multi-path QoS-aware web service composition using variable length chromosome genetic algorithm**. *Information Technology Journal*, **10**(1):113–119, 2011.

[106] GERARDO CANFORA, MASSIMILIANO DI PENTA, RAFFAELE ESPOSITO, AND MARIA LUISA VILLANI. **A Framework for QoS-aware Binding and Re-binding of Composite Web Services**. *Journal of Systems and Software*, **81**(10):1754–1769, oct 2008.

[107] GERARDO CANFORA, MASSIMILIANO DI PENTA, RAFFAELE ESPOSITO, AND MARIA LUISA VILLANI. **A lightweight approach for QoS-aware service composition**. In *Proceedings of the 2nd International Conference on Service-Oriented Computing*, pages 36–47. ACM, 2004.

[108] LIFENG AI AND MAOLIN TANG. **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 Modelling Control & Automation*, pages 738–743, Dec 2008.

[109] MAHMOOD ALLAMEH AMIRI AND HADI SERAJZADEH. **QoS aware web service composition based on genetic algorithm**. In *Proceedings of the 5th Intl. Sym.on Telecommunications*, pages 502–507, Dec 2010.

[110] WEN-YAU LIANG AND CHUN-CHE HUANG. **The generic genetic algorithm incorporates with rough set theory-An application of the web services composition**. *Expert System with Applications*, **36**(3):5549–5556, 2009.

[111] ZHIYONG CHEN, HAIYANG WANG, AND PENG PAN. **An Approach to Optimal Web Service Composition Based on QoS and User Preferences**. In *Proceedings of the Intl. Joint Conf. on Artificial Intelligence*, pages 96–101, April 2009.

[112] TIAN HUAT TAN, MANMAN CHEN, ETIENNE ANDRE, JUN SUN, YANG LIU, AND JIN SONG DONG. **Automated Runtime Recovery for QoS-based Service Composition**. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 563–574, 2014.

[113] XINFENG YE AND RAMI MOUNLA. **A hybrid approach to qos-aware service composition**. In *Proceedings of the IEEE International Conference on Web Services*, pages 62–69, 2008.

[114] ZHANG CHENGWEN, SU SEN, AND CHEN JUNLIANG. **Genetic algorithm on web services selection supporting QoS**. *Chinese Journal of Computers*, **29**(7):1029–1037, 2006.

[115] LIFENG AI AND MAOLIN TANG. **QoS-Based Web Service Composition Accommodating Inter-service Dependencies Using Minimal-Conflict Hill-Climbing Repair Genetic Algorithm**. In *Proceedings of the 4th Intl. Conf. on eScience*, pages 119–126, Dec 2008.

[116] HUI MA, ANQI WANG, AND MENGJIE ZHANG. **A Hybrid Approach Using Genetic Programming and Greedy Search for QoS-Aware Web Service Composition**. In *Proceedings of Trans. on Large-Scale Data & Knowledge-Centered Systems XVIII*, pages 180–205. Springer, 2015.

[117] AUGUSTO SAWCZUK DA SILVA ALEXANDRE, MA HUI, AND ZHANG MENGJIE. **A GP Approach to QoS-Aware Web Service Composition and Selection**. In *Proceedings of the 10th International Conference on Simulated Evolution & Learning*, pages 180 – 191. Springer, 2014.

[118] PABLO RODRIGUEZ-MIER, MANUEL MUCIENTES, MANUEL LAMA, AND MIGUEL COUTO. **Composition of web services through genetic programming**. *Evolutionary Intelligence*, **3**:171–186, 2010.

[119] ANQI WANG, HUI MA, AND MENGJIE ZHANG. **Genetic Programming with Greedy Search for Web Service Composition**. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 9–17. Springer, 2013.

[120] HONGBING WANG, PEISHENG MA, AND XUAN ZHOU. **A quantitative and qualitative approach for nfp-aware web service composition**. In *Proceedings of the IEEE 9th International Conference on Services Computing*, pages 202–209. IEEE, June 2012.

[121] YANG YU, HUI MA, AND MENGJIE ZHANG. **A Genetic Programming approach to distributed QoS-aware web service composition**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1840–1846, July 2014.

[122] YANG YU, HUI MA, AND MENGJIE ZHANG. **A Hybrid GP-Tabu Approach to QoS-Aware Data Intensive Web Service Composition**. In *Proceedings of the 10th International Conference on Simulated Evolution & Learning*, pages 106–118. Springer, 2014.

[123] LIANG BAO, FEN ZHAO, MENGQING SHEN, YUTAO QI, AND PING CHEN. **An Orthogonal Genetic Algorithm for QoS-Aware Service Composition**. *The Computer Journal*, **59**(12):1857–1871, 2016.

[124] PARVIN SHARIFARA, ALIREZA YARI, AND MOHAMMAD MANSOUR RIAHI KASHANI. **An evolutionary algorithmic based web service composition with quality of service**. In *Proceedings of the 7th Intl. Sym. on Telecommunications*, pages 61–65, 2014.

[125] WANGA WENBIN, ZHAOC QIBO SUNB, XINCHAO, AND YANGB FANGCHUN. **An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication**. *International Journal of Computational Intelligence Systems*, **3**(1):18–30, 2012.

[126] MAHMOOD ALLAMEH AMIRI AND HADI SERAJZADEH. **Effective web service composition using particle swarm optimization algorithm**. In *Proceedings of the 6th Intl. Symposium on Telecommunications*, pages 1190–1194, Nov 2012.

[127] JIAN LIU, JUN'E LI, KAIPEI LIU, AND WEN WEI. **A hybrid genetic and particle swarm algorithm for service composition**. In *Proceedings of the 6th Intl. Conf. on Advanced Language Processing & Web Information Technology*, pages 564–567, 2007.

[128] XIANGWEI LIU AND ZHIXIANG YIN. **Web Service Composition with Global Constraint Based on Discrete Particle Swarm Optimization**. In *Proceedings of the 2nd Pacific-Asia Conf. on Web Mining and Web-based Application*, pages 183–186, June 2009.

[129] WANG LI AND HE YAN-XIANG. **Web Service Composition Based on QoS with Chaos Particle Swarm Optimization**. In *Proceedings of the 6th International Conference on Wireless Communications Networking and Mobile Computing*, pages 1–4, Sept 2010.

[130] JIANXIN LIAO, YANG LIU, XIAOMIN ZHU, TONG XU, AND JINGYU WANG. **Niching Particle Swarm Optimization Algorithm for Service Composition**. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 1–6, 2011.

[131] YANG LIU, HUAIKOU MIAO, ZHUANG LI, AND HONGHAO GAO. **QoS-Aware Web Services Composition Based on HQPSO Algorithm**. In *Proceedings of the 1st International Conference on Computers, Networks, Systems and Industrial Engineering*, pages 400–405, May 2011.

[132] XINCHAO ZHAO, BOQIAN SONG, PANYU HUANG, ZICHAO WEN, JIALEI WENG, AND YI FAN. **An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition**. *Applied Soft Computing*, **12**(8):2208–2216, 2012.

[133] CAO JIUXIN, SUN XUESHENG, ZHENG XIAO, LIU BO, AND MAO BO. **Efficient Multi-objective Services Selection Algorithm Based on Particle Swarm Optimization**. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, pages 603–608, Dec 2010.

[134] LONG JUN AND GUI WEIHUA. **An Environment-Aware Particle Swarm Optimization Algorithm for Services Composition**. In *Proceedings of the Intl. Conf. on Computational Intelligence and Software Engineering*, pages 1–4, Dec 2009.

[135] SIMONE A LUDWIG. **Applying Particle Swarm Optimization to Quality-of-Service-Driven Web Service Composition**. In *Proceedings of the 26th Intl. Conf. on Advanced Information Networking and Applications*, pages 613–620, March 2012.

[136] HAMED REZAIE, NASER NEMATBAKSH, AND FARHAD MARDUKHI. **A multi-objective particle swarm optimization for web service composition**. In *Proceedings of the 2nd International Conference on Networked Digital Technologies*, pages 112–122. Springer, 2010.

[137] HONG XIA, YAN CHEN, ZENGZHI LI, HAICHANG GAO, AND YANPING CHEN. **Web Service Selection Algorithm Based on Particle Swarm Optimization**. In *Proceedings of the 8th IEEE Intl. Conf. on Dependable, Autonomic and Secure Computing*, pages 467–472, Dec 2009.

[138] JIANXIN LIAO, YANG LIU, XIAOMIN ZHU, JINGYU WANG, AND QI QI. **A multi-objective service selection algorithm for service composition**. In *Proceedings of the 19th Asia-Pacific Conf. on Communications*, pages 75–80, Aug 2013.

[139] JIANXIN LIAO, YANG LIU, XIAOMIN ZHU, AND JINGYU WANG. **Accurate sub swarms particle swarm optimization algorithm for service composition**. *Journal of Systems and Software*, **90**:191–203, 2014.

[140] TONGGUANG ZHANG. **QoS-aware Web Service Selection based on Particle Swarm Optimization**. *Journal of Networks*, **9**(3):565–570, 2014.

[141] ALEXANDRE SAWCZUK DA SILVA, HUI MA, AND MENGJIE ZHANG. **A Graph-Based Particle Swarm Optimisation Approach to QoS-Aware Web Service Composition and Selection**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3127–3134, July 2014.

[142] HAO YIN, CHANGSHENG ZHANG, BIN ZHANG, YING GUO, AND TINGTING LIU. **A hybrid multiobjective discrete particle swarm optimization algorithm for a SLA-aware service composition problem**. *Mathematical Problems in Engineering*, **1-1**, 2014. doi : 10.1155/2014/252934.

[143] SONDOS BAHADORI, SOMAYEH KAFI, KAMRAN ZAMANI FAR, AND MOHAMMAD REZA KHAYYAMBASHI. **Optimal web service composition using hybrid GA-TABU search**. *Journal of Theoretical & Applied Inf. Tech.*, **9**(1):10–15, 2009.

[144] JOSE ANTONIO PAREJO, PABLO FERNANDEZ, AND ANTONIO RUIZ CORTES. **QoS-aware services composition using Tabu search and hybrid genetic algorithms**. *Actas de Talleres de Ingeniera del Software y Bases de Datos*, **2**(1):55–66, 2008.

[145] NASTARAN JAFARPOUR AND MOHAMMAD REZA KHAYYAMBASHI. **A new approach for QoS-aware Web service composition based on Harmony Search algorithm**. In *Proceedings of the 11th IEEE International Symposium on Web Systems Evolution (WSE)*, pages 75–78, Sept 2009.

[146] MERZOUG MOHAMMED, MOHAMMED AMINE CHIKH, AND HADJILA FETHALLAH. **QoS-aware web service selection based on harmony search**. In *Proceedings of the 4th International Symposium on ISKO-Maghreb: Concepts & Tools for knowledge Management*, pages 1–6, Nov 2014.

[147] NASTARAN JAFARPOUR AND MOHAMMAD REZA KHAYYAMBASHI. **QoS-aware selection of web service composition based on Harmony Search algorithm**. In *Proceedings of the 12th International Conference on Advanced Communication Technology*, pages 1345–1350, Feb 2010.

[148] LONGFEI YAN, YI MEI, HUI MA, AND MENGJIE ZHANG. **Evolutionary web service composition: A graph-based memetic algorithm**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 201–208. IEEE, 2016.

[149] JIUYUN XU AND STEPHAN REIFF-MARGANIEC. **Towards Heuristic Web Services Composition Using Immune Algorithm**. In *Pro-*

ceedings of the IEEE International Conference on Web Services, pages 238–245, Sept 2008.

[150] GAO YAN, NA JUN, ZHANG BIN, YANG LEI, GONG QIANG, AND DAI YU. **Immune algorithm for selecting optimum services in Web services composition**. *Wuhan University Journal of Natural Sciences*, **11**(1):221–225, Jan 2006.

[151] XINCHAO ZHAO, ZICHAO WEN, AND XINGMEI LI. **QoS-aware web service selection with negative selection algorithm**. *Knowledge and Information Systems*, **40**(2):349–373, 2014.

[152] CRISTINABIANCA POP, VIORICAROZINA CHIFU, IOAN SALOMIE, AND MIHAELA DINSOREANU. **Immune-Inspired Method for Selecting the Optimal Solution in Web Service Composition**. In *Proceedings of the International Workshop on Resource Discovery*, pages 1–17, 2010.

[153] WANG YUNWU. **Application of Chaos Ant Colony Algorithm in Web Service Composition Based on QoS**. In *Proceedings of the International Forum on Information Technology and Applications & Applications*, pages 225–227, May 2009.

[154] YAMEI XIA, JUNLIANG CHEN, AND XIANGWU MENG. **On the Dynamic Ant Colony Algorithm Optimization Based on Multi-pheromones**. In *Proceedings of the 7th IEEE/ACIS Intl. Conf. on Computer & Information Science (ICIS)*, pages 630–635, May 2008.

[155] WEI ZHANG, CARL K CHANG, TAIMING FENG, AND HSIN-YI JIANG. **QoS-Based Dynamic Web Service Composition with Ant Colony Optimization**. In *Proceedings of the IEEE 34th Annual Computer Software & Applications Conf. (COMPSAC)*, pages 493–502, July 2010.

[156] WANG LI AND HE YAN-XIANG. **A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO**. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, pages 218–224, 2010.

[157] ZONGKAI YANG, CHAOWANG SHANG, QINGTANG LIU, AND CHENGLING ZHAO. **A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm**. *Journal of Computational Information Systems*, **6**(8):2617–2622, may 2010.

[158] CHENGYING MAO, JIFU CHEN, AND XINXIN YU. **An Empirical Study on Meta-Heuristic Search-Based Web Service Composition**. In *Proceedings of the IEEE 9th International Conference on e-Business Engineering*, pages 117–122, Sept 2012.

[159] DENGHUI WANG, HAO HUANG, AND CHANGSHENG XIE. **A Novel Adaptive Web Service Selection Algorithm Based on Ant Colony Optimization for Dynamic Web Service Composition**. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, pages 391–399, 2014.

[160] ZHAO SHANSHAN, WANG LEI, MA LIN, AND WEN ZEPENG. **An Improved Ant Colony Optimization Algorithm for QoS-Aware Dynamic Web Service Composition**. In *Proceedings of the 2012 International Conference on Industrial Control & Electronics Engineering*, pages 1998–2001, Aug 2012.

[161] ZHI-JIAN WANG, ZHI-ZHONG LIU, XIAO-FENG ZHOU, AND YUAN-SHENG LOU. **An approach for composite web service selection based on DGQoS**. *The Intl. Journal of Advanced Manufacturing Technology*, **56**:1167–1179, 2011.

[162] XIAO-QIN FAN, XIAN-WEN FANG, AND CHANG-JUN JIANG. **Research on Web service selection based on cooperative evolution**. *Expert System with Applications*, **38**(8):9736–9743, 2011.

[163] FLORIAN ROSENBERG, MAX BENJAMIN MÜLLER, PHILIPP LEITNER, ANTON MICHLMAYR, ATHMAN BOUGUETTAYA, AND SCHAHRAM DUSTDAR. **Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions**. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 97–104, July 2010.

[164] PRASHANTH PODILI, KK PATTANAIK, AND PRASHANTH SINGH RANA. **BAT and Hybrid BAT Meta-Heuristic for Quality of Service-Based Web Service Selection**. *Journal of Intelligent Systems*, 2016.

[165] YINGQIANG ZHOU, CHANGSHENG ZHANG, AND BIN ZHANG. **Multi-objective service compositon optimization using differential evolution**. In *Proceedings of the 11th International Conference on Natural Computation*, pages 233–238. IEEE, 2015.

[166] HAIFANG WANG, XIAOFEI XU, ZHIZHONG LIU, AND ZHONGJIE WANG. **The Configurability Study on Artificial Bee Colony Algorithm for QoS-aware Service Composition**. In *Proceedings of the 2015 International Conference on Service Science*, pages 106–112. IEEE, 2015.

[167] JUN HE, LIANG CHEN, XIAOLONG WANG, AND YONGGANG LI. **Web Service Composition Optimization Based on Improved Artificial Bee Colony Algorithm**. *Journal of Networks*, **8**(9):2143–2149, 2013.

[168] G. KOUSALYA, D. PALANIKKUMAR, AND P. R. PIRIYANKAA. **Optimal Web Service Selection and Composition Using Multi-objective Bees Algorithm**. In *Proceedings of the 9th IEEE Intl. Sym. on Parallel & Distributed Processing with Applications Workshops*, pages 193–196, May 2011.

[169] VIORICA ROZINA CHIFU, CRISTINA BIANCA POP, IOAN SALOMIE, MIHAELA DINSOREANU, ALEXANDRU NICOLAE NICULICI, AND DUMITRU SAMUEL SUIA. **Selecting the optimal web service**

composition based on a multi-criteria bee-inspired method. In *Proceedings of the 12th Intl. Conf. on Information Integration and Web-based Applications & Services*, pages 40–47, 2010.

[170] CRISTINA BIANCA POP, VIORICA ROZINA CHIFU, IOAN SALOMIE, AND MONICA VLAD. **Cuckoo-inspired hybrid algorithm for selecting the optimal Web service composition**. In *Proceedings of the IEEE Intl. Conf. on Intelligent Computer Communication & Processing*, pages 33–40, Aug 2011.

[171] SERIALRAYENE BOUSSALIA AND ALLAOUA CHAOUI. **Optimizing QoS-Based Web Services Composition by Using Quantum Inspired Cuckoo Search Algorithm**. In *Proceedings of the International Conference on Mobile Web and Information Systems*, pages 41–55. Springer, 2014.

[172] ARION DE CAMPOS JR, AURORA TR POZO, SILVIA R VERGILIO, AND TAYLOR SAVEGNAGO. **Many-Objective Evolutionary Algorithms in the Composition of Web Services**. In *Proceedings of the 11th Brazilian Sym. on Neural Networks*, pages 152–157, Oct 2010.

[173] YIWEN ZHANG, GUANGMING CUI, YAN WANG, XING GUO, AND SHU ZHAO. **An optimization algorithm for service composition based on an improved FOA**. *Tsinghua Science & Technology*, **20**(1):90–99, 2015.

[174] MELQUÍADES PÉREZ, FRANCISCO ALMEIDA, AND J MARCOS MORENO-VEGA. **A hybrid GRASP-path relinking algorithm for the capacitated p–hub median problem**. In *Proceedings of the International Workshop on Hybrid Metaheuristics*, pages 142–153. Springer, 2005.

[175] EYHAB AL-MASRI AND QUSAY H MAHMOUD. **Qos-based discovery and ranking of web services**. In *Proceedings of the 16th International Conference on Computer Communications and Networks*, pages 529–534. IEEE, 2007.

[176] MAHMOOD ALLAMEH AMIRI AND HADI SERAJZADEH. **QoS aware web service composition based on genetic algorithm**. In *Proceedings of the 5th International Symposium on Telecommunications (IST)*, pages 502–507, 2010.

[177] HUIYUAN ZHENG, WEILIANG ZHAO, JIAN YANG, AND ATHMAN BOUGUETTAYA. **QoS Analysis for Web Service Compositions with Complex Structures**. *IEEE Transactions on Services Computing*, **6**(3):373–386, July 2013.

[178] DAVID LAYZER. **Genetic variation and progressive evolution**. *American Naturalist*, pages 809–826, 1980.

[179] NARAYANASWAMY BALAKRISHNAN AND VALERY B NEVZOROV. **Discrete Uniform Distribution**. *A Primer on Statistical Distributions*, pages 27–37, 2005.

[180] ZIBIN ZHENG, YILEI ZHANG, AND MICHAEL R LYU. **Investigating QoS of Real-World Web Services**. *IEEE Transactions on Services Computing*, **7**(1):32–39, Jan 2014.

[181] YILEI ZHANG, ZIBIN ZHENG, AND MICHAEL R LYU. **WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services**. In *Proceedings of the IEEE 22nd Intl. Symposium on Software Reliability Engineering*, pages 210–219, Nov 2011.

[182] EYHAB AL-MASRI AND QUSAY H MAHMOUD. **Investigating web services on the world wide web**. In *Proceedings of the 17th International conference on World Wide Web*, pages 795–804. ACM, 2008.

[183] JOSÉ A VILLASENOR ALVA AND ELIZABETH GONZÁLEZ ESTRADA. **A generalization of Shapiro-Wilk's test for multivariate normality**. *Communications in Statistics Theory and Methods*, **38**(11):1870–1883, 2009.

[184] AHMED MOSTAFA AND MINJIE ZHANG. **Multi-Objective Service Composition in Uncertain Environments**. *IEEE Transactions on Services Computing*, 2015. doi : 10.1109/TSC.2015.2443785.

[185] DANDAN WANG, YANG YANG, AND ZHENQIANG MI. **A genetic-based approach to web service composition in geo-distributed cloud environment**. *Computers & Electrical Engineering*, **43**:129–141, 2015.

[186] XIANZHI WANG, ZHONGJIE WANG, AND XIAOFEI XU. **An improved artificial bee colony approach to qos-aware service selection**. In *Proceedings of the 20th International Conference on Web Services*, pages 395–402. IEEE, 2013.

[187] MIN LIU, MINGRUI WANG, WEIMING SHEN, NAN LUO, AND JUNWEI YAN. **A quality of service (QoS)-aware execution plan selection approach for a service composition process**. *Future Generation Computer Systems*, **28**(7):1080–1089, 2012.

[188] PABLO RODRIGUEZ-MIER, MANUEL MUCIENTES, AND MANUEL LAMA. **Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition**. *IEEE Transactions on Services Computing*, **10**(4):547–559, 2017.

[189] ALI E YILMAZ AND PINAR KARAGOZ. **Improved genetic algorithm based approach for qos aware web service composition**. In *Proceedings of the IEEE International Conference on Web Services*, pages 463–470. IEEE, 2014.

[190] ZHIJUN DING, JUNJUN LIU, YOUQING SUN, CHANGJUN JIANG, AND MENGCHU ZHOU. **A transaction and QoS-aware service selection approach based on genetic algorithm**. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **45**(7):1035–1046, 2015.

[191] VLADIMIR VAPNIK. *Statistical learning theory*. Wiley, New York, 1998.

[192] EDWARD G CARMINES AND RICHARD A ZELLER. *Reliability and validity assessment*. Sage publications, 1979.

[193] DONALD W ZIMMERMAN. **Teacher's corner: A note on interpretation of the paired-samples t test**. *Journal of Educational and Behavioral Statistics*, **22**(3):349–360, 1997.

[194] SIDNEY SIEGEL. **Nonparametric statistics for the behavioral sciences**. *McGraw-hill, New York*, 1956.

[195] FRANK WILCOXON. **Individual comparisons by ranking methods**. *Biometrics bulletin*, **1**(6):80–83, 1945.

[196] DN JOANES AND CA GILL. **Comparing measures of sample skewness and kurtosis**. *Journal of the Royal Statistical Society: Series D (The Statistician)*, **47**(1):183–189, 1998.

[197] THOMAS HILL, PAWEL LEWICKI, AND PAWEŁ LEWICKI. *Statistics: methods and applications: a comprehensive reference for science, industry, and data mining*. StatSoft, Inc., 2006.

[198] ALI REZA MEHRABIAN AND CARO LUCAS. **A novel numerical optimization algorithm inspired from weed colonization**. *Ecological informatics*, **1**(4):355–366, 2006.

[199] STAN BROWN. **Measures of Shape: Skewness and Kurtosis**. `http://www.tc3.edu/instruct/sbrown/stat/shape.html`.

[200] RAINER STORN AND KENNETH PRICE. **Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces**. *Journal of global optimization*, **11**(4):341–359, 1997.

[201] RICCARDO POLI, JAMES KENNEDY, AND TIM BLACKWELL. **Particle swarm optimization**. *Swarm Intelligence*, **1**(1):33–57, 2007.

[202] XIN-SHE YANG AND SUASH DEB. **Cuckoo search via Lévy flights**. In *Proceedings of the World Congress on Nature & Biologically Inspired Computing*, pages 210–214, 2009.

[203] UPPALA SHIVAKUMAR, VADLAMANI RAVI, AND G.R. GANGADHARAN. **Ranking cloud services using fuzzy multi-attribute decision making**. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 1–8. IEEE, 2013.

[204] SAURABH KUMAR GARG, STEVE VERSTEEG, AND RAJKUMAR BUYYA. **A framework for ranking of cloud computing services**. *Future Generation Computer Systems*, **29**(4):1012–1023, 2013.

[205] PUNAM BEDI, HARMEET KAUR, AND BHAVNA GUPTA. **Trustworthy service provider selection in cloud computing environment**. In *Proceedings of the 2012 International Conference on Communication Systems and Network Technologies*, pages 714–719. IEEE, 2012.

[206] HONG-KYU KWON AND KWANG-KYU SEO. **A decision-making model to choose a cloud service using Fuzzy AHP**. *Advanced Science and Technology Letters*, **35**:93–96, 2013.

[207] CHRISTIAN ESPOSITO, MASSIMO FICCO, FRANCESCO PALMIERI, AND ANIELLO CASTIGLIONE. **Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory**. *IEEE Transactions on Computers*, **65**(8):2348–2362, 2016.

[208] ANG LI, XIAOWEI YANG, SRIKANTH KANDULA, AND MING ZHANG. **CloudCmp: comparing public cloud providers**. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.

[209] PER ANDERSEN AND NIELS CHRISTIAN PETERSEN. **A procedure for ranking efficient units in data envelopment analysis**. *Management science*, **39**(10):1261–1264, 1993.

[210] ALI AZADEH, SF GHADERI, AND H IZADBAKHSH. **Integration of DEA and AHP with computer simulation for railway system improvement and optimization**. *Applied Mathematics and Computation*, **195**(2):775–785, 2008.

[211] LAWRENCE M SEIFORD AND JOE ZHU. **An investigation of returns to scale in data envelopment analysis**. *Omega*, **27**(1):1–11, 1999.

[212] RAJIV D BANKER, ABRAHAM CHARNES, AND WILLIAM WAGER COOPER. **Some models for estimating technical and scale inefficiencies in data envelopment analysis**. *Management science*, **30**(9):1078–1092, 1984.

[213] ABRAHAM CHARNES, WILLIAM W COOPER, AND EDWARDO RHODES. **Measuring the efficiency of decision making units**. *European journal of operational research*, **2**(6):429–444, 1978.

[214] WILLIAM W COOPER, LAWRENCE M SEIFORD, AND JOE ZHU. *Handbook on data envelopment analysis*. Springer Science & Business Media, 2011.

[215] THOMAS L SAATY. **What is the analytic hierarchy process?** In *Mathematical models for decision support*, pages 109–121. Springer, 1988.

[216] THOMAS L SAATY. *The analytic network process: decision making with dependence and feedback; the organization and prioritization of complexity*. RWS Publications, USA, 1996.

[217] GWO-HSHIUNG TZENG AND JIH-JENG HUANG. *Multiple attribute decision making: methods and applications*. CRC press, 2011.

[218] GOLAM KABIR AND M HASIN. **Comparitive Analysis of TOPSIS and FUZZY TOPSIS for the Evalution of Travel Website Service Quality**. *International Journal for Quality Research*, **6**(3), 2012.

[219] CHEN-TUNG CHEN. **Extensions of the TOPSIS for group decision-making under fuzzy environment**. *Fuzzy sets and systems*, **114**(1):1–9, 2000.

[220] GUO-DONG LI, DAISUKE YAMAGUCHI, AND MASATAKE NAGAI. **A grey-based decision-making approach to the supplier selection problem**. *Mathematical and computer modelling*, **46**(3):573–581, 2007.

[221] YONG-HUANG LIN, PIN-CHAN LEE, AND HSIN-I TING. **Dynamic multi-attribute decision making model with grey number evaluations**. *Expert Systems with Applications*, **35**(4):1638–1644, 2008.

[222] DA-YONG CHANG. **Applications of the extent analysis method on fuzzy AHP**. *European journal of operational research*, **95**(3):649–655, 1996.

[223] FRANCISCO RODRIGUES LIMA JUNIOR, LAURO OSIRO, AND LUIZ CESAR RIBEIRO CARPINETTI. **A comparison between Fuzzy AHP and Fuzzy TOPSIS methods to supplier selection**. *Applied Soft Computing*, **21**:194–209, 2014.

[224] AHMET CAN KUTLU AND MEHMET EKMEKÇIOĞLU. **Fuzzy failure modes and effects analysis by using fuzzy TOPSIS-based fuzzy AHP**. *Expert Systems with Applications*, **39**(1):61–67, 2012.

[225] VUONG XUAN TRAN, HIDEKAZU TSUJI, AND RYOSUKE MASUDA. **A new QoS ontology and its QoS-based ranking algorithm for Web services**. *Simulation Modelling Practice and Theory*, **17**(8):1378–1398, 2009.

[226] MANISH GODSE AND SHRIKANT MULIK. **An Approach for Selecting Software-as-a-Service (SaaS) Product**. In *Proceedings of the IEEE International Conference on Cloud Computing*, pages 155–158. IEEE, 2009.

[227] BARBARA PERNICI AND S HOSSEIN SIADAT. **Selection of Service Adaptation Strategies Based on Fuzzy Logic**. In *Proceedings of the IEEE World Congress on Services*, pages 99–106, 2011.

[228] SALAJA SILAS, ELIJAH BLESSING RAJSINGH, AND KIRUBAKARAN EZRA. **Efficient service selection middleware using ELECTRE methodology for cloud environments**. *Information Technology Journal*, **11**(7):868–875, 2012.

[229] SHIXING YAN, CHUNQING CHEN, GUOPENG ZHAO, AND BU SUNG LEE. **Cloud service recommendation and selection for enterprises**. In *Proceedings of the 8th International Conference on Network and Service Management and Workshop on Systems Virtualization Management*, pages 430–434. IEEE, 2012.

[230] ZIBIN ZHENG, XINMIAO WU, YILEI ZHANG, MICHAEL R LYU, AND JIANMIN WANG. **QoS ranking prediction for cloud services**. *IEEE Transactions on Parallel and Distributed Systems*, **24**(6):1213–1222, 2013.

[231] CHUNXIANG XU, YUPENG MA, AND XIAOBO WANG. **A Non-Parametric Data Envelopment Analysis Approach for Cloud Services Evaluation**. In *Proceedings of the Service-Oriented Computing-ICSOC 2014 Workshops*, pages 250–255. Springer, 2015.

[232] CHING-LAI HWANG AND KWANGSUN YOON. *Multiple attribute decision making: Methods and applications*. Springer-Verlag, 1981.

[233] CHI-CHUN LO, DING-YUAN CHEN, CHEN-FANG TSAI, AND KUO-MING CHAO. **Service Selection Based on Fuzzy TOPSIS Method**. In *Proceedings of the IEEE 24th International Conference on Advanced Information N/W & Applications Workshops*, pages 367–372, 2010.

[234] PRASAD SARIPALLI AND GOPAL PINGALI. **MADMAC: Multiple Attribute Decision Methodology for Adoption of Clouds**. In *Proceedings of the IEEE International Conference on Cloud Computing*, pages 316–323. IEEE, 2011.

[235] YANPING ZHANG, ZIHUI JING, AND YIWEN ZHANG. **MR-IDPSO: a novel algorithm for large-scale dynamic service composition**. *Tsinghua Science and Technology*, **20**(6):602–612, 2015.

[236] JEFFREY DEAN AND SANJAY GHEMAWAT. **MapReduce: simplified data processing on large clusters**. *Communications of the ACM*, **51**(1):107–113, 2008.

[237] MOHAMMAD ALRIFAI, DIMITRIOS SKOUTAS, AND THOMAS RISSE. **Selecting skyline services for QoS-based web service composition**. In *Proceedings of the 19th International conference on World Wide Web*, pages 11–20. ACM, 2010.

[238] HYUCK HAN, HYUNGSOO JUNG, SHINGYU KIM, AND HEON Y YEOM. **A skyline approach to the matchmaking web service**. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 436–443. IEEE, 2009.

[239] QI YU AND ATHMAN BOUGUETTAYA. **Computing service skyline from uncertain qows**. *IEEE Transactions on Services Computing*, **3**(1):16–29, 2010.

[240] FAN ZHANG, KAI HWANG, SAMEE ULLAH KHAN, AND QUTAIBAH M MALLUHI. **Skyline Discovery and Composition of Multi-Cloud Mashup Services**. *IEEE Transactions on Services Computing*, **9**(1):72–83, 2016.

[241] LIANG CHEN, LI KUANG, AND JIAN WU. **Mapreduce based skyline services selection for qos-aware composition**. In *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2035–2042. IEEE, 2012.

[242] JIAN WU, LIANG CHEN, QI YU, LI KUANG, YILUN WANG, AND ZHAOHUI WU. **Selecting skyline services for QoS-aware composition by upgrading MapReduce paradigm**. *Cluster computing*, **16**(4):693–706, 2013.

[243] XINCHAO ZHAO, BOQIAN SONG, PANYU HUANG, ZICHAO WEN, JIALEI WENG, AND YI FAN. **An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition**. *Applied Soft Computing*, **12**(8):2208–2216, 2012.

[244] TL SAATY. **Fundamentals of decision making and priority theory with analytical hierarchical process**. *RWS Publications*, **6**:3–95, 1980.

[245] QINGFU ZHANG, JIANYONG SUN, AND EDWARD TSANG. **An evolutionary algorithm with guided mutation for the maximum clique problem**. *IEEE Transactions on Evolutionary Computation*, **9**(2):192–200, 2005.

[246] ZIBIN ZHENG, YILEI ZHANG, AND MICHAEL R LYU. **Investigating QoS of real-world web services**. *IEEE Transactions on Services Computing*, **7**(1):32–39, 2014.

[247] AHMED MOUSTAFA AND MINJIE ZHANG. **Multi-objective service composition using reinforcement learning**. In *Proceedings of the 11th international conference on Service-Oriented Computing (ICSOC)*, pages 298–312. Springer, 2013.

[248] NOOR ELAIZA ABD KHALID, AHMAD FIRDAUS AHMAD FADZIL, AND MAZANI MANAF. **Adapting MapReduce framework for genetic algorithm with large population**. In *Proceedings of the IEEE Conference on Systems, Process & Control*, pages 36–41. IEEE, 2013.

[249] SIMONE A LUDWIG. **Clonal selection based genetic algorithm for workflow service selection**. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2012.

[250] DAAN WIERSTRA, TOM SCHAUL, JAN PETERS, AND JUERGEN SCHMIDHUBER. **Natural evolution strategies**. In *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387. IEEE, 2008.

[251] KN KRISHNANAND AND DEBASISH GHOSE. **Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications**. *Multiagent and Grid Systems*, **2**(3):209–222, 2006.

[252] SEYEDALI MIRJALILI, SEYED MOHAMMAD MIRJALILI, AND ANDREW LEWIS. **Grey wolf optimizer**. *Advances in Engineering Software*, **69**:46–61, 2014.

[253] R RAO. **Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems**. *International Journal of Industrial Engineering Computations*, **7**(1):19–34, 2016.

[254] HOJJAT EMAMI AND FARNAZ DERAKHSHAN. **Election algorithm: A new socio-politically inspired strategy**. *AI Communications*, **28**(3):591–603, 2015.

[255] HAMED SHAH-HOSSEINI. **Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation**. *International Journal of Computational Science and Engineering*, **6**(1-2):132–140, 2011.

[256] MICROSOFT. **Cortana Intelligence Gallery.** `https://gallery.cortanaintelligence.com/Collection/Cognitive-Services-1`. [Online; accessed 28-Mar-2016].

[257] MICROSOFT. **Microsoft Windows Azure.** `https://azure.microsoft.com/en-in/services/cognitive-services/`. [Online; accessed 15-Sep-2017].

[258] JAMES KOBIELUS. **Cognitive computing can take the semantic Web to the next level**. https://www.infoworld.com/article/2610447/big-data/cognitive-computing-can-take-the-semantic-\web-to-the-next-level.html. [Online; accessed 30-Jan-2017].

[259] LIDIA OGIELA AND MAREK R. OGIELA. **Towards cognitive service management and semantic information sharing in the Cloud**. *Multimedia Tools and Applications*, Oct 2017. doi : 10.1007/s11042-017-5302-9.

[260] DARRELL WHITLEY. **A genetic algorithm tutorial**. *Statistics and computing*, **4**(2):65–85, 1994.

[261] DAVID E GOLDBERG AND JOHN H HOLLAND. **Genetic algorithms and machine learning**. *Machine learning*, **3**(2):95–99, 1988.

[262] SACHCHIDA NAND CHAURASIA AND ALOK SINGH. **A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set**. *Applied Intelligence*, **43**(3):512–529, 2015.

[263] KAORU TONE. **A slacks-based measure of super-efficiency in data envelopment analysis.** *European Journal of Operational Research*, **143**(3):32–41, 2002.

[264] OMKARPRASAD S VAIDYA AND SUSHIL KUMAR. **Analytic hierarchy process: An overview of applications**. *European Journal of operational research*, **169**(1):1–29, 2006.

[265] THOMAS LORIE SAATY. *Decision making with dependence and feedback: The analytic network process*. in Analytic hierarchy process series. RWS publications Pittsburgh, 2001.

[266] THOMAS L SAATY. *The analytic network process*. Springer, 2006.

[267] RJ KUO, CW HSU, AND YL CHEN. **Integration of fuzzy ANP and fuzzy TOPSIS for evaluating carbon performance of suppliers**. *International Journal of Environmental Science and Technology*, **12**(12):3863–3876, 2015.

[268] LOTFI A ZADEH. **Fuzzy sets**. *Information and control*, **8**(3):338–353, 1965.

[269] LOTFI A ZADEH. **Outline of a new approach to the analysis of complex systems and decision processes**. *IEEE Tran. on Systems, Man and Cybernetics*, **3**(1):28–44, 1973.

[270] DIDIER DUBOIS AND HENRI PRADE. **Operations on fuzzy numbers**. *International Journal of systems science*, **9**(6):613–626, 1978.

[271] JU-LONG DENG. **Control problems of grey systems**. *Systems & Control Letters*, **1**(5):288–294, 1982.

[272] JU-LONG DENG. *Properties of relational space for grey systems*. in Essential Topics on Grey System – Theory and Applications. China Ocean Press, 1988.

[273] QINGYIN WANG AND HEQING WU. **The concept of grey number and its property**. In *Proceedings of the NAFIPS*, pages 45–49, 1998.

[274] JUAN MANUEL BENITEZ, JUAN CARLOS MARTÍN, AND CONCEPCIÓN ROMÁN. **Using fuzzy number for measuring quality of service in the hotel industry**. *Tourism management*, **28**(2):544–555, 2007.

[275] MING-CHYUAN LIN, CHEN-CHENG WANG, MING-SHI CHEN, AND C ALEC CHANG. **Using AHP and TOPSIS approaches in customer-driven product design process**. *Computers in Industry*, **59**(1):17–31, 2008.

[276] CHEN-TUNG CHEN, CHING-TORNG LIN, AND SUE-FN HUANG. **A fuzzy approach for supplier evaluation and selection in supply chain management**. *International journal of production economics*, **102**(2):289–301, 2006.

[277] PJM VAN LAARHOVEN AND WITOLD PEDRYCZ. **A fuzzy extension of Saaty's priority theory**. *Fuzzy sets and Systems*, **11**(1-3):229–241, 1983.

[278] BASAR OZTAYSI. **A decision model for information technology selection using AHP integrated TOPSIS-Grey: The case of content management systems**. *Knowledge Based Systems*, **70**:44–54, 2014.

Appendix A

Overview of Techniques Used

A.1 Statistical tests

A.1.1 T-test

A t-test is an evaluation of two populations means through the use of statistical examination [193, 194]. A t-test with two samples is commonly used and test the difference between the two samples when the variances of two normal distributions are not known. The t-test described as follows:

$$T - test = \frac{\mid \mu_1 - \mu_2 \mid}{\sqrt{\frac{\alpha_1^2}{n_1} + \frac{\alpha_2^2}{n_2}}} \tag{A.1}$$

where μ_1 and μ_2 represent the means of the samples, α_1 and α_2 represent the standard deviation of the samples, and n_1 and n_2 represent the number of samples.

A.1.2 Wilcoxon signed-rank test

The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test that can be used to determine the whether two independent samples were selected

from the population with the same distribution [195]. The wilcoxon signed-rank test described as follows:

$$Wilcoxon - test = \frac{R - \mu_R}{\alpha_R} \tag{A.2}$$

where

$$\mu_R = \frac{n_1(n_1 + n_2 + 1)}{2} \tag{A.3}$$

$$\alpha_R = \frac{n_1 n_2(n_1 + n_2 + 1)}{12} \tag{A.4}$$

R represent smallest of absolute values of sums, n_1 and n_2 represents smaller and larger of samples sizes respectively. μ and α represents the mean and standard deviation of the two samples.

A.2 Meta-heuristic Algorithms

A.2.1 Genetic Algorithm

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics process in which stronger individuals are likely the winner in a competitive environment. GAs are utilized for producing acceptable solutions to optimization problems when their search space cannot be traversed efficiency by conventional optimization methods (linear programming, heuristic, depth-first, and breath-first) [260, 261].

GA operates on a population of individuals. Each individual in the population represents a possible solution, and each individual is evaluated depending upon their fitness. The fitness shows how well an individual of the population. Initially, GA starts with a randomly generated population. Afterwards, an evolution process of the population takes place by using genetic operators: selection, crossover, and mutation. Through the selection process, the best fittest individuals will be selected to go to the next population. The crossover operator exchanges the genetic material of two individuals, generating two new individuals.

Mutation operator changes the genetic material of an individual. The application of the genetic operators upon the individuals of the population continues until a sufficient solution is found. The solution is usually achieved by a predefined stopping condition, i. e a certain number of generations is reached, or the amount of variation of individuals between different generations is agreed or a predefined value of fitness. The result may not be an optimal solution, but the algorithm can be calibrated so that it always produces a feasible result that satisfies certain criteria. The pseudocode of classical GA is presented in Algorithm A.1 .

Algorithm A.1 A classical genetic algorithm

Input: Individual, Population.
Output: Best individuals.
Generate an initial random population;
1: **while** $iteration \leq maxiteration$ **do**
2: iter = iter+ 1;
3: Evaluate the fitness of each individual;
4: Select the individuals according to their fitness for next generation;
5: Perform crossover with probability pc;
6: Perform mutation with probability pm;
7: population = selected individuals after crossover and mutation;
8: **end while**

A.2.2 Invasive Weed Optimization

In conventional Invasive Weed Optimization, the feasible solutions are represented as weeds and the set of all weeds are represented as the population. A finite number of weeds are spread over the search area. Each weed produces new weeds according to its fitness. The generated weeds are randomly distributed over the search space by normally distributed random numbers. This process continues until the maximum number of weeds is reached. Only the weeds with higher fitness can continue and produce seed; the others are eliminated. The process continues until the maximum number of iterations is reached or the weed with the higher fitness is close to the optimal solution. The algorithm procedure is as follows:

Step 1: Initialize a population An initial solution population is spread out over the D-dimensional search space with random positions.

Step 2: Reproduction The higher a weed's fitness is, the more seeds it produces. The formula for weeds producing seeds is

$$weed_n = \frac{f - f_{min}}{f_{max}}(s_{max} - s_{min}) + s_{min} \tag{A.5}$$

where f represents the current weed's fitness, f_{min} and f_{max} represent the highest and lowest fitness values in the current population, and s_{min} and s_{max} represent the highest and lowest fitness values of the weed.

Step 3: Spatial dispersal The generated seeds are randomly distributed over the D-dimensional search space by normally distributed random numbers with a varying variance. This ensures that seeds will be randomly distributed so that they remain near the parent plant. However, the standard deviation (α) of the random function is reduced in each generation from the previously defined initial value ($alpha_{init}$) to a final value ($alpha_{final}$). A nonlinear alteration that has shown satisfactory performance in simulations is given as follows:

$$\alpha_{cur} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n}(\alpha_{init} - \alpha_{final}) + \alpha_{final} \tag{A.6}$$

where $iter_{max}$ is the maximum number of iterations, α_{cur} is the standard deviation at the current time step, and n is the nonlinear modulation index, i.e., 3.

Step 4: Competitive exclusion After some iterations, the number of weeds in a colony will reach its maximum (P_{MAX}) by fast reproduction. At this time, each weed is allowed to produce seeds. The seeds produced are then allowed to spread over the search area. When all seeds have found their positions in the search area, they are ranked together with their parents (as a colony of weeds). Next, weeds with lower fitness are eliminated to arrive at the maximum allowable population in a colony. In this way, weeds and seeds are ranked together, and the ones with better fitness survive and are allowed to replicate. The population control mechanism also is applied to their offspring until the end of a given run, accomplishing competitive exclusion.

A.2.3 An evolutionary algorithm with guided mutation (EA/G)

The Evolutionary Algorithm with Guided Mutation (EA/G) [245] is a combination of the Estimation of Distribution Algorithm (EDA) and the Genetic Algorithm (GA). Traditional, GAs use crossover and mutation operators to produce an offspring from the selected parents. GAs directly employ only the location information of the solutions and do not make use of global information about the search space which can be collected by keeping track of all the solutions produced since the beginning of the algorithm. On the other hand, EDAs depends only on a probability model to produce an offspring. The probability model characterizes the distribution of the promising solutions in the search space and is updated at each generation using the global statistical information about the search space extracted from the present population at that generation. An offspring is generated by sampling this probability model. Contrary to GAs, EDAs do not directly use the location information of solutions. Here, by the location information of a solution, we mean the information that can uniquely identify a solution in the search space of all solutions [245, 262]. Considering these complementary aspects of GAs and EDAs, Zhang et al [245] devopled a novel evolutionary algorithm with guided mutation (EA/G) algorithm that uses a muation operator named as guided muation (GM) to produce offsprings. This algorithm makes use of both global and local information on solutions found so far to produce offspring. Global information is used to build a probability model of promising solutions. An offspring is produced by sampling from this model, in combination with partial reuse of parent solutions. The EA/G process is as follows:

Initially, an initial solution generated randomly. From this solution, the offspring is generated based on a probability guidance model in such a way that the newly generated solution retains similarity to the parent solution. To achieve this, a certain percentage (user-specified) of the elements of the parents in the offspring generation process. Thus, we are able to control the similarity between an offspring and its parent, and the resultant solution can fall within or close to a promising area as characterized by the probability model.

Initial solution Initially, the set of initial solutions are formed. In our proposed method we select some candidate services for the initial population based on the fitness value (calculation of the fitness function is given in section 5.2.1).

Initial probabilities A univariate marginal distribution model used for the estimation of the distribution of solutions in the search space. In this model, the probability vector $PV = (pv_1, \ldots, pv_R)$ is used to characterize the solution distribution, where R indicates the number of candidate services considered (i.e., $|V|$). PV is initialized with the N_c initial solutions. Once the initial solutions are generated, we count the number of solutions containing the candidate service s_i, and each pv_i is set to the frequency of occurrence of s_i, i.e., the count divided by N_c. We make use of this vector PV in the guided mutation process where it is further updated at each new generation.

Probability update At each generation, a parent population is formed by se lecting the best $N_c/4$ solutions from the current population set, where N_c is the sequence number of active solution. The parent population is used to update PV. The pseudocode for the probability updating is given as Algorithm A.2, where $\lambda \in [0, 1]$ is the learning rate that governs the parent population contribution. With high values of λ, the updated PV reflects more of the characteristics of the parent population, and vice versa. The change in probability for a candidate service depends on the frequency of occurrence of that service in the parent population.

Algorithm A.2 Probability Update

Input: initial PV, population, parent population.
Output: updated PV.
1: **for** each $s_i \in$ population **do**
2: $n \leftarrow$ number of solutions containing s_i in the parent population;
3: $pv_i' = (\lambda * (4n/N_c)) + (1 - \lambda) * pv_i$;
4: **end for**
5: $PV \leftarrow PV'$;

Guided mutation The guided mutation operator combines the global statistical information in PV with the generated solutions to generate new solutions. Algorithm A.3 explains the process of generating a new solution, wherein $\beta \in [0, 1]$ and d_t is the new solution generated, whose services are either sam pled from the model given by PV or retained as those of the best solution in the parent. This process is controlled by a parameter β, which specifies the contributions of PV and the parent population. As the value of β decreases, the parent population's contribution to the new solution increases whereas that of PV decreases, and vice versa. The generated solution d_t may not be valid. Hence, it needs to be checked for feasibility and, if required, a repair operator is applied to it.

Algorithm A.3 Guided Mutation

Input: population, PV, and best solution.
Output: Generated solution d_t.
initialize β ;
for each service s_i in population **do**
  generate a random value $r \in [0, 1]$;
  if $r < \beta$ **then**
    if p_{v_i} < threshold_probability **then**
      $d_t \leftarrow d_t \cup s_i$;
    else if $s_i \in$best solution **then**
      $d_t \leftarrow d_t \cup s_i$;
    end if
  end if
end for

Repair operator The repair operator is applied only on infeasible solutions produced through GM operator. In this repair operator, all the solutions from $\Re$ that are not present in the χ list are selected, and the $\Re$ list is reinitialized with those selected solutions. A solution from $\Re$ is selected randomly, and its nearest neighborhood is calculated. The repair operator selects a solution from this set and adds it to the infeasible solution set. Then, the selected service is deleted from $\Re$. This process is repeated until the set becomes feasible, i.e., until $\Re$ becomes null. The selection of a solution is based on a probability threshold value or is random. This helps to maintain the diversity of the population. The pseudocode for the repair operator is shown in Algorithm A.4.

Algorithm A.4 Repair Operator

Input: $\Re$, population, $Prob$.
Output: update d_t.
while $\Re \neq \phi$ **do** ▷ check for feasible solution in $\Re$
  $v \leftarrow$ random service from population;
  if v not in $\Re$ **then**
    if $V\_$ probability $<$ threshold_ probability **then**
      $d_t \leftarrow V$;
      delete V from population;
    end if
  end if
end while
return $\Re$; ▷ $\Re$ is an infeasible solution on which repair operator is applied

The pseudocode for the EA/G algorithm is presented in Algorithm A.5.

Algorithm A.5 Evolutionary Algorithm with Guided mutation

Input: population, P.
Output: best solution.
$iter \leftarrow 0$;
do
  $iter \leftarrow iter + 1$;
  Generate N_c initial solutions for initial population (g), is randomly generated;
  Compute initial probabilities for all the services in generation;
  Generate the parent population;
  Update P based on parent population by using algorithm A.2;
  do
    Apply Guided Mutation by using algorithm A.3;
  while (services $\neq$ end)
  Verify feasibility of the generated composition;
  if composition is not feasible **then**
    Apply repair operator on obtained solution using algorithm A.4;
  end if
while $iter \leq N$
Reinitialize generation g;

A.3 MCDM Techniques

A.3.1 Data Envelopment Analysis (DEA)

DEA is a multi-criteria decision-making model based on linear programming methodology. The "envelopment" in DEA stems from the way observations are enveloped in order to identify the "Pareto frontier" that is used to evaluate the

relative performance of all peer entities [212]. It was introduced as a "mathematical programming model applied to observational data that provides a new way of obtaining empirical estimates of relation" [213].

In the DEA model, decision-making units (DMUs) represent the operations or processes that convert multiple inputs to multiple outputs. The model maximizes the efficiency of a DMU subject to constraints relative to the best DMUs. The efficiency of all DMUs is, then, less than or equal to 1. Through performance evaluation, the model determines the "efficiency frontier" that represents the relative best practices. Inefficient strategies can be improved with the suggested directions [214]. The basic assumption is that if a DMU is inefficient, then a virtual DMU obtained via a combination of other efficient DMUs can either result in greater output for the same number of inputs or use fewer inputs to produce the same number of outputs, or some combination of both. There are two approaches to determining an efficiency frontier: input-oriented and output-oriented [214].

Input-oriented DEA. The DEA model is input-oriented when inputs are minimized while keeping outputs fixed at their current values. The linear programming formulation of the input-oriented DEA model is given as follows. Suppose there are N DMUs, of which DMUo is the one under evaluation. Let x_{ij} represent the value of the j^{th} input (of P inputs) used by DMUi and let y_{ik} denote the value of the k^{th} output (of Q outputs) produced by DMUi. The efficiency score of DMUo is given by

$$\theta^* = \min_{\theta, \mathbf{w}} \ \theta \tag{A.7}$$

subject to

$$\sum_{j=1}^{N} w_j x_{ij} \leq \ \theta x_{io} \qquad i = 1, \ldots, P \tag{A.8}$$

$$\sum_{j=1}^{N} w_j y_{kj} \geq \ y_{ko} \qquad k = 1, \ldots, Q \tag{A.9}$$

$$w_j \geq \ 0 \qquad j = 1, \ldots, N \tag{A.10}$$

where $\mathbf{w}$ is a weight vector (w_j represents the contribution of the j^{th} DMU to the virtual DMU). If $\theta^* = 1$, then the current input level cannot be reduced,

indicating that DMUo is on the "efficient frontier." Otherwise, if $\theta^* < 1$, then DMUo is inefficient, and there is room to increase efficiency by reducing input while maintaining output at the same level.

Output-oriented DEA. In the output-oriented DEA model, outputs are maximized while input levels are kept constant. The corresponding linear program is the same as in the input-oriented DEA, equations A.7–A.10, with the modifications that in equation A.7, maximization is substituted for minimization, and that the inefficiency score θ', the parameter corresponding to θ, appears in the constraint on outputs as a multiplier for y_{ko} and is no longer present in the constraint for inputs, yielding the following modified constraints in place of equations A.8 and A.9:

$$\sum_{j=1}^{N} w'_j x_{ij} \leq \quad x_{io} \qquad i = 1, \ldots, P \tag{A.11}$$

$$\sum_{j=1}^{N} w'_j y_{kj} \geq \quad \theta' y_{ko} \qquad k = 1, \ldots, Q \tag{A.12}$$

The two formulations are equivalent with the position

$$\theta = 1/\theta' \qquad w = w'/\theta' \tag{A.13}$$

There is also a possibility of improving a DMU that is located at the same level as another DMU on the efficient frontier. This can be done by varying the proportions with which inputs are utilized. Defining the (nonnegative) input excesses and output shortfalls as

$$s_i^- = \theta^* x_{io} - \sum_{j=1}^{N} w_j x_{ij} \qquad i = 1, \ldots, P \qquad \text{(input excess)} \tag{A.14}$$

$$s_k^+ = \sum_{j=1}^{N} w_j y_{kj} - y_{ko} \qquad k = 1, \ldots, Q \qquad \text{(output shortfall)} \tag{A.15}$$

and using them as slack variables in the following linear program:

$$\max \omega = \sum_{i=1}^{P} s_i^- + \sum_{k=1}^{Q} s_k^+ \tag{A.16}$$

subject to

$$s_i^- = \theta^* x_{io} - \sum_{j=1}^{N} w_j x_{ij} \qquad i = 1, \ldots, P \tag{A.17}$$

$$s_k^+ = \sum_{j=1}^{N} w_j y_{kj} - y_{ko} \qquad k = 1, \ldots, Q \tag{A.18}$$

$$w_j \geq 0,\ s_j^- \geq 0,\ s_j^+ \geq 0 \tag{A.19}$$

the optimal slacks (max-slack solution) can be computed. Now the notions of weak and strong efficiency can be given. A DMU for which $\theta^* = 1$ is said to be weakly efficient. A DMU is strongly efficient (Pareto–Koopmans efficient) if it is weakly efficient and all the optimal slacks are zero. With a strongly efficient DMU, any improvement of an input or output will produce a worsening of some other input or output. This is important from a service provider point of view, because for a given set of user-specific performance attributes, it is necessary not only to determine which one is performing better, but also to know the specific areas where efficiency is lacking and improvement is needed.

A.3.2 Super-efficiency Data Envelopment Analysis (SDEA)

Since the early 1980s, DEA has been used as an alternative method of classification for evaluating the relative efficiency of independent homogeneous units that use the same inputs to produce the same outputs [263]. However, DEA inconveniently leaves room for ties between units that have a relative efficiency equal to 100%. SDEA is used to rank the performance of efficient DMUs [209]. DEA evaluates the efficiency of a unit relative to a reference set comprising all units (including itself), whereas SDEA excludes each unit from its own reference set. It is, therefore, possible to obtain efficiency scores that exceed 1.

Input-oriented SDEA. The linear program for the input-oriented model used for Super-efficiency DEA is the same as the one in equations A.7–A.10, with the difference that DMUo is excluded from the summations in equations A.8 and

A.9, which become

$$\sum_{j=1,j\neq o}^{N} w_j x_{ij} \leq \theta x_{io} \qquad i = 1, \ldots, P \tag{A.20}$$

$$\sum_{j=1,j\neq o}^{N} w_j y_{ij} \geq y_{ko} \qquad k = 1, \ldots, Q \tag{A.21}$$

This will allow a super-efficiency score greater than 1, enabling distinguishability between efficient observations. Super-efficiency measures can be calculated for both inefficient and efficient observations, but in the case of inefficient observations, the values of the efficiency measure do not change, whereas efficient observations may attain higher values.

Output-oriented SDEA. The output-oriented version of SDEA is the same as the corresponding DEA version, with the same modifications as above, i.e., DMUo is excluded from the summations in the constraints, i.e., equations A.11 and A.12.

A.3.3 Analytical Hierarchical Process (AHP)

AHP is a multi-criteria decision making method to allow a decision makers to compute a ratio scale from preferences and model a complex problem in a hierarchical structure. This structure consists of goal, criteria (factors), sub-criteria (sub-factors), and alternatives [244]. In AHP, criteria are evaluated at top level and alternatives are evaluated at bottom level for each criteria. Each level and sub-level is evaluated separately by the decision makers. The decision makers should determine the weights of all criteria in order to do pairwise comparison among them. It helps to the decision makers to incorporate a group agreement using questionnaire for comparing each element and geometric mean to form a final solution [264]. The procedure of AHP is as follows [215, 265]:
(i) Model the problem structure by breaking down the decision problem (goal, criteria, sub-criteria, and alternatives)
(ii) Establish the priority of input data, by pairwise comparison matrix on each

element of the hierarchical structure.
(iii) Develop the pairwise comparison matrix at each level of hierarchy (The pairwise comparison determined according to the scale from 1 to 9).
(iv) Calculate vector of weights by using eigenvector procedure.
(v) Calculate the consistency ratio (CR) to check the consistency of the judgment. If CR < 0.1 then the pairwise comparison is consistent and acceptable.
(vi) Aggregate of the relative weights of decision elements and form a set of preferences for alternatives.

The consistency ratio (CR) of the pairwise comparison matrix A is defined as follows:

$$CR = \frac{CI}{RI} \tag{A.22}$$

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{A.23}$$

where CI is the consistency index, n is the order of the pairwise comparison matrix A and λ_{max} is its maximum eigenvalue, while the random index RI is the average CI value for random matrices.

A.3.4 Analytical Network Process (ANP)

Analytic Network Process (ANP) [216, 266] aims to surmount the limitations of AHP. AHP does not allow the elements of hierarchical model to have dependence and feedback between each criteria and alternatives. ANP allows interdependence interaction between involved elements that can be criteria and alternatives. ANP is not limited by the independent assumptions between the criteria and the alternatives of the decisions, or simply among the criteria or among the alternatives themselves. However, the importance of criteria and alternatives are determined. ANP shows the interdependencies with feedback in the structure which has two-way arrows and connected cycles of its clusters like a network instead of level hierarchy in AHP [267]. This approach has the potential to handling interdependent relationships among the criteria weights by procure composite weights through the development of a “supermatrix". The super matrix concept is similar to the Markov chain process [216] where respective

prime weights are forming a supermatrix from their eigenvectors. The supermatrix consists of control hierarchy and network hierarchy. The control hierarchy is an interaction between the criteria and sub-criteria and it also includes the goal of the problem, criteria and sub-criteria, each criterion and sub-criteria are independent at the same level. The weight of criteria is evaluated with AHP in the control hierarchy. The network hierarchy indicates the relationship between elements and clusters, and criteria interaction [267].

A.3.5 Fuzzy Sets and Fuzzy Numbers

The concept of fuzzy theory was developed by Zadeh [268] and is used for decision making problems based on vague and ambiguous information. Linguistic variables are qualitatively expressed by linguistic terms and quantitatively expressed by fuzzy sets and membership functions [269]. Denoting with U a nonempty set, a fuzzy set $\tilde{A}$ on U is a pair (U, μ_A) where$\mu_A(x) : U \to [0, 1]$ is the membership function of $\tilde{A}$. A crisp set can be viewed as a special case of a fuzzy set, where the membership function is restricted to the extremes $\{0, 1\}$ of the interval $[0, 1]$. In other words, if $\mu_A(x) = 0$ then $x \notin \tilde{A}$ with certainty. On the contrary, if $\mu_A(x) = 1$ then $x \in \tilde{A}$ with certainty. Otherwise, the membership value $\mu_A(x)$ basically indicates the degree to which x belongs to $\tilde{A}$. The fuzzy set $\tilde{A}$ is said to be convex if $\mu_A(\alpha x + (1-\alpha)y) \geq \mu_A(x) \wedge \mu_A(y)$ for all $x, y \in U$ with $\alpha \in [0, 1]$. The height $\mathrm{hgt}(\tilde{A})$ of $\tilde{A}$ is defined as the supremum of the image of U under μ_A, i.e., $\mathrm{hgt}(\tilde{A}) = \sup_{x \in U} \mu_A(x)$. A fuzzy set is normal if its height is 1.

A fuzzy real number is a fuzzy set $\tilde{A}$ on $\mathbb{R}$ which is both convex and normal and has a piecewise continuous membership function. A triangular fuzzy number $\tilde{q}$ is a special case of trapezoidal fuzzy number and it is commonly used in many real time decision making problems. It is associated to a triplet $\langle l_q, m_q, u_q \rangle$ of real numbers such that $l_q < m_q < u_q$ and its membership function is:

$$\mu_q(x) = \begin{cases} (x - l_q)/(m_q - l_q) & \text{if } l_q \leq x \leq m_q \\ (u_q - x)/(u_q - m_q) & \text{if } m_q \leq x \leq u_q \\ 0 & \text{otherwise} \end{cases} \tag{A.24}$$

The basic arithmetic operations on triangular fuzzy numbers were studied in [270]. If $\tilde{a} = \langle l_a, m_a, u_a \rangle$ and $\tilde{b} = \langle l_b, m_b, u_b \rangle$, the extensions of basic operations to triangular fuzzy numbers are

$$\begin{aligned} \tilde{a} + \tilde{b} &= \langle l_a + l_b, m_a + m_b, u_a + u_b \rangle \\ \tilde{a} \times \tilde{b} &= \langle l_a \times l_b, m_a \times m_b, u_a \times u_b \rangle \\ -\tilde{a} &= \langle -u_a, -m_a, -l_a \rangle \\ 1/\tilde{a} &= \langle 1/u_a, 1/m_a, 1/l_a \rangle \end{aligned} \tag{A.25}$$

A method for comparing fuzzy numbers is as follows: the degree of possibility that $\tilde{a} \geq \tilde{b}$ is

$$v(\tilde{a} \geq \tilde{b}) = \sup_{x \geq y} \{\min(\mu_a(x), \mu_b(y))\} \tag{A.26}$$

and, to compare $\tilde{a}$ and $\tilde{b}$, both $v(\tilde{a} \geq \tilde{b})$ and $v(\tilde{b} \geq \tilde{a})$ need be evaluated. For triangular fuzzy numbers, it holds that

$$v(\tilde{a} \geq \tilde{b}) = \begin{cases} 0 & \text{if } u_a \leq l_b \\ 1 & \text{if } m_a \geq m_b \\ \dfrac{u_a - l_b}{(m_b - m_a) + (u_a - l_b)} & \text{if } u_a > l_b \text{ and } m_a < m_b \end{cases} \tag{A.27}$$

with the last case being the ordinate of the highest intersection point between $\mu_a(x)$ and $\mu_b(x)$.

A.3.6 Grey Theory

Grey theory was proposed by Deng [271, 272] to cope with uncertainty in problems with small samples and incomplete information. Its emphasis is on constructing models strarting from small amounts of observed data. A grey number is an indeterminate number that takes value within an interval. It is denoted as $\otimes x \in [\underline{x}, \overline{x}]$, where $\underline{x}$ is a lower limit real number, and $\overline{x}$ is an upper limit real number for the grey number $\otimes x$. If both its limits are unknown, the number is called a black number (nothing is known). If both upper and lower limits are equal, then it is called a white number (complete information is available).

The basic operations for the grey numbers $\otimes x \in [\underline{x}, \overline{x}]$ and $\otimes y \in [\underline{y}, \overline{y}]$ are defined in [273]:

$$\begin{aligned}
(\oplus x + \oplus y) &\in [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
(- \oplus x) &\in [-\overline{x}, -\underline{x}] \\
(\oplus x \times \oplus y) &\in [\min\{\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y},\ \underline{x}\overline{y},\ \overline{x}\underline{y},\ \overline{x}\overline{y}\}] \\
(1/ \oplus x) &\in [1/\overline{x}, 1/\underline{x}]
\end{aligned} \tag{A.28}$$

The multiplication of a grey number by a scalar is done in accordance with

$$(h \oplus x) \in [h\underline{x}, h\overline{x}] \tag{A.29}$$

where h is a positive real number.

A.3.7 TOPSIS

Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), introduced by Hwang and Yoon [232], is a MCDM method that sorts the alternatives for a decision problem in order of distance from the positive and negative ideal solutions. TOPSIS is an efficient method due to its simplicity and ability to consider an unlimited number of alternatives and criteria in the process of making decisions. The best solution is the one which has a minimum distance from the positive ideal solution and maximum distance from the negative ideal solution [274]. The positive ideal solution is the one which maximizes the benefits and minimizes the costs while the negative ideal solution is the one which minimizes the benefits and maximizes the costs [275]. The steps of the TOPSIS method are described as follows [232]:

Step 1: Construct a Decision Matrix for the alternatives based on the criteria with alternatives as row indices and criteria as column indices. So element x_{ij} refers to the i-th alternative and j-th criterion:

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \dots & C_n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \dots & & & & \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{pmatrix} \end{array} \tag{A.30}$$

where $\{A_1, \ldots, A_m\}$ are the alternatives and $\{C_1, \ldots, C_n\}$ are the criteria.

Step 2: Normalize the decision matrix D using the following formula:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{A.31}$$

Step 3: Ascertain the weighted normalized decision matrix by multiplying weights of criteria w_j for $j = 1, \ldots\ldots, n$ with the associated columns. The weighted normalized value v_{ij} is calculated as follows:

$$v_{ij} = w_j \times r_{ij} \tag{A.32}$$

The weights w_j are calculated by using Analytic hierarchy Process (AHP) which is explained in section 5 of this paper.

Step 4: Determine the positive ideal solution (PIS) $A^* = \{v_1^*, v_2^*, \ldots, v_n^*\}$ and the negative ideal solution (NIS) $A^- = \{v_1^-, v_2^-, \ldots, v_n^-\}$, i.e., fictitious alternatives having the best and worst performance from each criterion, where:

$$v_j^* = \begin{cases} \max_i \ v_{ij} & \text{if } j \in J \\ \min_i \ v_{ij} & \text{if } j \in J' \end{cases} \tag{A.33}$$

$$v_j^- = \begin{cases} \min_i \ v_{ij} & \text{if } j \in J \\ \max_i \ v_{ij} & \text{if } j \in J' \end{cases} \tag{A.34}$$

and J is the set of benefit criteria (the larger the better), while J' is the set of cost criteria (the smaller the better).

Step 5: Calculate for each alternative the separation from the PIS and the NIS, according to equations (A.33) and (A.34):

$$S_i^* = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^*)^2} \tag{A.35}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2} \tag{A.36}$$

where $i = 1, 2, 3, \ldots, m$

Step 6: Determine for each alternative, the ratio C_i^* which is the relative closeness coefficient to the ideal solution as shown in equation (A.37):

$$C_i^* = \frac{S_i^-}{S_i^* + S_i^-} \tag{A.37}$$

Larger value of C_i^* indicates better performance of the alternatives.

Step 7: Rank the alternatives based on the values of C_i^*.

A.3.8 Fuzzy TOPSIS

It is often tough for a decision-maker to assign a precise performance rating to an alternative for the attributes under consideration. The merit of using a fuzzy approach is to allot the relative importance of attributes using fuzzy numbers instead of crisp numbers [218, 219]. Decision makers use linguistic variables to appraise alternative and the criteria weights. The values which are assigned are assumed to be triangular fuzzy numbers, and the operations defined on them are as listed in Sec. A.3.5. Thus, the triangular fuzzy number $\tilde{w}_j$ denotes the weight of the j-th criterion, and $\tilde{x}_{ij}$ is the rating of the i-th alternative with respect to the j-th criterion. The procedure of fuzzy TOPSIS is similar to the one described in Sec. A.3.7. Details are as follows [219, 223]:

Step 1: Normalize the fuzzy decision matrix $\tilde{X} = [\tilde{x}_{ij}]_{n\times m}$. Following [219], a simpler, linear scale transformation is used in place of the one in (A.31). The normalized fuzzy decision matrix is denoted by $R = [\tilde{r}_{ij}]_{n\times m}$ and its elements are given by

$$\tilde{r}_{ij} = \begin{cases} \left\langle \dfrac{l_{ij}}{u_j^+}, \dfrac{m_{ij}}{u_j^+}, \dfrac{u_{ij}}{u_j^+} \right\rangle & j \in J \\ \left\langle \dfrac{l_j^-}{l_{ij}}, \dfrac{l_j^-}{m_{ij}}, \dfrac{l_j^-}{u_{ij}} \right\rangle & j \in J' \end{cases} \tag{A.38}$$

where $l_j^- = \min\limits_i \; l_{ij}$ and $u_j^+ = \max\limits_i \; u_{ij}$.

Step 2: Construct the weighted normalized fuzzy decision matrix $\tilde{V} = [\tilde{v}_{ij}]_{n\times m}$ by multiplying the evaluation criteria weights and the elements of normalized fuzzy decision matrix:

$$\tilde{v}_{ij} = \tilde{r}_{ij} \times \tilde{w}_j \tag{A.39}$$

Step 3: To isolate the Fuzzy Positive Ideal Solution (FPIS) $\tilde{A}^*$ and the Fuzzy Negative Ideal Solution (FNIS) $\tilde{A}^-$, proceed as in [276]:

$$\tilde{A}^* = \begin{cases} \tilde{v}^* & j \in J \\ \tilde{v}^- & j \in J' \end{cases} \qquad \tilde{A}^- = \begin{cases} \tilde{v}^- & j \in J \\ \tilde{v}^* & j \in J' \end{cases} \tag{A.40}$$

where $\tilde{v}^* = \langle 1, 1, 1\rangle$ and $\tilde{v}^- = \langle 0, 0, 0\rangle$.

Step 4: The separation measures S_i^* and S_i^- of each alternative from $\tilde{A}^*$ and $\tilde{A}^-$ are computed according to the following equations:

$$S_i^* = \sum_{j=1}^{n} d(\tilde{v}_{ij}, \tilde{v}_j^*) \tag{A.41}$$

$$S_i^- = \sum_{j=1}^{n} d(\tilde{v}_{ij}, \tilde{v}_j^-) \tag{A.42}$$

where the distance $d(\tilde{x}, \tilde{y})$ between two triangular fuzzy numbers $\tilde{x} = \langle l_x, m_x, u_x\rangle$ and $\tilde{y} = \langle l_y, m_y, u_y\rangle$ is given by the vertex method as in [219]

$$d(\tilde{x}, \tilde{y}) = \sqrt{\frac{1}{3}\left((l_x - l_y)^2 + (m_x - m_y)^2 + (u_x - u_y)^2\right)} \tag{A.43}$$

noting that the distance function in (A.43) reduces to ordinary distance when applied to crisp numbers.

Step 5: Calculate the closeness coefficient (CC_i^*).

$$CC_i^* = \frac{S_i^-}{S_i^* + S_i^-} \tag{A.44}$$

Step 6: Rank the alternatives in decreasing order of CC_i^*.

A.3.9 Fuzzy AHP

A fuzzy extension to AHP was first proposed in [277]. Later, Chang presented a simpler approach to fuzzy AHP based on extent analysis [222], which is widely

used in MCDM problems. In this method, linguistic variables are used by decision makers to convey comparative judgments. Judgments from multiple decision makers are combined to obtain the fuzzy reciprocal decision matrices. In the following, a description is given of the first steps of Fuzzy AHP based on extent analysis [222], namely the part where the weight vector is determined. The sequencing of the reciprocal judgments was done in [222] with a similar procedure, which is omitted here for clarity, as in this work recourse is only made to Fuzzy AHP for the determination of weights, and TOPSIS and its fuzzy variant are used for subsequent processing.

Step 1: Populate the fuzzy decision matrix $\tilde{Y} = [\tilde{y}_{ij}]_{m\times m}$ describing the relative importance of each criterion with respect to the others. Thus, $\tilde{y}_{ij}$ indicates how much is criterion C_i important as compared with criterion j and $\tilde{y}_{ji} = 1/\tilde{y}_{ij}$.

Step 2: Calculate the fuzzy synthetic extent value with respect to the i-th object, defined in equation (A.45).

$$\tilde{s}_i = \frac{\sum_{j=1}^{m} \tilde{y}_{ij}}{\sum_{i,j} \tilde{y}_{ij}} \tag{A.45}$$

where the operations on triangular fuzzy number are to be carried out as detailed in equation (A.25)

Step 3: For each synthetic extent value $\tilde{s}_i$ $(i = 1, 2, \ldots, m)$, calculate the degree of possibility $g(i)$ that $\tilde{s}_i$ it is greater than all other synthetic extent values $\tilde{s}_j$, $j \neq i$ with the method in (A.26)

$$\begin{aligned} g(i) &= \\ v(\tilde{s}_i \geq \tilde{s}_1, \ldots, \tilde{s}_{i-1}, \tilde{s}_{i+1}, \ldots, \tilde{s}_m) &= \\ v\left((\tilde{s}_i \geq \tilde{s}_1) \text{ and } \ldots (\tilde{s}_i \geq \tilde{s}_{i-1}) \text{ and } (\tilde{s}_i \geq \tilde{s}_{i+1}) \text{ and } \ldots (\tilde{s}_i \geq \tilde{s}_m)\right) &= \\ \min_{j\neq i} v(\tilde{s}_i \geq \tilde{s}_j) & \end{aligned} \tag{A.46}$$

Step 4: Arrange the results (note that these are crisp numbers) into a vector and normalize it to obtain the weight vector W

$$W = \frac{1}{\sum_{i=1}^{m} g(i)}(g(1), g(2), \ldots, g(n))^T \tag{A.47}$$

A.3.10 Grey TOPSIS

Grey TOPSIS is a combination of Grey theory and TOPSIS method. The procedural steps of Grey TOPSIS for calculating the criteria weights are demonstrated as follows [220, 221]:

Step 1: Determine the grey decision matrix D.

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \ldots & C_n \end{array} \\ \begin{pmatrix} \otimes x_{11} & \otimes x_{12} & \otimes x_{13} & \ldots & \otimes x_{1n} \\ \otimes x_{21} & \otimes x_{22} & \otimes x_{23} & \ldots & \otimes x_{2n} \\ \ldots & & & & \\ \otimes x_{m1} & \otimes x_{m2} & \otimes x_{m3} & \ldots & \otimes x_{mn} \end{pmatrix} \end{array} \tag{A.48}$$

where $\otimes x_{ij}$ denotes the evaluation of grey numbers of the i-th alternative with respect to the i-th criteria.

Step 2: Calculate the criteria weights w_j using Table 4.4.

Step 3: Normalize the grey decision matrix according to equation (A.49) [278].

$$\otimes r_{ij} = \begin{cases} \dfrac{\otimes x_{ij}}{\max\limits_i \overline{x}_{ij}} & \text{for the benefit criteria} \\ 1 - \dfrac{\otimes x_{ij}}{\min\limits_i \underline{x}_{ij}} & \text{for the cost criteria} \end{cases} \tag{A.49}$$

Step 4: Identify the positive and negative ideal alternatives. The positive ideal alternative A^* and negative ideal alternative A^- are defined as follows.

$$A^* = \{r_1^*, \ldots, r_m^*\} \text{ and } A^- = \{r_1^-, \ldots, r_m^-\} \tag{A.50}$$

where

$$r_j^* = \begin{cases} \max\limits_i \overline{r}_{ij} & \text{for the benefit criteria} \\ \min\limits_i \underline{r}_{ij} & \text{for the cost criteria} \end{cases} \tag{A.51}$$

and

$$r_j^- = \begin{cases} \min\limits_i \underline{r}_{ij} & \text{for the benefit criteria} \\ \max\limits_i \overline{r}_{ij} & \text{for the cost criteria} \end{cases} \tag{A.52}$$

Step 5: Determine the separation measure of positive (d^*) and negative ideal (d^-) alternatives according to equations (A.53) and (A.54) [221]

$$d_i^* = \sqrt{\frac{1}{2}\sum_{j=1}^{n} w_i \left((r_j^* - \underline{r}_{ij})^2 + (r_j^* - \overline{r}_{ij})^2\right)} \tag{A.53}$$

$$d_i^- = \sqrt{\frac{1}{2}\sum_{j=1}^{n} w_i \left((r_j^- - \underline{r}_{ij})^2 + (r_j^- - \overline{r}_{ij})^2\right)} \tag{A.54}$$

Step 6: Calculate the relative closeness (C_i^*) to the positive ideal alternatives.

$$C_i^* = \frac{d_i^-}{d_i^* + d_i^-} \tag{A.55}$$

The larger indexed value is considered as the better alternative.

Appendix B

Annexure

Table B.1: Fact sheet for publication [1]

Title	Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review
Authors	Chandrashekar Jatoth, G. R. Gangadharan, and Rajkumar Buyya
Publication	IEEE Transactions on Services Computing, Vol. 10, No. 3, pages 475 - 492, 2017
ISBN/ISSN	1939-1374
DOI	10.1109/TSC.2015.2473840
Status	Published
Publisher	IEEE
Publication Type	Journal

Figure B.1: Publication [1]

Table B.2: Fact sheet for publication [2]

Title	Evaluating the efficiency of cloud services using modified data envelopment analysis and modified super-efficiency data envelopment analysis
Authors	Chandrashekar Jatoth, G. R. Gangadharan, and Ugo Fiore
Publication	Soft Computing, Vol. 21, No.23, pages 7221-7234, 2017
ISBN/ISSN	1432-7643
DOI	https://doi.org/10.1007/s00500-016-2267-y
Status	Published
Publisher	Springer
Publication Type	Journal

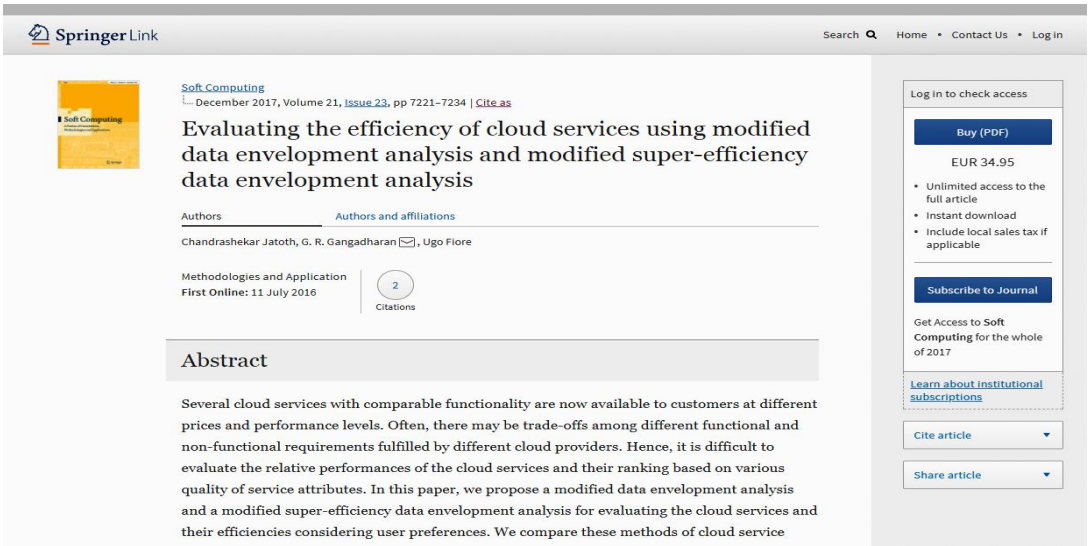


Figure B.2: Publication [2]

Table B.3: Fact sheet for publication [3]

Title	QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation
Authors	Chandrashekar Jatoth, G.R. Gangadharan, Ugo Fiore, and Rajkumar Buyya
Publication	Future Generation Computer Systems, 2017
ISBN/ISSN	0167-739X
DOI	https://doi.org/10.1016/j.future.2017.07.042
Status	Published (Available online)
Publisher	Elsevier
Publication Type	Journal

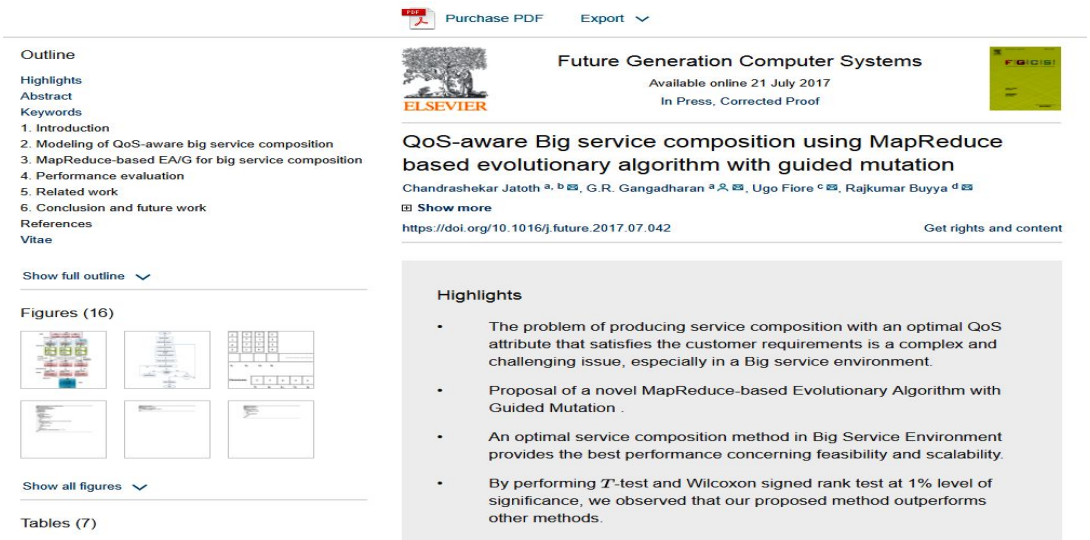


Figure B.3: Publication [3]

Table B.4: Fact sheet for publication [4]

Title	Fitness Metrics for QoS-aware Web Service Composition Using Metaheuristics
Authors	Chandrashekar Jatoth, and G. R. Gangadharan
Publication	Proceedings of the 7^{th} KES International Conference on Intelligent Decision Technologies KES-IDT'2015, pages 267-277, 2015
ISBN/ISSN	978-3-319-19856-9
DOI	https://doi.org/10.1007/978-3-319-19857-6-24
Status	Published
Publisher	Springer International Publishing
Publication Type	Conference

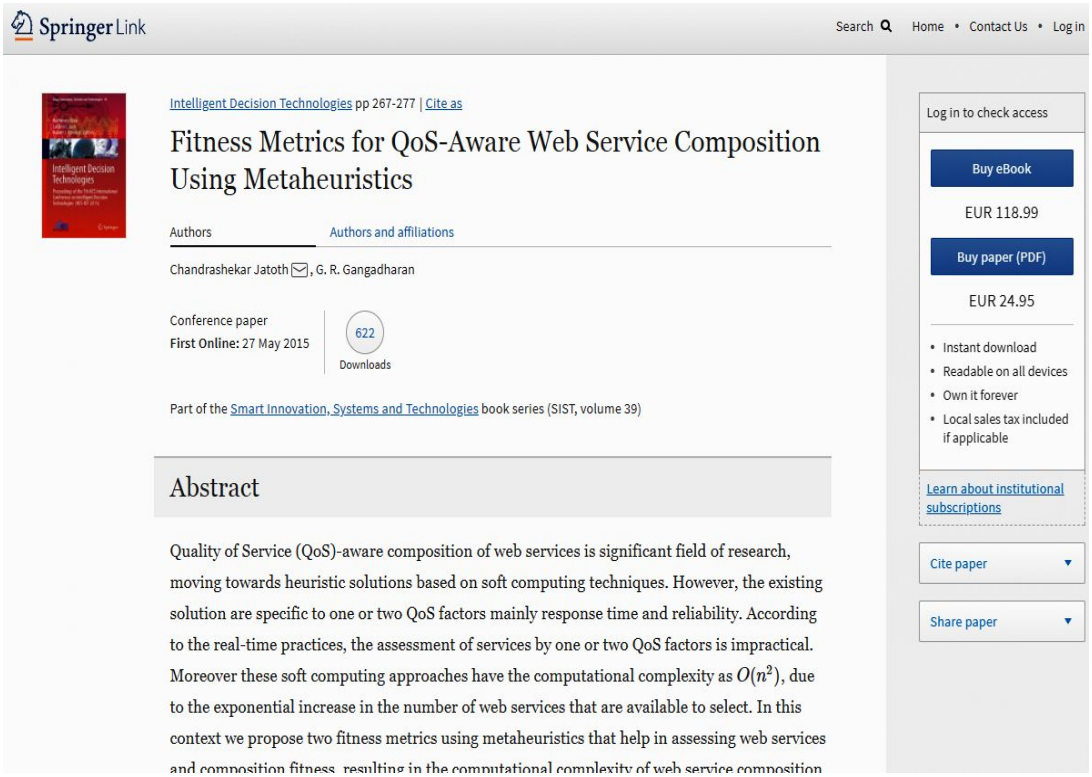


Figure B.4: Publication [4]

Table B.5: Fact sheet for publication [5]

Title	QoS-Aware Web Service Composition Using Quantum Inspired Particle Swarm Optimization
Authors	Chandrashekar Jatoth, and G. R. Gangadharan
Publication	Proceedings of the 7^{th} KES International Conference on Intelligent Decision Technologies, KES-IDT'2015, pages 267-277, 2015
ISBN/ISSN	978-3-319-19856-9
DOI	https://doi.org/10.1007/978-3-319-19857-6-24
Status	Published
Publisher	Springer International Publishing
Publication Type	Conference

Figure B.5: Publication [5]