Sudan University of Science and Technology (SUST)
College of Computer Science & Information Technology
Post Graduate Studies

Designing a Model for the Service Selection Aware-Quality of Service in the Internet of Things Environment

تصميم نموذج لأختيار الخدمات مع مراعاة جودة الخدمة في بيئة إنترنت الأشياء

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

By

Aghabi Nabil Abosaif Mitas

Supervised by

Dr. Salah Elfaki Elrofai Elfaki

Khartoum, 2020

(إنّ إله السّماء يعطينا النّجاح، ونحن عبيده نقوم ونبني)

(سفر نحميا 2 : 20)

Dedication

To my role model in life, my mother, without her words and supports, I couldn't succeed in my educational stages.
To my father, who always guides me and light my way.
To my companion and the reason for my success is my husband.
To my uncle Father Arsanious Abusafe, who kindly reassures my heart all the time.
To the reason for my happiness and the pursuit of success are my daughters and my son.
To the soul of my son Malak.
To all my wonderful brothers and sisters.
Lastly, everyone helped, supported, and advised me on this journey.

Acknowledgments

I want to express my sincere gratitude to my supervisor, Dr. Salah Elfaki Elrofai Elfaki, for my Ph.D. study's continuous support. His guidance helped me in all the time of research and writing of this thesis. I would also like to thank the first supervisor of this research, Prof. Haitham S Hamza, to find this research area and support me in the first two years.

Besides my advisors, I would like to thank Prof. Izzeldin Osman for his insightful comments and encouragement. My sincere thanks also go to SUST University and all Ph.D. program staff, which provided me with an opportunity to study and gave access to the laboratory and research facilities. Without their precious support, it would not be possible to conduct this research.

I thank all my colleagues, particularly Dr. Mohamed Yousif and Dr. Mutasim Adam, for their continuous support and discussion.

My most sincere thanks go to my family for always believing I could succeed.

Abstract

The Internet of Things (IoT) has been evolved over the last decade to provide machine-to-machine (M2M) interactions. There are many services in the IoT environment with similar functional but with different non-functional properties or Quality of Service requirements. Thus, the problem of selecting and combining the optimal service that matches the end-user constraints is challenging. Therefore, this research firstly reviews and analyzes the state-of-the-art algorithms for service selection process under QoS constraints in an IoT environment. That review aims to provide the research community with guidance and knowledge to identify the trends in the services selection problem. Secondly, the research proposed a model to select the optimal service that meets the end-user preferences. The proposed solution gradates into three models, where the last model includes the completed proposed solution for solving the selection problem.

The first introduced model used a friendly Likert type measurement method with an Improved Practical Swarm Optimization (Improved-PSO) algorithm to enhance the end-user requirements. The second model called RI-SSO, which implemented a Likert type measurement method to evaluate the reputation of the services from the end-users. It aims to improve bio-inspired optimizing algorithms, called a Social Spider Optimization (SSO) algorithm, by adding a reputation value to member's weight. The third model aims to classify the search space into sub-clusters. So when the end-user requires a service, the searching process is done in the appropriate cluster. For applying search space clustering, the model used a Fuzzy Logic System (FLS).

Lastly, the comparative study and experimental results were done to evaluate the proposed algorithm's performance with other related service selection models. The feasible efficiency has been obtained as output results from the proposed models in both condition maximization and minimization optimization states. They achieved better performance in terms of fitness values and execution time.

المستخلص

لقد تم تطوير إنترنت الأشياء (IoT) على مدى العقد الماضي لتوفير التفاعلات آلية (M2M). فهناك العديد من الخدمات في بيئة إنترنت الأشياء التى تتشابه فى خصائصها الوظيفة ولكن تختلف فى خصائصها الغير وظيفية أو تختلف فى متطلبات جودة الخدمة. وبالتالي ، فإن مشكلة اختيار ودمج الخدمة الأمثل التي تتوافق مع قيود المستخدم النهائي تمثل تحديًا.

لذلك ، يقوم هذا البحث أولاً بمراجعة وتحليل أحدث الخوارزميات لعملية اختيار الخدمة في ظل قيود جودة الخدمة في بيئة إنترنت الأشياء. حيث تهدف هذه المراجعة إلى تزويد مجتمع البحث بالإرشادات والمعرفة اللازمة لتحديد الاتجاهات المستقبلية في مشكلة اختيار الخدمة.

ثانيًا ، اقترح البحث نموذجًا لاختيار الخدمة الأمثل التي تلبي تفضيلات المستخدم النهائي. يتدرج الحل المقترح إلى ثلاثة نماذج ، حيث يتضمن النموذج الأخير الحل المقترح المكتمل لحل مشكلة الاختيار.

استخدم النموذج الأول طريقة قياس سهلة تسمى Likert type measurement مع خوارزمية Improved Practical Swarm Optimization (Improved-PSO) لتحسين متطلبات المستخدم النهائي. النموذج الثاني يسمى RI-SSO ، والذي طبق طريقة قياس نوع Likert لتقييم سمعة الخدمات من المستخدمين النهائيين. فهو يهدف إلى تحسين خوارزمية التحسين المستوحاة من الحيوية، تسمى خوارزمية Social Spider Optimization (SSO), وذلك عن طريق إضافة قيمة سمعة الخدمة إلى وزن الأعضاء. يهدف النموذج الثالث إلى تصنيف مساحة البحث إلى مجموعات فرعية. فعندما يحتاج المستخدم النهائي إلى خدمة ، تتم عملية البحث في المجموعة الفرعية المناسبة. و لتصنيف مساحة البحث ، تم استخدم نموذج Fuzzy Logic System (FLS) .

أخيرًا ، تم إجراء دراسة مقارنة وعرض النتائج التجريبية لتقييم أداء النماذج المقترحة مع خوارزميات اختيار الخدمة الأخرى ذات الصلة. حيث تم الحصول على كفاءة فى اداء النماذج المقترحة في كل من حالات maximization and minimization optimization . فلقد حققوا أداءً أفضل من حيث قيم دالة الهدف ووقت التنفيذ.

Contents

List of Figures

List of Tables

‘

List of Papers

Published papers

A. N. Abosaif and S. E. Elrofai, “QOS – Aware Meta-Heuristic Services Selection Algorithm and Likert Scale Measurement for IoT Environment,” Int. J. Comput. Sci. Trends Technol., vol. 8, no. 1, pp. 1–8, 2020.

A. N. Abosaif and H. S. Hamza, “Quality of service-aware service selection algorithms for the internet of things environment : A review paper,” Array, vol. 8, no. September, p. 100041, 2020.

Under Review

A. N. Abosaif and S. E. Elrofai, “An Efficient QoS-aware Services Selection in IoT using A Reputation Improved- Social Spider Optimization Algorithm,” Journal of Internet Services and Applications.

List of Abbreviations

Abbreviation	Definition
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AGP	Adaptive Genetic Programming
AHP	Analytic Hierarchy Process
ANN	Artificial Neural Network
Bi-SPO	Bi-Objective Shortest Path Optimization
CASSARAM	Context-Aware Sensor Search, Selection, And Ranking Model
CBSSMS	Collocation Based Sensor-Service Mapping Strategy
CCOA	Chaos Control Optimal Algorithm
CCSOS	Composited Cloud Manufacturing Service Optimal Selection
CEA	Cooperative Evolution Algorithm
CNCF	Context-Aware Neural Collaborative Filtering
DURD	Discrete Uniform Rank Distribution
DUSRD	Discrete Uniform Service Rank Distribution
EB	Energy Balancing
FBP	Flow-Based Program
FIFO	First In First Out
FQSA	Flexible QoS-Based Service Selection Algorithm
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
HABC	Hybrid Artificial Bee Colony
HOLA	Heuristic And Opportunistic Link Selection Algorithm
IIoT	Industrial IoT
ILP	Integer Linear Programming
IoT	Internet of Things
IPCC	Item-Based PCC
LRI	Linear Reward Inaction
LBFM	Location-Based Factorization Machine
MaOP	Many-Objective Optimization Problem
MOPs	Multi-objective optimization problem
MWIS	Maximum Weighted Independent Set
MWL	Maximum Weighted Link
NCF	Neural Collaborative Filtering
OGA	Orthogonal Genetic Algorithm
OPA	One-Pass Algorithm
PSO	Particle Swarm Optimization
PSS	Physical Service Selection
QCD	QoS Constraints Decomposition
QoS	Quality of Service
RR	Round-Robin Scheduling
RQWS	Randomly Generated QWS
SC	Service Composition
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSA	Service Selection Algorithm
TGA	Transactional Genetic Algorithm
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
UDDI	Universal Description, Discovery, and Integration
UPCC	User-Based PCC
W3C	World Wide Web Consortium
WS	Web Services
WSDL	Web Services Description Language
WSREC	Web Service Recommender

Chapter One

Introduction

Chapter One: Research Introduction

1.1 Introduction

In the last decade, interest has been increased in both academia and industry in the Internet of Things (IoT), for high economic impact [1]. Researchers predicted that devices and objects would be increasingly connected in the future [2], thereby leading to more services being exposed to end-users. These services have made the life of people more comfortable and more straightforward than it was before.

IoT environment provides a platform where objects become smarter every day, daily communication becomes informative, and the processes become intelligent. In this regard, to provide IoT services, numerous heterogeneous frameworks and protocols are proposed.

The fundamental feature of IoT systems includes selecting and combining services for matching the needs and preferences of end-users. Service selection is regarded as the core of Service Composition (SC) [3], and it is defined as selecting the most appropriate service from candidate independent services that match the end-user requirements[4]. These candidate services provide the same functionality, but they may have different non-functional properties. The non-functional properties are Quality of Service (QoS) factors with different values. Thus, it is essential to consider QoS factors that satisfy end-user requirements.

The service selection process is even more challenging when combined with specific QoS requirements. In this case, the problem does not merely involve identifying, selecting, and combining service with particular functionalities. However, it is necessary to select the best service that matches both the functionality and the end user's QoS requirements.

Also, due to IoT devices' mobility and the highly dynamic nature of the IoT environment, QoS requirements may change over time. Therefore the states of things in the physical world should be provided in real-time by IoT services.

All the above constraints are required to apply an efficient model to select the set of services with specific criteria, such as the energy consumption [5] [6] [7], response

time [8] [9], reliability [5] [8] [9] [10], availability [11] [12] [13], cost [14] [15], throughput [16], and accuracy [10].

1.2 Problem Statement

Based on the IoT environment's vision, the number of connected actuators and sensors is quickly growing to a massive number of small-embedded devices in the physical world. This growth leads to an increase in the number of services integrated one service to the end-users. These services are combined in the context of QoS constraints such as (response time, integrity, reliability, cost, throughput, and accuracy). There are vast services similar in their functionality and differ in their non-functionality requirement (QoS constraints). This constraint makes a selection of an optimal service that satisfies the end-users requirement an NP-hard problem.

This research investigates a model that can select an optimal abstract service from a set of concrete services. The selection criteria are built based on the QoS constraint that is defined by the end-user.

1.3 Research Questions

There are five main research questions used to map the work directions and outline the problem statement. This study tries to answer the following Research Questions (R.Q.):

R.Q.1 What are the fundamental components applied to design the services selection model in terms of process time, behavioral workflow management, and optimization objectives?

R.Q.2 How can the things or objects in the IoT environment connect, interact, and compose together to provide a service to the end-users?

R.Q.3 How can the end-user feedback affect the behavior of the services selection model?

R.Q.4 How can the QoS factors that can be preferred by the end-users in the IoT environment be adopted to support the dynamic service selection model?

R.Q.5 What techniques are used to evaluate selection algorithms' performance based on evaluation types, software used, and datasets applied?

1.4 Research Objectives

This study's primary goal is to select the optimal services that satisfy the end-user requirements in IoT environments based on QoS constraints. Three sub-objectives direct this study as follows**:**

1. To investigate and classify the state-of-the-art literature of optimization solutions that support end-users preference based on service selection and QoS constraints in the IoT environment.
2. To develop an efficient service selection model in the IoT environment that satisfies end-users preferences for specific constraints of QoS.
3. To conduct a comparative evaluation of the proposed service selection model's performance and the current state-of-the-art approaches.

1.5 Research Methodology and Tools

This research uses qualitative and quantitative research methodologies to achieve the research goals. The qualitative methodology is expressed in the literature review that surveys the state- the- arts of QoS–aware services selection algorithms for the IoT environment. The quantitative methodology is represented in the development of the services selection model in the IoT environment.

The methodology followed in this research is divided into two research methodologies. The inductive methodology aims to enhance Meta-Heuristic algorithms' behavior to select the optimal services that satisfy end-users requirements and apply a Fuzzy Logic System (FLS) to classify the search space into sub-clusters based on QoS factors. Second is the deductive methodology, which aims to test the existing state-of-the-art solutions**.**

The proposed model uses both objective information from the service providers and the service consumers' subjective information. It can develop a multi-objective

optimization algorithm, which imitates a social spider colony behavior. The optimization algorithm aims to help the end-users select the optimal services based on their QoS constraints, such as (response time, availability, reliability, cost, throughput, and reputation).

The proposed model has implemented in web services techniques integrated with the IoT platforms between the services and application layers.

The environment software that is used to simulate the proposed selection algorithm is MATLAB version 2017a. MATLAB is an interactive software that has become the most widely used software in academia and industry for modeling and simulation.

1.6 Research Contribution

This research contributed to improving QoS factors' fitness value, also called the objective function value of the optimization problem. It was introduced to enhance the performance of a meta-heuristic optimization algorithm called social spider optimization algorithm (SSO) by adding a reputation value to member's weight. This contribution increased the maximizing condition's fitness value and reduced it in the minimization condition by obtaining an optimal solution with less iteration.

Also, this study reduced the execution time of the selection process by reducing the search space of optimization algorithms; that has been done by classifying the search space into sub-clusters using a Fuzzy Logic System (FLS). Then when the end-user requires a service, the search is done only into the cluster that appropriates with QoS requirements.

1.7 Scope of This Study

There are many technologies implemented to provide the optimal services to the end-users based on their requirements. These technologies include discovery, matching, composition, and selection of the services. The scope of this research is confined to addressing the services selection optimization problem based on a set of

QoS constraints. The proposed model is concerned with collecting end-users preferences and selects the optimal services that satisfy their requirements. The selection process is performed on a set of services that are similar in their functional requirements and differ in non- functional requirements.

1.8 Research Outlines

The rest of the thesis is organized into six chapters as follows:

Chapter Two outlines the background and literature review related to IoT architecture and Web Services components. Also, it introduces the essential concepts needed to understand the service selection problem and QoS constraints.

Chapter Three reviews the state-of-the-art services selection solutions in IoT. It proposes a new classification to survey and review these proposed solutions. The results and future research directions for designing, implementing, and evaluating a selection algorithm are discussed in this chapter.

In Chapter Four, describes the research design and methodology. It involves the structural description of the proposed services selection model.

Chapter Five covers the performance analysis of the proposed QoS-aware service selection models compared to existing state-of-the-art algorithms using $MATLAB\ R2017b$.

Chapter Six presents the conclusion and future work recommendations for QoS-aware service selection algorithms in the IoT environment.

Chapter Two

Literature Review

Chapter Two: Literature Review

2.1 Introduction

This chapter recovers the import concepts and techniques related to the IoT environment's services selection process to get more knowledge about the research area and its components. The IoT architecture will be declared, followed by web services definitions, protocols, and components. The different Quality of Services (QoS) factors will be defined based on traditional QoS classification and IoT architecture. The three different types of optimization objectives algorithms and search optimization algorithms that are used to solve the selection process will be illustrated. At the End of this chapter, the fuzzy Logic System (FLS), Fuzzy Logic Controllers (FLC), and their components will be explained.

2.2 Internet of Things (IoT)

The first appeared in IoT terminology was introduced in 1999 [16]. It's a new idea that links radio-frequency identification (RFID) objects together. The concept of IoT is built based on different things or physical objects connected over the internet [16], is defined by IEEE as a network that connects uniquely identifiable Things to the Internet [17]. Another definition of IoT based on A and C elements [18], as shown in Figure (2.1), it is defined as things and peoples connected anytime, anyplace, with anything and anyone, ideally using any path/network and any service to address certain elements such as convergence, content, collections (repositories), computing, communication, and connectivity.

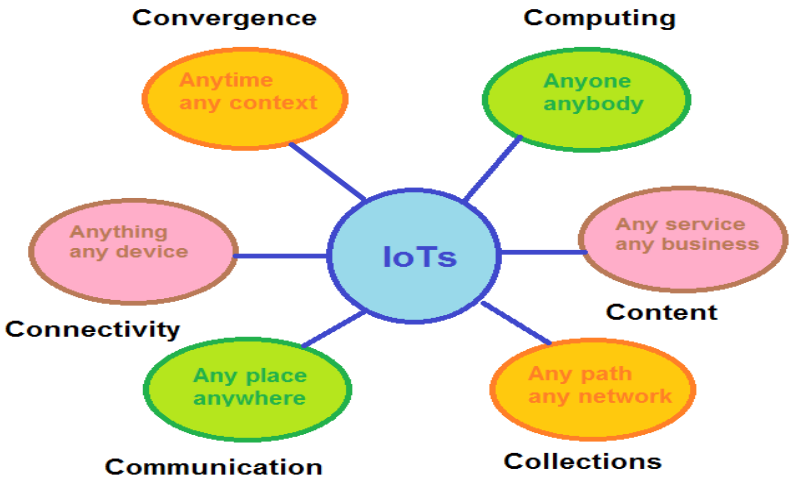


Figure (2. 1): IoT A& C Definition[17][18]

2.3 IoT Architecture

The architectural view provides a standardized way of structuring architectural descriptions. This section describes the structure of the IoT to provide a clear view of its different layers. The IoT architecture consists of five main layers these layers from down to up as defined as the following [19], shown in Figure (2.4):

2.3.1 Things

Things refer to smart physical objects that have their IP address connected and interacted together. These smart objects provide the end-users the required information collected from the surrounding environment. This information can be processing and storing by them. Things can be actuators, sensors, virtual objects, and people. Actuators are a mechanism for turning electrical data into energy or motion [20]. It can be categorized by the energy sources that require generating motion. For example, in smartphone speakers and screen, those are actuators see Figure (2.2).

Sensors are devices that transform useful energy detected from that physical environment [21] such as light, heat, motion, moisture, pressure, or anyone of a significant number of other environmental phenomena, into electrical data like a

camera, microphone as shown in Figure (2.2). For example, a signal is converted to a human-readable display at the sensor location or transmitted electronically over a reading and processing network.

Things can also be people or virtual objects such as an electronic ticket, agendas, books, and wallets.

The thing in IoT can round in cycle from the right side as a user require specific information from Services query through commands, these commands go to the things, which process needed information and passed them to the users as illustrated in Figure (2.3).

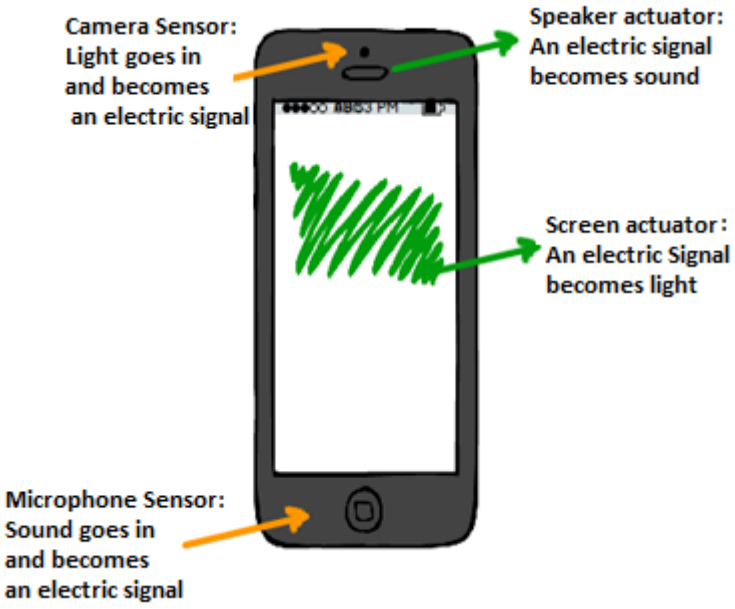


Figure (2. 2): Illustrate the Difference between Sensors and Actuators [20]

And from the left side, it's round as users monitors and observes things to keep track of their performance. When a new event occurs, things send a new event to services and services to send a recent change to the users.

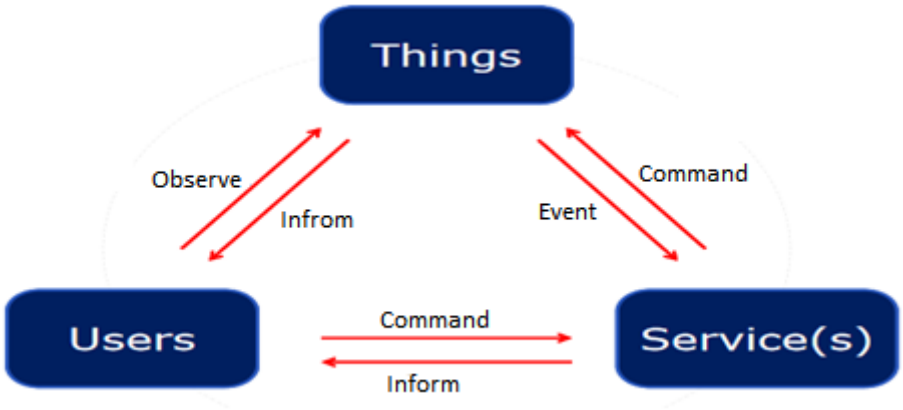


Figure (2. 3): The Things Cycle [22]

2.3.2 Connectivity

Connectivity represents a network communication layer, such as wireless and wireline technologies, standards, and protocols [19]. It concerns low power communication such as Wireless Fidelity (Wi-Fi), Near-Field Communication (NFC), RFID, and Bluetooth. Also, it includes gateways to transfer the collected data into the next layer. Thus, it has two interfaces, one connected to the things network and another related to the Internet.

2.3.3 IoT Middleware

The middleware is a software layer or a set of sub-layers that provides different services to an application and mediates between the technological and the application levels and more than one existing application [23]. Middleware provides an abstraction for the underlying infrastructure by hiding the details of different technologies, allowing them to deal with varying applications on various platforms and communicate together. It handles the context-management of the data and services such as service identification and discovery, composition, and changing the devices' status. It must differentiate between middleware, platform, and framework. Platform defined as a software platform allows the software to run, like a Java

virtual machine (JVM), operating software (OS), compilers, etc. In IoT, the environment platform represents the type of middleware used to connect the IoT components (objects, people, services, etc.) into a single output. As mention in [2]. The platform is a key to success IoT environment because IoT platforms can process many of the infrastructure components into a single product produced in the IoT system. The framework is defined as a software design concerted to various software components like the code libraries, scripting language, and any other software. It helps to develop and link elements of the software project.

2.3.4 Services

Services can be defined as actions performed to add value to the user. In IoT, Services are committed to providing cloud services to the data [19]. Cloud services can be emerging services sensing-as-a-service (S2aaS) and Object-as-a-service. These services can be used to store big data, perform information processing, Optimize business processes by integrating device data, and make decisions. The IoT services layer is also responsible for protecting the data using a trust, security, and privacy mechanisms. The output of this layer is passed to the next layer.

2.3.5 Application

The application presents the final output of data [19], such as social media, web page, mobile software. In an IoT environment, data can be processed in different fields simultaneously, such as data in a smart home, health care, smart city, and intelligent industry.

Figure (2. 4): IoT Architecture[19][24]

2.4 IoT Middleware Layer

IoT middleware is a software that serves as a mediator between layers of the IoT to communicate IoT elements together [25]. It extends to the whole of the IoT layers to connect the heterogeneous and complex existing applications. The role in IoT is that it allows (any Thing) to be connected and to communicate over a network.

Middleware for IoT is required for many different tasks [25]. It acts as connectivity joining the heterogeneous components in IoT together. Also, it enables and defines a common standard of huge devices from different domains in IoT. It hides the infrastructure details by providing Application Programming Interfacing (API) for physical layer communication. Besides, it provides desired services to heterogeneous applications from a different domain and allows them to integrate. Middleware architecture is shown in Figure (2.5). It presents a layered view of the

IoT middleware architecture. The main layers are interface protocols, device abstraction, central and management module, application abstraction module, other components like context analysis, and knowledge data from distributed architecture.

Interface protocols charges to provide technical interoperability by using the same communication protocols. It is defined as the association of software or hardware components and platforms that enable machine-to-machine communication.

Device abstraction is responsible for providing interoperation, resolving the syntax, and semantics associated with devices [25].

The Central and management module is the core component that performs the device discovery, management, and context detection and services composition. The application abstraction module provides the interface with the local and remote applications. Local applications mainly run as event-driven services.

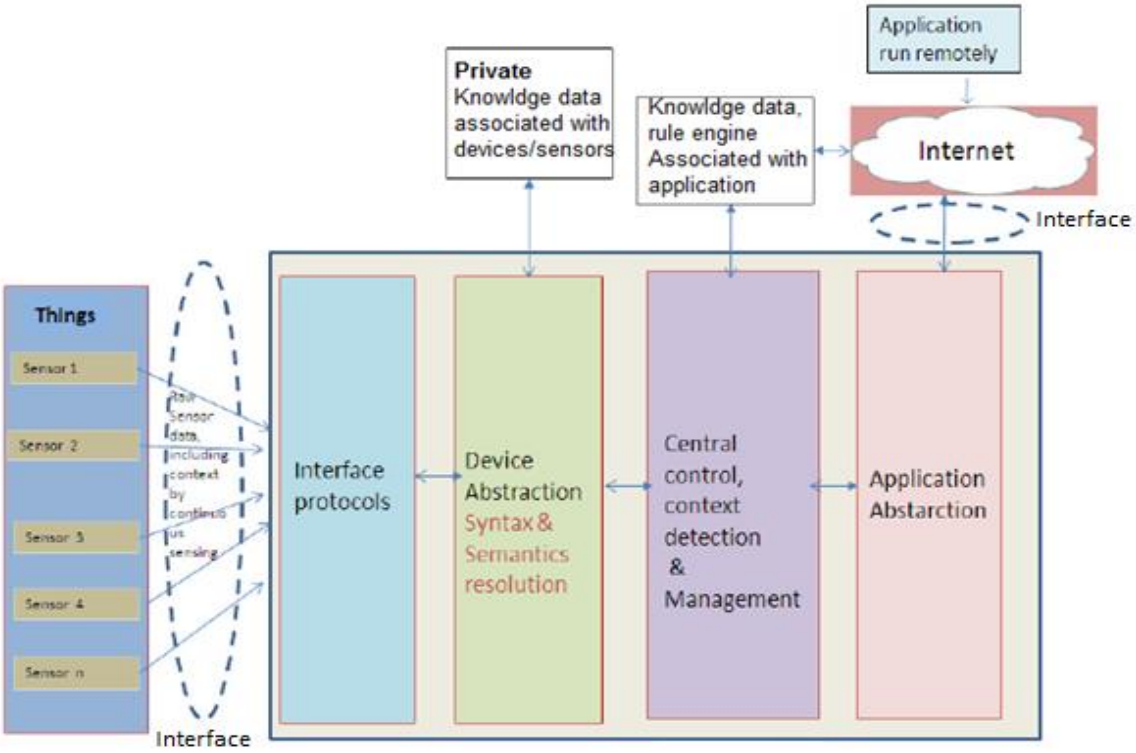


Figure (2. 5): Functional Components of IoT Middleware [25]

2.5 Web Services (WS)

There are many definitions of web services that vary from very public to very defined and specific definitions. Most consumers show a web service as an application or end services communicate to other applications or end services through the web. IBM described web services as a new breed of the web application, and they are self-contained, self-describing, modular applications that can be published, located, and invoked across the web [26]. Also, the World Wide Web Consortium (W3C) [27] introduced the concept of web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described specific format like Web Services Description Language (WSDL). Another demarcation of web services [28] is self-contained programs that can be executed through the internet's standard, global protocols. Generally, web services depend on three critical standard concepts web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP), and the Universal Description, Discovery, and Integration (UDDI).

This means a WS composes of different components connected and integrated into a more complex distributed environment.

2.5.1 Web service components

There are three main components of web services architecture [26], as shown in Figure (2.6): service provider, service consumer, and service registry.

Service Provider (SP): provides different services through the interfaces to create, implement, and publish a web service by using the (UDDI) specification.

Service Consumer (SC): requests and consumes the services. It regards the end-user of a web service. SC uses the service registry to gain information about services and access to it.

Service Registry (SR): contains information about different services provided based on the UDDI specification where services are listed and advertised for search.

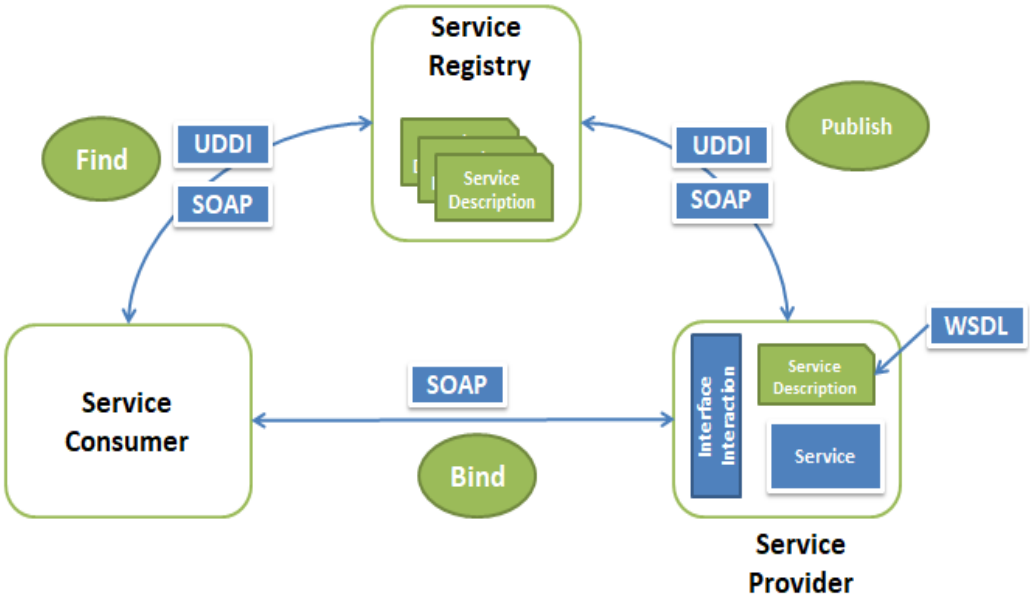


Figure (2. 6): Web Service Components [26]

2.6 Web Services Techniques

This section explains some essential concepts and techniques in web services. This research must know these concepts and understand the difference between them.

2.6.1 Atomic vs. Composite Web services

The main difference between atomic and composite web services is that; the atomic Service, also named elementary service, is a sold service that can't be divided into another web [29]. It has its interface based on SOAP and WSDL techniques.

Composite Service is the integration or collection of atomic and composite services; each has its function to implement a specific application or service [29]. An example of a composited service would be a travel outfit service; integrating services consist of many other services like booking appropriate flights, Reservations hotels, searching for Transportation, specific places, etc. Whatever atomic or composite a web service, it must be defined by an identifier that consists of a set of attributes of a

service to deliver information useful for the service's potential consumers and a set of operations to identify a service function.

2.6.2 Interaction Protocols

SOAP and REST are the primary protocols for developing web services. SOAP web service is a messaging protocol that allows applications to exchange information via hypertext transfer protocol (HTTP) and its extensible markup language (XML) [30]. SOAP has a standard message format for communication used to describing how information should be enveloped into an XML document. SOAP-based web services are defined as independent and state-full, requiring more computation resources, especially when handling SOAP messages. It used to call a Service registration, discovery, and invocation. SOAP-based web services are typically used to integrate the composed services and complex applications. Restful web Services where a REST stands for Representational State Transfer, which was introduced as an architectural style for building large-scale hypermedia distributed systems. RESTful web services develop the REST model by Uniform Resource Identifier (URI), which offers a global addressing space for resource and service discovery [30]. RESTful web services interact through a uniform interface, including a fixed set of operations (GET, PUT, DELETE, and POST) in the web and (HTTP) content. The services interact by exchanging requests and response messages, each containing sufficient information to describe how the message is handled. The difference to SOAP-based, RESTful web services is stateless and lightweight, suitable for merge or composition, and integration over the web.

2.7 Web Service Composition (WSC)

Web services composition involves integrating and collecting more than one service into a single service to perform more complex functions [29]. This section discusses important concepts and methods related to WSC.

In [30], the authors present an SOA-based middleware architecture, and a service composition layer comes on second top layers after the application layer, as in Figure (2.7). This layer creates applications by process and composes a single service offered by different networked objects. This layer doesn't care about heterogeneous devices and connects; it cares only about those devices' services. Service-oriented architecture (SOA) is defined as a modern paradigm to develop software systems often described as composite WS [31].

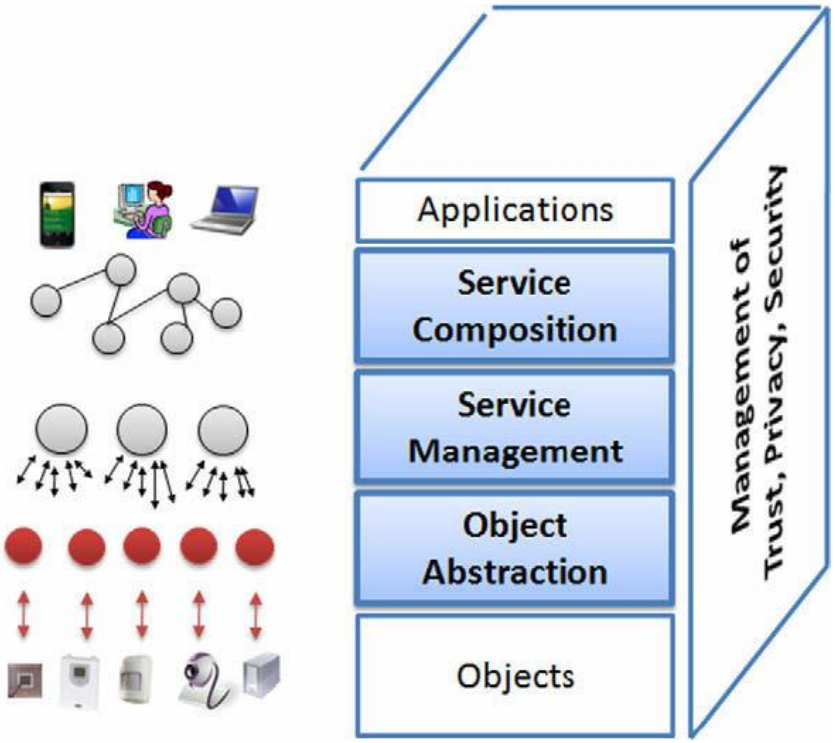


Figure (2. 7): SOA- Architecture [29][23]

2.7.1 Time Stage Composition

For deployment of services composition based on time stage, there are two main stages in which services can be consist of static and dynamic composition.

The static composition is a combination of services that occurs at design time. Different service components essential for composition are selected, collect together, and then deployed [29][30]. It depends on early service binding. So if there

are better alternative services offered or one service becomes unavailable, the static composition cannot provide a better service composition choice. Static web services composition doesn't become flexible and adaptable when there are frequent runtime changes; static WSC is more suitable when service components are never changed or changed rarely.

The dynamic composition is a combination of services that occurs at runtime. This composition allows determining and replacing service components during runtime [29][30]. It considers changes that can happen in-service components and the better alternative services offered of the composite services. Dynamic services composition is regarded as a more challenging task than static services composition due to essential issues that need to be cared about, such as time limits, composition services correctness, transactional support, etc.

2.7.2 Services Composition Behavior

There are two ways to organize and control the workflow and management of services in WSC. They are service orchestration and services choreography.

Service Orchestration represents a range of several services interaction; there is one centralized service responsible for managing and controlling the interaction and workflows among all other services. This central point, called the orchestrator, organizes the connection between the different services. It has a global and complete view of the logic interaction and obtains a result from various services connection as in Figure (2.8(A)).

Services Choreography represents distributed services. It has no central control point; service interacts and connects with other services with their logic and allows each involved party or service to describe its part in the interaction as in Figure (2.8(B)), which enables sensors to interact with actuators [32] directly.

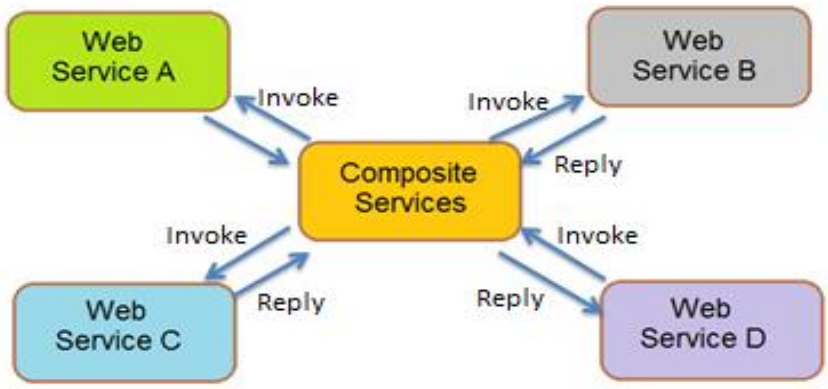


Figure (2. 8 (A)): Organization of Service Orchestration [29]

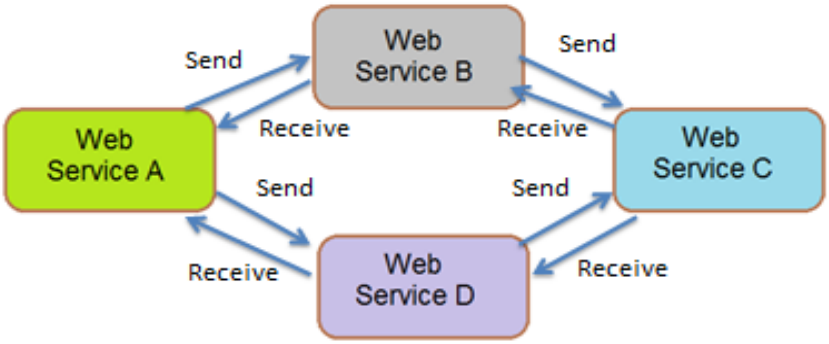


Figure (2. 8(B)): Organization of Service Choreography [29]

2.7.3 Services Composition Automation

Composite services need different services with different tasks to interact together and provide a required combined service or application to the end-user [29]. Web services composition can be classified into three categories manual composition, automated composition, and semi-automated composition.

Manual Composition applied the service providers to create abstract services composite. It binds the abstract web services manually according to desired outputs, which are performed [29]. The manual composition is a time-consuming, hard work,

and error-prone procedure with no guarantee that implementation will satisfy user requirements.

Automated Composition applied the services composited automatically from the start of composition to produce complete services or applications. The composition process depends on a set of existing services components and a previously specified requirement. However, to realize total automated services composition is considered very difficult, especially when it runs in a changeable environment. Semi-Automated Composition shares the ideas of both manual and automated services composition. It aims to assist the user at each step of the services composition procedure. The services composition is processed based on a specific schema; this schema allows the services components to be selected at runtime based on the non-functional properties (e.g., QoS parameters Time, cost accuracy, availability, and so on) and constraints specified by the user.

2.7.4 Services Composition Lifecycle

The lifecycle of services composition consists of the four primary levels [33] [29] [34], as shown in Figure (2.9). These levels are services definition, selection, deployment, and execution.

Services Definition is also called a services description or Need expression; this level is shown from different perspectives, providers' perspectives, and consumers' perspectives. Provider's Perspectives: a service task performance of the priorities of the functional and non-functional service is described using specific languages; a good description of services performance increases the importance of its selection and the validity of the resulting composition services or application. Consumers Perspectives: Information about a required performance and user's preferences for the composite service is described. These requirements collect together either semi-automatically or automatically, into an abstract model to determine a set of activities, the control and data flow among them, the definition of the QoS requirements, and the unusual behaviors.

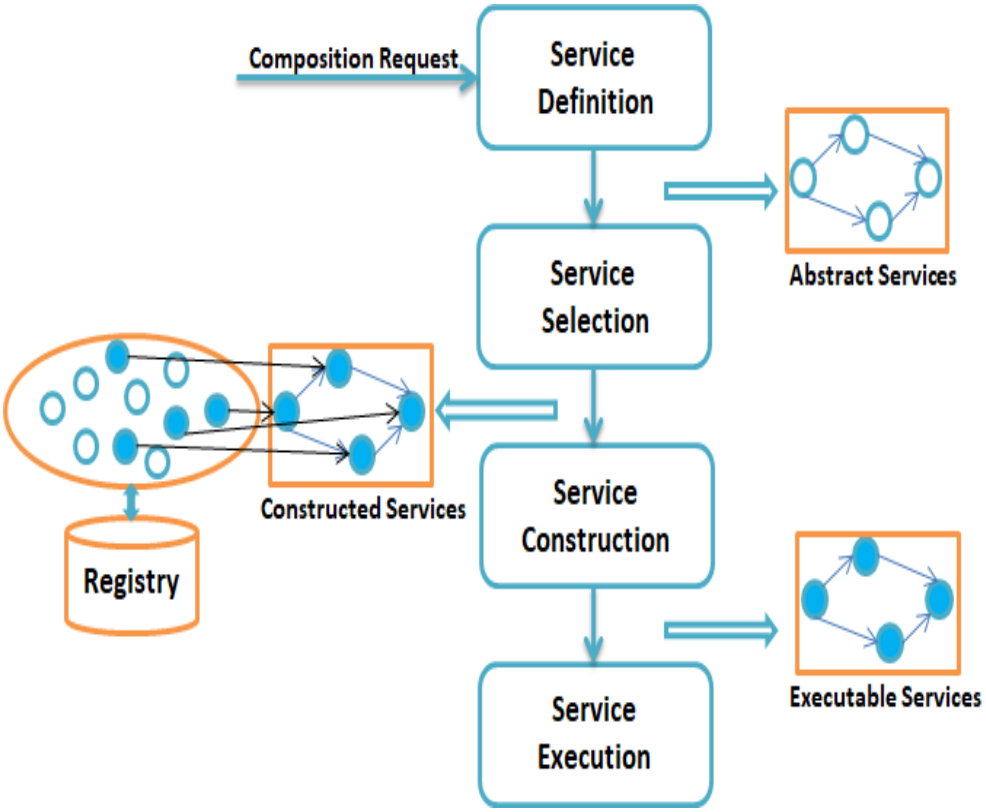


Figure (2. 9): Services Composition Lifecycle [33][3]

Services Selection regards the heart of service composition [3]. The appropriate services offered by different services providers in the services registry match the description of the required service by services consumers are be selected. These selected services consist of more than one candidate service, which meets the user preference. Therefore, the optimal matched service must be selected and grouped into a constructed composite service collection.

Services Deployment in this level, the constructed composite service is published to allow its use and invoke by end-users. As a result of this level, the executable composite service is produced. Services Execution represents the creation and execution of the composite service instance by the execution engine, either its orchestration engine or choreography engine, to run the individual service

components and result as End service or application. The monitoring tasks, including logging, performance measuring, exception handling, and execution tracking, should be performed at this level [29].

In some automated composition methods, the first two levels are merged, where the constructed composite service is directly generated according to the composition requirements [29].

2.8 Quality of Services (QoS)

The services requester needs to consume service from the services provider by searching for it in the services registry, as shown in Figure (2.6). There are many service providers with the same duty or functionality but with different QoS parameters in the services registry. This section explains in detail the concepts of QoS and its types. The QoS clarifies that quality cannot be defined; you know what it is [35]. Also, ISO-9126 defined it as the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs [36].

There are seven quality actors in web services: provider, consumer, stakeholder, developer, QoS broker, quality assurance, and quality manager, as shown in Figure (2.10) [37].

The provider defines as sides that provide an available WS with quality level to the consumer.

Consumers are defined as services requester or end-user of WS.

The stakeholder is responsible for defining the quality level requirements and requests the development of a WS to a developer. Developer: it is the side that develops the WS summiting the required QoS level.

QoS Broker is the side that gathers information on the QoS as consumer needs or requests.

Quality Assurer: it is the side that monitors the quality level of QoS to state the admiration for the Services Level Agreement (SLA) between consumers and providers.

The quality Manager is the side that manages the resources to ensure the promised QoS at the provider side.

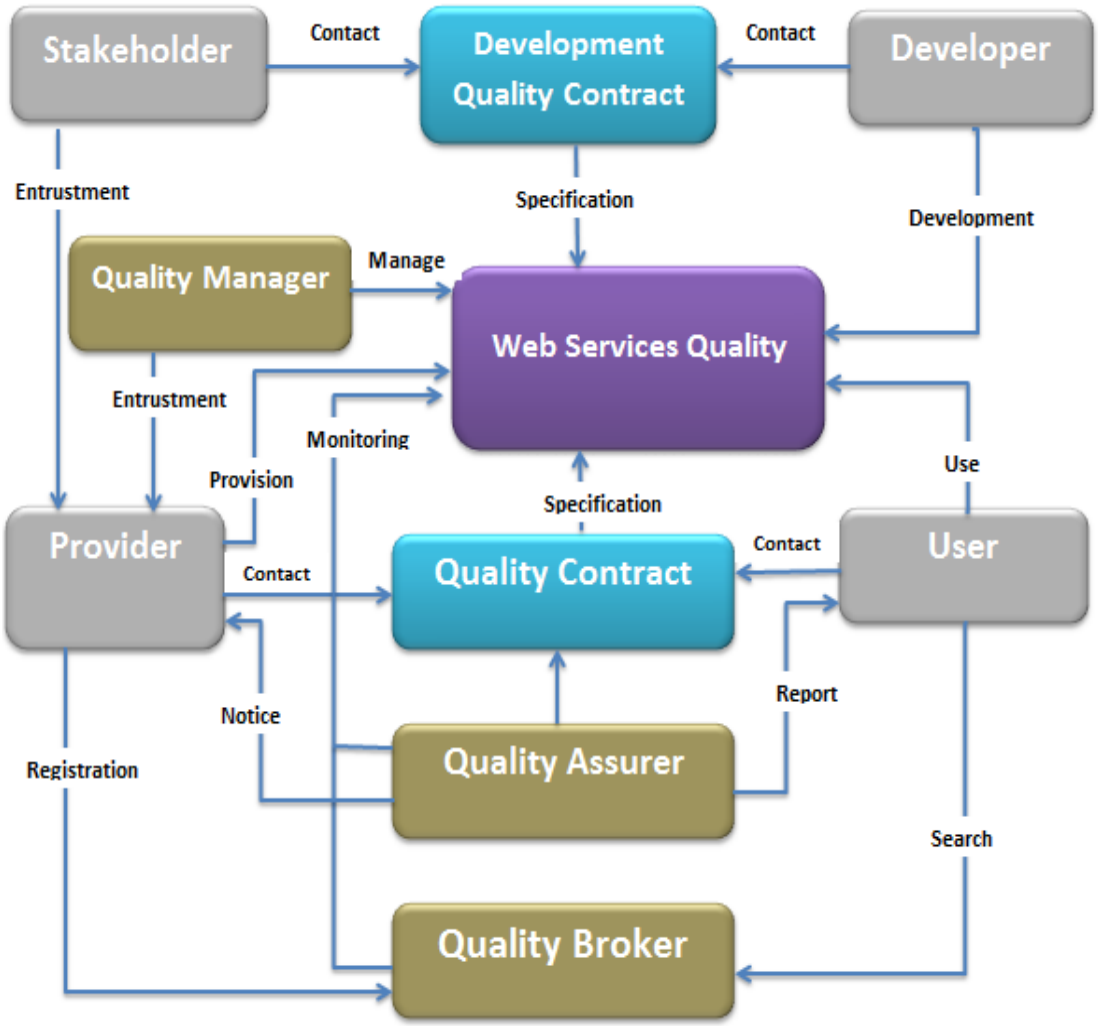


Figure (2. 10): Quality Actors of Web Services[38]

2.9 WS Quality Factors

Quality factors describe the non-functional properties of WS, which consist of two main groups [39], as shown in Figure (2.11):

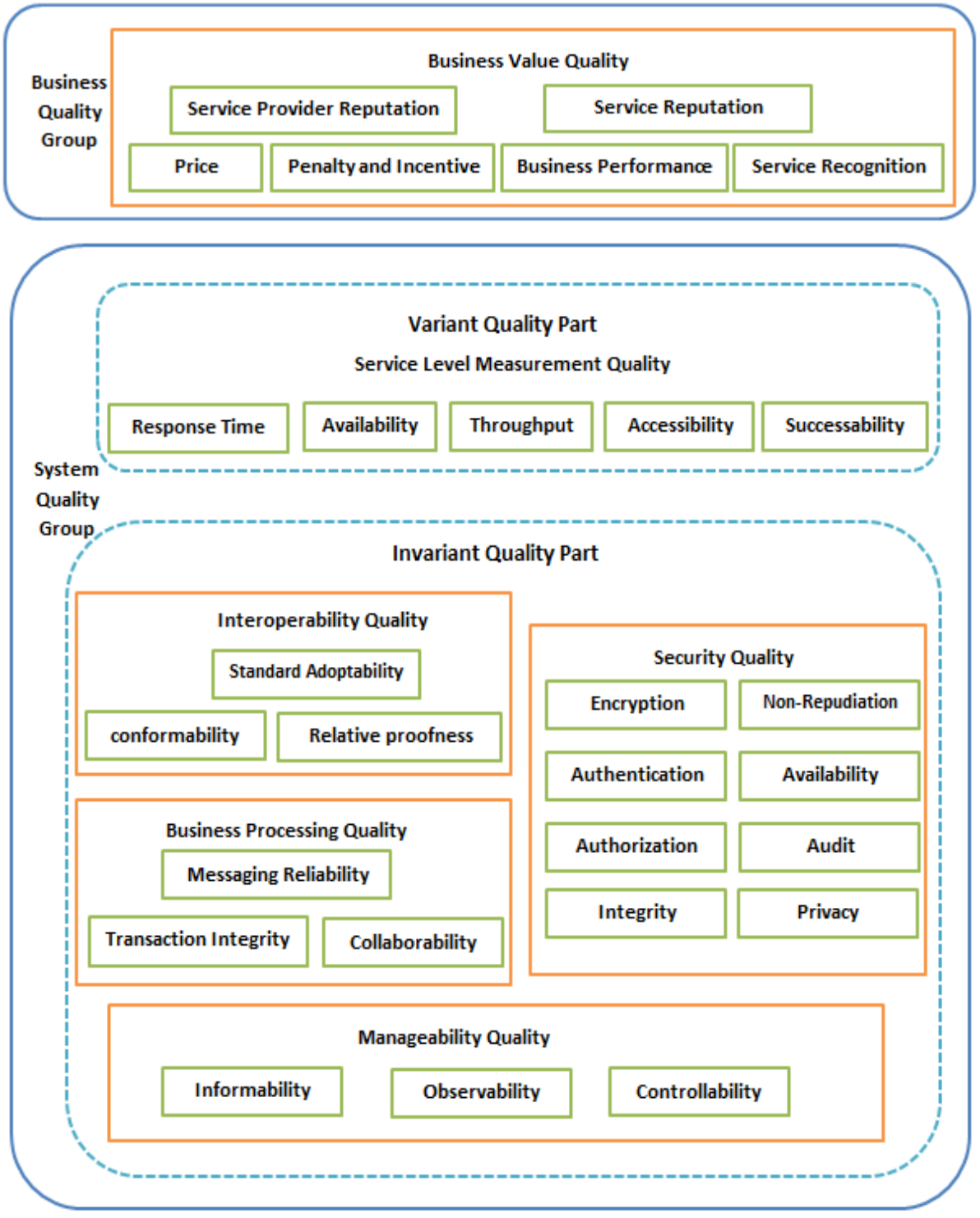


Figure (2. 11): Structure of Web Services Quality Factors [39]

2.9.1 Business Quality Group (BQG)

It means QoS is related to the Business. It consists of business value quality, which means an economic value offered by applying web services to a business. The examples of the BQG are price, penalty and incentive, business performance, service recognition, service reputation, and service provider reputation.

Price is a monetary value of service specified by a service provider and paid from the services requester to the services provider during or after using web services. Penalty and Incentive or compensation is the financial restitution for business losses due to the non-accomplishment of a contract between the service provider or a service consumer or failure to meet QoS's agreed level. Business Performance was the effect of business activity performance when services delivered to the consumer of the service in the real world. In contrast, the business value of web services can be affected directly. The overall service quality is related to the capability of the business structure as well as the QoS.

Service Recognition It is determined by the number of potential consumers who perceive a service. For example, it can be calculated by the number of page views on a service web page. Service Reputation is a social evaluation of service depending on the rates of services from different consumers after requesting services; the reputation of service is calculated to help users decide about using services based on its reputation rate.

Service Provider Reputation is the opinion of collecting service parties toward a service provider based on specific criteria. It gives the service provider a competitive advantage, whereas a good service provider reputation will be regarded as a credible, reliable, trustworthy, and responsible one for service requester.

2.9.2 System Quality group (SQG)

SQG means QoS related to system performance. It is classified into two parts based on the time stage applied to determine QoS properties; they are the variant quality part and invariant quality part.

2.9.2.1 Variant Quality Part

Variant Quality means their quality values can be changed dynamically at run-time while a service is being used. It consists of service level measurement quality: it represents a collection of quantitative parameters that describe the runtime service responsiveness from the requester's perspective and used to measure the rate of web services respond. The variant quality examples are response time, throughput, availability, and other factors such as accessibility and success-ability.

Response Time is the delay time between the service s obtains the input data, and the service s sends the output data in seconds.

Throughput represents the number of service invocations by service providers per second.

Availability is the possibility that the service can be accessed or called Up Time.

2.9.2.2 Invariant Quality Part

Invariant Quality means their quality values are determined once the development of the service is complete. Examples of Invariant Quality are interoperability quality, business processing quality, messaging reliability, transaction integrity, manageability quality inform-ability, controllability, security quality, encryption, authentication, authorization, integrity, availability, audit, non-repudiation, and privacy.

The interoperability quality means to exchange and use information between the service provider and service requester as if both can function adequately on the same platform without any technical or semantic problems in processing a message transmitted between them. Interoperability quality includes sub-quality factors that are standard conformability, standard adaptability, and relative proof-ness.

2.10 QoS based on IoT Architecture

This classification follows the QoS architecture proposed by Ling Li et al. to meet the IoT environment and optimize QoS in different IoT layers [40]. The architecture regards three IoT layers (Sensor Layer, Network Layer, and Application Layer). It integrated the traditional QoS attributes with other essential qualities in IoT Architecture (e.g., the cost of a network deployment, the information accuracy, energy consumption, coverage). The traditional QoS classification cannot meet the IoT heterogeneity and complexity architecture.

2.10.1 The Sensor Layer

It represents a physical infrastructure of IoT, including different edge-nodes, such as data center, RFID-tag, sensor networks, mobile devices, and other heterogeneous devices. It comes from the concept of sensing as an independent service (S2aaS). It enables the IoT environment to provide sensing and actuating capabilities modeled as services, using edge-node devices through the cloud computing system.

The QoS in this layer concerns selecting the sensor's necessary infrastructure based on the user/application requirements and the sensing capabilities. Thus layer aims to deal with the scheduling of information acquisition and resource allocation. For example, QoS in the sensor layer is (energy consumption, system lifetime, and resource optimization) [41].

This study discusses an optimization SSA proposed to optimize a QoS in sensor services. An optimal SSA in the Sensor layer is vital because there are multiple available devices with a different quality that can serve user or application requirements. The availability of services in this layer determines the success or failure of a service request.

2.10.2 Network Layer

It represents a network infrastructure of IoT which responsible for transferring data and information between sensors nodes. This layer includes different protocols, technologies, and edge-network, such as RFID, WSN, and WLAN, cellular. The network layer's importance comes from the network services in IoT, which require various network technologies. Like WSN, to provide universal coverage points and access to network nodes. In this layer, the QoS aims to schedule a heterogeneous network environment through traditional QoS and allocate various network resources. There are different protocols used in heterogeneous networks that largely depend on QoS requirements. For example, QoS in the network layer as (bandwidth, capacity, and throughput) [40][41]. An optimal SSA in the Network layer is essential, allowing service providers to provide optimal network technologies and protocols to serve user/application requirements. The efficient QoS depends on various network infrastructure in this layer becomes a challenge in the application development toward the IoT.

2.10.3 Application Layer

It represents the highest layer in IoT architecture, consisting of many distributed services composed to be one service to the end-user or an application. IoT environment deploys many applications in different domains, such as smart home, industrial automation, health care, traffic management, etc.

This layer aims to deal with the scheduling of services based on the QoS constraints. In QoS constraint, some network resources are allocated to services selected in an application layer, which affect the end services and user's requirements. For example, QoS in the application layer (accuracy, reliability, availability, and execution time) [41].

2.11 Types of Optimization Objectives

Optimization algorithms appeared for decades aimed to find the optimal maximum or minimum solution. It is used widely to solve complex services selection problems, especially QoS selection problems called with different names such as QoS-driven or QoS-aware. There are three main types based on the number of objective functions that can be solved by the optimization algorithms.

2.11.1 Single Objective Optimization (SOP)

SOP considers the standard optimization problem. It means optimization problems involving one objective function to optimize at one time. The single-objective optimization problem can be represented in mathematical terms as in equation (2.1) [42].

$$\text{Min/Max } F(x) = \text{Min/Max } f(x),\ x \in S \qquad (2.1)$$

Where: Min/Max represent the preference solution minimum solution or maximum solution, f (is an objective numerical function, S is the (implicit) set of constraints that can be defined as $S = \{x \in R_m: h(x) = 0, g(x) \geq 0\}$.

2.11.2 Multi-Objective Optimization (MOPs)

The MOPs aim to find an optimal solution in the presence of trade-offs between more than one conflicting objective concerning a set of specific constraints. There is no single solution [43] that can optimize all objectives at the same time. There are sets of infinite optimal solutions or sets of points that fit all predefined conditions for the optimal. These sets are called non-dominated or Pareto optimal solutions, defined in 1906, and became the most used optimization [23]. The Pareto optimal solution is required if none of the objective functions optimized the fitness value. It was done without affecting other objective values or making any of them worse off. When no further Pareto objective can be changed, it is called Pareto optimality. The main challenge can face MOPs that is no best unique solution [43]. It just found a

set of non-dominated solutions to get a good approximation to the real Pareto dominance. The multi-objective optimization problem can be represented in mathematical terms as in equation (2.2) [42]:

$$\text{Min/Max } F(x) = \text{min/Max } [f_1(x), f_2(x), f_3(x)], x \in S \quad (2.2)$$

Where: Min/Max represent the preference solution minimum solution or maximum solution, f (is an objective numerical function, S is the (implicit) set of constraints that can be defined as $S = \{x \in R_m : h(x) = 0, g(x) \geq 0\}$.

2.11.3 Many-Objective Optimization (MaOPs)

MaOPs is extended from MOPs, where the MOPs often optimize two or three objectives [43]. Whereas MaOPs were introduced to optimize more than three objectives with the increase in non-dominated solutions numbers and solution search space. This increase makes the services selection problem more complex and challenging to achieve diversity and convergence in IoT selection algorithms [44]. The many-objective optimization problem can be represented in mathematical terms as in equation (2.3) [42].

$$\text{Min/Max } F(x) = \text{min/Max } [f_1(x), f_2(x), f_3(x), \ldots, f_n(x)], x \in S \quad (2.3)$$

Where: Min/Max represent the preference solution minimum solution or maximum solution, n>1, f (is an objective numerical function, S is the (implicit) set of constraints that can be defined as $S = \{x \in R_m : h(x) = 0, g(x) \geq 0\}$.

2.12 Search Optimization Algorithms

Search optimization Algorithms refer to optimization algorithms used to select the optimal solution in a specific search space. The search optimization algorithms are classified into four main types: 1) Heuristic, 2) Meta-Heuristics, 3) Hyper-heuristic Algorithms, and 4) Non-heuristic Algorithms.

2.12.1 Heuristic Algorithms

The heuristic term came from Greek, which means that I find, discover. It is an artificial intelligence and dependent-optimization algorithm. It is designed to finding an approximate solution for complex problems when traditional algorithms cannot find any exact solution [45]. The main objective of heuristic is to give satisfaction and valuable solution in an acceptable time, not necessarily that the same solution. The main idea in heuristic algorithms is to order an alternative explanation for each step based on available information to decide the next step. This step is done by iterating the proposed algorithm rules and utilizing the previous step to input the next step until reaching the optimal solution.

2.12.2 Meta-Heuristic Algorithms

Meta-Heuristic is a term that was introduced by Glover in 1986. It came from the combination of the Greek prefix meta- (meta, which means that the high-level) plus heuristic [46]. It is a high-level independent-optimization problem that identifies concepts grouped in an algorithmic framework to provide a set of strategies or rules that develop and improve heuristic optimization algorithms and find a suitable solution to a specific problem [46]. It can be defined as a general-purpose heuristic method on the significant search space containing high-quality solutions.

2.12.3 Hyper-Heuristic Algorithms

Hyper-Heuristic is a heuristic search methodology introduced in1997 to indicate the combination of more than one artificial intelligence algorithms in the context of automated theory [47]. It was independently used in 2000 for combinatorial optimization context to denote heuristics to choose heuristics. The main idea in Hyper-heuristic algorithms is automating the use of high-level heuristic methodologies to select, combine, generate, and apply an instance of a specific

problem. It combines more than one appropriate heuristics methodologies at each decision point to efficiently solve computational search problems [48].

2.12.4 Non-heuristic Algorithms

Non-heuristics are optimization algorithms for optimizing the search space and obtaining an optimal solution. In this type, the solution is found without employing evolutionary algorithms, or an iteration process is not followed to find an optimal solution.

2.13 Fuzzy logic System (FLS)

The first introduced of the Fuzzy logic system (FLS) was in the 1960s by Dr. Lotfi Zadeh [49] to form a multi-valued logic rather than precise logic or classical proposition logic. It is an extension of Boolean or classical logic [49].

In the classical proposition logic, the value of variables is one of the truth values, which is either true or false. It is also called two-valued 0 or 1 [50].

In fuzzy logic, variables are not decoding into the absolute values of 0 and 1. It is an infinite valued logic that allows truth values to be any number in the interval [0, 1]. In other words, If Var is an atomic proposition, the truth value of Var Tv (Var) equals one; this means that the Var is true. Also, if the Tv (Var) equals zero, this means that Var is false. But when the Tv (Var) = 0.65, the truth of Var is 0.65.

For example, it is hard to define the truth of "James is old" as definitely true or false. If James's old equals 60 years. In some situations, he is old, being suitable for senior citizen benefits at many institutions. Still, in other cases, he is not old since he is not ideal for social security. So, in fuzzy logic, we would allow Tv (James is old) to take on other values in the interval [0, 1] besides just 0 and 1. One of the fuzzy logic's benefits is that; the fuzzy rules set in natural human language. For example, if the weather is cold, then turn on the heater. These Examples show that words like the cold is more compatible with human-logic than numbers. On the other hand, the sentence "if the weather is -1.4C°, then turn on the heater" is entirely unfamiliar to

people. Also, fuzzy logic applies as an artificial intelligence-based decision-making system with good performance in the pattern classification and the decision-making systems [49].

2.13.1 Probability vs. Fuzzy Logic

Fuzzy logic is not the same as probability. Probability and fuzzy logic are both terms used to describe uncertainty, but how each of these concepts deals with uncertainty is different. Probability measures the uncertainty present in the occurrence of an event. Simultaneously, fuzzy logic measures the uncertainty in the characteristics of an event that has occurred (i.e., fuzziness describes event ambiguity). It measures the degree to which an event occurs, not whether it happens.

2.13.2 Fuzzy Sets vs. a Crisp Set

A crisp set, also called a classical set, is a collection of distinct, well-defined objects. These objects are said to be elements or members of the set. The fuzzy sets are the generalization of the classical set. So, the classical set theory is a subset of the theory of fuzzy sets [49] see Figure (2.12).

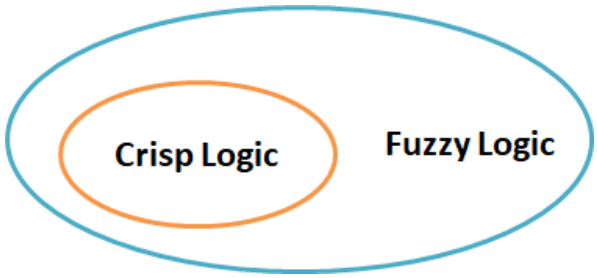


Figure (2. 12): Classical Sets Theory vs. Fuzzy Sets Theory [49].

An example of a crisp logic set is to answer the question, does John have a pen? The answer to the above question is definite value, either Yes or No, depending on the situation. If Yes is assigned a value one and No is assigned to zero, the outcome could have a 0 or 1. So, a logic which demands a binary (0/1) type of handling is known as Crisp logic in the field of fuzzy set theory.

Contrasting to a crisp logic set, a fuzzy logic set approximates human reasoning capabilities to apply it to the knowledge-based systems. The fuzzy logic theory provides a mathematical method to capture the uncertainties related to the human cognitive process.

An example of a fuzzy logic set is to answer the question, is the color of the specific object is blue or not. But the object can have any shade of blue depending on the intensity of the primary color. The answer would vary accordingly, such as royal blue, navy blue, sky blue, etc. The darkest shade of blue value assigns 1 and 0 to the white color at the lowest end of the values spectrum. Then the other shades will range from 0 to 1, according to intensities. Therefore, where any of the values can be accepted in a range of 0 to 1 is termed fuzzy.

The fuzzy set theory intended to introduce the imprecision and vagueness attempt to model the human brain in artificial intelligence. The significance of such an approach is increasing day by day in expert systems[51]. However, the crisp set theory was beneficial as the initial concept to model the digital and expert systems working on binary logic.

2.13.3 Membership Function

A membership function (μ_A) in the fuzzy logic set is a curve that defines how each point in the input space map to membership value, also called the degree of membership. The membership degree represents a value between [0, 1], as in equation (2.4).

$$\mu_A(X): U \rightarrow [0,1] \qquad (2.4)$$

But the membership function in crisp set is defines either value 0 or 1 as in equation (2.5).

$$\mu_A(X) = \begin{cases} 1, & X \in 0 \\ 0, & X \notin 0 \end{cases} \qquad (2.5)$$

One of the critical features of fuzzy logic is that a value can be a member of many sets (crisp sets) at the same time, see Figure (2.13).

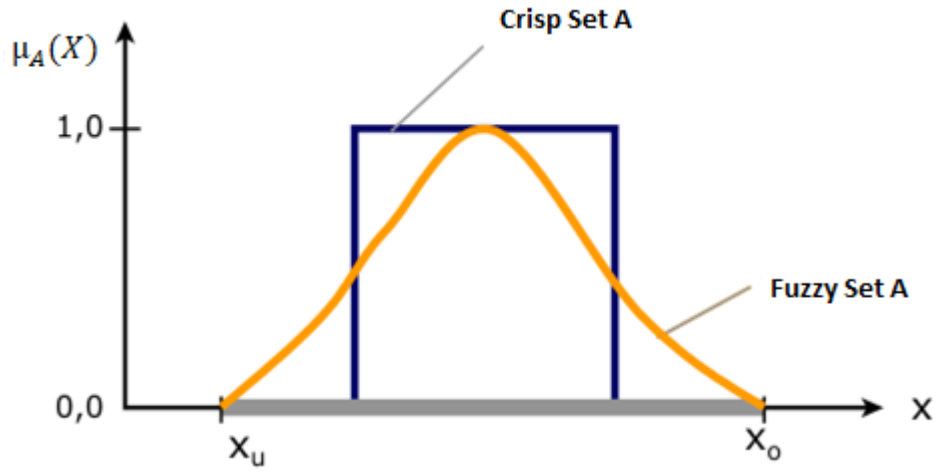


Figure (2. 13): $\mu_A(X)$ of Crisp Set and Fuzzy Set

An example to explain fuzzy membership is to define the tallness of to set of people. Suppose there is a set of people of various heights. The requirement is to classify them into two groups: a tall group and a short group. In the Crisp sets, the system considers that people taller than 6 feet are tall; otherwise, they are considered short. But this is unfair because people with 1 cm less than or more than 60 feet will consider in a different group. That is what a fuzzy membership aimed to clarify and evaluate in the final decision. See Figure (2.14) shows how the membership curve transfers from non-tall values to tall value in a clear view.

So the membership functions represent distributions of possibility rather than probability.

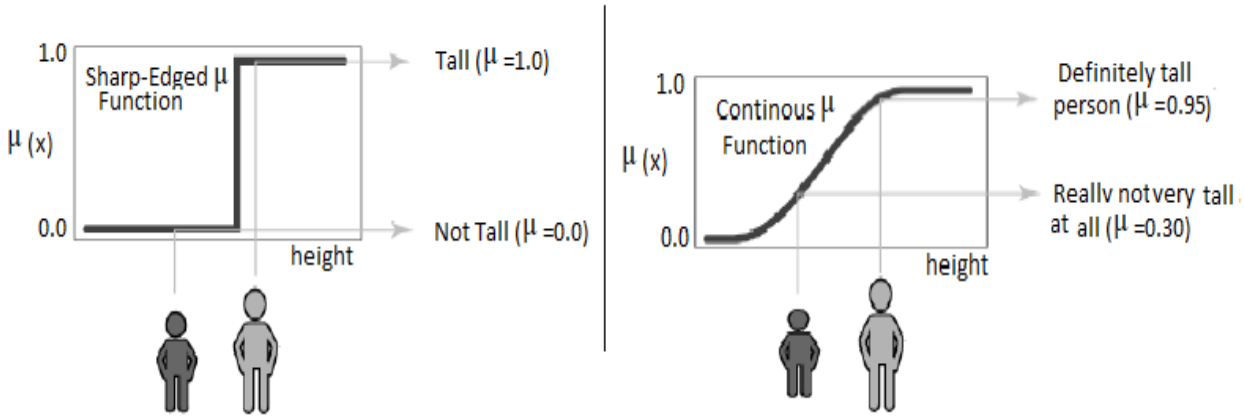


Figure (2. 14): μ (X) Function in Height Example

2.13.4 Membership Function Types

There are various types of membership functions. The most popular used ones (Triangular, Trapezoidal, and Gaussian) function. This research applied the Trapezoidal membership function. The three types can be defined as the membership function, Triangular function, Gaussian function, and Trapezoidal function.

Triangular Function is described by a lower limit a, an upper limit b, and a value m, where a < m < b, as shown in Figure (2.15).

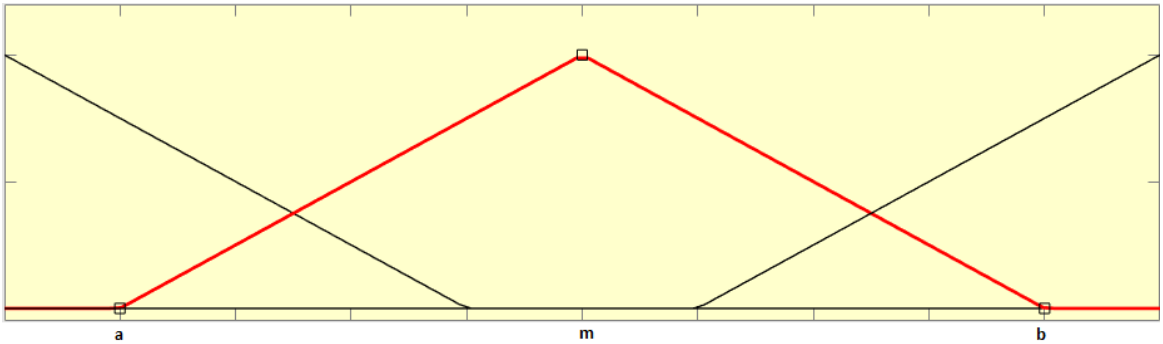


Figure (2. 15): Triangular Function Diagram

Gaussian Function describes by a central value m and a standard deviation k > 0. The smaller k is, the narrower the "bell" is, as shown in Figure (2.16).

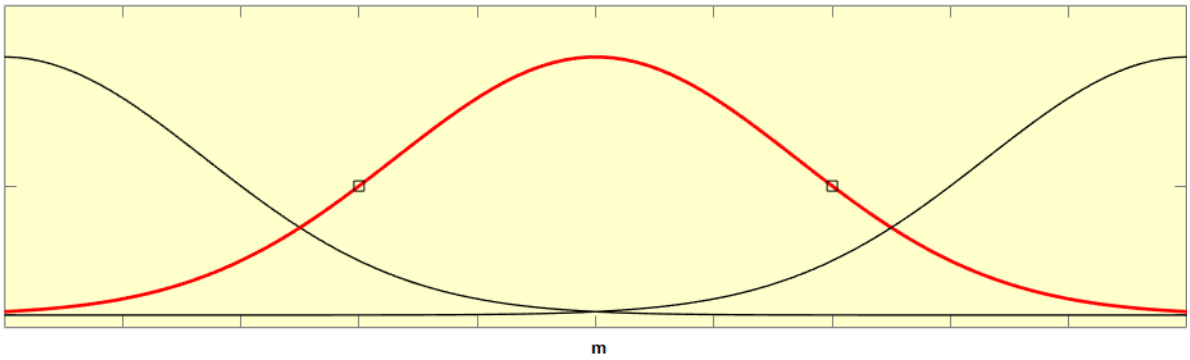


Figure (2. 16): Gaussian Function Diagram

Trapezoidal Function describes a lower limit a, an upper limit d, a lower support limit b, and an upper support limit c, where a < b < c < d.

There are two special cases of a trapezoidal function, which are called R-functions and L-functions: R-functions: with parameters a = b = - ∞, and L-Functions: with parameters c = d = + ∞, as shown in Figure (2.17).

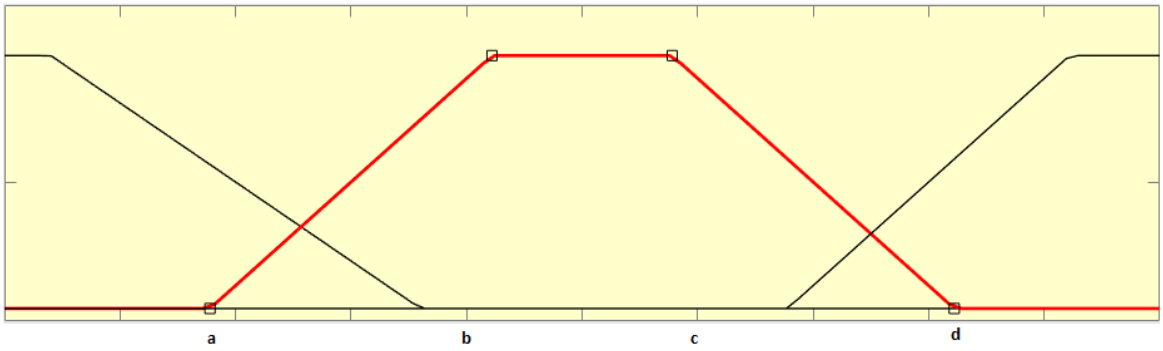


Figure (2. 17): Trapezoidal Function Diagram

2.13.5 Linguistic Variables

Linguistic variables refer to the non-numeric inputs and outputs variables of a fuzzy logic system, which are words or sentences from the Human natural language. For example, a person's height can have a value of tall or shot. Each linguistic variable can decompose into a set of linguistic terms.

Let V be a QoS variable, X the range of the variable's values, and Tv is the truth value of finite or infinite fuzzy sets [49]. A linguistic variable corresponds to the

triplet (V, X, Tv); V= QoS, X=[0,10] , and Tv={Poor, Good, Excellent}see Figure (2.18).

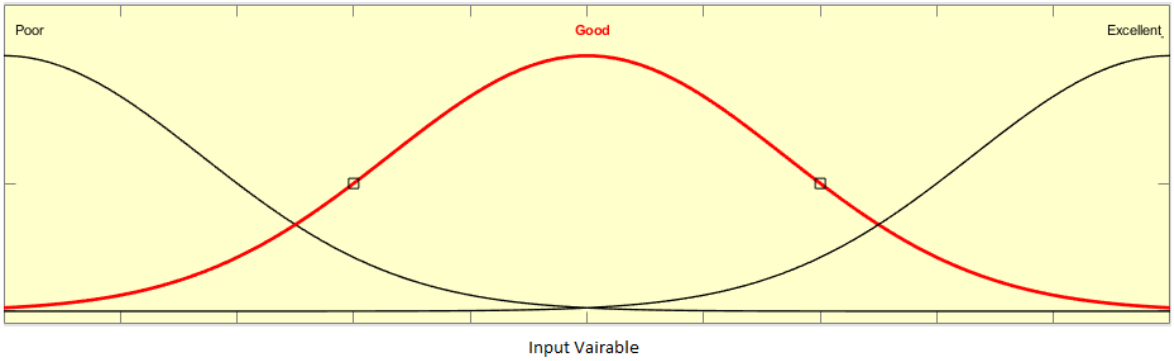


Figure (2. 18): Linguistic Variables for QoS Function

2.13.6 Fuzzy Operators

The crisp sets apply as traditional two-valued logic (bivalent) using Boolean operators (AND, OR, and NOT) to perform complement, union, and intersection operations, respectively. Table (2.1) show the traditional Truth table For Binary Logic.

Table (2. 1): Truth table For Binary Logic

X	Y	X AND Y	X OR Y	NOT X
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

These basic operations over crisp sets can be extended to costume the fuzzy sets. These standard operations' extensions are formalized based on Zadeh Operators [49], see Table (2.2).

Intersection extends from AND operator defined as:

$$\mu_{A\cap B}(X) = Min\ (\mu_A(X), \mu_B(X)) \qquad (2.6)$$

Union extends from OR operator defined as:

$$\mu_{A\cup B}(X) = Max\ (\mu_A(X), \mu_B(X)) \qquad (2.7)$$

Complement extends from NOT operator as:

$$\mu{-}_A(X) = 1\text{- } \mu_A(X) \qquad (2.8)$$

The Binary truth table applies for bivalent logic, but fuzzy logic needs more operators to cover all the possible fuzzy values (all the real numbers between 0 and 1).

Table (2. 2): Fuzzy Operators

standard Operators	Fuzzy Operators
X AND Y	Min(X,Y)
X OR Y	Max(X,Y)
NOT X	1-X

The fuzzy logic is not limited to a finite set of input values, and it needs to define more operations. Table (2.2) explains the convert of binary operators to fuzzy operators. Table (2.3) visualizes how fuzzy logic operations can cover both bivalent and fuzzy combinations.

Table (2. 3): Truth Table for Fuzzy Logic

X	Y	Min(X,Y)	Max(X,Y)	1-X
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0
0.2	0.5	0.2	0.5	0.8
0.7	0.2	0.2	0.7	0.3
0.6	0.6	0.6	0.6	0.4

2.14 Fuzzy Logic Controllers

Fuzzy controllers are very simple conceptually. They consist of three main stages, as in Figure (2.19) [51]. The input stage represents the crisp inputs data that convert to be a fuzzy set, such as maps sensor data, switches, and so on, to the appropriate membership functions and truth values.

The processing stage invokes each appropriate rule and evaluates each result, then combines the rules' results into a fuzzy inference machine.

Finally, the output stage is also called Defuzzifier. It converts the combined result back into a specific control output value. The output value is a crisp output value.

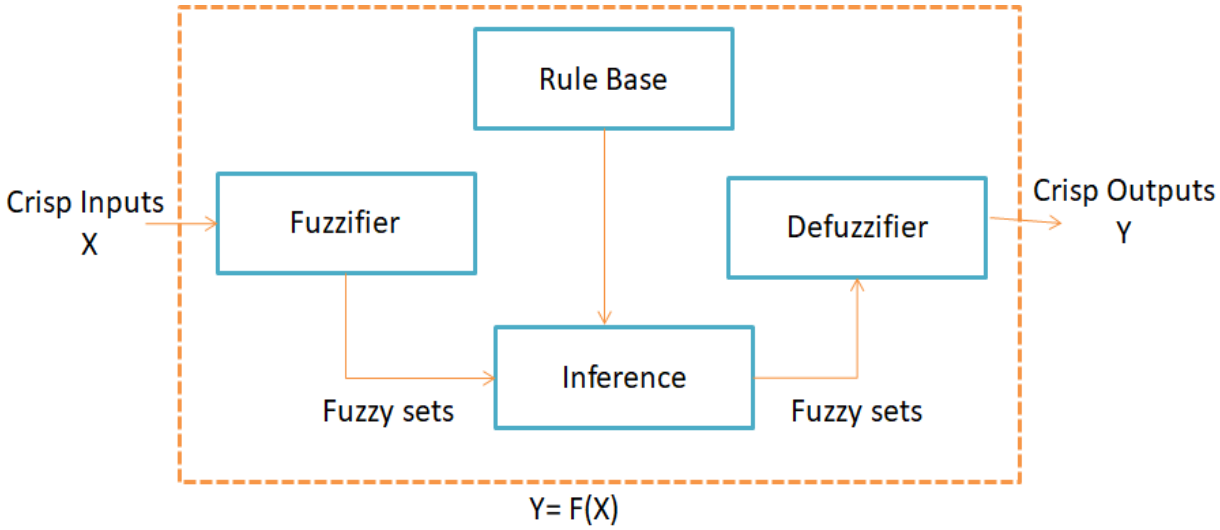


Figure (2. 19): Fuzzy Logic Controllers [51]

2.14.1 Fuzzification

Fuzzification is related to the vagueness and imprecision in a natural language. It is the method of mapping a crisp quantity into a fuzzy quantity. It achieves by recognizing the various assumed crisp and deterministic quantities. It is entirely non-deterministic and quite uncertain. It plays an essential role in dealing with uncertain information, which might be objective or subjective. The uncertainty might have emerged due to vagueness and imprecision, which lead the variables to be represented by a membership function as they could be fuzzy. It is a process that

translates accurate, crisp input values into linguistic variables defined by fuzzy sets. Then it applies membership functions to the measurements and determines the degree of membership. There are two main types of fuzzification operators:

2.14.1.1 Singleton Fuzzifier

It is widely used in fuzzy control applications because it is natural and easy to implement. The Singleton fuzzifier operator abstractly converts a crisp value into a fuzzy singleton within a certain discourse universe. It maps an object to the singleton fuzzy set centered the object itself (i.e., with support and core being the set containing only the given object) [52]. It represents an accurate value, and hence no fuzziness is introduced by fuzzification in this case. In a singleton fuzzifier, the input X_0 interprets as a fuzzy set A with the membership function $\mu_A(X)$ equal zero except at the point x_0 at which $\mu_A(X_0)$ equals one as in Figure (2.20). This research applied the singleton Fuzzifier operator.

2.14.1.2 Probabilistic Fuzzifier

A fuzzifier operator maps data disturbed by random noise by converting the probabilistic data into fuzzy numbers. It maps an object to a fuzzy set generally centered the object itself (i.e., the core of the fuzzy set contains the object) and with support having the object but being a set bigger than only the object itself. It leads to enhance computational efficiency since fuzzy numbers are much easier to manipulate than random variables. For example, an isosceles triangle can choose to be the fuzzification function. This triangle's vertex corresponds to the mean value of a data set, while the base is twice the standard deviation of the data set. In this way, a triangular fuzzy number formed convenient to manipulate, see Figure (2.20).

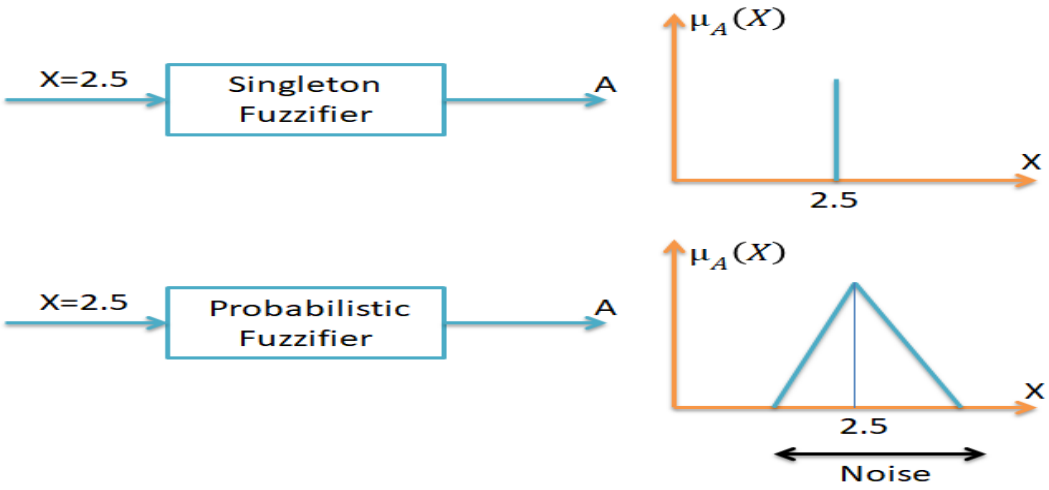


Figure (2. 20): Singleton and Probabilistic Fuzzifier

2.14.2 Rule Bases in Fuzzy Logic

The rule base represents the heart of FLS and forms a mapping of input on the output [53]. As in classical logic, the fuzzy logic rules include as:

$$\begin{cases} \text{If p then q} \\ \text{p true then q true} \end{cases}$$

Based on fuzzy rules, fuzzy reasoning, are expressed in natural language using linguistic variables. A fuzzy rule has the form: If x ∈ A and y ∈ B, then z ∈ C, with A, B, and C are fuzzy sets. For example, if (the weather outside is hot) and (Number of people is many) , then (the conditioner volume should be high). The variable' conditioner volume' belongs to the fuzzy set' high' to the degree that depends on the degree of validating the hypothesis, i.e., the membership degree of the variable' weather outside' to the fuzzy set' hot.' The idea is that the more propositions in hypothesizing are checked, and then the suggested output actions must be applied. The field experts' fuzzy rules may be designed or extracted from a set of numeric data [53].

2.14.3 Fuzzy Inference Engine

It combines the fuzzy input set and rule bases to produces the fuzzy output set. For example, to determine the proposition's degree of truth, fuzzy' conditioner volume will be high,' the fuzzy implication or fuzzy inferences must be defined. The fuzzy implication, like other fuzzy operators, does not have just a single definition. The fuzzy system designer must choose among the wide choice of fuzzy implications. There are many various definitions of fuzzy inferences; the two most commonly used are Mamdani, and Larsen [49], as mentioned in Table (2.4).

Table (2. 4): Definitions of fuzzy implications

Name	Truth value
Mamdani	min $(f_a(x), f_b(x))$
Larsen	$f_a(x) \times f_b(x)$

Figure (2.21) shows the result of the rule "if (the weather outside is hot), then (the conditioner volume should be high)." is applied using a Mamdani implication.

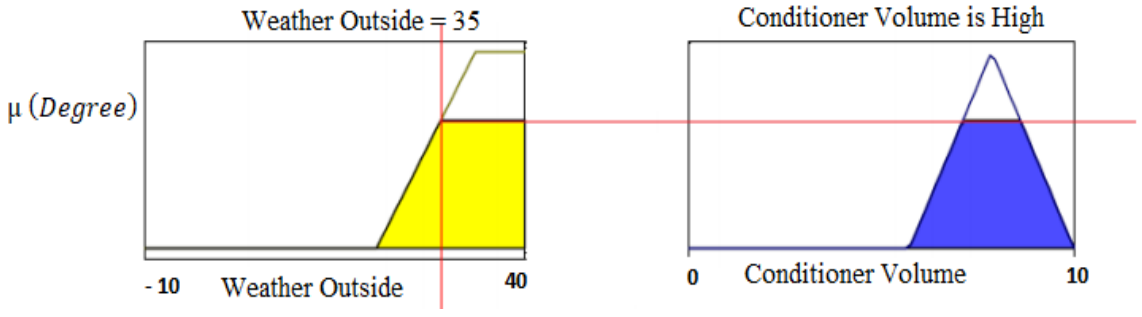


Figure (2. 21): Mamdani Implication of Weather Rule[49]

Let's take a clear example of the Mamdani implications if there is a customer in a restaurant, and he wants to give tips to the server. To determine a tip amount, the

FLS used a Mamdani implication. The tips amount is determined to depend on service quality, and food quality sees Figure (2.22) to see possible fuzzy sets. Figure (2.22) defined the input set as 'the quality of the service =3' and 'the food quality = 8' the output determines 'the amount of the tips = 16.7%'. To express this in a natural language using linguistic variables: 'the quality of the service is low' and 'the food quality is high,' the output determines 'the amount of the tips is average.'

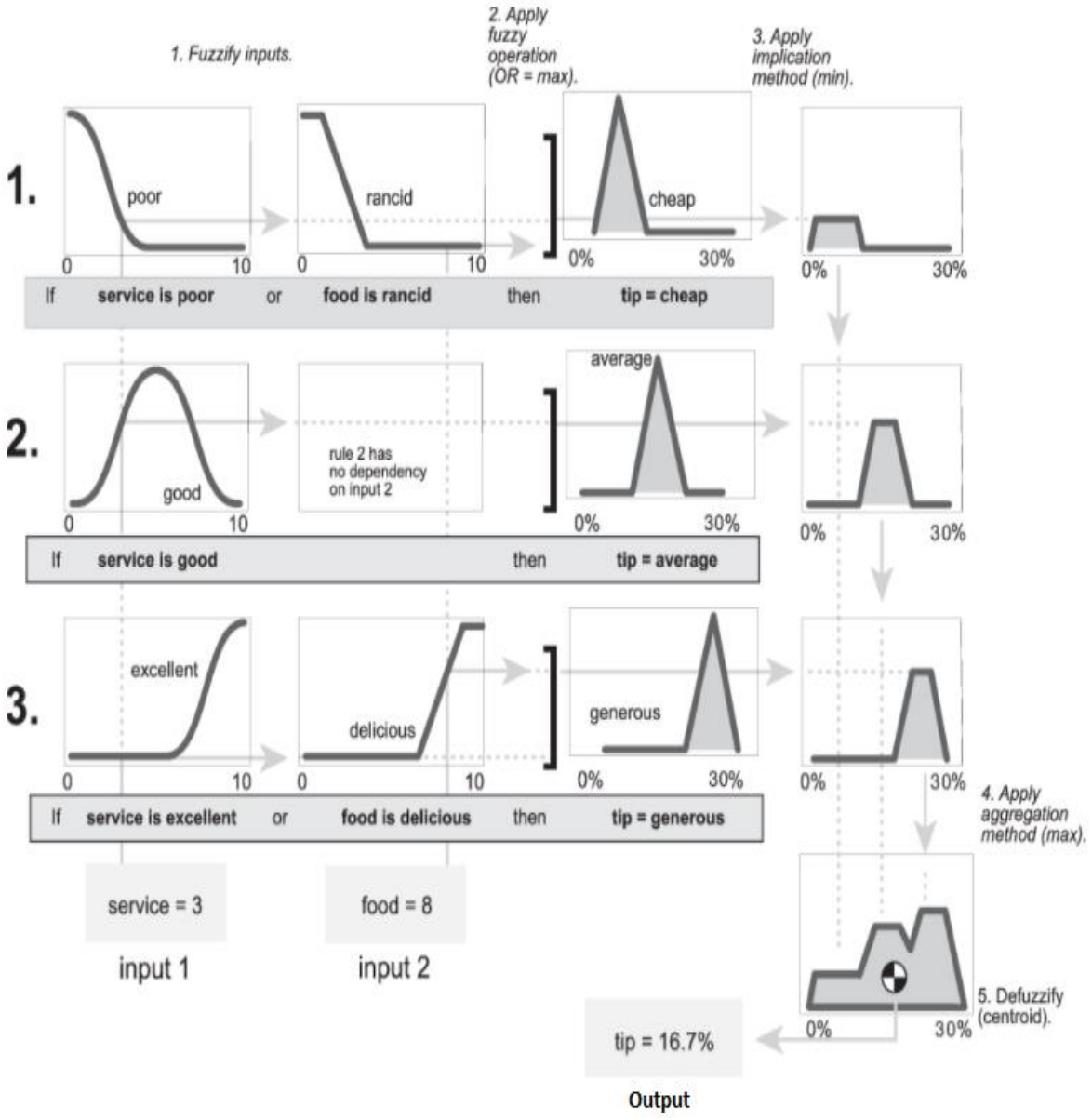


Figure (2. 22): Mamdani Implication of Tips Amount [54]

The rule bases hypothesize by combining various fuzzy propositions while the operators of AND, OR, and NOT participate in the combining process. The result of applying a fuzzy rule depends on three factors: the definition of fuzzy implication chosen, the definition of the membership function of the fuzzy set of the proposition located after the fuzzy rule, and the degree of validity of propositions found hypothesize [49].

2.14.4 The Defuzzification

Defuzzification is the inverse process of fuzzification in FLS. It maps the fuzzy results from a space of fuzzy control actions, defined over an output universe of discourse into a space of non-fuzzy (crisp) control actions. The process can generate a non-fuzzy control action that illustrates the possibility distribution of an inferred fuzzy control action. The defuzzification process considers as the rounding off the Fuzzification process, where a fuzzy set with a group of membership values on the unit interval may be decreased to a single scalar quantity. The reason behind this is the situations that occur when the outcome of the fuzzy process requires being a single scalar quantity as restrained to a fuzzy set. It is employed because, in many practical applications, a crisp control action is required. Unfortunately, there is no systematic procedure for choosing a defuzzification strategy. There is no scientific basis for any of them (i.e., no Defuzzifier has been derived from a first principle, such as maximization of fuzzy information or entropy). Consequently, defuzzification is an art rather than a science.

2.14.4.1 Defuzzification Methods

There are many methods used in defuzzification to transform the fuzzy set results into a crisp set. Some of them are the maximum Defuzzifier method (MDM), first of maximum method (FOM), last of the maximum method (LOM), mean of a maximum method (MOM), and center of gravity (COG) or centroid of area (COA) method.

In the maximum Defuzzifier method (MDM), the Defuzzifier examines the fuzzy set B and chooses as its output the value of Y* for which μ_B (Y) is a maximum. There are two types of MDM, as in Figure (2.23).

The maximum method (FOM) determines the smallest value of the domain with maximum membership value.

The last maximum method (LOM) defines the largest value of the domain with maximum membership value.

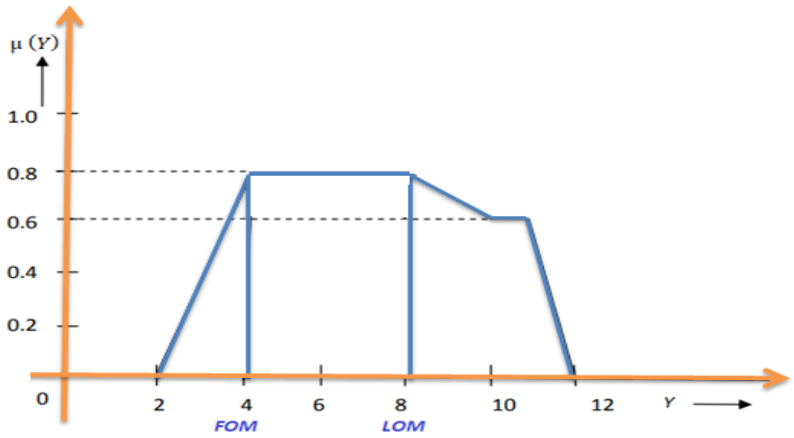


Figure (2. 23): First and Last Maximum Methods

In the mean of the maximum method (MOM): the Defuzzifier examine the fuzzy set B and first determines the values of Y for which $\mu_B(Y)$ is a maximum, then compute these values' mean as its output as in Figure (2.24). The Defuzzifier Y* is calculated as in equation (2.9):

$$Y^* = \sum \frac{Yi \in M^{Yi}}{|M|} \qquad (2.9)$$

Where M= {Yi | $\mu_B(Y)$ is equal to the height of the fuzzy set B} and |M| is the set M's cardinality.

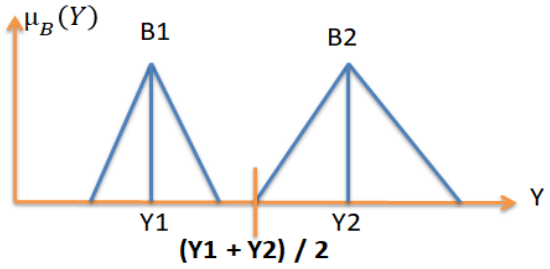


Figure (2. 24): Mean of Maximum Method (MOM)

The center of gravity (COG) / centroid of area (COA) method determine a crisp value based on the center of gravity of the fuzzy set. The membership function distribution's total area used to represent the combined control action divides into several sub-areas. The area and the center of gravity or centroid of each sub-area calculates, and then the summation of all these sub-areas is taken to find the defuzzified value for a discrete fuzzy set see Figure (2.25). The defuzzified value y^* for discrete membership function using COG is defined as in equation (2.10)

$$y^* = \frac{\sum_{i=1}^{n} yi \,.\mu\,(yi)}{\sum_{i=1}^{n} \mu\,(yi)} \qquad (2.10)$$

The defuzzified value y^* for continuous membership function using COG is defined as in equation (2.11):

$$y^* = \frac{\int \mu_A(y).\ y\ dx}{\int \mu_A(y).dx} \qquad (2.11)$$

Where, $\mu_A(y)$indicates the weightiness of the output membership functions, y means the center of each of output membership functions, and the Center of gravity (COG) means the crisp amount of the Defuzzifier output.

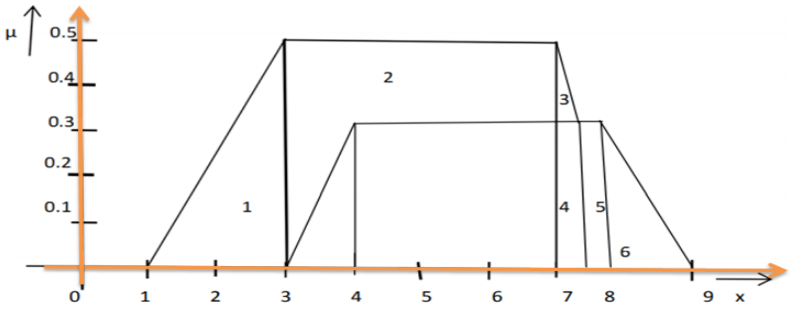


Figure (2. 25): Centroid of Area (COA) Method

Chapter Three

Related Work

Chapter Three: Related Work

3.1 Introduction

This chapter provides a classification to review the QoS-aware SSAs used in the IoT environment. The state-of-the-art research approaches can be classified into three main types, as shown in Figure (3.1). The classification criteria were obtained by analyzing the state-of-the-art methods. In order to develop a new method to solve the selection problem in the IoT environment, three main points should be considered. First, it is necessary to design an appropriate environment that allows the implementation of the proposed solution. Second, the proposed solution type is applied for selection in the QoS layer. Third, it is necessary to determine the methods used to evaluate the proposed solutions.

Thus, based on these three requirements for developing a new SSA, the proposed classification approach is described below. First, the design of SSAs for IoT must adhere to three basic concepts. SSAs are based on the process time phase, where the time when the SSA is performed is specified. Workflow management behavior represents how the services are controlled and connected. The optimization algorithm's objective is to establish the number of goals to satisfy when the algorithm is implemented. Second, it considers two basic concepts related to the implementation of SSAs. Based on the QoS layer, defining the layer where the SSA will be applied in the QoS architecture [40].

Moreover, the algorithm type can be divided into heuristics, meta-heuristics, hyper-heuristics, and other algorithms (non-heuristic algorithms). Third, the evaluation approaches, software, and performance management can be used to evaluate proposed algorithms' performance. Each of these sub-classifications is defined in Chapter 2.

Based on the definitions and concepts described in Chapter 2, the state-of-the-art methods are class.

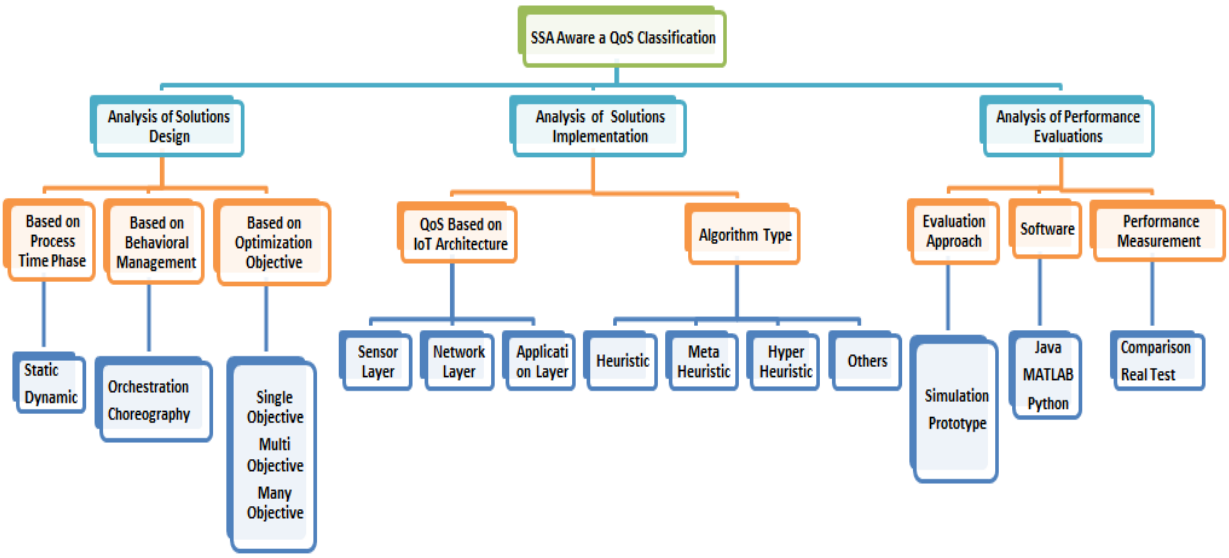


Figure (3. 1):Classification of QoS-aware SSAs in the IoT Environment.

3.2 Solution Designs Analysis

To design a new solution for the selection problem, researchers must first define an appropriate environment. Thus, this research reviewed the state-of-the-art methods and identified three common categories comprising SSAs based on the process time phase, behavioral workflow management, and optimization objective type. The methods analyzed according to these three categories are shown in Table (3.1).

3.3. Solution Implementations Analysis

Many methods have been proposed to solve the QoS-aware selection problem in the IoT environment. This research identified these solutions based on the three QoS layers introduced for the IoT architecture by Li. et al.[40], i.e., the sensor layer, network layer, and application layer. It classified the implementations of the methods for each layer according to four optimization algorithms comprising heuristics, meta-heuristics, hyper-heuristics, and non-heuristic algorithms. Moreover, it considered the traditional QoS factors optimized in each proposed solution.

Table (3. 1): Categorization of state-of-the-art methods according to the solution design

Reference	Process Time Phase		Behavioral Workflow Management		Optimization Objective Type	
	Dynamic	Static	Choreography	Orchestration	MOPs	MaOPs
Huang et al., 2014[5]	✓		✓			✓
Liu et al., 2013[11]	✓		✓			✓
Na et al., 2015[55]	✓		✓		✓	
Zhou et al., 2016[12]	✓		✓			✓
Sun et al., 2017[6]	✓		✓		✓	
Reddy et al., 2017[56]	✓			✓		✓
Mejri et al.,2017[8]	✓		✓		✓	
Yin et al., 2014[57]	✓		✓		✓	
Huang et al., 2015[14]	✓			✓	✓	
Dhondge et al.,2016 [7]	✓			✓	✓	
Anas et al., 2016[58]	✓		✓		✓	
Gao et al., 2014 [59]	✓		✓			✓
Abinaya et al., 2017[60]	✓		✓			✓
Nwe et al., 2014[61]	✓			✓	✓	
Jin et al., 2014,2016 [62], [63]	✓			✓		✓
Perera et al., 2014[10]	✓		✓			✓

Table (3.1) Continued

Categorization of state-of-the-art methods according to the solution design

Reference	Process Time Phase		Behavioral Workflow Management		Optimization Objective Type	
	Dynamic	Static	Choreography	Orchestration	MOPs	MaOPs
Quan et al., 2019[64]	✓		✓			✓
Yu et al., 2014[65]	✓		✓			✓
Huang et al., 2014 [66] [67]	✓		✓		✓	
Shukla et al., 2018 [68]	✓		✓			✓
Elhoseny et al.,2018[69]	✓		✓			✓
Alsaryrah et al., 2018 [70]	✓			✓	✓	
Huang et al., 2014[15]	✓		✓		✓	
Lin et al., 2017[44]	✓		✓			✓
Khanouche et al., 2016 [13]	✓	✓		✓	✓	
Yuan et al., 2019 [71]	✓		✓			✓
Hosseinzadeh et al. 2020 [72]	✓		✓	✓	✓	
Gao et al., 2020 [73]	✓	✓	-	-	✓	
Jatoth et al., 2019[74]	✓		✓		-	-
Khan et al., 2019[75]	✓			✓		✓
Abu-safe et al. 2019[76]	✓		✓			✓
Singh et al. 2020 [77]	✓		-	-		✓

3.3.1 Sensor Layer (SL)

Most researchers [57], [68], [70], [10], [44], and [13], who applied the selection algorithms in the sensor layer, didn't regard the feedback information or historical data in the implementation of their algorithms or models. The feedback information is the evaluation information collected from the end-users after they used a service to evaluate the services' performance from the specific QoS side.

There is only one paper [55] that took into account the historical information related to the service selection history of end-users requesting services.

3.3.1.1 Heuristic Algorithms in SL

Yin et al. [57] addressed the single-source many-target k shortest paths problem in Map-Reduce, where they used a graph of web services to find k shortest paths (selected services) from a candidate set of services for one source node while providing acceptable QoS. They focused on reducing the execution time and consuming less power. They proposed an efficient pruning algorithm for the breadth-first search (BFSKNN) called PruningBFSKNN algorithm. The proposed algorithm was built based on finding the shortest paths between nodes in a graph using DijkstraKNN and BFSKNN for the single-source many-target shortest path problem.

Shukla et al. [68] introduced a collocation-based strategy for hosting IoT services and applications in devices where they considered a smart home scenario. They aimed to find the correct sensor for the required services from a set of sensors while satisfying the QoS objectives by mapping flow-based process components on the system sensors. They identified three key factors comprising minimizing the latency time when collecting and transferring data from IoT devices and communicating it to the gateways, reducing the system's energy, and balancing the system energy requirements to increase its lifetime. They proposed a collocation-based sensor-service mapping strategy (CBSSMS) to link the IoT services to appropriate sensors.

The approach used two algorithms where one found the path that reduced the latency time, and a collection algorithm assembled the components that formed a link on the same sensor.

Alsaryrah et al. [70] aimed to select an appropriate set of smart objects by considering the traditional QoS and the energy consumed to form a service. They divided their objectives into minimizing the traditional QoS (execution time, network latency, and cost) and reducing energy consumption by the combined service, i.e., maximizing the sensors' battery lifetime. They proposed a bi-objective shortest path optimization (Bi-SPO) algorithm with four pruning techniques comprising pruning by the cycle, nadir point, efficient set, and label.

Perera et al. [10] investigated sensors' properties and the information associated with data streams to search, select, and large-scale rank sensors with the same functionalities and capabilities to satisfy the user's requirements. They considered the user preferences and sensor characteristics, such as accuracy, reliability, battery life, location, and other features, to identify appropriate sensors for data collection approaches. They designed and implemented an ontology-based context-aware sensor search, selection, and ranking model (CASSARAM). To improve the performance of CASSARAM, they proposed a comparative-priority-based weighted index.

They removed the sensors with a lower weighted context property based on the user preferences technique called Top-K selection. The comparative priority-based heuristic filtering was done t reduce the number of sensors ranked by removing sensors placed away from the user.

Aso, the relational expression-based filtering was done to speed up the sensor selection by specifying an acceptable range of context property values using relational operators in semantics. Three different methods were determined to search distributed sensors based on query/data transformation over the network, i.e., chain processing, parallel processing, and hybrid processing.

3.3.1.2 Meta-Heuristic Algorithms in SL

Lin et al. [44] introduced a sensor selection algorithm for specifying multiple sensor devices in a large-scale environment. They defined their optimization parameters for energy and distance, minimizing the energy consumed by communication between two sensor devices, balancing the energy among different sensors to reduce overloading on some sensor devices, maximizing the total energy harvested by supplementing the sensor's battery energy with natural energy (e.g., wind and solar), and green index optimization to reduce the whole pollution level. Satisfying the QoS involved optimizing the cost, reliability, and availability of IoT services. Their proposed algorithm based on many-objective evolutionary algorithm decomposition (MOEA/D) solved larger-scale problems by decomposing them into multiple sub-problems and then finding the optimized solution for each sub-problem.

Na et al. [55] conducted service selection for IoT based on physical resources by using platform-independent middleware. They focused on increasing the IoT system's lifetime by using the power on all devices equally to reduce energy consumption and costs. To balance the energy consumption, they developed an evolutionary game approach by defining a fixed point of the replicator dynamics where the payoffs are equal for all players in the same group. They also presented some options for improving service selection behavior. The initialization step was improved by estimating the remaining lifetime for all devices at the beginning instead of selecting them randomly. This approach may require a long time to find the optimal initialization solution, but it will save time in the following steps by maintaining communication with the other selection process. The decision-making step was improved by allowing the algorithm to select a longer estimated remaining lifetime, which could accelerate the algorithm and reject the optimal solution when selecting a service with a shorter lifetime.

3.3.1.3 Non-Heuristic Algorithms in SL

Khanouche et al. [13] aimed to solve MOPs during service selection by managing the energy consumption and maintaining the availability of services while slightly

reducing the QoS level but without affecting user satisfaction. They designed a QoS model by describing the QoS of an atomic service divided into quantitative attributes comprising the traditional QoS, such as the cost, response time, reputation, reliability, and availability, and qualitative attributes, such as security, privacy, and comfort. The QoS of a composite service is dependent on the structure of its atomic services connected through a sequential structure. The relative dominance of services conforms to the Pareto optimality set, which comprises the collection of possible solutions where at least one objective is optimized without affecting other goals. They proposed an energy-centered and QoS-aware service selection algorithm (EQSA) by effectively selecting a user-centered service from the most appropriate services that match the user's preferences while satisfying the specified QoS level. The proposed solution is executed in two main phases comprising the pre-selection of services that provide the required QoS level for the users and static selection before the runtime. The most appropriate services for SC are selected according to the relative dominance of the services.

3.3.2 Network Layer (NL)

Based on our review of the state-of-the-art papers there is no researcher applied a feedback information of historical data for a selection algorithms in network layer [67], [66], [65], [15], [14] , [7], [6], [56], [60], and [75].

3.3.2.1 Heuristic Algorithms in NL

Huang et al. [67], [66], [65], [15], and [14] presented a service merging approach that maps and co-locates neighboring virtual service on the same physical devices to reduce the communication energy costs and to balance the energy consumption by sensors to prolong the system lifetime. They applied WuKong middleware to automatically discover and manage smart sensors and actuator devices, which could support flexible and interoperable IoT systems by selecting from predefined flow-based programs (FBP) to find the appropriate mapping to the abstraction of an application onto physical smart devices and actuators according to the QoS

requirements. They proposed an energy sentient algorithm called the maximum weighted link (MWL) algorithm that treats selection as a two-co-locating problem and ignored the distance between devices [67]. They also updated their model [66] to consider the distance between devices because two remote devices require more energy for communication than closer devices. They treated the problem as a quadratic programming problem and proposed a reduction method to transform the problem into an integer linear programming (ILP) problem.

Huang et al. [65] presented a mapping model that considers the distance and runtime QoS requirements, such as the accuracy response time. Moreover, they attempted to reduce the total communication energy in IoT systems during each new update. They modeled the service matchmaking problem as a maximum weighted bipartite problem and solved it using the ILP model. Huang et al. [15] and [14] also used strategies for solving the maximum weighted independent set (MWIS) in their selection framework, which considered all possible co-location combinations for services. They implemented this method only in single-hop networks and treated the problem as a data clustering problem [15]. Heuristic algorithms were employed to find the maximum weight for the independent set, which comprised the fundamental decisions regarding service co-location. However, the method was subsequently implemented in a single-hop network and multi-hop network by modeling the problem as a quadratic programming problem and solving it with the ILP model[14].

Dhondge [7] presented a study of industrial IoT (IIoT) systems, where they focused on collecting and controlling communication data and parameters obtained from sensors on factory floors. They aimed to reduce and balance the energy consumption in the IoT by proposing a heuristic and opportunistic link selection algorithm (HOLA) to maintain the energy efficiency in the IoT sensors by opportunistically transporting the IoT sensor data to smart devices. These intelligent devices had multiple radio links (3G/4G LTE, Wi-Fi, and Bluetooth) to transmit the received data to the cloud by using HOLA to select the best radio link based on the quality preserved by the Services Level Agreements and the energy cost of the relationship.

3.3.2.2 Meta-Heuristic Algorithms in NL

Sun et al.[6] proposed a solution for integrating and co-operating with smart IoT functionalities to satisfy the user requirements, which could be applied to more than one smart thing. They considered the conflicts between the device's features and balancing its energy consumption. Spatial constraints were defined by the physical location and communication radius of a smart thing on the IoT. The temporal constraint was defined as the specific time duration required to meet a user's requirement because smart things should only be available for the predefined time duration. The energy efficiency was considered by balancing the energy load of intelligent devices or things and avoiding excessive consumption. Configurability of the IoT services occurred when two IoT services instantiated on the same smart device, and their functionalities conflicted. Thus, the composition and selection of the services demanded consideration of the delay between two sequential services by building a set of aggregated alternative smart things. The two main QoS factors involved reducing and balancing the energy consumption and prolonging the network lifetime. A two-tier framework was proposed, and three different meta-heuristic algorithms (ant colony optimization (ACO), GA, and PSO) were implemented to search for the optimal IoT SCs. The two-tier framework was configured, as shown in Figure (3.2). The IoT smart thing tier encapsulated the functionalities of devices in IoT services. The services class tier categorized IoT services into service class chains using traditional web service composition techniques. The service network was created between the two tiers by considering the possibility of calling between service classes.

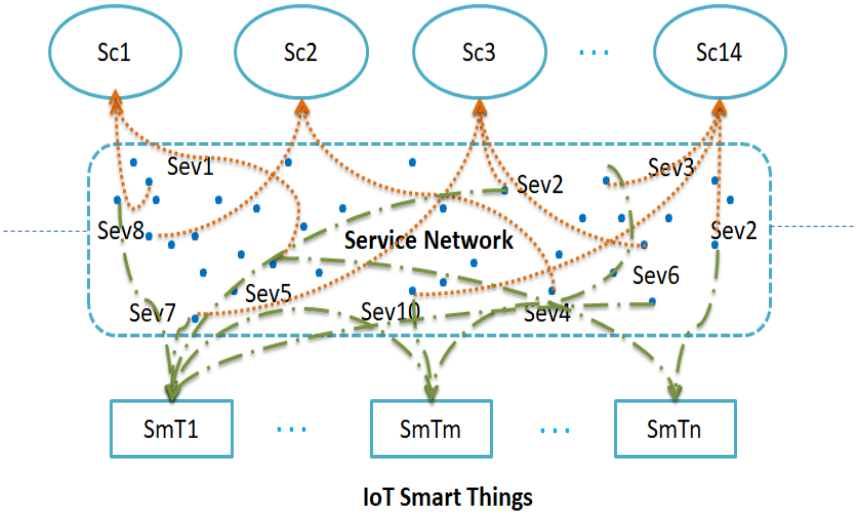


Figure (3. 2): Two-tier Framework [6].

Reddy et al.[56] proposed a clustering method for dividing a WSN-based IoT network into a small network, where each was called a cluster head. By using a meta-heuristic algorithm to optimize the network communication, the clustering method could divide the WSN into a small, reliable, and manageable network with efficient data transmission. The proposed algorithm used five parameters comprising the distance, energy, delay, network load, and temperature of the IoT devices. To efficiently select the cluster heads, a novel method was proposed by combining the gravitational search algorithm (GSA) with the artificial bee colony (ABC) algorithm.

Abinaya et al. [60] also aimed to minimize the energy, time, and loss of packets during the transfer of data among nodes when selecting and combining services. They sought to increase energy efficiency, time utilization, and throughput without any loss of data or reduction in the packet delivery ratio. They proposed a meta-heuristic algorithm (ACO) to find the shortest path between the nodes using the network routing protocol approach. The algorithm clustered the data nodes before then transferring data between the nodes with low power consumption.

3.3.2.3 Nov-Heuristic Algorithms in NL

Khan et al.[75] proposed a QoS-aware secured communication approach for IoT-based networks called QoS-IoT. The Sybil attack detection mechanism was used for identifying Sybil nodes during multi-hop communication. To ensure the fair and efficient utilization of the available bandwidth, an optimal contention window (CW) was selected for QoS provisioning. The optimal CW size was chosen by using the binary exponential back-off mechanism. The performance of QoS-IoT was evaluated based on measurements of Sybil attack detection, fairness, throughput, and buffer utilization.

3.3.3 Application Layer (AL)

Most authors that applied SSA algorithms in an application layer didn't regard feedback information, as [11], [59], [58], [5], [9], [74], [69], [71], and [72].

Few of them regarded feedback information or historical data in their implementations of SSA, such as Nwe et al. [61] kept the histories of end-user feedback in the UDDI repository with documents containing the QoS information.

Zhou and Yao [12] regarded the feedback information of composite CMfg services.

Mejri et al. [8]] proposed SSA based on end-user feedback. They classified a historical date or feedback information into two categorized: Centralized systems have the responsibility of gathering and recording QoS information from consumers. The centralized systems rely on a single entity to manage historical data like collecting, calculating, and updating all participating parties' scores. However, the centralized systems have some limitations like the single point of failure and the bottleneck problems.

Decentralized systems are characterized by the absence of the central authority node, unlike centralized ones. So, to control the historical data, all members must cooperate and communicate with each other. They are more scalable than centralized ones with a substantial gain of bandwidth. The decentralized systems are more complicated to design and implement than centralized ones. However, when a

failure occurs in the system, stored data are always accessible and retrievable from any node.

Authors in [64] took into accounts the subjective assessments from the latest end-users feedback and objective assessments of service. Also, to reduce the effect of unreasonable feedback from end-users in different contexts, users' similarity is considered in their solution.

3.3.3.1 Heuristic Algorithms in AL

Nwe et al. [61] introduced a matching, ranking, and selection model to satisfy the distributed needs of dynamic networks in IoT environments. They selected services based on two factors to optimize the QoS, i.e., objective information supplied by the service providers and the service consumers' subjective data. To select a service, they proposed a flexible QoS-based service selection algorithm (FQSA). They calculated the subjective factors for the user with a similarity aggregation method (SAM)) to evaluate the creditability of different users.

Moreover, the user's input was extracted using the QoS ontology, WordNet, and ontological reasoning. To help understand the QoS characteristics, they analyzed a separate language glossary and evaluated the consistency among the end-users' QoS criteria. The FQSA algorithm employed an artificial neural network back-propagation algorithm (ANN-BP) to find the objective factors and improve the selection performance rate for acceptable real-time service selection. They also provided a flexible, user-friendly assessment form to allow users to request any number of QoS criteria.

Mejri et al. [8] investigated the scalability of service selection in the IoT by using a self-adaptive approach based on a combination of a QoS prediction model and the technique for preference by similarity to ideal solution (TOPSIS) model. The prediction model was considered the user context, service context, and network context by using the ANN. They optimized two QoS parameters comprising the response time and reliability.

Quan et al. [64] introduced a reinforcement learning approach called the linear reward inaction (LRI) algorithm in real-time. They considered the dynamic IoT environment by calculating the user's mobility, which could affect the accessibility and connection location of services, reducing the search space for service discovery. The latest subjective assessment obtained from user feedback concerning the user context similarity was used to estimate the QoS in a similar environment. A subjective evaluation was conducted by calculating four factors: privacy, reliability, availability, and response time. Three objective attributes comprising the availability, response time, and calculation speed were determined by obtaining a score for each service. The service with the highest score was selected.

3.3.3.2 Meta-Heuristic Algorithms in AL

Liu et al. [11] designed a cooperative evolution algorithm (CEA) for service composition and selection to solve MOPs when selecting an optimal service from a group of services with similar functionalities and diverse QoS requirements. They aimed to develop an efficient and robust approach by considering non-functional attributes comprising the cost, time, availability, and reliability. A heuristic optimization approach was developed by integrating GA and PSO in CEA. Their approach was characterized by improving the best local first strategy to select a service candidate, enhancing the global best policy, and fitting the learning rate's self-adaptive mechanism.

Gao et al. [59] conducted global optimization for event SCs by using a meta-heuristic method based on GA but without the need to consider all possible combinations. The non-functional attributes were represented by QoS properties for latency, price, energy consumption, bandwidth consumption, availability, completeness, accuracy, and security.

In particular, they provided a QoS aggregation schema for complex event service (QoS-AS for CES) composition in CES networks by treating complex event processing as reusable services where reusability was determined by examining

intricate event patterns and primitive event types. The abstract architecture of these complex networks is shown in Figure (3.3).

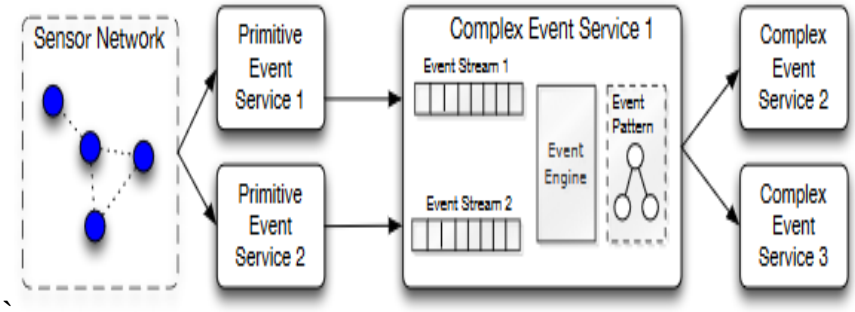


Figure (3. 3): Architecture of CES Networks[59].

A GA was also developed to efficiently create optimal event SCs with the same standard GA steps (select, crossover, and mutate until the termination conditions are satisfied) but some differences from the standard implementation. An event reusability forest maintained a tree encoding schema.

Anas et al. [58] aimed to simulate human thinking when deciding about multiple choices and using data collected from IoT sensors. They considered an example of a human deciding while driving when faced with two paths that lead to the same place but with different trade-offs regarding the time, distance, or cost. Their framework collected data to help systems to select their future path. The main problem was how to capture and use human heuristic information. The final solution reduced the total time and obtained more accurate results. They used the heuristic-IoT framework for enhancing heuristic search algorithms and collecting data from IoT sensors. They implemented their framework with a GA using data regarding drivers' habits and behavior collected from sensors deployed in taxis to solve the traveling salesman problem (TSP) with hidden edge costs. The proposed framework used heuristic information to generate smarter initial solutions for the GA to solve the TSP instead of generating it randomly.

On the cloud manufacturing side, Huang et al. [5] focused on solving the MOPs for cloud SC optimal selection (CSCOS) while considering non-functional QoS factors. They determined four parameters comprising the cost, execution time, energy consumption, and reliability. Non-functional attributes were evaluated for three cloud services: manufacturing software, hardware, and human resource services. They introduced a new chaos control optimal algorithm (CCOA) to solve the CSCOS problem in large-scale solution spaces.

Li et al. [9] focused on a cloud logistics platform based on IoT and cloud computing environments to study a logistics center by considering service encapsulation and resource virtualization. They defined the logistics center's primary requirement in terms of service selection to find the best essential web services rather than the best combination of abstract web services. Non-functional constraints were used to compute the QoS for composite services by applying Canfora [9] for an aggregation function and four QoS parameters comprising the time, cost, availability, and reliability. Besides, a dynamic service selection model was proposed based on PSO.

Abu-Safe et al. [76] proposed a service selection model that ranked services based on end-user feedback and reputation value. The Likert scale was employed as a user-friendly method for acquiring feedback from end-users. An improved-PSO was used to select the optimal service from ranked services. Two quality groups used to calculate the QoS factors comprised the business quality group (BQG), i.e., reputation and execution price, and the system quality group (SQG), i.e., reliability, availability, and response time.

Jatoth et al.[74] introduced a meta-heuristic model using an adaptive genotype evolution-based GA (AGEGA). They balanced the QoS parameters and connectivity constraints to perform SC in a cloud environment. The discrete uniform rank distribution (DURD) and discrete uniform service rank distribution (DUSRD) were proposed to determine the service fitness and SC fitness, respectively, thereby allowing services to be pruned from the non-optimal solutions and reduce the search

space. The specific QoS parameters employed were not defined. However, they used a synthetic data set with QoS parameters such as accessibility, cost, availability, throughput, response time, security, integrity, and reliability.

3.3.3.3 Hyper-Heuristic Algorithms in AL

Elhosenya et al. [69] considered health services applications where they proposed a new cloud-IoT based model for efficiently managing large amounts of data in an integrated industry (4.0) environment. They aimed to satisfy five factors, reducing the medical requests time, optimizing the storage space for patient data, improving task scheduling, providing a real-time data retrieval mechanism for health care applications, and maximizing the utilization of resources. They proposed a new model for optimizing virtual machine selection using three optimization algorithms (GA, PSO, and Parallel PSO (PPSO)) to build the proposed model.

Zhou and Yao [12] focused on cloud manufacturing by introducing a solution for composited cloud manufacturing service optimal selection (CCSOS) under multi-objective using four QoS parameters comprising time, cost, availability, and reliability. They introduced the hybrid ABC (HABC) algorithm for CCSOS problems with three main steps. First, the HABC was initialized before outputting feasible solutions at each iteration, and these solutions were ordered from large to small settlements. The onlooker strategy was improved by updating all the solutions with the chaos algorithm, which had irregular properties in all states, and it could help the worst bees. The bee colony's search space was searched more efficiently based on knowledge of the problem structure and social colony information by updating small solutions with Archimedean copula estimation of distribution.

Yuan et al. [71] proposed a dynamic approach using a fuzzy logic technique and cultural GA to adopt global QoS constraints. The global QoS constraints were decomposed into near-optimal local QoS constraints before independently selecting a service component for each abstract service. They aimed to satisfy five QoS

factors comprising the price, response time, availability, throughput, and successful execution rate.

Hosseinzadeh et al.[72] combined a machine learning method with a meta-heuristic algorithm in a hybrid ANN-PSO algorithm. They aimed to improve the execution time and reachability rate for a service selection model in cloud-edge computing. A labeled transition system was proposed based on a verification approach to check the proposed model's correctness. Three QoS factors were considered comprising the response time, availability, and prices.

3.3.4 Aggregate Layers (GL)

Some solutions aggregated more than one layer together, such as in the sensor and application layers [62] and [63].

Jin et al. [62] considered the services provided by IoT devices and designed a physical service model (PSM) to describe various physical IoT services and a method for selecting a candidate physical service that satisfies a user's requirements. The PSM model included three main components (devices, resources, and services), and the relationships between them were defined. The following four QoS properties were determined based on the features of physical services: the available time when environmental services may be accessible, the service area comprising descriptions of physical services for on-device resources that contain information about an entity, the processing time representing the computational time capacity of IoT devices, and the reputation calculated for a service to help users decide whether to use a service-based depending on the service ratings given by different users (equipment or service) after requesting services. A physical service selection (PSS) algorithm was proposed in terms of Spatio-temporal features to rate candidate physical services according to user preferences based on individual QoS rating functions. The PSS algorithm comprised pre-sorting, filtering, and final sorting phases.

In 2016, Jin et al. [63] improved their PSM to dynamically rate QoS values and select a physical service based on the user's preference. They added the following

three types of QoS attributes to reflect the features of physical services: spatial-temporal attributes related to the problems that affect the mobility and availability of physical services due to network, energy-saving, or privacy issues, i.e., available time and service area; positive features that preferably have higher values, i.e., reputation and reliability; and negative attributes that preferably have lower values, i.e., processing time and execution cost.

Gao et al. [73] aggregated the application layer and network layer to provide recommendations for services then select the services. They proposed a holistic framework for predicting the QoS values in the IoT. The proposed framework employs fuzzy C-means to cluster contextual information related to users and services, such as the network location and geographic location, and neural collaborative filtering (NCF) is applied as a neural network model to learn the in-depth latent features. NCF utilizes local features such as similar users or similar services with historical QoS values and global elements comprising user latent vectors and latent service vectors. NCF then combines the contextual information with the historical QoS values to perform both the prediction and selection processes.

Jatoth et al.[74] also aggregated the application layer and network layer with a meta-heuristic model using AGEGA, where they balanced the QoS parameters and connectivity constraints to perform SC in a cloud environment. DURD and DUSRD were applied to determine the service fitness and the SC fitness, thereby allowing services to be pruned from the non-optimal solutions to reduce the search space. However, they did not define the specific QoS parameters used in their study, although they described using a synthetic data set containing QoS parameters, such as accessibility, cost, availability, throughput, response time, security, integrity, and reliability.

Singh et al. [77] introduced a framework based on multi-criteria decision-making to direct the selection process. The framework aggregated the sensor, network, and application layers, and they combined two multi-criteria decision-making methods

comprising the analytic hierarchy process (AHP) and TOPSIS. AHP was used to calculate the weights for the QoS criteria, and TOPSIS ranked the service providers. They describe a QoS parameter based on three IoT components, i.e., things, communication entity, and computing entity. Nine QoS parameters were considered: operating temperature range, resolution, accuracy, delay, jitter, pricing, availability, throughput, and response time.

In this layer, the authors applied their selection algorithms in more than one layer; based on our research, most of the researchers didn't consider the feedback information in their implementation of SSA, such as [62],[63], [74], and [77]. However, only one proposed solution in an aggregate layer considered the feedback information; Gao et al. [73] employed the historical QoS information of similar users or similar services for help in the prediction process.

3.4 Analysis of Implementing SSAs

Abbreviations: AppL: application layer; SenL: sensor layer; NwL: network layer; Heu: heuristic; M-Heu: Meta-heuristic; H-Heu: Hyper-Heuristic; Non-Heu: Non-heuristic.

Table (3. 2):

Categorization of selected state-of-the-art methods based on the proposed solution, QoS layer, algorithm type, and QoS parameters

Proposed Solution	QoS Layer				Algorithm Type			QoS Parameters
	AppL	SenL	NWL	Heu	M-Heu	H-Heu	N-Heu	
New CCOA to solve CSCOS [5]	✓				✓			Cost Execution time Energy consumption Reliability
Integrating GA and PSO algorithms [11]	✓				✓			Cost Time Availability Reliability
Evolutionary game approach with platform-independent middleware [55]		✓			✓			Save energy Cost
HABC algorithm for CCSOS [12]	✓					✓		Time Cost Availability Reliability
A novel method that combines GSA and ABC [56]			✓		✓			Reliable and manageable network Efficient data transmission
Self-adaptive approach including ANN and TOPSIS models [8]	✓			✓				Response Time Reliability
DijkstraKNN, BFSKNN, and PruningBFSKNN Algorithms [57]		✓		✓				Reduce execution time Consume less power
HOLA in IIoT systems [7]			✓	✓				Reduce energy consumption Balance energy consumption across the IoT network
Heuristic IoT framework based on GA [58]	✓				✓			Reduce the total time Obtain more accurate results

Table (3.2) Continued

Proposed Solution	QoS Layer			Algorithm Type				QoS Parameters
	AppL	SenL	NWL	Heu	M-Heu	H-Heu	N-Heu	
ACO [60]			✓		✓			Increase: Energy efficiency Time utilization Throughput Packet delivery ratio
FQSA comprising ANN-BP and SAM [61]	✓			✓				Specific QoS not defined
PSM model and PSS algorithm [63][62]	✓	✓					✓	In[62]: Available time Service area Processing time Reputation In[63]: Spatial-temporal attributes Positive attributes Negative attributes
CASSARAM [10]		✓		✓				Accuracy Reliability Battery life Location
Dynamic PSO model [9]	✓				✓			Response time Cost Availability Reliability
Fuzzy logic and cultural GA [71]	✓					✓		Price Response time Availability Throughput Execution rate
LRI [64]	✓				✓			Subjective: Privacy Reliability Availability Response Time Objective: Availability Response time

	Speed

Table (3.2) Continued

Proposed Solution	QoS Layer			Algorithm Type				QoS Parameters
	AppL	SenL	NWL	Heu	M-Heu	H-Heu	N-Heu	
Fuzzy C-means and NCF [73]	✓		✓				✓	Response time Throughput
Hybrid ANN-PSO[72]	✓					✓		Response time Availability Prices
QoS-IoT[75]			✓				✓	Sybil attack detection Fairness Throughput Buffer utilization
Likert-Improved-PSO [76]	✓				✓			BQG: Reputation Execution price SQG Reliability Availability Response time
AHP- TOPSIS [77]	✓	✓	✓		✓			Temperature range Resolution Accuracy Delay Jitter Pricing Availability Throughput Response time

3.4.1 Simulation to Evaluate SSAs

Huang et al. [5] demonstrated the high performance of their proposed CCOA algorithm in CSCOS based on simulations. They found that their algorithm was better at searching large-scale solution spaces than GA and typical chaotic GA, where it reduced the time required and energy consumption. They recommended improving the effectiveness of CCOA to solve other combinatorial optimization problems by balancing the search capacity and time consumption and tested the effects of other QoS factors.

Liu et al. [11] developed CEA by integrating GA and PSO. They conducted simulations and generated a data set at different scales based on real scenarios. They showed that CEA was a highly efficient search approach with more excellent stability and more rapid convergence compared with canonical PSO (CPSO) and the improved discrete immune algorithm based on CPSO (IDIPSO).

Na et al. [55] showed that their method decreased the time required for implementation and increased service utilization rate. When the service utilization reached 100%, the algorithm could not make any changes after initialization. They recommended focusing on group-based service selection to reduce the communication energy requirements**.**

Zhou et al.[12] evaluated the performance of HABC based on comparisons with GA, PSO, and the basic ABC algorithm using 15 different end-users QoS preferences. They randomly generated the data set, and the results showed that HABC exhibited a high search capacity and stability with acceptable time complexity. They recommended studying the performance of HABC in detail according to the characteristics of the cloud manufacturing environment and integrating HABC with other heuristic algorithms.

Reddy et al. [56] assessed the GSA and ABC algorithm's performance based on the trends in the network sustainability of live IoT nodes in the network and by evaluating its convergence compared with PSO, GA, ABC, and GSO. The results demonstrated that their approach performed better than the other cluster head selection methods for IoT devices.

Yin et al. [57] showed that the pruning algorithm was more efficient than the breadth-first search shortest path algorithm. Furthermore, the DijkstraKNN algorithm was suitable for small shortest paths, but the PruningBFS algorithm was better when the candidate set was small, and the shortest paths were significant. The execution times were more stable for the BFSKNN and PruningBFSKNN algorithms according to tests using two real-world data sets comprising the Epinions Social network and LiveJournal social network. They recommended identifying

approximation algorithms that can handle more significant graphs where time is required to compute several nearest neighbors.

Anas et al. [58] used the T-drive data set based on 10,357 taxi drivers and compared the proposed Heuristic-IoT framework with GA. They found that the TSP results improved by up to 49% compared with the traditional TSP.

Abinaya et al. [60] compared the ACO algorithm with load balancing clustering for data clustering. They also reached the number of nodes versus the data throughput for time-efficiency prediction, where ACO reduced the time and energy required. ACO could find an almost optimal solution, and it could be scaled to large-scale IoT environments.

Nwe et al. compared FQSA and other SSAs including (genetic algorithm and fuzzy logic-based service selection algorithm (GAFLSS), agent proxy on user preference approach (APUP), and fuzzy linear programming approach (FLP)) [61], from the reliability of the trust mechanism-based service selection algorithm. They calculated the symmetric mean of each system's recall and precision using the frequency for various evaluation metrics (QoS aggregation, QoS reasoning, QoS scalability, personalized confidentiality, and user-friendliness). The results showed that FQSA improved service selection performance, user satisfaction level, and user-friendliness rates. However, many computations were required to select the services, which was time-consuming.

Jin et al. [62] [63] evaluated the PSS method against the skyline-based algorithm called the one-pass algorithm (OPA) in terms of the execution time and user preferences for physical services. They used a random data set based on the Climatography of the United States Number 81 series (CLIM8144) data set. The results showed that PSS was efficient with a large number of candidate physical services [62]. It performed better than OPA in the filtering step by reducing the number of CPUs to improve the selection performance [63]. For future research, they suggested creating and implementing an IoT service platform that allows users to register their devices, discover and select required physical services, address

privacy and security issues, and reduce the search space. They suggested the use of pruning methods and heuristic techniques.

Huang et al. [67], [66], [65], [15], [14] conducted simulations to assess the performance of their algorithm. Their algorithm reduced the total communication energy by about 20% in IoT systems [67], [66]. They compared the performance of greedy matching and the ILP solution at service matching and found that the ILP solution was optimal, but it required more time, and it might not be scalable to large-scale IoT systems [65]. In addition, they compared ILP [15], [14] and the MWIS framework with MWL [67], [66], [65], as well as with other selection strategies(GWMAX, GWMIN, and GWMIN2)[15]. The results were improved [15], and the total communication energy was reduced by 10% compared with other methods [67]. They also implemented MWIS in a multi-hop network [14], and the total communication energy was reduced by more than 10% [15]. They planned to develop heuristic algorithms, test them with Adriano-based devices, and study more complex applications by checking the automatic configuration module to support more users interacting with IoT systems.

Shukla et al. [68] presented CBSSMS to link IoT services with appropriate sensors. They conducted comparisons with existing collocation distance algorithms based on the random mapping of services on any IoT device, where they tested linear, random, and star FBP networks. The results showed that the CBSSMS algorithm reduced the latency and energy consumption between devices compared with the collocation distance algorithms [66].

Elhosenya et al. [69] conducted a comparative study based on the execution time, system efficiency, and data processing speed. They evaluated the effectiveness of their model against GA, PSO, and PPSO. The results showed that the proposed model improved the total implementation time by 50%. Moreover, the efficiency of the system at real-time data recovery improved significantly by 5.2%.

Alsaryrah et al. [70] evaluated the Bi-SPO algorithm against QoSC, which only considers the QoS, and EPC, which only assumes the energy profile. The results

showed that Bi-SPO achieved the ideal balance between the traditional QoS level and energy consumed, and it performed better than the other algorithms.

Lin et al. evaluated their MOEA/D approach based on a sensor selection problem [44]. They showed that increasing the problem size led to increased energy consumption, energy balancing, energy harvesting services, and pollution level, but it did not affect the QoS. They generated their data set.

Khanouche et al. [13] assessed the performance of EQSA by simulating the A2NEts European project scenario, which involves monitoring and smart metering for buildings. They synthetically generated data sets based on QoS factors and realistic energy models to specify IoT services' energy profiles. The simulation results demonstrated the efficient performance of EQSA in terms of energy efficiency, selection time, composition lifetime, and optimality of the solution.

Li et al. [9] simulated their PSO method and showed that it was more efficient than GAs. PSO optimized different fitness parameters and maximized the availability or reliability while maintaining a low cost and response time. They considered a real-world scenario involving the transport of furniture among countries by combining five web services related to shipping cargo services. They applied their method to a previously reported data set (Mao data set) of QoS values. The feasibility of applying PSO was confirmed by implementing the simulation program in Java. They recommended further research into logistics and web service selection, improving the efficiency of SSA based on PSO, comparing PSO with other algorithms, and developing methods to confirm the consistency of QoS between service consumers and service providers.

Yuan et al. [71] evaluated the performance of a fuzzy logic technique and cultural GA by comparing it with a QoS constraints decomposition (QCD)approach based on cultural GA and an integer programming(IP)-based method. The experimental results showed that using the fuzzy logic technique and cultural GA was appropriate in terms of the adaptability and scalability to the environment and satisfying the user preferences and increasing the number of candidate services. They used the QWS

data set, which contains 2508 real web services with 10 QoS attributes factors[78]. Also, they randomly generated other data set according to QWS (RQWS) using the Eclipse programming tool. They recommended increasing the number of fuzzy sets and formulating more appropriate fuzzy rules before applying their approach in a distributed environment where a group of distributed QoS registries maintains the QoS values.

The hybrid ANN-PSO algorithm obtained better fitness values than PSO, GA, and PSOGA [72]. They evaluated their method based on simulations using the C# language as an integrated development environment (IDE). The PAT model checker was employed to prove the correctness of the proposed algorithm. They employed QWS data set containing 2500 web services. They recommended using deep learning methods to avoid the space explosion problem in the SC model.

Satisfactory experiments were conducted based on real-world WS-Dream data sets by Gao et al. [73]. The prediction performance was evaluated using the root mean squared error and mean absolute error. Also, the experimental results verified the effectiveness of the introduced frameworks; NCF and context-aware NCF (CNCF). They compared with the well-known QoS prediction methods, comprising user-based PCC (UPCC), item-based PCC (IPCC), web service recommender (WSRec), and location-based factorization machine (LBFM). They recommended implementing work-based models in the QoS prediction task, such as a recurrent neural network and convolutional neural network, and studying the time factor during QoS prediction.

The dynamic LRI model was compared with another based on user feedback [64]. The results showed that the LRI model improved the effectiveness in a real-time scenario because it considers the similarity between users, although the time consumption was higher. The data set and scenario were generated in the study. For future research, they recommended applying a user-centric service management system based on the user's preferences in the IoT environment.

Jatoth et al. [74] compared the performance of AGEGA with other methods based on GA, i.e., GA, orthogonal GA (OGA), adaptive genetic programming (AGP), and transactional GA (TGA). The experimental results showed that AGEGA obtained better fitness values within a lower execution time. They used QWS as the data set of QoS parameters and randomly generated some of the QoS parameters and their corresponding values. They recommended considering multiple service connectivity constraints and multiple QoS parameters in future research, as well as developing an efficient approach for various parallel data processing platforms.

Khan et al. [75] simulated QoS-IoT and compared FIFO, round-robin (RR), and cross-layer scheduling based on the utilization of the CW via adaptation using the network simulator NS-2. They simulated IoT-based networks that covered a city of 100×100 km^2 using a system model produced with the proposed approach. The total area was divided into smaller IoT-based networks, where each network comprised Sybil, mobile, static, and high power nodes. The simulation results showed that QoS-IoT was resilient against the Sybil attack, as well as improving network performance with a large amount of data. They recommended studying the effect of Sybil node detection-aware QoS on the Internet of Vehicles and flying ad hoc networks.

Abu-Safe et al. [76] simulated their Likert-Improved-PSO model and evaluated its performance based on comparison with the original PSO and Improved-PSO. The proposed model had a lower execution time, and it obtained better fitness value. For future research, they recommended testing more QoS parameters and combining with more than one meta-heuristic algorithm with respect to the end-user feedback

Singh et al. [77] applied their AHP–TOPSIS framework and existing AHP–AHP framework to a health care case study and compared the results based on the execution time for the selected value. AHP–TOPSIS required a lower execution time than AHP–AHP to obtain the same selection value. They assembled real data sets from three different providers but did not describe them. The robustness of the

proposed framework was measured, but the sensitivity toward changes in the user or decision-maker was not analyzed.

3.4.2 Prototypes to Evaluate SSAs

Sun et al. [6] evaluated three heuristic algorithms (ACO, GA, and PSO) by building prototypes to calculate the fitness, minimum, and difference in the residual energy for smart devices. The results showed that PSO performed better than GA and ACO at the optimization problem.

Mejri et al. [8] developed a parallel implementation of the ANN model and TOPSIS model to evaluate the scalability of SSA in the IoT. They used the mean absolute error to measure the quality of the prediction model. As the number of services increased, the mean absolute error decreased, and the accuracy increased, but there was no significant increase in the execution time. They recommended using an evolutionary technique and pruning methods. A limitation was that the proposed approach was applied to a training set built during search steps that did not meet all Internet requirements. In addition, they only considered the response time and reliability as QoS factors in their study.

Gao et al. [59] proposed QoS-AS for CES and GA, which they compared with a brute-force enumeration algorithm in terms of the execution time and optimization degree, where the proposed algorithm improved the optimized results from 79% to 97%. The performance of CASSARAM was also evaluated based on the change in the storage requirements according to the sensor data descriptions, the requirements for sensor selection and indexing, the memory required to select sensors, and the change in the accuracy rate. They evaluated the processing time and memory requirements based on sensor selection and relational expressions during the semantic querying phase. The data sets employed were from the Linked Sensor Middleware project. They showed that CASSARAM could reduce the processing time and minimize storage requirements. In the future, they plan to merge their algorithm with leading IoT middleware solutions such as SenseMA and Open-IoT to

improve the automated sensor selection functionality in the IoT environment. They also recommended enhancing the efficiency of CASSARAM to integrate automated machine learning techniques using cluster-based sensor search and heuristic algorithms.

3.4.3 Simulations and Prototypes to Evaluate SSAs

Dhondge et al. [7] validated HOLA based on simulation studies and designed the HOLA IoT sensor prototype with Adriano. In practical experiments, they measured the energy consumption of the HOLA IoT sensors in different operational scenarios and communication settings. In particular, they compared the energy consumption of HOLA and the Vanilla System and showed that HOLA could reduce the energy consumed in IoT sensors by reducing the internal communication in the IoT device. The time consumption with HOLA was better compared with the Vanilla System. They recommended detecting the maximal energy efficiency that satisfies the SLA agreement and evaluating HOLA using different smartphone densities in the future.

3.5 Analysis Performance Evaluation for SSAs

Table (3. 3):
Categorization of state-of-the-art algorithms according to evaluation methods and performance measurement based on prototypes.

Ref	Software used	Performance measurement	Data set	Results
[6]	Java program, with Intel i7-6700 CPU, 8-GB of memory, and 64-bit Windows 7	Compared with ACO, GA, and PSO	Generated their own data set	PSO performed better than GA and ACO
[8]	Java program			Increased the accuracy, but with no significant increase in execution time

Table (3.3) Continued

Ref	Software used	Performance measurement	Data set	Results
[10]	Java on a computer with an Intel(R) Core i5-2557M, 1.70 GHz CPU and 4 GB RAM	Comparison based on execution time, memory required, and accuracy	Linked Sensor Middleware project	Reduced processing time and minimized storage requirements
[59]	Java using MacBook Pro with 2.53 GHz duo core CPU and 4 GB 1067 MHz memory	Compared with a brute-force enumeration algorithm	Built their own data set	Improved execution time and degree of optimization, and optimized results from 79% to 97%

Table (3. 4):

Categorization of state-of-the-art algorithms according to evaluation methods and performance measurement based on simulations.

Ref.	Software used	Performance measurement	Data set	Results
[57]	Java on the Hadoop platform using Intel Core 2 Duo CPU and 1GB of RAM, running CentOS v6.0	Compared DijkstraKNN, BFSKNN, and PruningBFSKNN algorithms	Epinions Social network LiveJournal social network	BFSKNN and PruningBFSKNN algorithms obtained more stable execution times
[60]	Java using Intel Core, Window 32-bit system	Compared ACO with load balancing clustering	Not stated	Reducing the time and energy required, and more efficient for large scale IoT
[68]	MATLAB R2017b i3-5005U CPU @ 2.00 GHz (4 CPUs) and 4 GB RAM	Compared CBSSMS with algorithms at randomly mapping services on devices	Generated their own data set	Reduced latency and energy balance between devices

Table (3.4) Continued

Ref.	Software used	Performance measurement	Data set	Results
[69]	MATLAB + CloudSim package	Compared hyper-model with GA, PSO, PPSO	Generated their own data set	Enhanced implementation time and the system's efficiency in real-time
[61]	Not stated	Compared FQSA with GAFLSS, APUP, and FLP	Random QoS data sets generated from [0,1]	Improved selection performance and user satisfaction level
[67] [66] [65]	Not stated	Compare the performance of three models	Generated their own data set by randomly generating services dataset	Reduced communication energy consumption and increased system lifetime
[15] [14]	Not stated	Compared MWIS with MWL, GWMAX, and GWMIN	Generated their own data set	Reduced communication energy and increased system lifetime
[44]	C++ programming language run on a PC with Intel Core i7-6700 CPU and 8 GB memory	Compared with sensor selection algorithms	Generated their won data set	Ran more efficiently
[7]	Python with NetworkX, SciPy, and NumPy libraries	Compared HOLA system with Vanilla System	Not stated	Reduced energy consumption by IoT sensors by reducing the internal communication in IoT devices, as well as reducing the time required

Table (3.4) Continued

Ref.	Software used	Performance measurement	Data set	Results
[73]	-	Compared NCF and CNCF against QoS prediction methods (UPCC, IPCC, WSRec, and LBFM)	WS-Dream	Superior prediction performance demonstrated
[64]	Mac-OS 10.14.3 within an Intel i5-7500U 2.30 GHz CPU, 8 GB RAM	Compared LRI with another method that considered user feedback	Generated their own data set	Verified the similarity between users in real-time
[74]	Java and R language on Intel (R) Core (TM) i5 2.60-GHz processor and 8 GB of memory, running Windows 8.1	Compared AGEGA with GA, OGA, AGP, and TGA	QWS plus some random data	Obtained better fitness values in lower execution time
[75]	Network simulator NS-2	Compared QoS-IoT against FIFO, RR, and cross-layer based utilization of CW for scheduling	Designed their scenario	Improved network performance with a large amount of data
[76]	MATLAB on Windows 10, 2.90 GHz processor, and 8 GB RAM	Compared Likert-Improved-PSO with original PSO and Improved-PSO	Generated a random data set	Better fitness values and lower execution time

Table (3.4) Continued

Ref.	Software used	Performance measurement	Data set	Results
[77]	-	Compared with AHP–AHP	Health care case study using real data collected from various sources available online	Lower execution time and robust to changes in user or decision-maker
[13]	Used JVM, JRE 1.6, for Windows 64-bit running on Intel Core i7-4712HQ CPU random memory	–	Synthetically generated data set based on QoS and EP	Improved energy efficiency, selection time, composition lifetime, and optimality of the solution
[9]	Simulation program in Java	Compared with GA	Mao Data set	Increased availability and reliability and maintained low cost and response time
[11]	MATLAB 7.0 + Intel Core2 Duo 2.10 GHz CPU	Compared CEA with CPSO and IDIPSO	Data generated from real scenarios	High performance in terms of search convergence and stability
[12]	MATLAB R2013b for Windows 7 on 2.50-GHz PC with 4-GB RAM	Compared HABC with GA, PSO, and basic ABC	Randomly generated data set among [0.7, 0.95]	High performance in terms of search stability within an acceptable time
[56]	MATLAB R2015a	Compared GSA and ABC with PSO, GA, ABC, and GSO	Real-time data acquisition read through Xively IoT API	Improved cluster head performance

Table (3.4) Continued

Ref.	Software used	Performance measurement	Data set	Results
[71]	Microsoft Visual C. Net on PC with an Intel Core i5 (1.6 GHz) CPU and 4 GB RAM	Compared fuzzy logic technique and cultural GA with QCD and WS-IP	QWS and RQWS generated randomly using the Eclipse programming tool	Reduced runtime and approximation ratio

3.6 Results and Future Research Directions

SSAs are essential for the IoT environment in order to satisfy the preferences of end-users by selecting the required services based on QoS factors. In the following, the discussion of the results, possible future research directions, and limitations of the state-of-the-art solutions, thereby highlight the basic requirements for a robust SSA structure.

Based on our proposed classification, the structure of SSA can be divided into the design process to determine the appropriate environment for building the SSA, the implementation stage involving the definition of the structure required to implement the SSA, and the evaluation step to measure the performance of the SSA.

In order to design an appropriate SSA structure for the IoT system, the following specific features should be considered. The time allocated to the selection process is called the process time phase. The design time is rarely static before a user requires a service [13]. Thus, most of the state-of-the-art methods are dynamic during the runtime [5] [11] [7]. In terms of workflow management, IoT is a large-scale environment that requires complex management or orchestration[56][14][7], where the IoT devices and network communication structure have specific properties. Thus, most studies preferred to select a choreography workflow[6] [58] [59]. Moreover, to the best of our knowledge, no algorithms involved single-objective

optimization because the QoS factors are related to others, and thus MOPs [55] [6] and MaOPs [5][11] required optimization.

In order to implement and build an appropriate SSA for the IoT system, most of the state-of-the-art methods focused on selecting the QoS factors that need to be optimized or satisfied. We classified the QoS factors based on the IoT architecture proposed by Li et al. [40] in the application layer, sensor layer, and network layer. In future research, it would be useful to focus more on search in the network layer because its properties affect the selection of the required services. Moreover, the traditional QoS factors considered in most studies comprised the optimization time, cost, availability, reliability, and energy consumption, as shown in Figure (3.5).

In addition, most of the solutions proposed for SSA used search optimization algorithms, particularly meta-heuristic algorithms [44] [60][58] based on evolutionary algorithms (e.g., GA, PSO, and ACO), as well as heuristic algorithms [57] [7] [67] [66] [65]. To the best of our knowledge, very few studies have implemented hyper-heuristic algorithms for making selections in the IoT system, as shown in[69]. The remaining state-of-the-art methods employed other types of algorithms, such as Pareto optimality [13] and the PSS method [62],[63].

Thus, researchers have tended to produced improved algorithms by combining more than one to obtain more efficient solutions. Therefore, new solutions can be obtained for selection problems by considering other methods such as fuzzy logic.

A new trend is the use of prediction in the selection process to enhance end-user preferences [73]. We consider that using a recommendation system that predicts the behavior and preferences of end-users could result in a more effective selection process.

In order to evaluate and measure the performance of SSAs, studies have generally compared the proposed algorithms and models with others, as shown in Table 4. Most studies conducted simulations but some involved building prototypes. The most commonly used language is Java, followed by MATLAB and other programming languages, as shown in Figure (3.6). In most studies, data sets were

generated for the experimental evaluations [12], [61], [67], [66], [65], [68], although some used existing data sets that were not constructed specifically for IoT environments [9], [62], [63], [57]. Thus, there is a need to provide an appropriate data set that satisfies the QoS requirements for services in IoT environments.

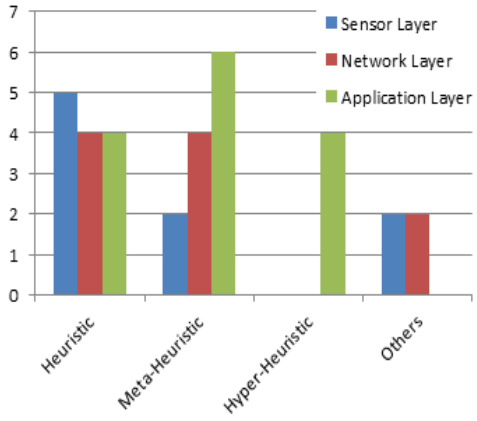


Figure (3. 4): Implementations Algorithms in Different IoT Layers.

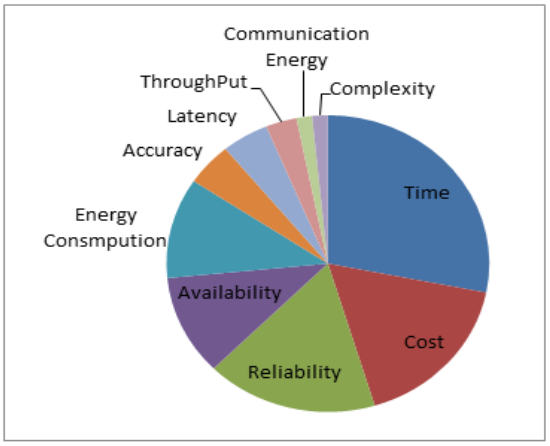


Figure (3. 5): Common QoS Factors Considered.

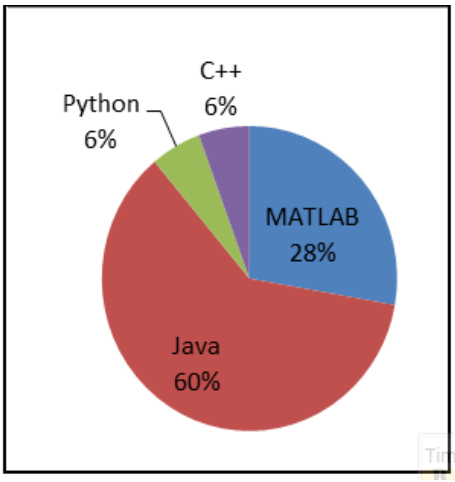


Figure (3. 6): Programming Languages Used for Performance Evaluations.

The results are shown in Figure (3.4), Figure (3.5), and Figure (3.6) were extracted by analyzing reports of state-of-the-art methods. Figure (3.4) was derived by analyzing the implementations of solutions for SSAs, as shown in Table (3.2).

Figure (3.6) was extracted by analyzing the evaluations and performance measurements for SSAs, as shown in Tables (3.3) and (3.4). Figure (3.4) - (3.6) should help researchers to extract useful information to guide their research into SSAs in IoT environments.

Chapter Four Research Methodology

Chapter Four: Research Methodology

The Proposed Services Selection Models

4.1 Introduction

Internet of Things (IoT) is a coherent environment, which aims to link physical things together. It senses things by using a massive number of sensors and actuators. These actuators are connected to create new applications that facilitate end-users, lives. The provided applications are composited of any number of abstract services connected as one service for the end-user to perform more complex functions [79]. In the IoT environment, the number of sensors continuously increases; this leads to an increase in the number of services. The provided services from the same sensor type are similar in their functional properties, but they differ in their non-functional properties.

The service selection algorithm is considered an NP-hard problem. It aims to select an optimal abstract service from a set of concrete services. The selection criteria are built based on the QoS constraints that are accepted by the end-users. ITU-T E.860 defines QoS as the degree of conformance of the service delivered to a user by a provider with an agreement between them [80]. In the IoT environment, QoS is formed in several architectures. Authors in [40] proposed a QoS architecture to be appropriate with IoT architectures. They divided QoS into three main layers (Sensor layer, Network layer, and Application layer). These layers integrated the traditional QoS attributes with other essential qualities in IoT Architecture (e.g., network deployment cost, Energy efficiency management, information accuracy). This research focused on QoS from the Application Layer. It represents the highest layer in IoT architecture, which consists of many distributed abstract services composed to be one service or application for the end-users. Examples of QoS in the application layer are (Execution Time, Availability, Services Perform Price, and Reliability) [40].

The methodology of this research aims to solve a service selection problem in the IoT environment. It is divided into three leading solutions that are proposed to solve this selection problem. In general, the proposed solutions followed a horizontal adaptation to compose the abstract services together.

The three proposed solutions models are built based on using two main ideas: firstly, they aim to use a reputation value, which is called historical information [61] or end-user feedback information [8]. This information is gathered from the end-users after using a service. The proposed models introduce an ease and friendly method to gather this historical information by using a Likert Scale measurement.

Secondly, they aim to use a bio-inspired Meta-Heuristic Optimization Algorithm (M-HOP) as a key to solving the selection problem. M-HOP algorithms are one of the algorithms used to address the services selection problem. It is an artificial intelligence algorithm introduced by Glover in 1986 [81]. It comes from the combination of two Greek terms meta-(Meta, which means the high-level), and heuristic, which means "I find, discover".

The first model is called Likert-IPSO, which is proposed to improve the performance of the selection process and selection time. It was done based on End-users evaluation of QoS for each service by using a user-friendly way called a Likert scale measurement. The services are ranking in the services registry based on end-users evaluation as reputation factors. Then use the PSO algorithm to search for ranked services that match the reputation value of the required services. This led to reducing the search space for the required service. The second model aims to improve the selection process by using a more precise and robust M-HOP algorithm and solve the exploitation problem in the PSO algorithm [82]. It is proposed to enhance the performance of an existing M-HOP algorithm called an SSO. The proposed model improves the behavior of the original SSO by enhancing the member weights in a colony. The member weights increase or decrease based on the service reputation takes from the end-user. It is called a Reputation Improved-Social

Spider Algorithm (RI-SSO) because it improved the behavior of SSO based on gathered reputation information.

In the end, the third model called FL-RISSO aims to reduce the search space by classifying it into sub-clusters. Then the search process applies only in the cluster that is appropriates with the end-user requirement. It is proposed to use a fuzzy logic system to cluster the search space. Then appling the proposed RI-SSO algorithm into the sub-cluster to find the optimal result based on end-user preference.

4.2 Problem Formulation

In order to formalize the process of service selection based on QoS constraints decomposition, some basic concepts and definitions are listed below.

Definition 1: Component service (S): $S = \{WS_1, WS_2, \ldots, WS_n\}$. It is the basic unit in service composition, providing services to users. Each service S composes of n abstract web services.

Definition 2: A web service class (WS_i): $WS_i = \{cs_{i1}, cs_{i2}, cs_{i3}, \cdots, cs_{iK}\}$ denotes the i^{th} abstract service of a composite web service. It has k candidate services, which have the same functionality but differ in non-functionality (QoS factors) as in Figure (4.1).

Definition 3: QoS vector of concrete web service CS_{ij} is declared as $Q(CS_{ij})=\{Q_1(CS_{ij}), Q_2(CS_{ij}), Q_3(CS_{ij}), \ldots., Q_R(CS_{ij})\}$ contains R QoS attributes of each concrete service.

Definition 4: QoS aggregation for a composite service (CQ): $CQ = \{C_{Q1}, C_{Q2}, \cdots, C_{QR}\}$ contains R QoS attributes of a composite service. It can be calculated in terms of QoS values of component services and the composition structures.

Definition 5: User preferences also called services weight or QoS utility, W: $W = \{w_1, w_2, \cdots, w_R\}$ which represents user preferences, where w_k $(1 \leq k \leq R)$ is the user's preference for the k^{th} QoS attributes. $\sum_{k=1}^{R} W_k = 1$, $0 < W_k < 1$.

Definition 6: Global QoS Constraints (C): $C = \{C_1, C_2, \cdots, C_R\}$ is the set of user's global QoS constraints, which contains R constraints and C_k $(1 \leq k \leq R)$ is a global

constraint over Q_k. C_k can be shown according to upper or lower bounds for the QoS aggregation value C_{Qk}.

4.3 Quality of Services Model

The selection of an optimal service that meets end-user requirements depends on QoS factor values distinguished from one service to another. This section illustrates the QoS model used in the proposed solution including QoS types and QoS normalization.

4.3.1 Quality of Services Types

The proposed QoS model identifies five QoS factors are related to the application layer. These factors are selected from two quality types: Business Quality Type (BQT), and System Quality Type (SQT) [39].

4.3.1.1 Business Quality Type (BQT)

BQT is an economic value that is offered by applying services. This value is used to evaluate the right service based on business value. The proposed model studies two BQT factors: reputation $q_{RP}(s)$, and execution price $q_{EP}(s)$.

Reputation $q_{RP}(s)$ is a social evaluation of service depending on the rates coming from different users after requesting the services.

Execution Price $q_{EP}(s)$ is a value that the user pays for the service invocation to a provider during or after using the service.

4.3.1.2 System Quality Type (SQT)

SQT indicated to the QoS related to the system performance. There are three SQT factors considered in the proposed model, which are reliability $q_{re}(s)$, availability $q_{av}(s)$, and response time $q_{rt}(s)$.

Reliability $q_{RE}(s)$ is the probability ratio to complete the services successfully.

Availability $q_{AV}(s)$ refers to the probability rate that the service is running and accessed when invoked.

Response Time q_{RT} (s) refers to the time between the service request, and the service response is received. It is measured by seconds.

4.3.2 Quality of Services Normalization

To optimize the QoS value, the behavior of factors are varied from one factor to another [82]. Some factors are optimized by getting the minimum values; they are called negative QoS factors. Other factors are optimized by getting the maximum values; they are called positive QoS factors.

The optimal results for the negative QoS factors (execution price and response time) are the smallest values. The negative factors can be normalized as in equation (4.1).

$$Q_i(CS) = \begin{cases} \frac{Q_i^{max} - Qi(CS)}{Q_i^{max} - Q_i^{min}} & Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & Q_i^{max} - Q_i^{min} = 0 \end{cases} \quad (4.1)$$

Moreover, the optimal results for positive QoS factors (Reputation, Reliability, and Availability) are the highest values. The positive factors can be normalized as in equation (4.2).

$$\bar{Q}_i(CS) = \begin{cases} \frac{Q_i(CS) - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & Q_i^{max} - Q_i^{min} = 0 \end{cases} \quad (4.2)$$

Where i, ($1<i<5$) indicates the number of QoS factors. CS indicates concrete services. Q_i^{max} and Q_i^{min}represent the maximum and minimum values of the i-th QoS factor, respectively. The following equation gives QoS vector of the concrete service CS:

$$Q(\text{CS}) = (Q_{\text{RP}}(\text{CS}), Q_{\text{EP}}(\text{CS}), Q_{\text{RE}}(\text{CS}), Q_{\text{AV}}(\text{CS}), Q_{\text{RT}}(\text{CS})) \quad (4.3)$$

4.4 Services Selection Model

End-users always implement different types of applications in the IoT environment. These applications consist of numerous abstract services that are composed together to satisfy the end-users requirements. There are many concrete services for each

abstract service. This section includes the service selection adaption followed by the composed service selection structure and aggregation function declaration applied in the services selection model.

4.4.1 Services Selection Adaption

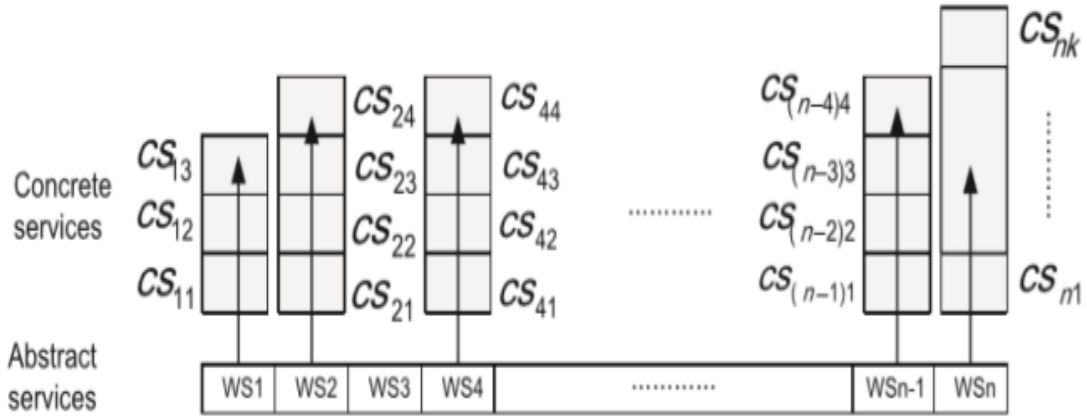
There are two goals to adapt the service selection process: vertical adaptation and horizontal adaptation [83],[9].

4.4.1.1 Vertical Adaptation

The vertical adaptation aims to find the optimal services composition path for each abstract service—the binding path of abstract services workflow adequate for all the existing interdependent restrictions. In the vertical adaption, the main task is decomposed into many subtasks. Then a sequence of services that accomplish each subtask is searched. The result is the best combination of abstract web services, which can differ in their functional requirements. At the same time, each service is an abstract web service that is not bound to any concrete service.

4.4.1.2 Horizontal Adaptation

The horizontal adaptation aims to find the ideal concrete service from a set of functionality equivalent candidate services for each abstract service separately, as in Figure (4.1). It is more appropriate for the IoT environments, containing an enormous number of sensors that provide services equivalent in functional properties and different in non-functional properties. Horizontal adaptation provides greater flexibility for user intervention [83]. It enables the user to modify the abstract workflow when required. This means that the end-user is not obligated to follow a specific workflow. Also, it simplifies the composition problem by reducing its computational complexity. Hence, this research applies horizontal adaptation to optimize the selection problem to satisfy the different end-users' preferences.



Figure(4. 1): Service Selection Workflow[9]

4.4.2 Composed Services Selection Structure

To compose the services together using general web services technologies, there are four main composition structures: sequential, parallel, cycle, and branch [11] [12], as shown in Figure (4.2).

Sequential composition is to execute the composition of services in the sequential order one following the other.

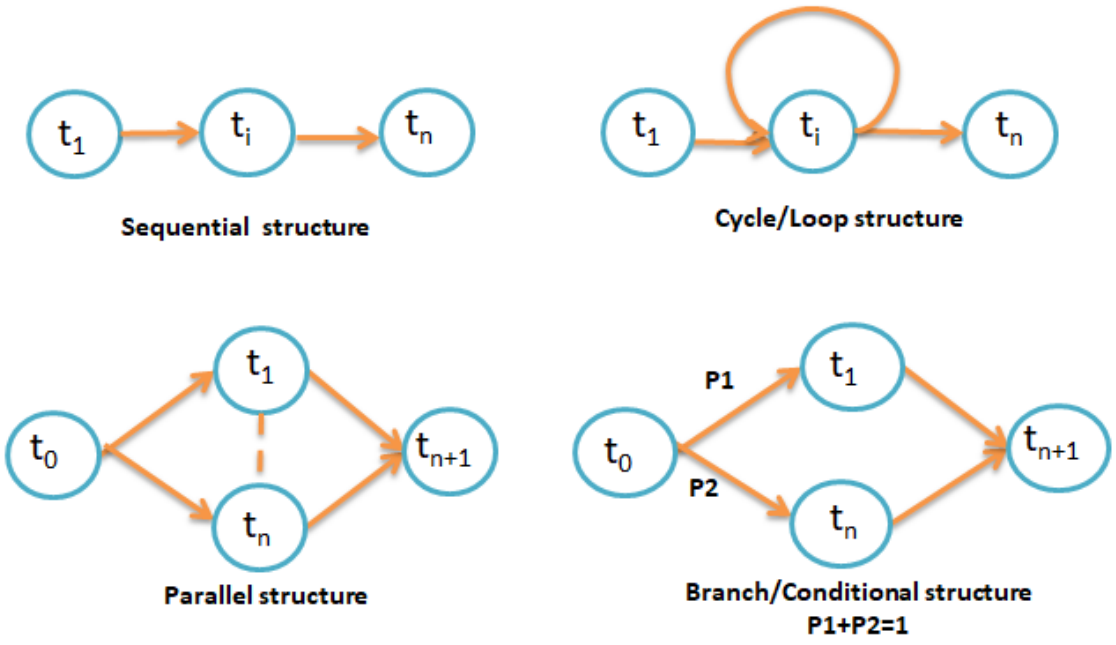
Parallel compositions tasks are performed simultaneously go to the next task, until all of these parallel tasks are achieved.

In a cycle composition, at least one task must be performed more than once.

The branch composition selects only one task from a set of optional ones and goes to the next step. The difference between the parallel and branch structure is very distinct. For example, all the tasks t_1, . . ., t_n have to be completed before the execution of task tn+1 in a parallel structure. In the case of branch structure, it goes through only one of the tasks t_1, . . . and t_n from task t_0 to t_{n+1}.In the proposed model, the sequential workflow of service composition is considered. Other composition structures can be converted into sequential structures using existing techniques [84].

In a sequential workflow, the QoS value for each concrete service is calculated by aggregating each factor's corresponding values. The sequence-structure applies two types of QoS aggregation functions illustrated in Table (4.1). The additive function

applies for response time qRT(s), execution price qEP (s), and reputation qRP(s) factors. The multiplicative function applies for availability qAV(s) and reliability qRE (s) factors.



Figure(4. 2): Basic Patterns of Web Service Composition[85][11]

4.4.3 Aggregation Function

To get the optimal selection process, the aggregation function also called objective function or fitness value is calculated by considering the five QoS factors for candidate concrete services in all abstract services. The objective function or fitness value is calculated based on an optimization type to maximize or minimize the services selection as in Equation (4.4):

Objective Function =

$$[\text{Min}(W_{RT}*F_{RT}(CS_{ij}))+\text{Min}(W_{EP}*F_{EP}(CS_{ij}))+\text{Max}(W_{AV}*F_{AV}(CS_{ij}))+\text{Max}(W_{RE}*F_{RE}(CS_{ij}))+\text{Max}(W_{RP}*F_{RP}(CS_{ij}))] \quad (4.4)$$

Where, $F_{QoS}(CS_{ij})$ represents the summation or multiplicative function for each factor. For simplicity's sake, the model calculates the summation for all QoS factors. W_{QoS} represents the weight for each factor, calculated as in equations (4.5) and (4.6).

$$\sum_{i=1}^{5} W_{QoS} = 1\ ,\ 0< W_{QoS}<1. \quad (4.5)$$

$$W_{QoS}= W_{\text{RT}} + W_{\text{EP}}+ W_{\text{AV}}+ W_{\text{RE}}+ W_{\text{RP}}=1 \quad (4.6)$$

Where, i represents the number of QoS factors ($1< i <5$).

Table (4. 1): Aggregation Function To Compute The QoS Factors[86]

Structure	Response Time	Execution Price	Availability	Reliability	Reputation
Sequence	$q_{RT}(\text{Seq}) = \sum_{i=1}^{n} q_{RT}(CS_i)$	$q_{EP}(\text{Seq}) = \sum_{i=1}^{n} q_{EP}(CS_i)$	$q_{AV}(\text{Seq}) = \prod_{I=1}^{N} q_{AV}(CS_i)$	$q_{RE}(\text{Seq}) = \prod_{I=1}^{n} q_{RE}(CS_i)$	$q_{RP}(\text{Seq}) = \frac{1}{n}\sum_{i=1}^{n} q_{RP}(CS_i)$

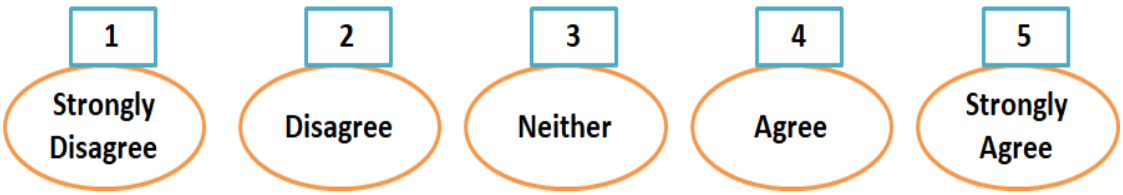
4.5 Likert-Type Measurement

Likert-Type explained in [87] is a psychometric response scale primarily used in questionnaires to obtain a participant's preferences or degree of agreement with a statement or set of statements. The scale measurement was named by Dr. Rensis Likert in 1932 to improve a means of measuring psychological attitudes directly in a scientific method. There are many structures introduced to measure the levels of granularity. Some researchers use the 7-point scales and 9-point scales, which added an additional level of granularity. Other researchers use 4-point scales, the point determination based on questionnaire requirements. The proposed solution implemented the most commonly used structure which is a 5-point scale [88]. The scales are ranging from Strongly Disagree, Disagree, Neither, Agree, and Strongly Agree, as shown in Figure (4.3). Each scale is assigned to a code, like using numeric value or alphabet value. This value is used to measure the attitude under investigation, usually starting at one and incremented by one for each level.

The introduced model uses a Likert-type to measure the agreement level of the service taken from end-users after using the service. This research uses a Likert-type and does not use a Likert scale—the Likert-type items are identified as stand-alone

questions that use the 5-points as an alternative to the end-users response. However, a Likert scale is composed of a series of Likert-type items that are composite into a single score during the data analysis process [88]. The agreement level provides the feedback value about the end-user preference saved as reputation value to each service. In the beginning, the reputation value is set to zero. To analyze the collected feedback data, the median value is not the mean value of each service calculated to get a more precise answer. The median used because the solution used the Likert-type does not use a Likert scale [88].

For example if the service $S_{i,}$ is evaluated ten times from end-users as follow: 3,1,3,2,4,3,4,2,1,1. To calculate the median value, first, reorder the evaluation list in ascending order as 1,1,2,2,3,3,3,3,4,4. Then find the middle one's position by dividing the (list-length / 2): ($10/2=5^{th}$ position). The result is rounded to the nearest integer number for odd list length. So the median is equal to 3, which indicates "Neither". The reputation value is saved as a decimal number as in Table (4.2) — thus the reputation value for service S_i is equal to 0.6.



Figure(4. 3): Five-Point Scales Used in Likert-type Measurement [88]

Table (4. 2): Reputation Values of 5-Points Agreement Levels

Agreement levels	Equal Number	Reputation Values
Not evaluated services	0	0
Strongly Disagree	1	0.2
Disagree	2	0.4
Neither	3	0.6
Agree	4	0.8
Strongly Agree	5	1.0

4.6 First Proposed Model Likert-IPSO

The first proposed model applied the Practical Swarm Optimization Algorithm (PSO), and Likert Scale Measurement named Likert-IPSO. This section defines and explains the original PSO behavior and then illustrates the proposed model Likert-IPSO to solve the services selection problem in the IoT environment.

4.6.1 Practical Swarm Optimization (PSO)

PSO is a nature-inspired multi-objective optimization algorithm developed by J. Kennedy and R. Eberhart in 1995 [89]. PSO is a stochastic optimization technique, commonly used on continuous nonlinear optimization function [90] [91]. It mimics the behavior of a fish or birds swarm to search for food [11] in the search space. The swarm represents the population, and when an individual practical finds a direction for food in search space, it shares this information with other practice in a swarm. The other particles direct to the correct path towards the food. The PSO is directed by personal (individual) practical solution (xPBest), global practical solution (xGBest), and the present movement of the particles to decide their next positions in the search space. It means that if a particle finds a new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process. The improved-PSO (canonical PSO, CPSO) was proposed by Shi and Eberhart in 1995 [11]. It hurries up the progress of traditional PSO in a dynamic environment. The Improved-PSO has an inertia weight W, which balances between exploitation and exploration. The basic equations of PSO are:

$$x_i(t+1) = x_i(t) + v_{ij}(t+1) \tag{4.7}$$

$$\begin{aligned} v_{ij}(t+1) = {} & w(t) \times v_{ij}(t) + c_1 \times r_{1j}(t) \times \left(xPBest_{ij}(t) - x_{ij}(t)\right) \\ & + c_2 \times r_{2j}(t) \\ & \times \left(xGBest_{ij}(t) - x_{ij}(t)\right) \end{aligned} \tag{4.8}$$

The iterations directed by two factors c_1 is a variable to weigh the particle's knowledge, and c_2 is a variable to weigh the swarm's knowledge. They control how far a particle will move in a single iteration, and two random numbers r_1 and r_2 generated between [0, 1]. In contrast, the present movement multiplied by an inertia factor W varying between [W_{min}, W_{max}] which is used to weigh the final velocity, t a point in time or iteration number, xPBest is the best solution the particle ever visited, and xGBest is the best location any particle in the swarm ever seen.

The initial population of size N and dimension j is denoted as $X = [X_1,X_2,...,X_N]^T$, where 'T' denotes the transfer operator. Each individual (particle) X_i (i = 1,2,...,N) is given as $X_i=[X_{i,1},X_{i,2},...,X_{i,j}]$; X represents a particle and i denotes the particle's number.

Also, the initial velocity of the population is denoted as $V = [V_1,V_2... V_N]^T$. Thus, the velocity of each particle X_i (i = 1,2,...,N) is given as $V_i=[V_{i,1},V_{i,2},...,V_{i,j}]$.

4.6.2 Proposed Solution Using Likert-IPSO

In the normal process, a customer requests a service and selects it from the services registry. The registry contains a vast number of services from different providers with the same functional but different in non-functional properties.

The proposed solution ranks services in the registry based on their reputation.

The services are classified on five Likert scale measurements (Strongly Agree, Agree, Neither, Disagree, and Strongly Disagree). It means that services with the same function are classified into five parts with the five values (A, B, C, D, and F) sequentially based on its measurement value. This value reduces the search space when a customer requests a service; the service is only selected from a part that meets the required constraints.

For example, if a customer requests a service S with QoS q and value A, the search is done only in part A from the same services. This behavior improves the performance, increases the response time, and reduces the cost. For the first time,

when a new customer requests a service, it is selected from the registry in a usual way. This research proposes to use an improved PSO algorithm. This step is done with all new customers; customers request service for the first time only before they evaluate any service evaluation. After a customer uses a service, they evaluate the QoS for each service based on a Likert scale. The research depends on five QoS (Cost, Time, Reputation, Reliability, and Availability). This evaluation keeps the reputation of each service.

If there are more than one customer use the same service from the same provider for the first time — the mean of reputation scales are calculated and kept with a service. The next time when a customer requests a service, it will be selected based on its previous registry feedback.

4.7 Second Proposed RI-SSO Model

The second proposed model improved the behavior of the original Social Spider Algorithm (SSO) named RI-SSO. RI-SSO is aimed to add the service's reputation value to the spider weight to enhance the service selection process. This section defines the behavior of the original SSO and explains in details the proposed RI-SSO model to solve the services selection problem in the IoT environment.

4.7.1 Social Spider Optimization (SSO)

SSO is a swarm intelligence algorithm that emulates the collective behavior of spider swarms. It was proposed by Cuevas et al. in [92] to find an optimal solution to complex optimization problems in continuous search space. The original SSO and proposed RI-SSO are used to solve the service selection problem, which is discrete, namely the Nearest Integer method [82]. There are two primary behaviors in the SSO colony: First: catches a prey behavior is achieved by utilizing the vibrations on the spider web to determine its positions. Second: the mating behavior; where females use the vibrations of the male spiders over the web to determine heavier spider fitness. This research followed the second behavior.

4.7.1.1 SSO Components

The social spider optimization colony consists of two main components, which are the communal web and social members.

The communal web represents a search space of the optimization problem of the SSO algorithm, on which all spiders have a position. Each position represents an available solution to the optimization problem on the web. When a spider leaves the web, its position represents an unavailable solution to the optimization problem [93]. Social Members are spiders on the web which are agents of SSO to perform optimization. This represents the complete population (pop_s). Social members are divided into two members groups [92][93], as in Figure (4.4). Females group: F represent 65 - 90 of the total colony members, F= $\{f_1, f_2, f_3, \ldots, f_{Nf}\}$. Males group: M represent 35 - 10 of the total colony members, M= $\{m_1, m_2, m_3, \ldots, m_{Nm}\}$. Whereas $pop_s = F \cup M$.=$F \cup M$, S= $\{s_1, s_2, s_3, \ldots, s_N\}$, so pop_s ={ $s_1=f_1$, $s_2=f_2$, …, $s_{Nf}=f_{Nf}$, $s_{Nf+1}=m_1$, $s_{Nf+2}=m_2$, ..., $s_N = m_{Nm}$}. On the basis of gender, each individual is calculated through a set of different evolutionary operators that emulate different cooperative behaviors. Also a spider receives a weight according to the fitness value of the solution on the web.

4.7.1.2 Fitness Evaluation

On the web, each spider s_i has a weight SW_i, which represents the solution quality. The weight value of each solution s_i in population pop_s is calculated by equation (4.9):

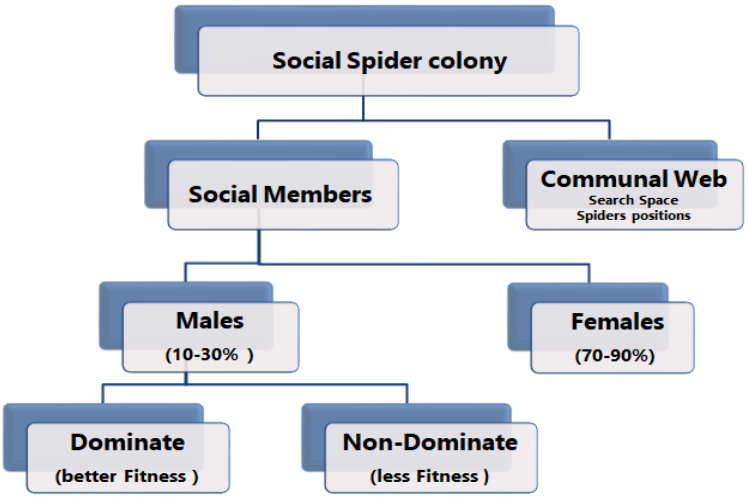
$$SW_i = \frac{f(s_i) - Worst_s}{best_s - Worst_s} \qquad (4.9)$$

Where $f(s_i)$ is the fitness value obtained by the calculation of the spider position s_i concerning the fitness or objective function in equation (4.4). The best and worst values are the maximum and the minimum values of the population's solutions.

4.7.1.3 Social Members (Spiders)

The spiders on the web, male or female, are the primary agents to perform the optimization selection of the SSO Algorithm. Each spider S on the web has a memory to store two types of information: individual situation information and new positions information.

Individual situation information is used to describe the spider S, which consists of the position of spider S on the search space or population pop_s. It represents the fitness value in the current position of spider S. New position information is used to guide a spider S to new position and consists of the target vibration of spider S in the previous iteration, the number of iterations since spider S last changed its target vibration, the movement that spider S performed in the previous iteration, and the dimension mask that spider S employed to guide its movement in the previous iteration.



Figure(4. 4): Social Spider Members

4.7.1.4 Vibrations through the Communal Web

The communal web is used as a mediator to transmit information among the spiders. This information is encoded as small vibrations V for the collective coordination of all individuals on the web. The strength of vibrations strong depends on two factors. First is the weight factor calculated by fitness function as in equation (4.10),

whereas members have the higher weights generate stronger vibrations than members having lower weights. Second is the distance factor that is the distance between an individual that generates the vibrations and the member that detects them as in equation (4.11). Members located near the individual that generates the vibrations receive stronger vibrations in comparison with members located in further positions. Three relationships considered when computing vibrations between any pair of individuals. As shown in Figure (4.5):

Vibrations (V_ibc_i) is the transmitted information between the individual i(S_i) and the member c(S_c), which is the nearest member to (i) with a higher weight compared to i ($SW_c > SW_i$).

Vibrations (V_ibb_i) is the transmitted information between the individual i(S_i) and the member b(S_b), which is the best weight member in the population (S).

Vibrations (V_ibf_i) is the transmitted information between individual i(S_i) and the nearest female individual f(S_f).

Vibrations are affected by two properties, which are the intensity source and the intensity attenuation. The intensity source of the vibration is calculated by equation (4.14) in the range [0, + ∞] [93]. For each time *t*, a spider s at position P_a moves to a new position, it generates a vibration at its current position. This intensity source at the position P_a is affected by the fitness value of its position $f(P_a)$, or spider weight at position P_a.

$$I_S(P_a , P_b , t) = \log(\frac{1}{f(P_a)-C} + 1) \qquad (4.10)$$

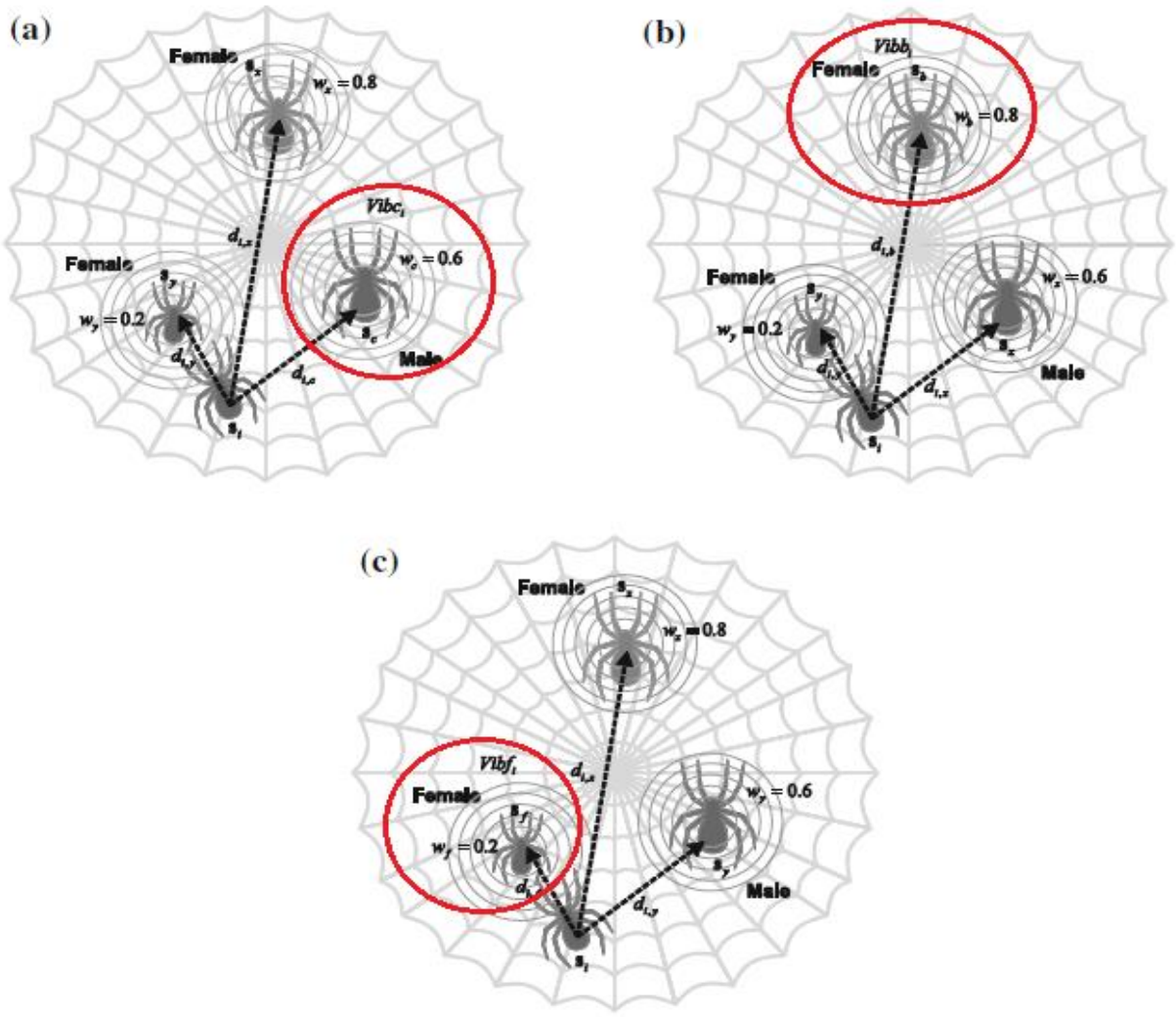
Where, C is a confidently small constant. In minimization problems, all possible fitness values are more significant than the C value.

The spider positions with better fitness values are the larger value from maximization problems or smaller value for minimization problems with higher vibration intensity sources than those with worse fitness values.

The intensity attenuation also called source position, I_d over a distance $D(P_{a,}\, P_b)$ between spiders at position P_a and P_b is calculated as equation (4.11) with consideration of vibration attenuation's physical energy phenomenon during the propagation process over the web.

$$I_d(P_a\,, P_b\,, t) = \; exp\left(-\frac{D(P_a, P_b)}{\bar{\sigma} * r_a}\right) \qquad (4.11)$$

Where, $\bar{\sigma}$ is a standard deviation of all spiders' positions.



Figure(4. 5): Configuration of Each Special Relation: (a) Vibci, (b) Vibbi And (c) Vibfi [92]

So the vibration intensity receives $I(P_a,\ Pb, t)$ value sensed by a spider in position P_b and generated by a spider in source position P_a at time *t* is calculated by equation (4.12) and (4.13) that regarding both intensity source I_S and intensity attenuation *Id.*

$$I(P_a\ , P_b\ , t) = I_S(P_a\ , P_b\ , t)\ \times\ I_d(P_a\ , P_b\ , t) \qquad (4.12)$$

$$I(P_a\ , P_b\ , t) = \log(\frac{1}{f(P_a)-C} + 1)\ \times\ exp\left(-\frac{D(P_a,P_b)}{\bar{\sigma} * r_a}\right) \qquad (4.13)$$

4.7.1.5 SSO Algorithm levels

To obtain an optimization selection solution by using SSO Algorithm, there are three levels: initialization level, iteration level, and final level.

Initialization level is a start step in optimization processes done by initializing the following steps:

Step1 is the optimization search space, which represents the hyper-dimensional spider web.

Step2 is the spider's positions (feasible solution for service selection) represent the population pop_s over the web. They are randomly generated with their fitness, which represents the quality of each offered solution.

Step3 is for the objective function that is used to select an optimal solution based on end-user preference.

Step4 is for the end-user that defined a value for the QoS factors to be used in SSA, representing female attraction.

Iteration level: SSO performs searching iteratively until finding the optimal solution in the offered solutions. In any iteration, spiders on the population pop_s move to a new position and perform the following steps:

Step1 evaluate the fitness values of each spider S in the population pop_s.

Step2 generate of the vibrations V for all spiders by using equation (4.10).

Step3 propagate these vibration intensities over the web using equation (4.13).

Step4 selects the strongest vibration value V^{best} from V, in maximization problems, the strongest vibration means the largest value and vice versa in minimization problems.

Step5 compare the intensity value of V^{best} with the sorted intensity value of the target vibration V^{tar}. If V^{tar} is less than V^{best}, the inactive degree IN^d is reset to zero. Otherwise, V^{tar} value retained, and IN^d is incremented by one.

Step6 move a random walk towards V^{tar} by using dimension mask M to direct the spider movement. M is a binary vector $\in [0,1]$ of length equal to the web dimension D of the optimization problem. Its value is changed based on the probability $1 - P\_C^{IN^d}$. If the M value is changed, all vector elements have a probability of P_m to be an equal one, and a probability of $1 - P\_m$ is zero. Whereas, P_C is a user pre-defined parameter that detects the probability of changing the mask. Moreover, P_m is also a user pre-defined controlled parameter $\in [0,1]$.

Step7 generate a new following position $P_{s,i}^{folw}$ based on the mask for S as in equation (4.14):

$$P_{s,i}^{folw} = \begin{cases} P_{s,i}^{tar}, & M_{s,i} = 0 \\ P_{s,i}^{r}, & M_{s,i} = 1 \end{cases} \qquad (4.14)$$

Where, $P_{s,i}^{tar}$ is the *ith* element of the source solution of V^{tar}, $P_{s,i}^{r}$ is a random solution's position, r is a random integer value $\in [1, |\mathrm{pop_s}|]$, and $M_{s,}$ *i* is the *ith* dimension mask *M* of the spider S.

Step8 calculate the random walk of a new position $P_s(t+1)$ using the following formula (4.15):

$$P_s(t+1) = P_s(t) + \left(P_s(t) - P_s(t-1)\right) \times r + \left(P_s^{fo} - P_s\right) \odot R \qquad (4.15)$$

Where, $\odot$ operator indicates the element-wise multiplication operator, and $P_s(t)$ is a current position.

The final level is the last level, which handles any constraint that can happen during the iteration level lead to violating the optimization problem, such as spiders can move out of the web (maximum and minimum bounds) during the random walk step, which means the offered solution will be unavailable.

$$P_{s,i}(t+1) = \begin{cases} \left(\overline{X_\iota} - P_{s,i}(t)\right) \times r, & P_{s,i}(t+1) > \overline{X_\iota} \\ (P_{s,i}(t) - \underline{X_i}) \times r, & P_{s,i}(t+1) < \underline{X_i} \end{cases} \quad (4.16)$$

Where, $\overline{X_\iota}$ is the upper bound on the search space, and $\underline{X_i}$ is the lower bound.

4.7.2 Proposed Solution Using RI - SSO

The proposed model is built based on the two natural behaviors in the social spider colony. In the proposed solutions, these behaviors were used to enhance the selection process. They are explained as follows:

First is the natural behavior of mating between social spider members, which is done between female and dominant males (males with better fitness) [93]. As a result of the mating, a new offspring is generated with new fitness values. The generated fitness values are based on the strength of the dominated male, which performs the mating. Fitted male mating generates fitted offspring and vice versa [94]. To represent this behavior in the optimization selection problem, the females represent the end-users, and dominated males represent the candidate services.

The proposed solution aims to enhance and emulate this natural behavior by regarding to a subjective factor. It evaluates the feedback values that are taken from the end-user for each service. Then this evaluation value is added to the colony member or services in search space as a reputation factor. The reputation values are gathered by using an easy, friendly method called a Likert-type measurement. Then the new fitness value is generated based on the collected information. The new fitness value of the service represents the new offspring with a new fitness value. This added value leads to improving the next service selection process based on the collected evaluation from a previous selection process.

The second behavior is the attraction or dislike between females and males. It is evolved based on the propagation of the vibrations across the web from males to females. In the original SSO, the strength of vibration intensities are sensed based on two properties [94] weight and distance as physical energy phenomena as in equations (4.10), (4.11), and (4.13). The more energetic vibration are generated either by large spiders or neighboring members on the web.

Generally, in physics, the intensity is defined as "the quantity of energy the wave conveys per unit time across a surface of unit area" [95]. The intensity formula is:

$$I = \frac{P}{A} \qquad (4.17)$$

Where P; is the power, and A; is the area.

From the above equation, the relation between intensity and power is directly proportional. So the increase in the power value leads to an increase in the intensity value. In a social spider colony, the power is represented by the member's weight. So the proposed model focuses on the effect of members' weights based on the collected evaluation information. If the end-users are satisfied with selected services, they evaluate it with a high agreement point, for example (Strongly Agree, or Agree) and vice versa. This evaluation is converted to a reputation value (1.0, 0.8), respectively. The reputation value is added to the member's weight or its fitness value as in equation (4.18). The service with a high reputation value meets the spider with high intensity that enhances the attraction of dominated males and vice versa.

The optimization model is summarized in the following steps as in Figure (4.6), where the blue processes represent the Original SSO, and the red processes are the additional improvement:

Step1: In the initial steps, the selection process is implemented by the original SSO, and the reputation value is set to equal zero.

Step2: The selected service is delivered to the end-user and evaluated based on QoS constraints. The Likert scale measurement is used based on 5-points scales to get the evaluation value from the end-user.

Step3: The reputation value is calculated from the evaluation information is in Table (4.2). The reputation value added to each service as a reputation property.

Step4: Evaluate the intensity source value by adding the reputation value to the QoS fitness value as in equation (4.18(a)) and (4.18(b)). This update applied in the equation (4.10) of the original SSO and is calculated as follows:

For maximization optimization problem, the intensity source is calculated as:

$$I_S(P_a, P_b, t) = \log\left(\frac{1}{(f(P_a)-(f(P_a)*Rep\_val))-C} + 1\right) \quad (4.18(a))$$

For minimization optimization problem, the intensity source is calculated as:

$$I_S(P_a, P_b, t) = \log\left(\frac{1}{(f(P_a)+(f(P_a)*Rep\_val))-C} + 1\right) \quad (4.18(b))$$

Where, $f(P_a)$ is a fitness value of spider at position a, C is a confidently small constant.

Step5: Evaluate the vibration intensity value by updating the original SSO equation (4.19) to be calculated as follows:

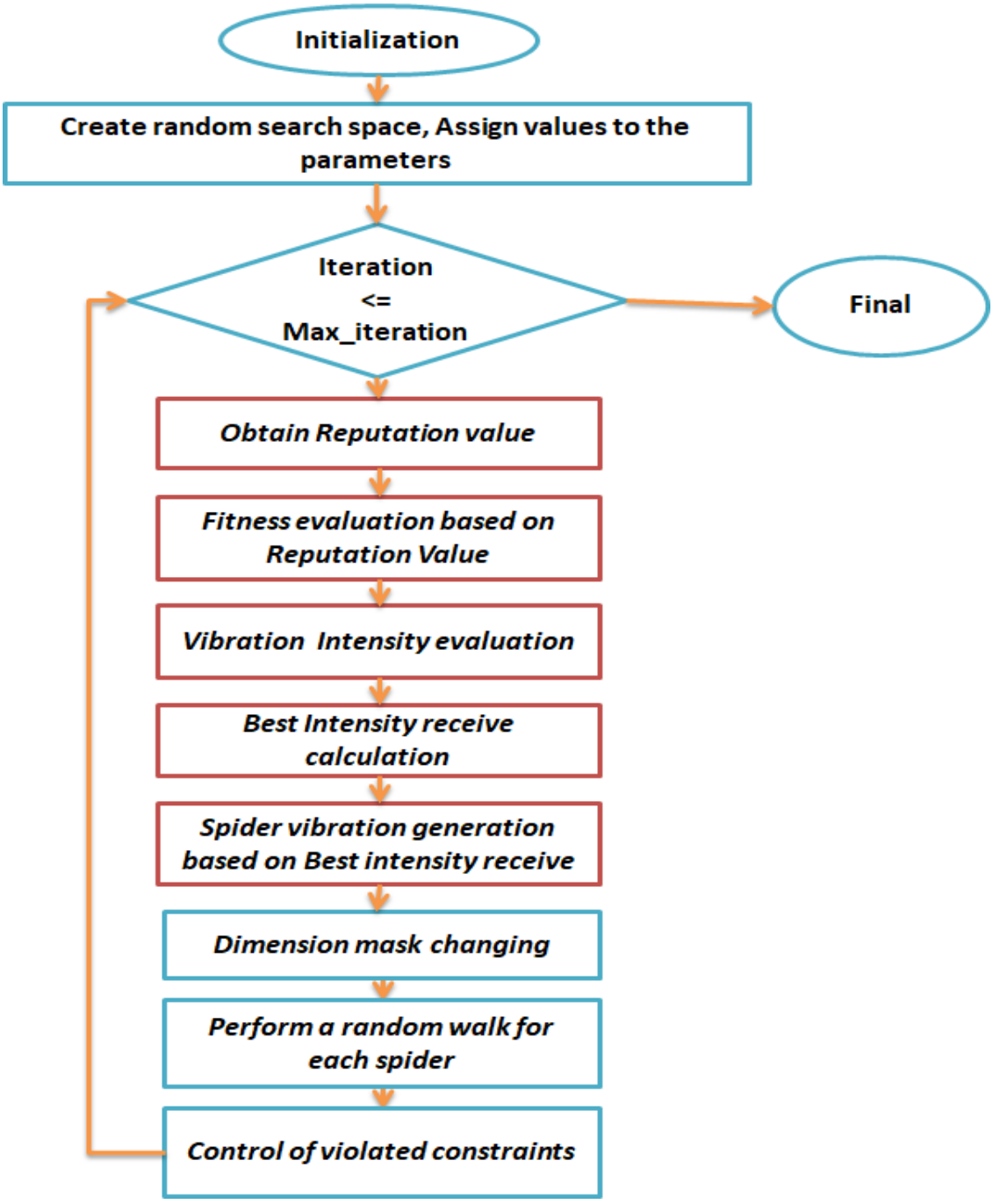
$$(P_a, P_b, t) = \quad I_S(P_a, P_b, t) \times exp\left(-\frac{D(P_a, P_b)}{\bar{\sigma} * r_a}\right) \quad (4.19)$$

Then calculate the best vibration intensity received in the population and save it.

Step6: Add the best vibration intensity received value to the *target vibration* position as in this equation (4.20):

$$Target\ viberation = (((Target\ viberation * Target\ Matrix) + (Max\ Position * 1 - Target\ Matrix) + Best\ Intensity\ Recive) \quad (4.20)$$

In the original SSO, the target vibration is calculated without adding the best vibration intensity received.



Figure(4. 6): Flow Chart RI-SSO Operations.[96]

4.8 Third Proposed FL-RISSO Model

The third proposed model aims to reduce the search space for the required service. This is done by classifying the search space into sub-clusters. Hence when the end-

user requires service, the search is done only in the cluster that is appropriates his requirements. To apply search space clustering, the introduced model proposes to use a Fuzzy Logic System (FLS). Then it implements the RI-SSO algorithm in an appropriate cluster to select the optimal service that meets the end-user constraints. The RI-SSO model has illustrated in Section 4.7. This section explains FLS used techniques and illustrates in details the proposed FL-RISSO model to solve the services selection problem in the IoT environment. In the end, the case study of the car parking system that applied the proposed FL-RISSO will be discussed.

4.8 .1 Proposed Solution Used FL-RISSO

To solve the service selection process in the IoT environment. The proposed model considers the dynamic IoT environment requirements because the QoS value for the provided services it's different from one location to another based on data retrieved from a massive number of sensors. The proposed FL-RISSO model applies two main steps: firstly, it aims to use a FLS to classify the search space into sub-clusters. The purpose of clustering is to identify groupings of data from an extensive data set to produce identical subsets of service behavior. Then implement the RI-SSO optimization algorithm into the sub-cluster that matches the end-user constraints. The actual behavior of the selection algorithm is that the required services must be selected from a large search space, which contains a set of services similar in their functionality but different in non-functional constraints. In the proposed FL-RISSO, the required services will be selected from a sub-cluster that is similar to the end-user requirements.

This research implements the singleton Fuzzifier operator. In a singleton Fuzzifier the input X_0 interprets as a fuzzy set A with the membership function $\mu_A(X)$ equals zero except at the point x_0 at which $\mu_A(X_0)$ equals one. During the development of the fuzzy inference engines, the input and the output variables are interpreted by membership functions. There are various types of membership functions. The proposed solution applies the trapezoidal (Trap) membership function. The

Trapezoidal function is defined by a lower limit a, an upper limit d, a lower support limit b, and an upper support limit c, where a < b < c < d. The Trap membership function is calculated as in equation (4.21) is utilized for illustrating the membership degree of the input and an output variable used in the fuzzy interface approach.

$$\mu_A(X) = \begin{cases} 0, & (x < a)\ or\ (x > d) \\ \frac{x-a}{b-a} & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c} & c \le x \le d \end{cases} \qquad (4.21)$$

The rule bases mapping the fuzzy input set to the fuzzy output set through the classical logic IF-THEN rules. It determines the degree of truth of the proposition fuzzy. The number of the rule bases equal (n^m) rules, where n, and m are the number of membership functions described via the inputs variables. Then the fuzzy inference engine combines the fuzzy input set and rule bases to produces the fuzzy output set. Also, this research is applied the Mamdani fuzzy implication.

At the last, the defuzzification is applied to map the fuzzy results from the space of fuzzy the control actions, defined over an output universe of discourse into a space of non-fuzzy (crisp) control actions. The Center of Gravity (COG) approach is applied to the suggested pattern in this research.

Table (4. 3): Applied FLC in the proposed solution

FLC Components	Applied Method
Fuzzification	Singleton Fuzzifier Operator
Membership Function	Trapezoidal Function
Rule Bases	IF-THEN Rules
Fuzzy Inference Engine	Mamdani Fuzzy Implication
Defuzzifier Method	Center of Gravity (COG)

The proposed model can be illustrated in the following steps:

Step1: End-user requires the specific service s_x with the set of QoS constraints $C1, C2, \ldots, Cn$. The FLS first evaluates these requirements as a single input set to the Fuzzifier components based on the IF-THEN rule base and FIE.

Step2: Then generate the crisp output value O_x of the required service by linking the input membership functions with the output membership function.

Step3: The FLS evaluates the QoS of each service in the search space separately, as the Fuzzifier input sets: $\{s_1, s_2, s_3, \ldots\ldots, s_N\}$, where N represents the number of available services in the search space.

Step4: Then, based on the IF-THEN rule base and FIE by using a Mamdani Fuzzy Implication; the output crisp sets :$\{O_1, O_2, O_3, \ldots\ldots, O_M\}$ are generated. It links the input membership functions with the output membership function.

Step5: The Defuzzifier using COG classifies the output crisp sets into the M sets or clusters based on the Defuzzifier design using equation (2.11). These processes of FLS are done dynamically in the IoT environment every time when an end-user requires a service to satisfy the dynamic natural of the IoT environment.

Step6: Then the crisp output value O_x is compared with the output crisp sets: $\{O_1, O_2, O_3, \ldots\ldots, O_M\}$ to identify the specific set O_S that matched the End-user requirement.

Step7: At the end, the RI-SSO optimization algorithm illustrated above is applied for O_S set only; this leads to reducing the search space for the selection process.

4.9 Case Study for SSA in Smart Parking Systems

To illustrate the proposed model FL-RISSO, the case study of a smart parking system in the IoT environment will be tested. The increase in car ownership makes it difficult to find parking spots, especially in densely populated areas. Searching for an appropriate parking spot with specific constraints; takes up potentially productive time, increases the number of cars circulating the roads, and introduces new challenges to the infrastructure. Searching for the appropriate parking results in

significant losses in productive time and money. In contrast, 35% of the overall commute time is allocated to a car parking spot. According to [97], drivers spend an average of 17 hours a year searching for parking spots. Also, the cost of wasted time and fuel per driver in the year is $345.

To help the drivers find comfortable parking spot with various quality metrics such as parking cost, the distance from a given point, and available time. This case study finds out how the service selection model is used to shorten the search time and increase the level of drivers' comfort in the dynamic IoT environment.

Let's say there is a driver Di who searches for an appropriate parking spot based on different QoS requirements. In this scenario, the functionality of the required service is the place to park the car, but there are different non-functional quality metrics requirements that must be considered. In this situation, three quality metrics will be considered:

Parking Cost (PC): It represents the amount paid for the renting of the parking spot PX per hour, such as 5SD per hour.

Distance length (DL): It represents the distance from a given point. It is a dynamic variable as its value changes based on the driver's distance from available parking spots.

Available time (AT): It represents the driver's time to park the vehicle in the parking.

There are many stakeholders ST= {S_1, S_2, …. S_n}, they have parking spots with various values of the three quality metrics. This study considers a hundred stakeholders.

$$QM = \begin{bmatrix} S_1Q_1, S_1Q_2, S_1Q_3 \\ S_2Q_1, S_2Q_2, S_2Q_3 \\ S_3Q_1, S_3Q_2, S_3Q_3 \\ . \\ . \\ S_{100}Q_1, S_{100}Q_2, S_{100}Q_3 \end{bmatrix} \qquad (4.23)$$

To find the optimal parking spot that satisfies the driver requirements, the proposed FL-RISSO model is applied. In this model, the FLS is applied to cluster the search space for the selection process. The main objective of clustering is to reduce the search space for required services, which reduces execution time and improves the selection process's performance. The proposed model declares in the following five steps:

Step1: Fuzzifier and evaluate the driver input values

The driver D_i input the quality metrics values for a Parking cost (PC), Distance length (DL), and Available time (AT) to the required parking spot. These input values can be crisp inputs such as a driver D_i defining PC equals 5SD, DL equal 150 Km, and AT equal 10 Hours. These three metrics are determined by the membership functions, as shown in Figure (4.8), (4.9), and (4.10) to generate a suitable the fuzzy input set.

Step2: Evaluate the driver input set

Then FLC evaluates these metrics based on the IF-THEN rule base, as shown in Table (4.4) and FIE. Then generate the crisp output value O_i of the required parking spot by linking the input membership functions with the output membership function.

Step3: Fuzzifier the available stakeholders' quality metrics values.

Secondly, the FLS evaluates the quality metrics for the one hundred available stakeholders, as Fuzzifier input sets :$\{s_1, s_2, s_3, \ldots\ldots, s_N\}$, where N represents the number of available stakeholders in the search space, which considers equal hundred in this case study.

Step4: Evaluate the available stakeholder's input sets.

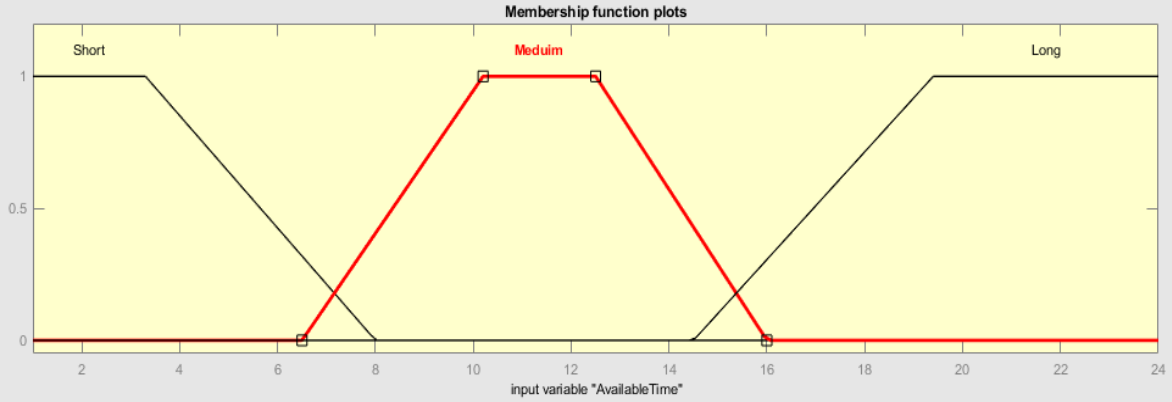
Then based on IF-THEN rule base as shown in Table (4.4) and FIE by using a Mamdani Fuzzy Implication; the output crisp sets :$\{O_1, O_2, O_3, \ldots\ldots, O_M\}$ are generated.

Step5: The Defuzzifier using COG classifies the output crisp sets into the M sets or clusters based on the Defuzzifier design. These FLS operations take place

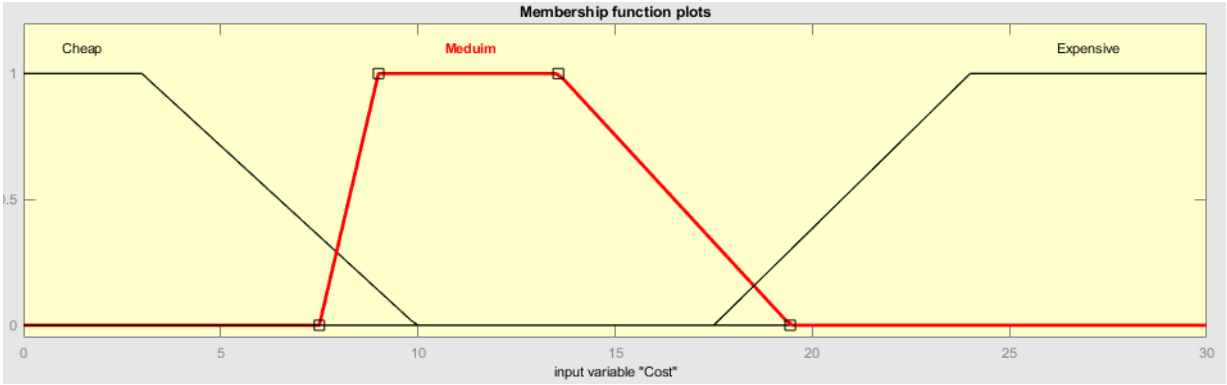
dynamically in the IoT environment every time the driver needs a parking place to satisfy the IoT environment's natural dynamics. Such as the distance length, which changes dynamically based on the driver's distance from available parking spots.

Step6: Then the crisp output value O_i is compared with the output crisp sets :$\{O_1, O_2, O_3, \ldots\ldots, O_M\}$ to identify the specific set O_S that matches the driver requirements.

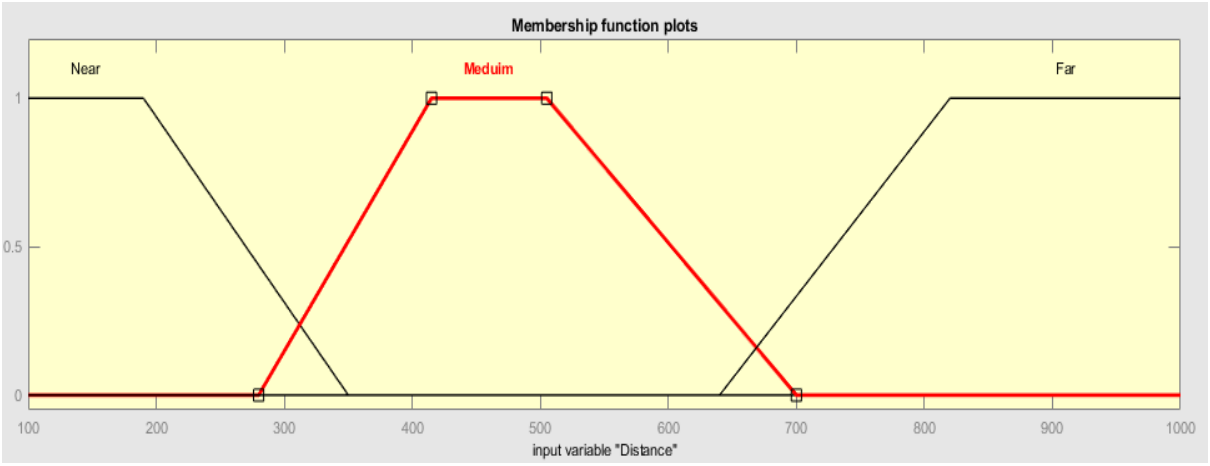
Step7: At the end, the RI-SSO optimization algorithm which illustrated above is applied for O_S set only; this leads to reducing the search space for the selection process.



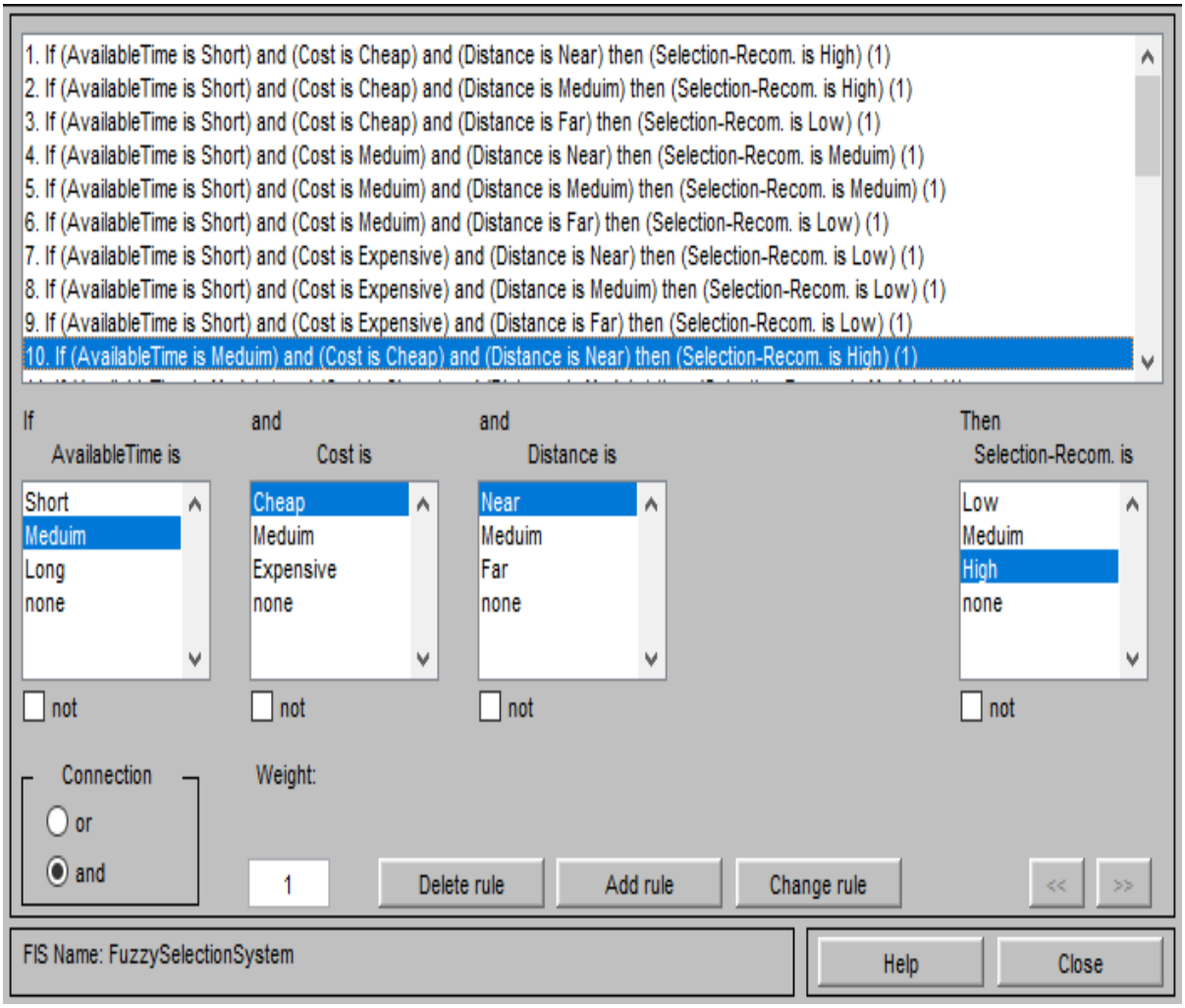
Figure(4. 7): Membership Function of the Available Time Input



Figure(4. 8): Membership Function of the Cost Input



Figure(4. 9): Membership Functions of the Distance Input



Figure(4. 10): Sample of Fuzzy Rules Editor Using MATLAB

Table (4. 4): Fuzzy (IF-THEN) Rule-Base Sets

Input			Output
Available Time	**Cost**	**Distance**	**Services Evaluation**
Short	Cheap	Near	High
Short	Cheap	Medium	High
Short	Cheap	Far	Low
Short	Medium	Near	Medium
Short	Medium	Medium	Medium
Short	Medium	Far	Low
Short	Expensive	Near	Low
Short	Expensive	Medium	Low
Short	Expensive	Far	Low
Medium	Cheap	Near	High
Medium	Cheap	Medium	Medium
Medium	Cheap	Far	High
Medium	Medium	Near	Medium
Medium	Medium	Medium	Medium
Medium	Medium	Far	Medium
Medium	Expensive	Near	Low
Medium	Expensive	Medium	Medium
Medium	Expensive	Far	Low
Long	Cheap	Near	High
Long	Cheap	Medium	High
Long	Cheap	Far	High
Long	Medium	Near	High
Long	Medium	Medium	Medium
Long	Medium	Far	Medium
Long	Expensive	Near	High
Long	Expensive	Medium	Low
Long	Expensive	Far	Low

Chapter Five Results and Discussion

Chapter Five: Results and Discussion

5.1 Introduction

The experiment settings and results of this thesis will be introduced in this chapter. The experimental settings include the applied software, dataset, and required QoS factors considered in the proposed solutions. The experiment results include the performance comparisons and feasibility validation of the proposed solutions.

In the beginning, the chapter discusses the efficiency of the first proposed model Likert-IPSO by comparing its performance with the performance of PSO and Improved-PSO. Then show and discuss the results of the second proposed model RI-SSO. Finally, the proposed FL-RISSO model results will be deliberated to demonstrate that FL-RISSO is an appropriate solution that reduces the selection process's execution time.

The three proposed models were performed on a laptop with Windows 10, 2.90 GHz processor, and 8GB RAM, and the algorithms were implemented using MATLAB R2017b. Also, to evaluate the proposed models in terms of efficiency and feasibility, an integer array-coding scheme is designed where the number of items in the array denotes the dimension or search space SP of the selection problem. Each element in the array represents an index of candidate service S with specific QoS value Q as in equation (5.1).

$$SP = \begin{bmatrix} S_1Q_1, S_1Q_2, S_1Q_3, \dots\dots\dots\dots, S_1Q_n \\ S_2Q_1, S_2Q_2, S_2Q_3, \dots\dots\dots\dots, S_2Q_n \\ S_3Q_1, S_3Q_2, S_3Q_3, \dots\dots\dots\dots, S_3Q_n \\ \vdots \\ S_mQ_1, S_mQ_2, S_mQ_3, \dots\dots\dots\dots, S_mQ_n \end{bmatrix} \quad (5.1)$$

Where *m* represents the number of abstract services, and *n* represents the number of concrete services for each abstract service *m*.

5.2 Discussion of the Proposed Solution Likert-IPSO

5.2.1 Likert-IPSO Experiment Settings

This section discusses the experimental settings of the Likert-IPSO that applies the Likert type measurement with improved-PSO. The dataset is randomly generated between 0 and 1.

The performance of regular PSO is compared to the Improved-PSO, and with the Likert-IPSO. The values of PSO's parameters are w= 1, c1= 1.50, and c2=2. The values of improved PSO's parameters are w= 0.729 and c1=c2= 1.49. In the experiments, the population size is set to 20.

5.2.2 Likert-IPSO Experimental Results

To analyze the efficiency of the proposed model Likert-IPSO, the execution time through the number of iterations of the three compared algorithms (PSO, Improved-PSO, and Likert Scale with Improved-PSO) is calculated. The result shows that the proposed approach takes less execution time compered to Improved-PSO and PSO only as in Figure (5.1). This result approves the efficiency of the proposed model. Moreover, it shows that the gradual increase in the number of iterations increases the disparity of time between the Likert-IPSO, Improved-PSO, and PSO.

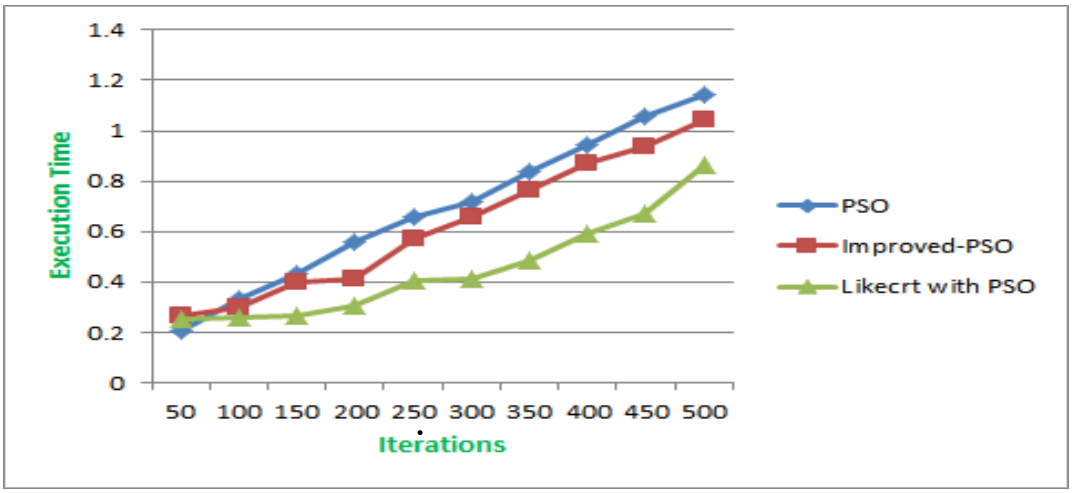


Figure (5. 1): Evolution Curves of Execution Time for PSO, Improved-PSO, and Likert-IPSO

To validate the Likert-IPSO model's feasibility, its fitness optimization values are compared with PSO and Improved-PSO over the previous setting. The results show that Likert Scale with PSO achieves better fitness values to the other algorithms, as shown in Figure (5.2).

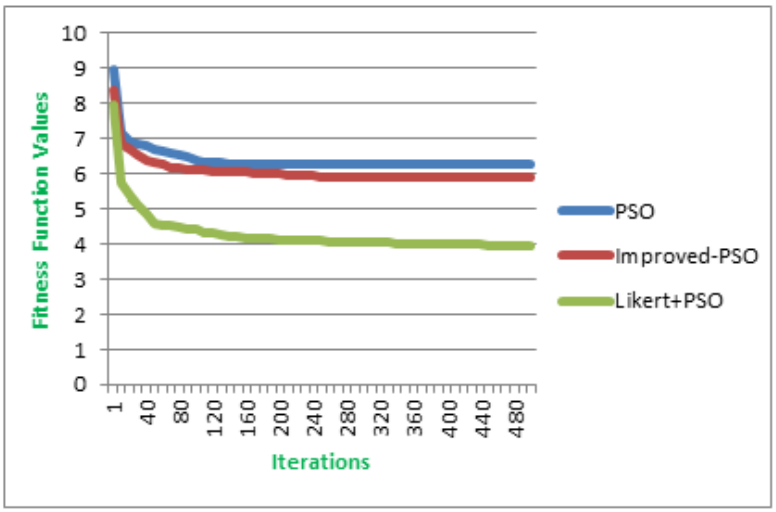


Figure (5. 2): Evolution Curves of Fitness Value for PSO, Improved-PSO, and Likert-IPSO

5.3 Discussion of the Proposed Solution RI-SSO

5.3.1 RI-SSO Experiment Settings

To evaluate the proposed model RI-SSO's performance, it is compared with the original SSO proposed by (Mousa and Bentahar, 2016) [82] to solve the selection problem. The selection problem's dimension is represented by the number of items in an integer array-coding scheme, as in equation (5.1). The indexes of candidate service are represented by the array elements. The parameter's value that is used in SSO and RI-SSO are [93]:

r_a = 1 represents the parameter that controls the attenuation rate of the vibration intensity over the distance.

p_c = 0.7, represent the user-defined attribute that describes the probability of changing mask.

p_m = 0.1, represent the user-controlled parameter defined between (0, 1).

In the experiments, the number of stopping criteria (iterations) is set to 150, and the population size is set to 20 for five abstract services. Each type of abstract service has a group of concrete services, as shown in Figure (5.3). These services have similar functional properties but differ in non-function properties (QoS factors). The introduced model, RI-SSO, regards five QoS factors (execution time, availability, cost, reliability, and reputation) for each concrete service. The dataset values for the first four factors (execution time, availability, cost, and reliability) are acquired from (Li 2013) [3], and the values of reputation factor are randomly generated based on five Likert-type values (0.2, 0.4, 0.6, 0.8, and 1.0) as shown in Figure (5.3).

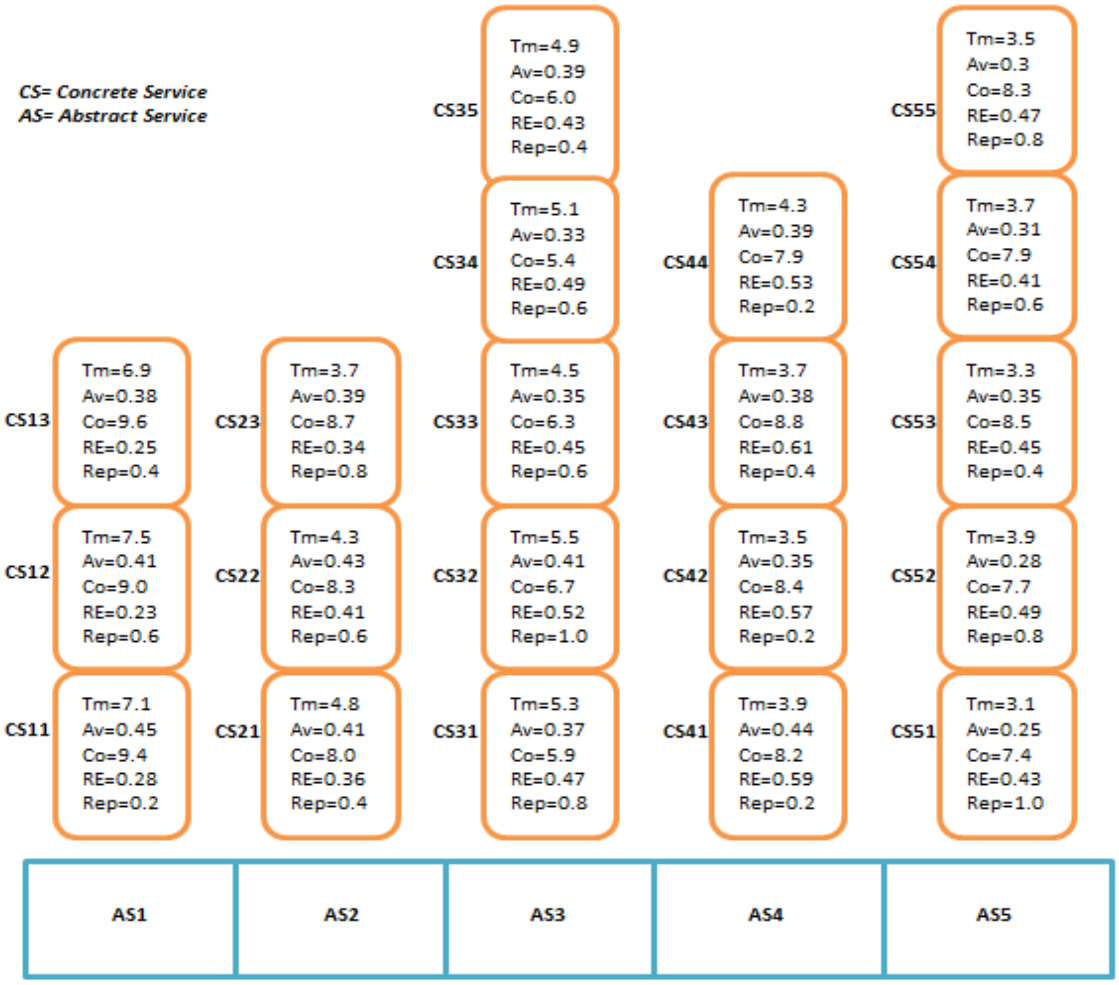


Figure (5. 3): Data Set of Services Selection Workflow [9]

5.3.2 RI-SSO Experimental Results

To analyze the RI-SSO model's performance, the behavior of Original SSO is compared with RI-SSO in two situations, one when the reputation value equals 0.2, which is the minimum point in Likert-type measurement. The other is when the reputation value equals 0.8, which is the previous maximum point in Likert-type measure.

The comparison is made based on three evaluations: firstly, it recorded the best intensity receive values obtained from the three comparisons in minimization and maximization problems. Results show that the increase in reputation value lead to an increase in the best intensity received values in the maximization problem, as in Figure (5.4). Furthermore, it leads to a decrease in the best intensity receive values in the minimization problems, as in Figure (5.5).

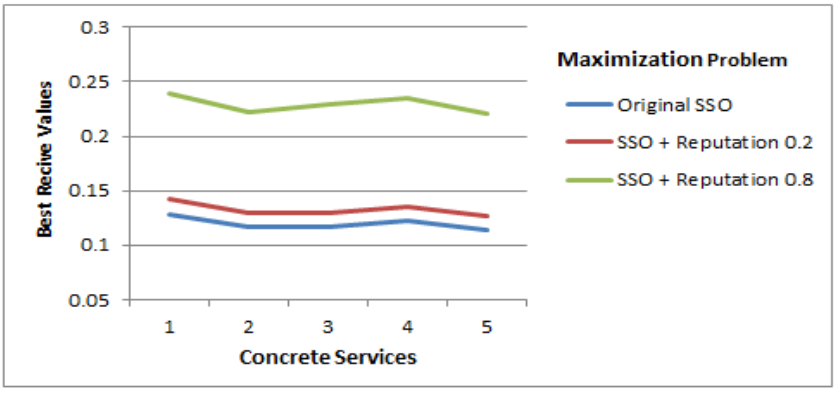


Figure (5. 4): Evolution Curves of Best Intensity Receive in Maximization Problem

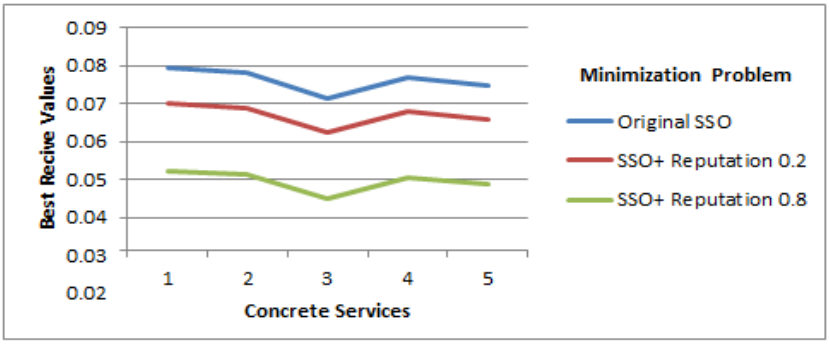


Figure (5. 5): Evolution Curves of Best intensity Receive in Minimization Problem

Secondly, the fitness values are observed over the two optimization types: It marked the availability factor's fitness value for the maximization problem. The results in Figure (5.6) show that the RI-SSO with reputation 0.8 gets the best performance by obtaining the highest fitness values compered to reputation 0.2 and Original SSO. The RI-SSO with reputation value equals 0.2 improves the availability of fitness value by 2%-6%. Moreover, when the reputation value equals 0.8, the fitness value improves by 17%-57%; (see Table (C) in the appendix for more details).

For the minimization problem, the fitness value of the cost factor is observed. The results in Figure (5.7) show that the RI-SSO with reputation 0.8 gets the best performance by obtaining the less fitness value than RI-SSO with reputation 0.2 and Original SSO. The RI-SSO with reputation value equals 0.2 improves the cost fitness value by 9%-17%, and when the reputation value equals 0.8, the fitness value improves by 16%-26%; (see Table (D) in the appendix for more details).

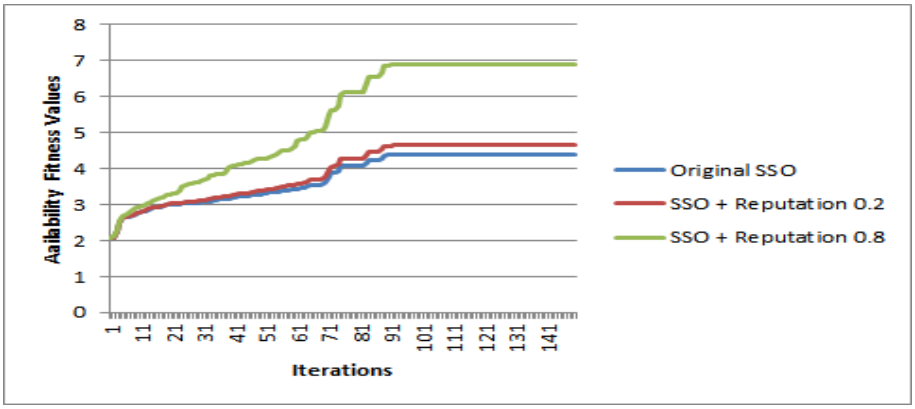


Figure (5. 6): Evaluation of Availability Fitness Value

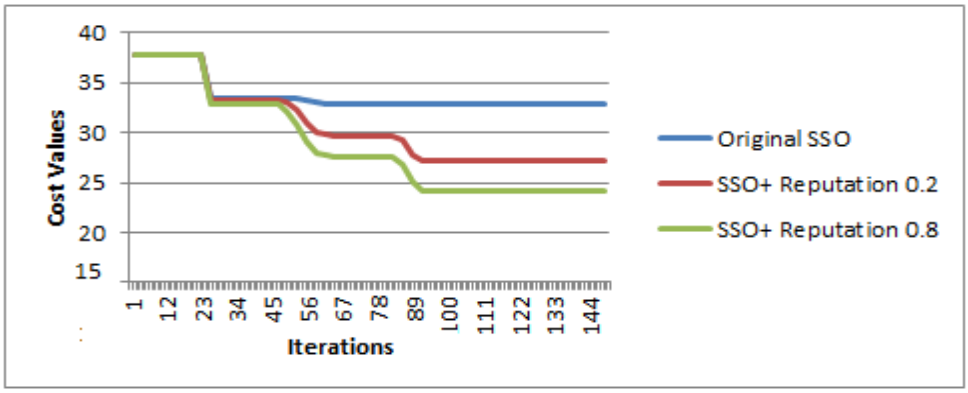


Figure (5. 7): Evaluation of Cost Fitness Value

Thirdly, to approve the feasibility and efficiency of RI-SSO, the QoS fitness values are calculated to optimize five factors. The weights of the factors are 0.25, 0.25, 0.15, 0.15, and 0.20 for response time(RT), availability(AV), execution price(EP), reliability(RE), and reputation(RP), respectively, as in equation (5.1).

QoS fitness function =

$$[\text{Min}(0.25*F_{RT}(CS_{ij}))+\text{Max}(0.25*F_{AV}(CS_{ij}))+\text{Min}(0.15*F_{EP}(CS_{ij}))+\text{Max}(0.15*F_{RE}(CS_{ij})) + \text{Max}(0.20*F_{RP}(CS_{ij}))] \quad (5.1)$$

Where, F_{QoS} (CS_{ij}) represents the summation function for each factor.

The results is shown in Figure (5.8) demonstrate that SSO with a high reputation value can achieve better fitness values in this selection problem than SSO with low reputation value and original SSO. As shown in Figure (5.8), SSO with a high reputation obtained the fitness value 14.6 in iteration eleven, the SSO with a low reputation obtained it at iteration eighteen, and the original SSO obtained it at iteration forty-four. The results show that the RI-SSO with a reputation value equals to 0.2 can improve the selection process by 7%-10%. While it can increases the performance by 21%-26% when the reputation value equals 0.8. (For more details, see Table (E) in the appendix).

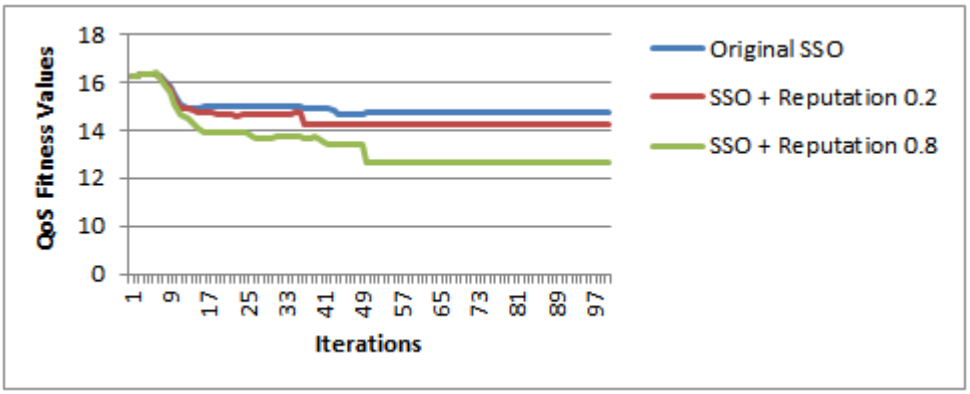


Figure (5. 8): Comparing Results of Fitness Function Values

Also, to validate the fitness function obtained from the RI-SSO, its value was compared with the obtained fitness value proposed by (Wenfeng.et al., 2013) [9]. The implementation applied the same dataset, which shown in Figure (5.3), and the exact weights of QoS factors. The weights of QoS factors are 0.28, 0.24, 0.3, and 0.18 for response time, availability, cost, and reliability, respectively. Thus demonstrate that the RI-SSO can perform more efficiency to optimize a selection problem. It got high availability and reliability values while keeping low cost and execution time values.

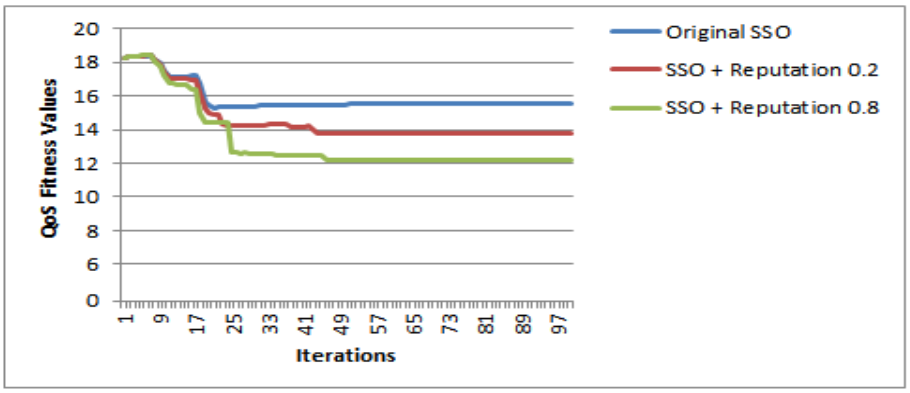


Figure (5. 9): Results of Fitness Function Values

The results show that the RI-SSO performes better than PSO, which was proposed in [9]. In [9], the minimum objective function value is 18.08 when the number of iterations is 15. This value 18.08 is obtained in iteration 8 when applied to the RI-SSO as in Figure (5.9). This means that the number of iterations in PSO is more than that in RI-SSO.

5.4 Discussion of the Proposed Solution FL-RISSO

5.4.1 FL-RISSO Experiment Settings

To evaluate the performance of the proposed model FL-RISSO, the experimental testing was done on a case study of the smart parking system in the IoT environment, explained in section (4.8.3). It applied a FLS on the search space area, which represents the available parking spots. FLS classifies the available parking spots into sub-clusters based on deriver requirements, containing three QoS factors

(parking cost, distance length, and available time). Then perform the selection process in the cluster that matches the driver requirements. In this case study, the available parking spots were set to 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000, (see Table (F) in the appendix form more details). The rule-based sets are implemented as in table (4.4). The driver requirements are classified into one of three clusters (low, medium, and high) as in Figures (5.10), (5.11), and (5.12), respectively.

Figure (5. 10): Sample of Fuzzy Rule Viewer Using MATLAB (Low-Output)

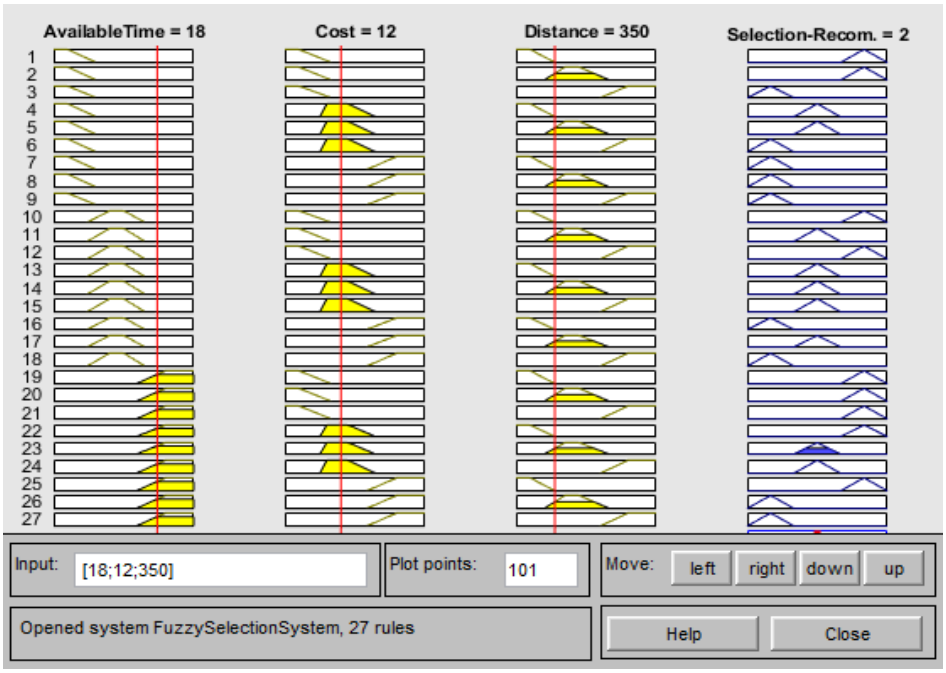


Figure (5. 11): Sample of Fuzzy Rule Viewer Using MATLAB (Medium-Output)

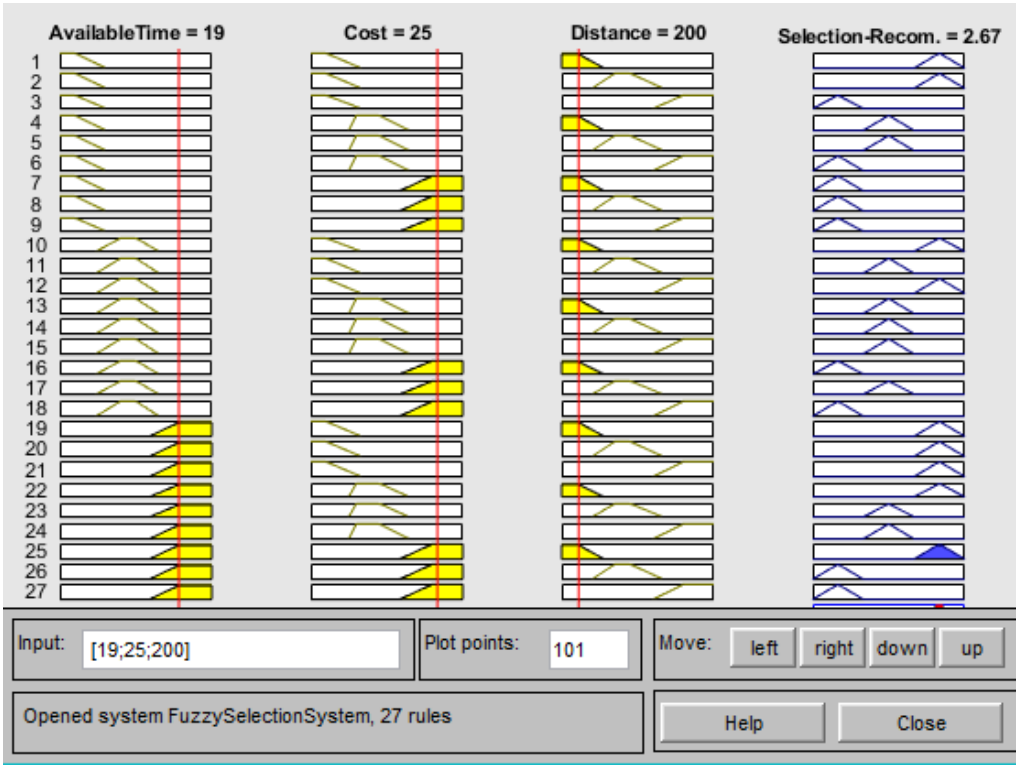


Figure (5. 12): Sample of Fuzzy Rule Viewer Using MATLAB (High-Output)

5.4.2 FL-RISSO Experimental Results

The experimental results show that applying of FLS on the search space leads to reducing it to 64%-71%, (see Table (F) in the appendix for more details). The comparison was done between implementing the RI-SSO model overall search space and implementing it in a sub-cluster that matched the end-user requirements. The results show that the gradual increase in the number of concrete services increases the disparity of execution time between the proposed FL-RSSO and RI-SSO models. It indicates that the FL-RSSO reduces the execution time by 15%-85%, as in Figure (5.10). For instance, when the concrete service before clustering is 200, the execution time is decreased by 29% after clustering. When the concrete service before clustering is 900, the execution time is reduced by 83% after clustering, (see Table (G) in the appendix).

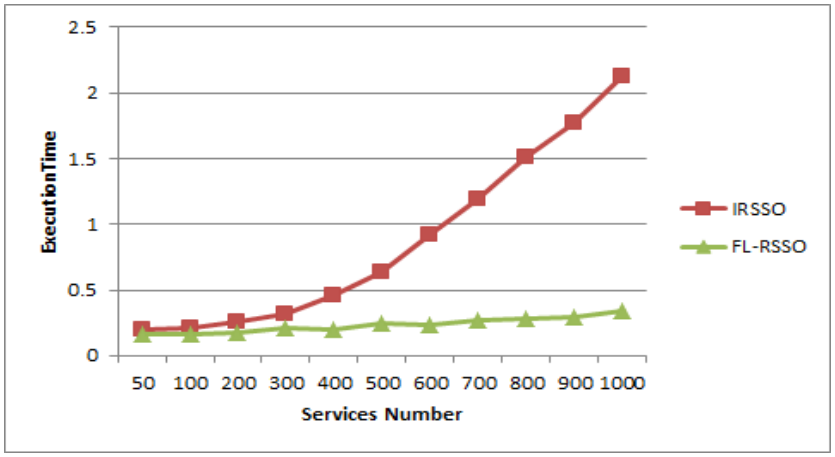


Figure (5. 13): Comparing Results of Execution Time

5.5 Summary of results and discussion

In the end, the experimental results prove that using feedback or reputation value improves the performance of selection algorithms. Utilizing a reputation with PSO improves PSO's performance by increasing the fitness value and reducing the execution time. Also, using the reputation value with SSO leads to increasing the fitness value in the maximization and reducing it in the minimization problems. Moreover, the experiment shows that using a FLS to classify the search space to sub-clusters reducing execution time by a high percentage.

Chapter Six Conclusions and Future Work

Chapter Six: Conclusions and Future Work

6.1 Conclusions

IoT environments content a massive number of actuators that provide different services for end-users to facilitate their life. Similar actuators can provide services with similar functionality but various non-functional requirements. So the end-users can request the same service with different QoS criteria. This makes the process of selecting the required service with the optimal QoS an NP-hard problem. This research proposed gathering feedback about services from end-users who used it to improve the selection process. This information was collected through a user-friendly tool named Likert-type measurement. The average value of end-users feedbacks saves as reputation value for each service.

This research also proposed a novel model called Improved-Social Spider Optimization (RI-SSO) that enhances the behavior of an existing meta-heuristic algorithm called SSO. It adds the reputation value to the new offspring in the SSO model by updating its fitness weight values based on an average calculation of feedback information. The RI-SSO upgrades the search process's reliability by selecting a service that appropriates the end-user preference. Comparisons found that the proposed model RI-SSO improves the fitness function value compared to the Original SSO.

Moreover, this research was introduced to classify the search space into sub-clusters using the FLS. Then the selection process was performed into the sub-cluster that matches the end-user requirements.

The experimental results prove the efficiency of the proposed models in the context of fitness value and execution time.

6.2 Future Work

In future work, we aim to build an entire data set that satisfies the QoS requirements for services in IoT environments such as smart parking spots, smart cities, smart homes, industrial automation, health care, traffic management, and others. This is because in experimental evaluations, most of the applied data sets were generated randomly, and some used existing data sets that were not explicitly constructed for IoT environments.

We recommend using historical data with prediction tools, such as machine learning, to predict the quantities and qualitative QoS factors, which end-user will require in the future.

Biography

[1] R. Nicolescu, M. Huth, P. Radanliev, and D. De Roure, “State of The Art in IoT - Beyond Economic Value,” 2018.

[2] D. Evans, “The Internet of Things - How the Next Evolution of the Internet is Changing Everything,” 2011.

[3] A. Maaradji, “End-user service composition from a social networks analysis perspective,” *Inst. Natl. des Télécommunications*, 2012.

[4] Y. Xia, P. Chen, L. Bao, and M. Wang, “2011 IEEE International Conference on Web Services,” in *A QoS-Aware Web Service Selection Algorithm Based On Clustering*, 2011, p. 8.

[5] B. Huang, C. Li, and F. Tao, “A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system,” *Enterp. Inf. Syst.*, vol. 8, no. 4, pp. 445–463, 2014.

[6] M. Sun, Z. Shi, S. Chen, Z. Zhou, and Y. Duan, “Energy-Efficient Composition of Configurable Internet of Things Services,” *IEEE Access*, pp. 1–14, 2017.

[7] K. Dhondge, R. Shorey, and J. Tew, “HOLA: Heuristic and opportunistic link selection algorithm for energy efficiency in Industrial Internet of Things (IIoT) systems,” in *2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016*, 2016, pp. 1–6.

[8] M. Mejri and N. Ben Azzouna, “Scalable and Self-Adaptive Service Selection Method for the Internet of Things,” *Int. J. of computer Appl.*, vol. 167, no. 10, pp. 43–49, 2017.

[9] W. Li, Y. Zhong, X. Wang, and Y. Cao, “Resource virtualization and service selection in cloud logistics,” *J. Netw. Comput. Appl.*, pp. 1–9, 2013.

[10] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, “Sensor search techniques for sensing as a service architecture for the internet of things,” *IEEE Sens. J.*, vol. 14, no. 2, pp. 406–420, 2014.

[11] J. Liu *et al.*, “A Cooperative Evolution for QoS-driven IoT Service Composition,” *Autom. – J. Control. Meas. Electron. Comput. Commun.*, vol. 54, no. 4, pp. 438–447, 2013.

[12] J. Zhou and X. Yao, “A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition,” *Int. J. Adv. Manuf. Technol.*, 2016.

[13] M. E. Khanouche, Y. Amirat, A. Chibani, M. Kerkar, and A. Yachir, “Energy-Centered and QoS-Aware Services Selection for the Internet of Things,” *IEEE Trans. Autom. Sci. Eng.*, pp. 1–14, 2016.

[14] Z. Huang, K. Lin, S. Yu, and J. Y. Hsu, “Co-locating services in IoT systems to minimize the communication energy cost,” *J. Innov. Digit. Ecosyst.*, 2015.

[15] Z. Huang, K. J. Lin, S. Y. Yu, and J. Y. J. Hsu, “Building energy efficient internet of things by Co-locating services to minimize communication,” in *MEDES 2014 - 6th International Conference on Management of Emergent Digital EcoSystems, Proceedings*, 2014, pp. 101–108.

[16] “internet-of-things-history.” [Online]. Available: https://www.postscapes.com/internet-of-things-history .

[17] D. Roberto#Minerva,#Abyi#Biru, “Towards a Definition of the Internet of Things (IoT),” *IEEE Org.* [Online]. Available: http://iot.ieee.org/definition.html.

[18] V. Bhuvaneswari and R. Porkodi, “The internet of things (IoT) applications and communication enabling technology standards: An overview,” *Proc. - 2014 Int. Conf. Intell. Comput. Appl. ICICA, 2014*, pp. 324–329, 2014.

[19] W. Muschik, “Future Intelligent Vehicular Technologies,” *Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng.*, vol. 24, no. 6, pp. 876–882, 2016.

[20] M. Vladimer, “Sensors, Actuators and IoT.” [Online]. Available: https://www.linkedin.com/pulse/sensors-actuators-iot-mike-vladimer.

[21] M. Burhan, R. A. Rehman, B. Khan, and B. S. Kim, “IoT elements, layered architectures and security issues: A comprehensive survey,” *Sensors (Switzerland)*, vol. 18, no. 9, pp. 1–37, 2018.

[22] R.Minerva, “Toward a Definition of Internet of Things,” 2016.

[23] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[24] M. O. Elbasheer, “Architecture Proposal for MCloud IoT Architecture Proposal for MCloud IoT,” no. December 2018, 2017.

[25] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti, and Subhajit Dutta, “Role Of Middleware For Internet Of Things: A Study,” *Int. J. Comput. Sci. Eng. Surv.*, vol. 2, no. 3, pp. 94–105, 2011.

[26] IBM® and I. K. Center, “Web services: Key roles.”.

[27] W. Hugo Haas and M. (until J. 2002) Allen Brown, “Web Services Glossary,” *W3C Working Group*. [Online]. Available: https://www.w3.org/TR/ws-gloss/. [Accessed: 20-Jan-2018].

[28] F. Mustafa and T. L. McCluskey, “Dynamic web service composition,” in *Proceedings - 2009 International Conference on Computer Engineering and Technology, ICCET 2009*, 2009, vol. 2, pp. 463–467.

[29] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, “Web services composition: A decade’s overview,” *Inf. Sci. (NY).*, vol. 280, pp. 218–238, 2014.

[30] A. L. Lemos, F. Daniel, and B. Benatallah, “Web Service Composition: A Survey of Techniques and Tools,” *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–41, 2015.

[31] Z. Z. Liu, X. Xue, J. Q. Shen, and W. R. Li, “Web service dynamic composition based on the decomposition of global QoS constraints,” *Int. J. Adv. Manuf. Technol.*, vol. 69, no. 9–12, pp. 2247–2260, 2013.

[32] S. Cherrier, Y. M. Ghamri-Doudane, S. Lohier, and G. Roussel, “D-LITe: Building Internet of Things Choreographies,” 2016.

[33] S. Elfirdoussi and Z. Jarir, “An integrated approach towards service composition life cycle: A transportation process case study,” *J. Ind. Inf. Integr.*, vol. 15, no. May 2018, pp. 138–146, 2019.

[34] M. Elqortobi, J. Bentahar, and R. Dssouli, “Framework for Dynamic Web Services Composition Guided by Live Testing,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2018, vol. 206, no. 5, pp. 129–139.

[35] M. Speegle, *Quality Concepts for the Process Industry*, 2nd ed. Gengage Learning.

[36] ISO 9126, “Information technology — Software product quality,” *Iso/Iec Fdis 9126-1*, vol. 2000, pp. 1–26, 2000.

[37] P. Milano and P. Plebani, “Quality of Web services,” *Quality*, no. May 2006.

[38] E. Kim and Y. Lee, “Quality Model for Web Services,” no. September, pp. 1–45, 2005.

[39] OASIS, “Web Services Quality Factors Version 1. 0 Candidate OASIS Standard 01,” *OASIS Open 2012*, 2012.

[40] L. Li, S. Li, and S. Zhao, “QoS-Aware scheduling of services-oriented internet of things,” *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1497–1507, 2014.

[41] M. Marjanović and L. Skorin-kapov, “Quality of service (QoS) for IoT services,” 2014.

[42] M. M. Rai, “Introduction to Optimization and Multidisciplinary Design Single-and Multiple-Objective Optimization with Differential Evolution and Neural Networks,” *NASA Ames Res. Cent.*, pp. 1–32, 2006.

[43] P. Caramia, Massimiliano, Dell’Olmo, “Multi-objective Optimization,” in *Multi-objective Management in Freight Logistics*, 2008, pp. 11–37.

[44] C. C. Lin, D. J. Deng, and A. L. Y. Lu, “Many-Objective Sensor Selection in IoT Systems,” *IEEE Wirel. Commun.*, vol. 24, no. 3, pp. 40–47, 2017.

[45] N. M. H. Jiménez-Sáez, “Multi-Criteria Decision-Making, Evolution and Characteristics,” in *International Series in Operations Research & Management Science*, Springer, Cham, 2019, pp. 3–13.

[46] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, “A survey on metaheuristics for stochastic combinatorial optimization,” *Nat. Comput.*, vol. 8, no. 2, pp. 239–287, 2009.

[47] P. R. and S. S. Edmund Burke, Emma Hart, Graham Kendall, Jim Newall, “Hyper-Heuristics: An Emerging Direction in Modern Search Technology,” *ResearchGate*, no. January, pp. 0–23, 2003.

[48] M. Gendreau and J.-Y. Potvin, “A Classification of Hyper-heuristic Approaches Edmund,” *Springer-Verlag*, vol. 272, no. July 2014, 2010.

[49] F. Dernoncourt, “Introduction to fuzzy logic1,” *IECON Proc. (Industrial Electron. Conf.*, vol. 1, no. January, pp. 50–56, 2013.

[50] J. J.Buckley, *An Introduction to Fuzzy Logic and Fuzzy Sets*, vol. 91, no. 5. Advances in soft

computing, 2002.

[51] N. P. Says, “Difference Between Fuzzy Set and Crisp Set,” *Tech Differences*. [Online]. Available: https://techdifferences.com/difference-between-fuzzy-set-and-crisp-set.html#:~:text=A fuzzy set is determined, the location of the set boundaries.

[52] P. Vojtáš, “Fuzzy logic as an optimization task,” *Proc. - 4th Conf. Eur. Soc. Fuzzy Log. Technol. 11th French Days Fuzzy Log. Appl. EUSFLAT-LFA, 2005, Jt. Conf.*, no. 1, pp. 781–786, 2005.

[53] A. Orooji, M. Langarizadeh, M. Hassanzad, and M. R. Zarkesh, “A Comparison Between Fuzzy Type-1 and Type-2 Systems in Medical Decision Making: A Systematic Review,” *Crescent J. Med. Biol. Sci.*, vol. 6, no. 3, pp. 246–252, 2019.

[54] Matlab Workers, “types-of-fuzzy-inference-systems,” *MatlabWorkers*, 2013. [Online]. Available: https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html. [Accessed: 13-Aug-2020].

[55] J. Na, K. Lin, Z. Huang, and S. Zhou, “An Evolutionary Game Approach on IoT Service Selection for Balancing Device Energy Consumption,” in *IEEE 12th International Conference on e-Business Engineering*, 2015, pp. 331–338.

[56] M. P. K. Reddy and M. R. Babu, “Energy Efficient Cluster Head Selection for the Internet of Things Energy Efficient Cluster Head Selection for the Internet of Things,” *New Rev. Inf. Netw.*, vol. 22, no. 1, pp. 54–70, 2017.

[57] X. Yin and J. Yang, “Shortest paths based web service selection in the internet of things,” *J. Sensors*, vol. 2014, 2014.

[58] M. Anas, S. Khatab, and A. Badr, “HeuristicIoT: A framework for augmenting heuristic search algorithms by internet-of-things data,” *Int. J. Comput. Inf. Sci.*, vol. 12, no. 1, p. 4, 2016.

[59] F. Gao, E. Curry, M. I. Ali, S. Bhiri, and A. Mileo, “QoS-aware complex event service composition and optimization using genetic algorithms,” 2014, vol. 8831, pp. 386–393.

[60] S. Abinaya, G. Akshaya, and J. Dhivya, “Minimizing Energy Consumption Using Internet of Things,” *Int. J. Recent Innov. Trends Comput. Commun.*, no. March, pp. 67–70, 2017.

[61] C. G. 1. NWE Nwe Htay Win1(?), BAO Jian-min2, “Flexible user-centric service selection algorithm for the Internet of Things services NWE,” *J. China Univ. Posts Telecommun.*, pp. 64–70, 2014.

[62] X. Jin, S. Chun, J. Jung, and K. Lee, “IoT Service Selection based on Physical Service Model and Absolute Dominance Relationship,” in *IEEE International Conference on Service-Oriented Computing and Application*, 2014, pp. 65–72.

[63] X. Jin, S. Chun, J. Jung, and K. Lee, “A fast and scalable approach for IoT service selection based on a physical service model,” *Inf. Syst. Front.*, 2016.

[64] H. Quan, R. Takahashi, and F. Yoshiaki, “Dynamic service selection based on user feedback in the IoT environment,” in *CITS 2019 - Proceeding of the 2019 International Conference on Computer, Information and Telecommunication Systems*, 2019, pp. 1–5.

[65] S. Y. Yu, C. S. Shih, J. Y. J. Hsu, Z. Huang, and K. J. Lin, "QoS oriented sensor selection in IoT system," in *Proceedings - 2014 IEEE International Conference on Internet of Things, iThings 2014, 2014 IEEE International Conference on Green Computing and Communications, GreenCom 2014 and 2014 IEEE International Conference on Cyber-Physical-Social Computing, CPS 20*, 2014, no. iThings, pp. 201–206.

[66] Z. Huang, K. J. Lin, C. Li, and S. Zhou, "Communication energy aware sensor selection in IoT systems," in *Proceedings - 2014 IEEE International Conference on Internet of Things, iThings 2014, 2014 IEEE International Conference on Green Computing and Communications, GreenCom 2014 and 2014 IEEE International Conference on Cyber-Physical-Social Computing, CPS 20*, 2014, no. iThings, pp. 235–242.

[67] Z. Huang, K. J. Lin, and L. Han, "An energy sentient methodology for sensor mapping and selection in IoT systems," in *IEEE International Symposium on Industrial Electronics*, 2014, pp. 1436–1441.

[68] J. Shukla, P. Maiti, and B. Sahoo, "Low latency and energy-efficient sensor selection for IoT services," in *International Conference on Technologies for Smart City Energy Security and Power: Smart Solutions for Smart Cities, ICSESP 2018 - Proceedings*, 2018, vol. 2018-Janua, pp. 1–5.

[69] M. Elhoseny, A. Abdelaziz, A. S. Salama, A. M. Riad, K. Muhammad, and A. K. Sangaiah, "A Hybrid Model of Internet of Things and Cloud Computing to Manage Big Data in Health Services Applications," *Futur. Gener. Comput. Syst.*, 2018.

[70] O. Alsaryrah, I. Mashal, and T. Chung, "Bi-Objective Optimization for Energy-Aware Internet of Things Service Composition," *IEEE Access*, vol. 6, pp. 26809–26819, 2018.

[71] Y. Yuan, W. Zhang, X. Zhang, and H. Zhai, "Dynamic service selection based on adaptive global QoS constraints decomposition," *Symmetry (Basel).*, vol. 11, no. 3, 2019.

[72] M. Hosseinzadeh *et al.*, "A Hybrid Service Selection and Composition Model for Cloud-Edge Computing in the Internet of Things," *IEEE Access*, vol. 8, pp. 85939–85949, 2020.

[73] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-Aware QoS Prediction with Neural Collaborative Filtering for Internet-of-Things Services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, 2020.

[74] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm," *Futur. Gener. Comput. Syst.*, vol. 94, pp. 185–198, 2019.

[75] F. Khan *et al.*, "A quality of service-aware secured communication scheme for the internet of things-based networks," *Sensors (Switzerland)*, vol. 19, no. 19, pp. 1–18, 2019.

[76] A. N. Abu-safe and S. E. Elrofai, "QOS – Aware Meta-Heuristic Services Selection Algorithm and Likert Scale Measurement for IoT Environment," *Int. J. Comput. Sci. Trends Technol.*, vol. 8, no. 1, pp. 1–8, 2020.

[77] M. Singh, G. Baranwal, and A. K. Tripathi, "QoS-Aware Selection of IoT-Based Service," *Arab. J. Sci. Eng.*, vol. 20, no. 2, 2020.

[78] E. Al-Masri's, "QWS-Datasets," *University of Guelph & University of Washington Tacoma*,

2007. [Online]. Available: https://qwsdata.github.io/.

[79] P. Asghari, A. M. Rahmani, H. Haj, and S. Javadi, “Service Composition approaches in IoT: A Systematic Review,” *J. Netw. Comput. Appl.*, 2018.

[80] ITU-T, “Recommendation E.860: Framework of a Service Level Agreement,” *Ser. E Overall Netw. Oper. Teleph. Service, Serv. Oper. Hum. Factors*, 2002.

[81] K. Sörensen, M. Sevaux, and F. Glover, “A history of metaheuristics,” in *Handbook of Heuristics Springer, 2016*, vol. 2–2, 2016, pp. 791–808.

[82] A. Mousa and J. Bentahar, “An Efficient QoS-aware Web Services Selection using Social Spider Algorithm,” *Procedia - Procedia Comput. Sci.*, vol. 94, no. MobiSPC, pp. 176–182, 2016.

[83] and T. I. Ahlem Ben Hassine, Shigeo Matsubara, “A Constraint-Based Approach to Horizontal Web Service Composition,” *Int. Semant. Web Conf.*, vol. 4273, no. November, pp. 172–186, 2006.

[84] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, “Quality of service for workflows and web service processes,” *Web Semant.*, vol. 1, no. 3, pp. 281–308, 2004.

[85] Y. Li, Z. Luo, and J. Yin, “A LOCATION-AWARE SERVICE SELECTION MODEL,” *Serv. Trans. Serv. Comput.*, vol. 1, no. 1, pp. 52–63, 2013.

[86] X. Mei, F. Zheng, A. Jiang, and S. Li, “QoS aggregation evaluation of web services composition with the transaction,” in *Proceedings - 2009 International Conference on Information Technology and Computer Science, ITCS 2009*, 2009, vol. 2, pp. 151–155.

[87] D. Bertram, “Likert Scales. Topic Report,” *Fac. Math. – Univ. Belgrade – Serbia*, 2009.

[88] A. D. Averin, A. A. Yakushev, O. A. Maloshitskaya, S. A. Surby, O. I. Koifman, and I. P. Beletskaya, “Synthesis of porphyrin-diazacrown ether and porphyrin-cryptand conjugates for fluorescence detection of copper(II) ions,” *Russ. Chem. Bull.*, vol. 66, no. 8, pp. 1456–1466, 2017.

[89] M. Nabab, “Particle Swarm Optimization: Algorithm and its Codes in MATLAB,” *ResearchGate*, no. 1, pp. 8–12, 2016.

[90] A. L. Ballardini, “A tutorial on Particle Swarm Optimization Clustering,” *arXiv Prepr.*, vol. arXiv:1809, 2018.

[91] S. A. Ludwig, “Applying particle swarm optimization to quality-of-service-driven web service composition,” in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 2012, pp. 613–620.

[92] E. Cuevas, M. A. Díaz-Cortés, and D. A. Oliva Navarro, “A swarm global optimization algorithm inspired in the behavior of the social-spider,” in *Studies in Computational Intelligence*, vol. 629, no. 16, 2016, pp. 9–33.

[93] J. J. Q. Yu and V. O. K. Li, “A social spider algorithm for global optimization,” *Appl. Soft Comput. J.*, vol. 30, pp. 614–627, 2015.

[94] E. Cuevas, M. Cienfuegos, E. Cuevas, and M. Cienfuegos, “A new algorithm inspired in the

behavior of the social-spider for constrained optimization," *Expert Syst. Appl.*, 2013.

[95] "Byjus learning App," *Byjus learning App*. [Online]. Available: https://byjus.com/intensity-formula/. [Accessed: 30-May-2020].

[96] U. P. Shukla and S. J. Nanda, "A Binary Social Spider Optimization algorithm for unsupervised band selection in compressed hyperspectral images," *Expert Syst. Appl.*, vol. 97, pp. 336–356, 2018.

[97] K. McCoy, "Drivers spend an average of 17 hours a year searching for parking spots," *USA TODAY*. [Online]. Available: https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/.

Appendix

Table (A): Parking Spots Dataset

Available Time	Parking Cost	Distance Length
10	25	645
10	19	362
20	19	255
14	2	818
14	3	348
1	11	630
16	17	108
10	5	678
16	23	123
13	4	553
14	11	446
4	13	349
16	25	417
5	27	350
15	28	746
21	28	533
12	25	906
10	23	713
21	16	381
21	18	705
10	27	347
20	3	519
3	25	792
21	8	226
24	19	207
20	24	725
22	0	691
20	6	255
7	14	853
1	21	806
21	14	948
1	6	617
4	11	625
24	5	971
10	2	100
15	11	247
7	0	650
14	26	138
2	22	391
6	5	502
8	17	103

Table (A): Continued

Available Time	Parking Cost	Distance Length
5	15	750
24	9	960
13	4	323
10	7	108
15	14	720
3	25	330
6	12	547
22	10	205
6	16	954
4	3	598
8	27	918
14	23	914
13	18	429
14	10	856
2	18	134
11	27	826
1	1	533
21	8	168
20	4	223
3	17	133
3	17	402
24	17	226
2	24	633
2	14	204
1	23	541
15	28	632
5	10	679
3	22	281
9	16	744
6	22	302
16	4	962
13	3	277
5	12	415
15	23	769
11	17	668
11	16	669
19	14	229
2	27	664
9	25	255
10	1	667

Table (A): Continued		
Available Time	Parking Cost	Distance Length
24	27	240
11	17	850
19	26	110
20	20	511
14	0	865
3	11	388
4	25	165
4	12	473
1	4	672
21	24	710
10	12	877
20	10	460
8	2	927
22	11	548
8	14	732
17	4	665
5	4	394
8	18	984

Table (B): Parking Spot Dataset with FLS Evaluation

Available Time	Parking Cost	Distance Length	FLS-Evaluation
24	1	890	2.67
24	27	889	1.33
10	7	227	2.67
23	24	801	1.33
16	1	422	2.67
14	26	891	1.33
21	0	952	2.67
16	8	239	2.67
13	20	862	1.33
1	6	693	1.5
11	24	242	1.33
4	3	423	2.67
6	29	638	1.33
14	21	308	1.63
17	12	284	2.61
17	1	636	2.67
15	16	263	2.19
11	2	173	2.67
3	30	294	1.33
7	5	324	2.4
16	9	144	2.67
10	0	416	2
7	29	670	1.63
2	2	735	1.33
17	0	606	2.67
11	14	124	2
16	6	505	2.67
21	13	519	2
2	28	131	1.33
12	22	709	1.33
7	23	125	1.33
23	8	511	2.31
2	13	226	2
5	20	872	1.33
17	20	960	1.33
12	1	100	2.67
10	15	104	2
15	30	395	1.81
12	17	418	2
22	17	556	2

Table (B) Continued

Available Time	Parking Cost	Distance Length	FLS-Evaluation
13	9	245	2.15
5	1	715	1.33
12	17	435	2
22	9	424	2.14
16	30	909	1.33
12	23	748	1.33
9	19	595	2
8	29	214	1.33
19	23	810	1.33
11	17	874	2
6	19	832	1.33
16	13	579	2
2	21	298	1.33
14	5	186	2.67
6	17	242	2
16	18	867	1.83
11	16	358	2
24	12	956	2
12	29	152	1.33
2	10	856	1.33
7	22	744	1.33
3	20	790	1.33
15	29	535	1.81
8	22	318	1.72
7	28	360	1.6
8	8	581	2
24	13	901	2
24	17	938	2
14	12	554	2
11	29	521	2
10	30	931	1.33
13	15	306	2
17	9	370	2.17
17	3	116	2.67
18	12	871	2
6	22	149	1.33
17	15	140	2.67
8	5	765	2.67
8	17	498	2
10	26	172	1.33

Table (B) Continued

11	26	669	1.67
8	23	504	2
22	24	746	1.33
4	8	492	2.31
18	4	434	2.67
2	16	414	2
22	4	645	2.67
4	20	915	1.33
24	15	344	2.06
19	6	286	2.67
19	12	663	2
14	29	822	1.33
18	1	627	2.67
5	25	124	1.33
24	30	781	1.33
24	21	555	1.33
4	26	622	1.33
5	5	222	2.67
16	11	634	2
17	14	760	2

Table (C): Differentiate Percentages in the Availably Fitness Values by using Original SSO, and RI_SSO

Iteration	Original SSO	SSO+0.2	SSO+0.8	Improvement Percentage for Reputation equal 0.2	Improvement Percentage for Reputation equal 0.8
1	2.08	2.08	2.08	0%	0%
25	3.027057	3.060361	3.555905	1.1%	17.4%
50	3.300107	3.394835	4.280225	2.8%	29.6%
75	4.076443	4.287478	6.027024	5.1%	47.8%
100	4.374072	4.640585	6.909961	6.0%	57.9%
125	4.374072	4.640585	6.909961	6.0%	57.9%
150	4.374072	4.640585	6.909961	6.0%	57.9%

Table (D): Differentiate Percentages in the Cost Fitness Values by using Original SSO, and RI_SSO

Iteration	Original SSO	SSO+0.2	SSO+0.8	Improvement Percentage for Reputation equal 0.2	Improvement Percentage for Reputation equal 0.8
1	2.08	2.08	2.08	0%	0%
25	3.027057	3.060361	3.555905	9.5%	16.0%
50	3.300107	3.394835	4.280225	17.3%	26.3%
75	4.076443	4.287478	6.027024	17.3%	26.3%
100	4.374072	4.640585	6.909961	17.3%	26.3%
125	4.374072	4.640585	6.909961	17.3%	26.3%
150	4.374072	4.640585	6.909961	17.3%	26.3%

Table (F): Differentiate Percentages in the QoS Fitness Values

Iteration	Original SSO	SSO+0.2	SSO+0.8	Improvement Percentage for Reputation equal 0.2	Improvement Percentage for Reputation equal 0.8
1	18.301	18.301	18.301	0	0
25	15.37265	14.24543	12.68458	7.3%	21.1%
50	15.47905	13.81229	12.22896	10.6%	26.5%
75	15.50312	13.81229	12.22896	10.9%	26.7%
100	15.50312	13.81229	12.22896	10.9%	26.7%
125	15.50312	13.81229	12.22896	10.9%	26.77%
150	15.50312	13.81229	12.22896	10.9%	26.7%

Table (G): Differentiate Percentage between Concrete Services No. Before and After Clustering

Concrete Services No. Before Clustering	Concrete Services No. After Clustering	Differentiate Percentages
100	36	64%
200	59	70.5%
300	95	68.4%
400	124	69%
500	168	66.4%
600	184	96.4%
700	218	68.9 %
800	234	70.8%
900	272	69.8%
1000	318	68.2%

Table (F): Differentiate Percentage between IR-SSO and FL-RISSO in Execution Time

Concrete services No.	IR-RSSO Execution Time	FL-SSO Execution Time	Differentiate Percentages
50	0.196829	0.165398	15.9%
100	0.207175	0.165245	20.2%
200	0.252805	0.178503	29.3%
300	0.313979	0.209876	33.1%
400	0.459242	0.202395	55.9%
500	0.631116	0.247211	60.8%
600	0.920759	0.235937	74.3%
700	1.187881	0.264824	77.7%
800	1.51265	0.28445	81.1%
900	1.77474	0.294313	83.4%
1000	2.130648	0.341708	83.9%