Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Networks

OPTIMIZING THE RESOURCE ALLOCATION TO ENHANCE THE QUALITY OF SERVICE IN IOT-FOG ENVIRONMENT

A Dissertation

Submitted to the Council of the College of Information Technology at University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctorate of Philosophy in Information Technology/ Information Network

By

Balasem Allawi Hussein

Supervised by

Prof. Dr. Soukaena Hassan Hashem

2023 A.D. **1445 A.H.**

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

ٱقۡرَأۡ بِٱسۡمِ رَبِّكَ ٱلَّذِي خَلَقَ (1) خَلَقَ
ٱلۡإِنسَٰنَ مِنۡ عَلَقٍ (2) ٱقۡرَأۡ
وَرَبُّكَ ٱلۡأَكۡرَمُ (3) ٱلَّذِي عَلَّمَ بِٱلۡقَلَمِ (4)
عَلَّمَ ٱلۡإِنسَٰنَ مَا لَمۡ يَعۡلَمۡ (5)

صدق الله العلي العظيم

سورة العلق (1-5)

Supervisor Certification

I certify that this dissertation was prepared under my supervision at the Department of Information Networks / Collage of Information Technology / Babylon University, by **Balasem Allawi Hussien** as a partial fulfillment of the requirements for the degree of **Ph.D. in Information Technology**.

Signature:

Name: **Dr. Soukaena Hassan Hashem**

Title: **Professor**

Date: / / 2023

The Head of the Department Certification

In view of the available recommendation, we forward this dissertation for debate by the examining committee.

Signature:

Name: **Dr. Alharith A. Abbdullah**

Title**: Asst. Prof.**

Date: / / 2023

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**Optimizing the Resource Allocation to Enhance the Quality of Service in Iot-Fog Environment**) presented by the student (**Balasem Allawi Hussein**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Distinction**) standing as a thesis for the degree of Doctor of Philosophy in Information Technology-Information Networks.

Signature:	Signature:
Name: Dr. Saad Talib Hasson	Name: Dr. Ali Kadhum Idrees
Title: Prof.	Title: Prof.
Date: / / 2023 (**Chairman**)	Date: / / 2023 (**Member**)
Signature:	Signature:
Name: Dr. Rafah	Name: Dr. Mehdi Ebady Manaa
Title: Prof.	Title: Asst. Prof
Date: / / 2023 (**Member**)	Date: / / 2023 (**Member**)
Signature:	Signature:
Name: Dr.	Name: Dr. Soukaena Hassan Hashem
Title: Asst. Prof.	Title: Prof.
Date: / / 2023 (**Member**)	Date: / / 2023(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:

Name: Dr. Wesam S. Bhaya

Title: Professor

(**Dean of Collage of Information Technology**)

Date: / / 2023

Declaration

I hereby declare that this dissertation, **Optimizing the Resource Allocation to Enhance the Quality of Service in Iot-Fog Environment** submitted to University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology-Information Network has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for reports and summaries whose sources are appropriately cited in the references

Signature:

Name: **Balasem Allawi Hussien**

Date: / / 2023

Acknowledgements

I would like to begin by expressing my deepest gratitude to all those who have supported and guided me throughout this journey.

First, my deep appreciation for my supervisor

Prof. Dr. Soukaena Hassan Hashem,

for her continuous guidance, supervision and determination during the course of this work.

My extended gratitude to my beloved wife and daughters for their patience and love.

Balasem Allawi Hussien Al-Isawi

Dedication

To my family…

Thank you for being my pillars of strength during the challenging times, for celebrating every small triumph, and for always cheering me on. Your presence in my life has given me the courage to dream big and reach for the stars.

Abstract

Resource allocation for applications is one of the most challenging and crucial problems in the IoT-Fog environment. Generally, efficient distribution of tasks over fog nodes benefits application users by getting better quality of service (QoS) in terms of metrics like latency.

This study adopted a hybrid meta-heuristic and evolutionary optimization approaches to tackle the resources allocation problem. The hybridization was made by two well-known algorithms namely the Non-dominated Sorting Genetic Algorithm II (NSGA-II), and the Multi-Objective Grey-Wolf Optimization (MOGWO) algorithms. The proposed algorithm passes into two phases, first an NSGAII-MOGWO hybrid algorithm which has a modest performance comparing with the base algorithms. The second phase is an elite version of NSGAII-MOGWO (E NSGAII-MOGWO) that outperform the base algorithms in addition to the NSGAII-MOGWO. The performance of both algorithms was evaluated using ten benchmarking functions with six selection methods (i.e., Roulette Wheel, Tournament, Stochastic Universal Sampling, Boltzmann, Ranking, and Linear Ranking).

The comparisons show the superiority of E NSGAII-MOGWO over all the considered algorithms. The best performance of E NSGAII-MOGWO was in the tournament selection method with average wining points of 53.33% and ratio of superiority scores (0.7868, 0.6874, and 0.6563) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. These results suggests that E NSGAII-MOGWO is the best candidate algorithm to be used in this study to solve the IoT-Fog environment resource allocation problem.

Additionally, E NSGAII-MOGWO was compared against five multi-objective optimization algorithms. The results show that for the Inverted Generational Distance (IGD) indicator, the E NSGAII-MOGWO scores 70% better performance, and 60% for Hypervolume indicator for the ten benchmarking functions.

The simulation environment in this work was used to validate the feasibility of the generated solution by the proposed optimization algorithms. When the proposed algorithm simulated, it reduces the requests waiting time, response time, application loops latency by (23.97%, 67.2%, and 95.27%) respectively.

Declaration Associated with this Thesis

Some of the works presented in this thesis have been published or accepted as listed below.

(**First Paper**)

Journal Name: *Lecture Notes in Networks and Systems*

Paper Title*: NSGAII-MOGWO: A Novel Hybrid Algorithm for IoT-Fog Environment Resources Allocation.*

Scopus: site Score 0.7.

Authors: Balasem A. Hussein[1*], and Soukaena H. Hashem[2]

[1]University of Babylon, Babil, Iraq. [2]University of Technology, Baghdad, Iraq

Email: balasem@uobabylon.edu.iq

(**Second Paper**)

Journal Name: *AIP Conference Proceedings*

Paper Title*: Combined Elitism Multi-Objective Grey Wolf Optimization for Solving Resource Allocation Problems*

Scopus: site Score 0.7.

Authors: Balasem A. Hussein[1*], and Soukaena H. Hashem[2]

[1]University of Babylon, Babil, Iraq. [2]University of Technology, Baghdad, Iraq

Email: balasem@uobabylon.edu.iq

(**Third Paper**)

Journal Name: *Mustansiriyah journal of Pure and applied sciences*

Paper Title*: iFogger: A New Framework to Simulate Fog Computing Resource Placement inside OMNeT++ Environment.*

Authors: Balasem A. Hussein[1*], and Soukaena H. Hashem[2]

[1]University of Babylon, Babil, Iraq. [2]University of Technology, Baghdad, Iraq

Email: balasem@uobabylon.edu.iq

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BBO	Biogeography-Based Optimizer
BCPA	Bee Collecting Pollen Algorithm
BW	Bandwidth
CC	Cloud Computing
CPU	Central Processing Unit
CS	Cuckoo Search
DAG	Directed Acyclic Graph
DDF	Distributed Data Flow
DNSGA-II	Dynamic NSGA-II
DPO	Dolphine Partner Optimization
DRL	Deep RL
EA	Evolutionary Algorithm
EGA	Elitism-based Genetic Algorithm
EPSO	Extended PSO
FA	Firefly Algorithm
FC	Fog Computing
FCFS	First Come First Served
FIFO	First In First Out
FSA	Fish Swarm Algorithm
FWA	Fireworks Algorithm
GA	Genetic Algorithm
GD	Generational Distance
GD+	Generational Distance Plus
GOA	Grasshopper Optimization Algorithm
GP	Genetic Programming
GWO	Grey Wolf Optimization
HEFT	Heterogeneous Earliest Finish Time

HFSGA	Hybrid Flamingo Search with GA
HM-DA	Hybrid Monarch-Dragon Algorithm
HV	Hypervolume
IaaS	Infrastructure as a Service
IGD	Inverted GD
IGD+	Inverted GD+
IGWO	Improved GWO
ILP	Integer Linear Programming
IoT	Internet of Things
IoV	Internet of Vehicles
IPT	Instruction Per Time
IT	Information Technology
ITS	Intelligent Transportation Systems
JSSP	Job Shop Scheduling Problem
LAN	Local Area Network
LP	Linear Programming
MCC	Mobile Cloud Computing
MCSO	Modified Cat Swarm Optimization
MEC	Mobile Edge Computing
MI	Millon Instruction
MILP	Mixed ILP
MINLP	Mixed Integer Non-Linear Programming
MOEA	Multi-Objective Evolutionary Algorithm
MOGWO	Multi-Objective Grey Wolf Optimization
MOO	Multi-Objective Optimization
MPSO	Modified PSO
NP-hard	Non-deterministic Polynomial-time hardness
NPSO	Non-dominated PSO
NSGA	Non-dominated Sorting GA
NSGA-II	Non-dominated Sorting GA-II
PaaS	Platform as a Service
PAES	Pareto-Archived Evolutionary Strategy
P-GA-PSO	Prioritized hybrid Genetic PSO
PS	Percentage of Superiority
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality-of-Service
QTCS	Queuing Theory-based Cuckoo Search
RAM	Random Access Memory

RAN	Radio Access Network
RL	Reinforcement Learning
SaaS	Software as a Service
SI	Swarm Intelligence
SLA	Service-Level Agreement
SOO	Single-Objective Optimization
SOS	Symbiotic Organisms Search
SPEA	Strength Pareto EA
WAN	Wireless Area Network
WORA	Whale Optimization Algorithm
YAFS	Yet Another Fog Simulator

CHAPTER One
General Introduction

1.1 Introduction

In the rapidly evolving digital landscape, three transformative technologies have emerged as the driving forces behind the advancement of modern computing: Cloud Computing, Fog Computing, and the Internet of Things (IoT). These technologies have revolutionized the way data is processed, stored, and accessed, creating a more interconnected and intelligent world.

Cloud Computing refers to the delivery of computing services over the Internet. Instead of relying on local servers and physical infrastructure, organizations and individuals can access a shared pool of computing resources, such as servers, storage, databases, networking, and software, through a network of remote data centers. These data centers, managed by cloud service providers, offer on-demand access to resources, scalability, and flexibility, allowing users to pay for only what they use (pay-as-you-go model) [1]. The cloud has enabled the development and deployment of applications and services without the need for significant upfront investments in hardware and maintenance [2]. As a result, cloud computing has become the backbone of numerous online services, from web hosting and data storage to enterprise applications and AI-powered services.

While cloud computing has revolutionized the way data is processed on a global scale, it faces some challenges when it comes to the growing demand for real-time and low-latency applications, especially in environments where data is generated at the edge of the network.

Fog Computing addresses these challenges by extending the cloud's capabilities closer to the data source. In fog computing, computing resources are distributed along the network edge, in proximity to IoT devices and sensors [3]. This architecture allows data to be processed locally, reducing latency and conserving network bandwidth. Fog computing is particularly beneficial in scenarios where real-time data analysis is crucial, such as in autonomous vehicles, industrial automation, healthcare monitoring, and smart cities [4].

IoT refers to the vast network of interconnected devices, sensors, and objects that collect and exchange data over the Internet[5]. These "smart" devices range from consumer products like smartwatches and home automation systems to industrial machinery and environmental sensors. IoT enables the seamless integration of the physical and digital worlds, leading to improved efficiency, automation, and enhanced user experiences. IoT devices generate massive amounts of data, and cloud and fog computing play vital roles in processing and deriving valuable insights from this data. The combination of IoT and cloud/fog computing facilitates the development of smart applications that enhance our daily lives, optimize industrial processes, and support data-driven decision-making [6] [7], [8].

Resource allocation in the context of fog computing aims to achieve two primary objectives: satisfying users' Quality of Service (QoS) requirements and enhancing the benefits for both fog providers and users [9]. The infrastructure of fog computing poses several challenges. These challenges encompass client resource utilization, task scheduling, resource allocation, and ensuring Quality of Service (QoS) levels [10].

Optimization of resource allocation addresses the problem of efficient management of the environment’s resources. The decision-making strategy is decided by the optimization algorithm to dedicate resources to the requests’ class [11].

This work uses a hybrid approach to combine two well-known evolutionary algorithms, first a genetic-based algorithm called the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), and the bio-inspired multi-objective Grey-Wolf Optimization (MOGWO) algorithm. The algorithms were enhanced by choosing the best selection method among six candidate selection methods (i.e., Roulette Wheel, Tournament, Boltzmann, Stochastic Universal Selection, Ranking, and Linear Ranking). Next, all the implementations were evaluated against ten benchmarking test functions from the IEEE CEC09 functions set.

Finally, the algorithms were further validated inside a simulation environment.

1.2 Problem Statement

The problem of resources allocation is a crucial issue in the IoT-Fog environment. Optimizing the resource allocation process in Fog devices can minimize latency, processing cost, network utilization, resource utilization, and power consumption. Which ensure the sustainability of the Fog environment while efficiently allocating limited resources to multiple applications.

This problem also elevates issues related to satisfying the user requirements including meeting specific deadlines, budget constraints, and response time. Furthermore, the problem of inefficient resource allocation can degrade the dependable deployment of applications' modules in the IoT-Fog environment resources.

The problem of efficient resource allocation in computing paradigms was addressed by many relevant research papers. The most dominant strategy is the use of metaheuristic methods. This work adopts a hybrid algorithm to solve the IoT-Fog resource allocation.

1.3 Dissertation Aims and objectives

The main aim of this dissertation is to build a hybrid algorithm to solve the problem of IoT-Fog environment resources to get the desired quality-of-service. This was accomplished by following a number of sub-objectives:

1. To build a hybrid algorithm with multi objectives to address the requirements of IoT-Fog environment QoS.
2. To enhance the suggested algorithm using performance enhancement methods.

3. Increasing the users' QoS when deploying the resulting allocation map.
4. Minimizing the network resources utilization, energy consumption, and monetary cost.

1.4 Dissertation Contributions

This dissertation makes several significant contributions to the field of resource allocation and Quality of Service (QoS) enhancement in IoT-Fog computing environments. In comparison with previous studies, this study offers the following contributions:

1- Proposed a hybrid algorithm that combines two well-known optimization techniques, NSGA-II and MOGWO, to address the fog resource allocation problem challenges.
2- An Elite version (E NSGAII-MOGWO) was built that control the wolf's pack size in the MOGWO part of the proposed algorithm. The elite method combines Adaptive Elitism-Based Immigration with Elite Opposition-Based Learning Strategy.
3- Adding stagnation mitigation to the E NSGAII-MOGWO using adaptive parameters from the elitism approach to control the diversity of the next-generation population.
4- The E NSGAII-MOGWO hybrid algorithm can overcome the limitations of NSGA-II, such as premature convergence and lack of diversity, by incorporating the diversity-enhancing techniques of MOGWO. The proposed algorithm showed potentials to enhance the quality and diversity of the Pareto Front and produce better solutions than NSGA-II alone.

1.5 Related Work

In this section, we will delve into the existing literature and provide a comprehensive overview of the related works in resource allocation for IoT-Fog environments. Through this exploration, we aim to lay the foundation for our proposed novel approaches and contribute to the body of knowledge in this rapidly evolving field.

The presented works are related to resource allocation and optimization in fog computing and IoT environments, with a focus on improving quality of service (QoS) metrics such as makespan, cost, energy consumption, delay, and task time computation. These works propose various algorithms and techniques to address the challenges of resource allocation in these environments.

Tian et al. (2018) [12] suggested the use of inverted generational distance indicator with a multi-objective evolutionary algorithm (MOEA). The proposed algorithm was guided by an empirical methos to select a reference point to guide the search process. This method lacks scalability in the size of objective-space. Furthermore, this approach adds another hyperparameter, which is the number of reference points.

X. Zhang et al. (2018) [13] proposed a multi-objective particle swarm hat does not store best particles in a global archive. The particle selection method depends on a tournament between pairs of particles in the same generation. In this algorithm, the number of particles might increase the time complexity because of the constant comparison of particles against each other. This drawback can lead to scalability issues.

Panichella (2019) [14] proposed an adaptive algorithm to estimate the geometry of the solutions front. The selection of next generation individuals is based on their contribution to the diversity of the objective-space. The proposed algorithm is sensitive to the size of the problem and objective spaces.

Xu et al. (2019) [15] proposed an improved particle swarm optimization (PSO) based workflow scheduling algorithm for cloud-fog environments. They use PSO to solve the mapping process between tasks and computing resources, considering objectives such as cost and makespan. The authors suggest using more objectives for enhancing performance.

Zhuang & Zhou (2020) [16] presented a hyper-heuristic resource allocation algorithm for fog computing, using ant colony optimization (ACO) as the high-level strategy. They consider objectives related to delay and energy consumption. However, they do not model computing offloading and collaborative cloud-fog environments.

Anu & Singhrova (2020) [17] proposed a prioritized hybrid genetic particle swarm optimization (P-GA-PSO) algorithm for resource allocation in fog computing. They consider objectives related to delay, waiting time, and energy consumption. However, the algorithm has a tendency to get stuck in local optima and is not scalable for a large number of requests.

Yang et al. (2020) [18] presented a multi-objective task scheduling method for fog computing in cyber-physical-social services. They use an adaptive multi-objective optimization algorithm and consider objectives such as makespan and task resource cost. However, they do not consider energy consumption in their approach.

Zhang et al. (2021) [19] proposed a novel edge server selection method using a combined genetic algorithm and simulated annealing algorithm. They focus on objectives related to time latency and energy consumption. However, the results for latency and energy consumption are not provided.

Potu et al. (2021) [20] proposed an extended particle swarm optimization (EPSO) algorithm for task scheduling in fog computing environments. They consider objectives related to makespan and cost. However, they do not take energy consumption into consideration, and application tasks flow is not considered.

Natesha & Guddeti (2021) [21]adopted an elitism-based genetic algorithm (EGA) for IoT service placement in fog computing environments. They consider objectives related to service time, energy consumption, service cost, and average CPU utilization of fog nodes. However, they do not consider the application workflow in their approach.

Ali et al. (2022) [22] proposed an automated task scheduling model using non-dominated sorting genetic algorithm II (DNSGA-II) for fog-cloud systems. They consider objectives related to makespan and total costs in a fog-cloud environment. However, they do not consider energy consumption and application workflow.

Sing et al. (2022) [23] proposed a resource allocation scheme based on the Whale Optimization Algorithm (WORA) for cloud-fog-based IoT applications. They consider objectives related to cost, energy consumption, makespan, and completion of task ratio. However, they do not consider QoS and task migration.

Salimian et al. (2022) [24] proposed an evolutionary multi-objective optimization technique for deploying IoT services in fog-enabled networks. They use Particle Swarm Optimization (PSO) and consider objectives related to fog resource utilization and QoS metrics such as response time, service cost, utilization of fog, and throughput. However, they do not model energy consumption.

Ramzanpoor et al. (2022) [25]proposed a multi-objective fault-tolerant optimization algorithm for the deployment of IoT applications on fog computing infrastructure. They consider objectives related to power consumption and total latency between application components. However, they do not consider cost, resource mobility, and application tasks workflow.

Harika & Krishna (2022) [26]proposed a hybrid monarch-dragon algorithm (HM-DA) for resource allocation in the fog environment. They consider objectives related to credibility score, concurrency, price affordability, and task time computation. However, they do not consider energy consumption.

Iyapparaja et al. (2022) [27]proposed a queuing theory-based cuckoo search (QTCS) model for resource allocation in fog computing. They consider objectives related to QoS metrics such as resource availability, resource allocation time, and energy consumption. However, they do not consider cost.

He et al. (2022) [28] presented a two-step adaptive offspring selection method. The algorithm was experimented with large-scale multi-objective benchmarking functions. The direction vectors were selected from reference points, which raise the problem of choosing the best points from the solutions fronts. The method is dependent on the deployed evolutionary algorithm, as noticed from the paper presented results.

Shu et al. (2022) [29] suggested an augmented algorithm to adopt the velocity of particles in particle swarm optimization algorithm. The algorithm adopts the particles velocity using the hypercube indicator and the minimum distance between particles in the same front. The proposed algorithm suffers from scalability problems for complex and large objective spaces.

Saif, Latip, Hanapi, & Shafinah (2023) [30] proposed a multi-objective grey wolf optimizer (MOGWO) algorithm for task scheduling in cloud-fog computing. They consider objectives related to QoS metrics such as delay and energy consumption.

Saif, Latip, Hanapi, Alrshah, et al. (2023) [31] proposed a workload allocation algorithm using the non-dominated particle swarm optimization (NPSO) algorithm. They consider objectives related to energy consumption and delay. However, they do not model cost or consider QoS.

Reffad et al. (2023) [32] proposed a dynamic adaptive bio-inspired multi-agent system for healthcare task deployment. They combine improved grey wolf optimization (IGWO) with PSO and consider objectives related to energy consumption, makespan, and waiting time. However, the cost is only modelled for energy consumption, and they compare their approach with priority queueing algorithms only.

Mohammedzadeh et al. (2023) [33] proposed an energy-aware workflow scheduling algorithm in fog computing using a hybrid chaotic algorithm. They combine Symbiotic Organisms Search (SOS) with the Grasshopper Optimization Algorithm (GOA) and consider objectives related to makespan and energy consumption. However, they do not consider cost and QoS.

Shinu et al. (2023) [34] proposed a resource provisioning model using meta-heuristic methods for IoT microservices with mobility management. They consider objectives related to latency, energy consumption, network utilization, and cost. However, they do not consider QoS or application tasks workflow.

Table 1.1: Summary of the related works

Reference	Approach	Objectives	Limitations	Environment
[12]	Adapted Reference-point Multi-Objective Evolutionary Algorithm (ARMOEA)	Multi-objective	Lacks scalability in the size of objective-space.	Benchmarking
[13]	Competitive mechanism based Multi-Objective Particle Swarm Optimization (CMOPSO)	Multi-objective	- Does not adopt with increased population. -Scalability issue	Benchmarking
[15]	Improved particle swarm optimization (IPSO)	Makespan, cost	The economic cost was the same as the original PSO.	Cloud/Fog
[14]	Adaptive Geometry Estimation based Multi-Objective Evolutionary Algorithm (AGEMOEA)	Multi-objective	Sensitive to the size of the problem and objective spaces.	Benchmarking
[16]	Hyper-heuristic resource allocation algorithm	Delay, energy consumption	There is no cost modeling.	Fog
[17]	Hybrid Prioritized Genetic Particle Swarm Optimization (P-GA-PSO)	Delay, waiting time, energy consumption	There is no cost modeling.	Fog

[18]	adaptive multi-objective optimization task scheduling method for fog computing (FOG-AMOSM)	Makespan, cost	The optimization method used was a simple GA. No energy consumption modeling.	Fog/IoT
[19]	Hybrid Genetic algorithm and simulated Annealing algorithm for edge Server Selection	Time latency, energy consumption	The task completion probability of edge servers and precise user mobility path should be pre-known for the algorithm to work.	MCC/Edge server
[20]	Extended particle swarm optimization (EPSO) with an extra gradient parameter	Makespan, cost	The energy consumption was not taken under consideration.	Cloud/Fog
[21]	multi-objective Elitism-based Genetic Algorithm (EGA)	Service cost, energy consumption, service time	Does not consider the interdependent IoT applications to check the performance of the proposed EGA	IoT/Fog
[22]	Discrete NSGA-II (DNSGA-II)	Makespan, cost	There is no energy consumption modeling.	Fog/Cloud
[23]	Whale optimized resource allocation (WORA)	Cost, energy consumption, makespan, completion of task ratio	The performance was against resource scheduling algorithms despite that the algorithm was to optimize resource allocation.	Fog/Cloud/ IoT

[24]	Particle Swarm Optimization (PSO)	Response time, Service cost, Utilization of fog, Throughput	There is no energy consumption model. The model objectives were given weight in the fitness function but they were set to 1.0 (i.e., have no effect).	Fog/cloud/ IoT
[35]	Multi-objective cuckoo search algorithm (MOCSA)	power consumption, total latency	There is no cost modeling. The resource requests are static. Clients are stationary.	IoT/Fog
[26]	Hybrid monarch-dragon algorithm (HM-DA)	Credibility scores, price affordability, concurrency, task time computation	The paper does not consider energy consumption.	Fog
[27]	Queuing Theory based Cuckoo Search (QTCS)	resource availability, resource allocation time, and energy	No cost modeling.	IoT/Fog
[28]	Direction Guided Evolutionary Algorithm (DGEA)	Multi-objective	Dependent on the deployed evolutionary algorithm	Benchmarking
[29]	Hypercube and Distance Multi-Objective Particle Swarm Optimization (HDMOPSO)	Multi-objective	Scalability problems for complex and large objective spaces.	Benchmarking
[30]	Multi-Objectives Grey Wolf Optimizer (MGWO)	Delay, energy consumption	There is no cost modeling. The environment is very limited (7 sensors, 3 fog nodes, and 1 cloud).	Fog/cloud/ IoT
[31]	Non-dominated Particle Swarm Optimization (NPSO)	Delay, energy consumption	There is no cost modeling. NSGA-II was better in terms of delay.	Cloud/Fog
[32]	Improved Grey Wolf Optimization (IGWO) + PSO	energy consumption, makespan, waiting time	The cost was modeled for energy consumption only.	Fog/cloud/ IoT

[33]	HDSOS-GOA combines the search qualities of Symbiotic Organisms Search and the Grasshopper Optimization Algorithm (GOA) algorithms	makespan, energy consumption.	The cost was not modeled.	Fog/IoT
[36]	Meta-Heuristic Methods	latency, energy consumption, network utilization, cost	A case-study. Application tasks workflow was not considered.	Edge/Fog/ Cloud

1.6 Dissertation Outline

The remaining thesis is organized as follows:

Chapter 2: presents an exploration of various computing concepts and highlights the relevant optimization and elitism algorithms used in this dissertation.

Chapter 3: provides the detailed implementation of the proposed architecture and discusses the key design decisions made for solving resource allocation in the IoT-fog environment.

Chapter 4: focuses on the comparative evaluation of the developed algorithms. The results from testing the selection methods with the benchmarking functions were presented. Additionally, simulation experiment results were investigated.

Finally, ***Chapter 5*** concludes the findings of this thesis and presents future works recommendations.

CHAPTER Two
Theoretical Background

2.1 Overview

This chapter represents the theoretical concepts of designing optimization algorithms in general and in resource allocation for IoT-Fog environment in specific. The algorithms in this work implements an evolutionary approach to achieve the best allocation of resources. In addition, the practical aspects of the deployed algorithms are presented in this chapter. Finally, a description of the performance measures that used to evaluate the proposed algorithms is provided.

2.2 Cloud Computing

Cloud computing is a computing architecture characterized by high processing and compute power, high latency, the ability of AI processing, high standards of cybersecurity and equipped with massive storage capacity (Figure 2.1.).

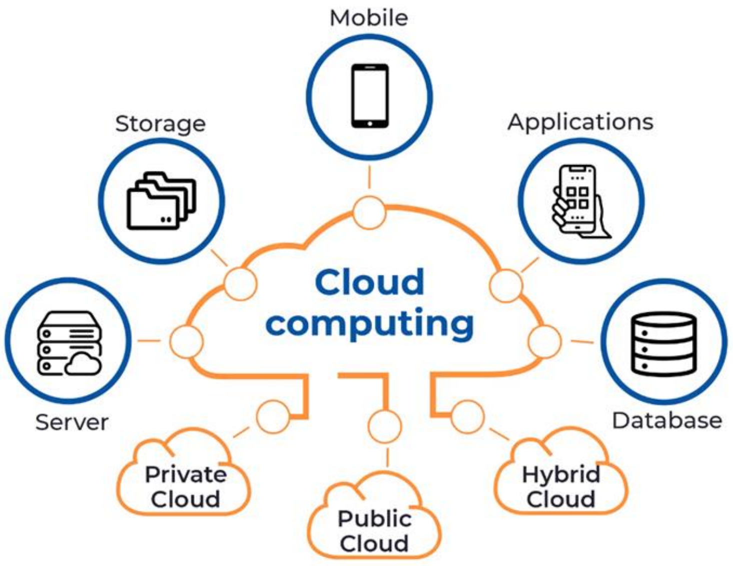

Figure 2.1: Cloud Computing Paradigm [37]

In this highly scalable applications deployment, when using IoT devices, they will be many kilometers away from the compute and storage devices. IoT services providers such as Google, IBM, Amazon, and Microsoft, have bundles of ready-to-use applications, which makes this environment very attractive for IoT customers. Whichever was the cloud service provider, the common feature is the potential to reach elastic IT resources with lower investment expenditures in hardware and software and their management (Laroui et al., 2021). Cloud computing uses the same infrastructure for conventional Internet, as such, CC can be considered as a complement to traditional Data Centers and not a replacement

The primarily dominant cloud services include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The special disperses between the cloud and IoT devices will result latency and temporal delays which bring out different types of communicational problems like a degradation in service-quality, network traffic congestion, and an increase in channel errors.

2.3 Edge Computing

Solving the Cloud Computing problems, mentioned above, was achieved by Edge Computing, which is a heterogenous architecture combining peer-to-peer and cloud servers on one side and mobile devices on the other, Figure 2.2.

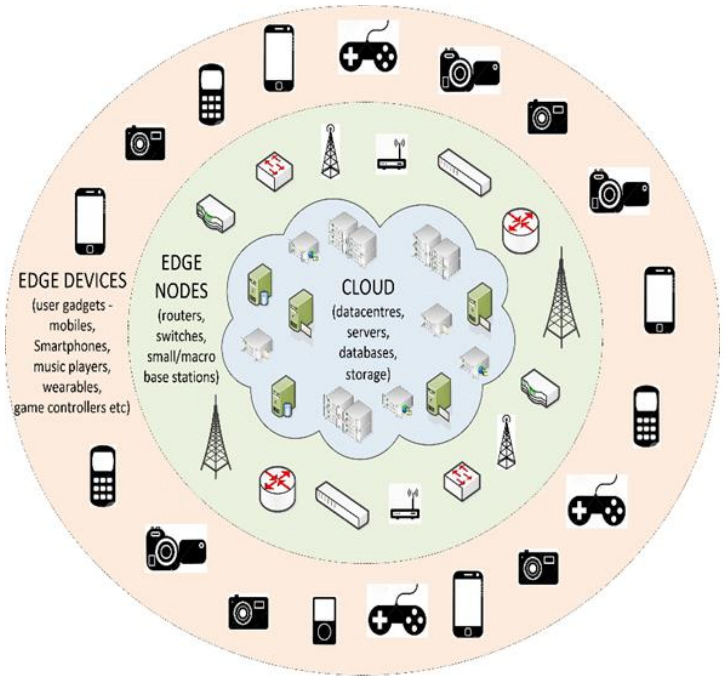
EDGE DEVICES
(user gadgets - mobiles, Smartphones, music players, wearables, game controllers etc)
EDGE NODES
(routers, switches, small/macro base stations)
CLOUD
(datacentres, servers, databases, storage)

Figure 2.2: The relationship between Cloud and Edge devices. [38]

Edge computing deploys compute and storage resources at locations near data collectors, on the network edge. All edge components must have the capabilities to perform computations on the edge (Varghese et al., 2016).

2.3.1 Mobile Cloud Computing (MCC)

Devices like smartphones characterized by a constrained compute and storage capabilities. These devices can compensate this drawback by means of the cloud, which can virtually provide unlimited compute and storage services on-demand. This feature is not built by default in smartphones; therefore, the technology of Mobile Cloud Computing was evolved to fulfil this new technological trend, Figure 2.4.

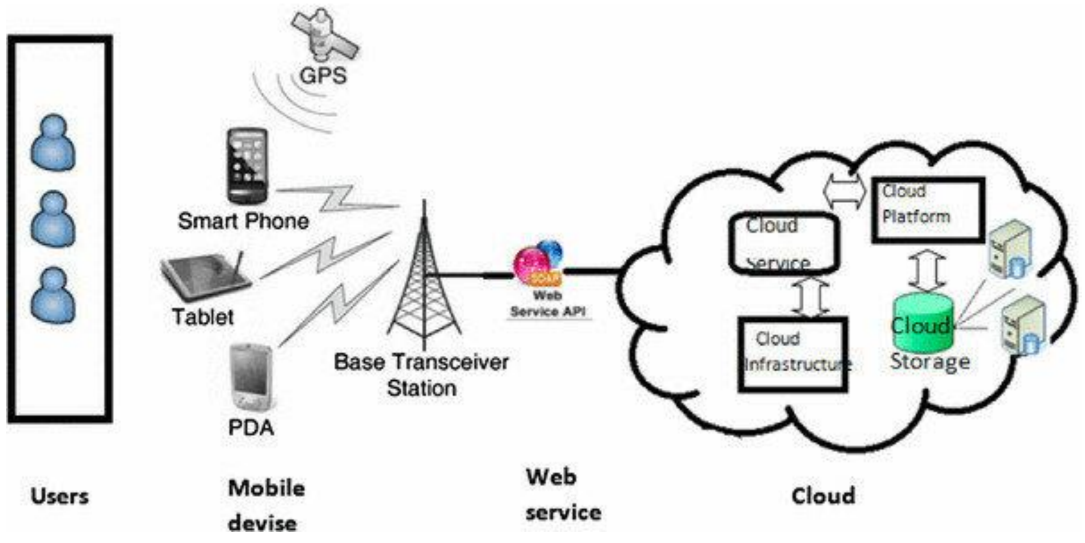

Figure 2.4: Heterogeneous environment architecture of MCC [39]

Application models for MCC aim to reach objectives like the ability of running applications beyond the comprehension of smartphones in terms of resources for in place execution, optimizing execution time, or minimizing energy consumption for heavy computational applications when executed locally (Capponi et al., 2019). The aims above are not mutually exclusive, i.e., multiple aims can be achieved by a single application model.

MCC runs in a heterogenous environment, thus it needs methods to bring it together (Capponi et al., 2019). This diversity in vendors and applications, such methods include compatibility, interoperability and the ability of integration between the environment components. The massive number of mobile devices generates unpredictable number of requests, and it is very important to provide redundant devices to absorb the traffic. The solution came by light-weight redundant servers called cloudlets and placed on the network edge.

2.3.2 Mobile Edge Computing (MEC)

Mobile Edge Computing (Figure 2.3) is a service environment with CC capabilities positioned at the edge of the mobile network, within the RAN and consequently near the mobile equipment. MEC technology aims to lower latency, delivering network functions and services with high efficiency, and enhanced QoE. The MEC service scenarios generally include Augmented Reality, Intelligent Video Acceleration, Connected Cars, and Internet of Things Gateway (Hou et al., 2022; Noor et al., 2018).

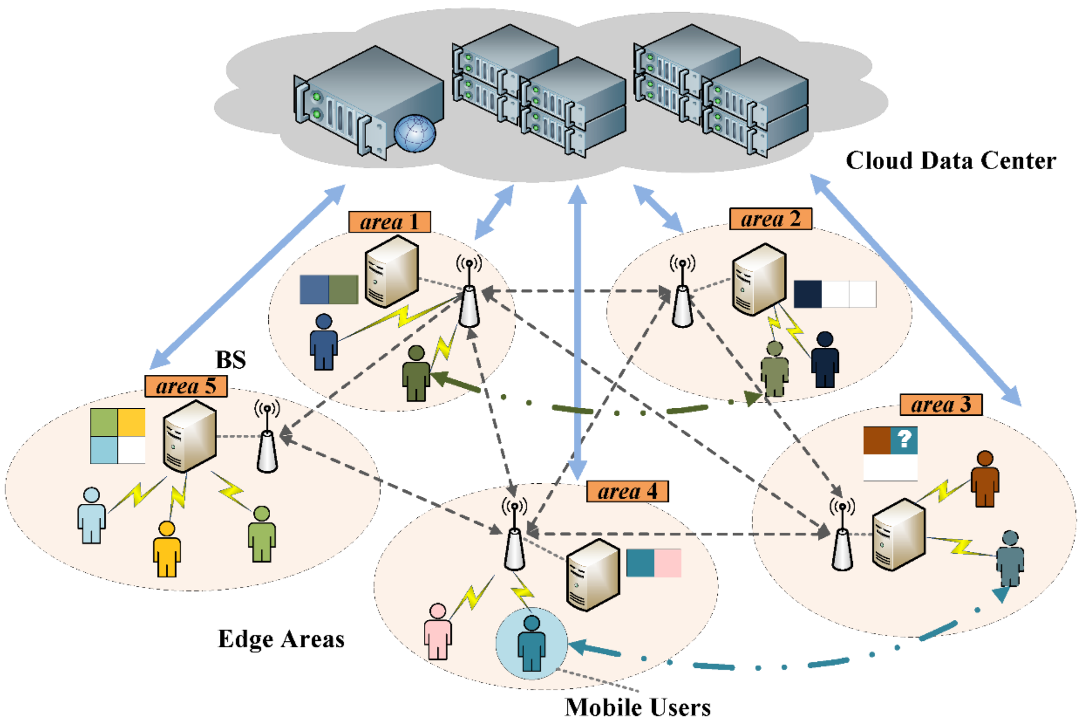

Figure 2.3: Mobile edge computing architecture [40]

The selection of variables plays a critical role in traffic flow forecasting models, as it has a direct influence on their performance and efficiency. Indirect approaches, such as employing mutual information derived from entropy theory, have been employed for the purpose of extracting information from unprocessed feature values. Variables commonly taken into account include the volume of traffic flow, the duration of travel, and the speed data obtained from on-site sensors such as loop detectors and laser sensors. Various models incorporate input parameters such as traffic density, speed, incident severity, road delays, and lane blocking duration (Mchergui et al., 2022).

2.4 Fog Computing

In contrast to CC, Fog Computing is a computing architecture, in it, fog nodes are near the client with out-of-the-shelf computes having a good multi-core computation power (Figure 2.5). Achieving the required fog functionality requires installing several types of smart devices, such as networking devices and smart phones to act as fog compute nodes.

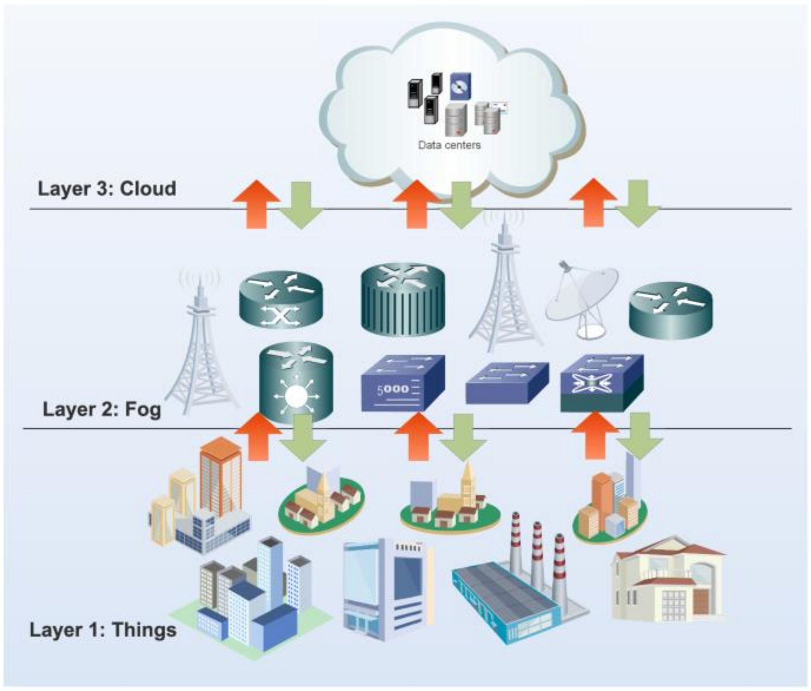

Figure 2.5: Fog computing architecture [41]

Fog computing was defined in many researches depending on the research context. The OpenFog Consortium (https://www.openfogconsortium.org) define Fog Computing as a framework that can mediate network resources (i.e., compute, storage, and networking) from IoT devices all the way to the cloud. To overcome these limitations, fog computing can be employed as it integrates application-specific logic in intermediate infrastructure along with edge devices and remote clouds [42].

More generally, Fog computing is a geographically distributed shared heterogenous compute resources that can serve applications on the approximate without frequent requests to the cloud [42]. The Fog core contains components such as routers, WAN switches, and servers (compute devices), now these devices provide interfaces for easy integration with the Fog. The near proximity of Fog devices to the edge reduces delay times radically to reach near-real-time service requests and responses. The diversity on both organizational and connectivity domains, lead to several challenges. One of these challenges is how does Fog tolerate network failure, also the way it will manage the diversity in deployed environments.

The layout by which Fog computing is formulated allow massive number of IoT entities to be assimilated by the Fog. In addition, the nature of Fog components makes it easy to position those components near the IoT entity, comparing to other cloud or mobile computation models. The cut-off round trip between the IoT device and the cloud core by the Fog enables more rapid response for critical sensors or devices requests. Fog computes, as mentioned, are located nearer to these entities and equipped with the necessary computing and storage powers to work on the surrounding environment of the IoT entities. These features appear to be promising to use Fog Computing with IoT context and tasks.

The presumed features of Fog Computing result an offloading of heavy and battery exhaustive processes to be lifted to the Fog, yielding an optimized usage of device battery. Another advantage of Fog architecture is the mirrored cloudlets, which can be utilized to reduce traffic congestion. For example, when a device needs some stored data the provisioning of data can be calculated to be taken from the nearest cloudlet and avoiding congested paths [43]. This approach is crucial to time-sensitive tasks. Fog computing can achieve anonymity and anti-censorship using Tor overlay and onion-routers, these routers can be hosted as in-network service inside cloudlets. After hosting these routers, other Fog nodes will be aware of their addresses and ultimately add them as intermediate communication path nodes to add privacy to the mobile node identity.

Providing localized services is a very important aspect to increase customer satisfaction. Service providers can dedicate a localized virtual infrastructure on some cloudlets and install service on these cloudlets. Furthermore, service providers can additionally provide a domain name for these services, and the local ISP will resolve these addresses to the nearest local virtual infrastructure cloudlet, which will increase the quality of experience (QoE) for the end user.

2.4.1 Features of Fog Computing

When Fog computing was introduced and defined it was expected to fulfill number of desirable features. These features include, low latency and real time interactions, save bandwidth, support for mobility, geographical distribution and decentralized data analytic, heterogeneity, interoperability, data security and privacy protection, and low energy consumption (Table 2.1).

Principally, data produced by IoT entities located at the edge of a LAN are computed and stored by the Fog nodes. The locality of Fog nodes operation accelerates the quality of the in-network services. Hence, low latencies can be gained to reach real-time support to cover the specifications of latency and time sensitive applications [44]. Researches studies this property includes [45] and [46].

Table 2.1: Features of different computing paradigms [47]

Feature	CC	EC	MEC	MCC	FC
Heterogeneity support	Y	Y		Y	Y
Infrastructure need	Y	Y		Y	Y
Geographically distributed		Y			Y
Location awareness		Y	Y		Y
Ultra-low latency		Y			Y
Mobility support		Y	Y	Y	Y
Real-time application support		Y			Y
Large-scale application support	Y	Y			Y
Multiple IoT applications	Y				Y
Virtualization support	Y				Y

CC: Cloud Computing, **EC**: Edge Computing, **MEC**: Mobile Edge Computing, **MCC**: Mobile Cloud Computing, **FC**: Fog Computing.

Furthermore, Fog nodes can perform several computations locally like data cleaning and filtering, features extraction. The cloud will receive only the necessary resulted data. Machine learning was used in [46] with the aim to reduce the transmissions of data to the cloud. The concluded advantage is an optimized usage of bandwidth, and with the current massive data size bouncing in the internet this feature gaining growing importance.

Fog environments usually comprise several types of mobile and stationary entities at the device-layer, this also true for Fog nodes on the fog-layer. Some exemplar applications can be found in [48]–[50]. Fog computes and mobile devices can have inter- and intra-communications between each other [51], [52].

Furthermore, Mobile nodes can support some sort of low range communication technology to avoid communication fluctuation caused by mobility [53]. In addition, on the fly data transmission between static edge entity and nearby mobile smart device can take place using Bluetooth just when it is in range, and then the smart device will surrender that information to the Fog or cloud [54], [55][56].

Moreover, Fog environment disperse wide geographical area range and it maintain the trajectory and localize its mobile entities. In consequence, it is important to have a decentralized structure to ensure the communication paths will be minimum. This approach can shorten processing time, better location-based services, and better support for real-time decision making. Example of wide spread mobile entities can be found in Internet of Vehicles (IoV) [57].

Normally, Fog nodes are diverse in vendors and type of usage weather the nodes are physical or virtual [58]. Fog nodes can be cutting-edge servers, edge network equipment and base stations. The hardware of these nodes varies in compute and storage abilities with wide range of operating systems and applications. The nature of Fog computing inherently support virtualization which means virtual computes and virtual network devices can be utilized as Fog nodes [59]. In consequence, fog nodes are heterogenous with different types of communication media and speed links to connect to edge devices [60]. The dynamic and heterogeneity of Fog environment extend to several levels of design architectures to cope with the demands of highly distributed applications that acquire low latency [61].

Following for the heterogenic nature, fog computes must be integrated and embedded to work transparently with different types of services [62], for example streaming services [63]. Another example is real-time data analysis in ITS (Intelligent Transportation Systems) in the Fog environment [64], [65].
Additionally, services hosted by the Fog are naturally near the edge entities. Fog devices placement gives the opportunity for data security and privacy protection, where, sensitive data privacy protection is maintained by different layers of security measures [66].

Finally, Fog nodes have low spatial density which implies low heat dissipation and additional cooling systems are not a necessity. Furthermore, mobile fog nodes can use optimized energy management schemes and short-range communication mode [67]. The previous approaches lead to power consumption reduction, energy preserving [68], and lower costs. In other words, a greener computing structure [45].

A comparison between cloud and fog computing can be found in table 2.2.

Table 2.2: Comparison Between Fog and Cloud Computing [69]

	Fog Computing	Cloud Computing
Workflow	Distributed	Centralized
Compute nodes placement	Local to their environment	Within the web
Latency	Small	High
Hops	Single hop	Multiple hops
Safety measure	Hard to measure	Measurable
Physical entities	Constrained computing and storage capabilities.	Vast computing and storage capabilities.
Vulnerability	More vulnerable	Less vulnerable
Geolocation discovery	Allowed	Not allowed

2.4.2 Fog Computing Architecture

Works on Fog architecture include [70][71], arrange Fog environment in a seven layers reference framework, Figure 2.6. Generally, these layers include; Physical and Virtual Sensors (Layer 1), Fog devices (Layer 2), Monitoring (Layer 3), Pre and post processing (Layer 4), Storage and resource management (Layer 5), Security (Layer 6), and Application (Layer 7). The following represent a preview of those seven layers.

First, the physical and virtual sensors layer. Sensors are the primary source of data in the Fog Computing. This layer considered as the environment data entry point. The data characteristics depends on the sensor type and field of application [72]. Virtual sensors are software sensors that process sensors history data and get insights from those readings.

Next, the physical devices layer (Layer2). Generally, devices in the Fog are IoT devices (Fog server, Fog device, or a gateway) [73]–[75]. In practice, Fog servers must possess higher computation powers configurations than both the gateways and sensor nodes in their coverage area. Fog networks make heavy use of virtualization to cluster the network entities. Physically and virtually clustered IoT devices can communicate with each other on demand. This formulation can assist in developing intelligent applications, for example in intelligent transportation systems, fuel-aware path planning can be deployed using a collective of Fog servers and devices [76].

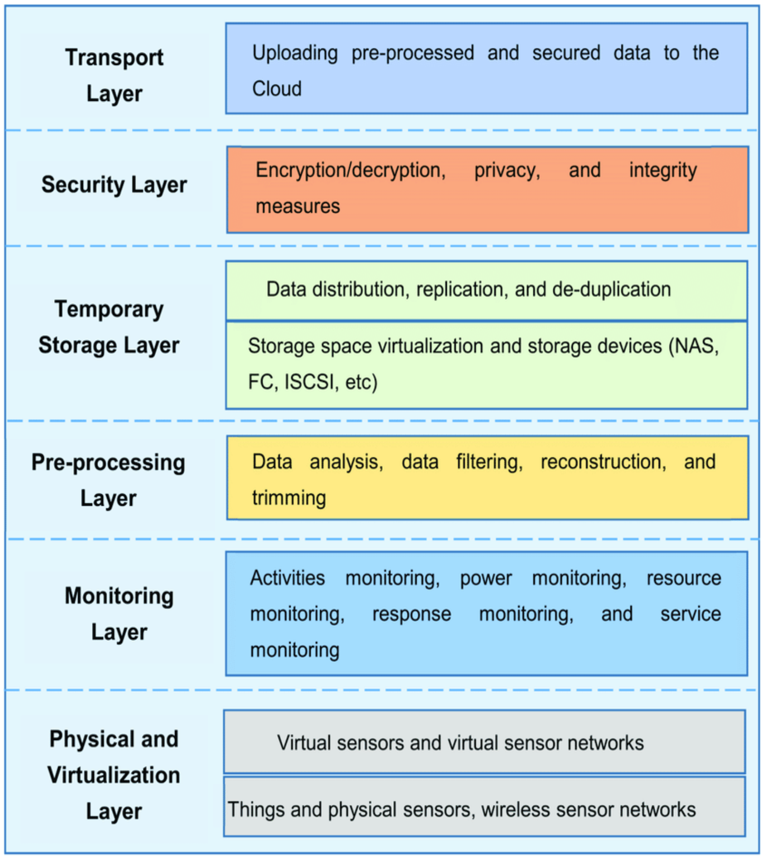
Transport Layer
Uploading pre-processed and secured data to the Cloud
Security Layer
Encryption/decryption, privacy, and integrity measures
Temporary Storage Layer
Data distribution, replication, and de-duplication
Storage space virtualization and storage devices (NAS, FC, ISCSI, etc)
Pre-processing Layer
Data analysis, data filtering, reconstruction, and trimming
Monitoring Layer
Activities monitoring, power monitoring, resource monitoring, response monitoring, and service monitoring
Physical and Virtualization Layer
Virtual sensors and virtual sensor networks
Things and physical sensors, wireless sensor networks

Figure 2.6 A layered Fog computing architecture [77]

The third layer is the monitoring layer. The monitoring processes may include tracking the performance and resources efficiency of the environment [78]. When devices seek help to locate resources the Fog environment management server will take these requests and response with an optimized allocation. Furthermore, efficient resource audit can predict future demands depending on current requests, this can maintain the acquired QoS as depicted in the Service-level agreement (SLA) and minimize cost of banalities when violating the SLA.

The pre and post processing is the fourth layer in this architecture and serve as the data analysis level. Processes can be utilized in this layer include data analysis, filtering and cleaning the collected data. The resulted data will be in the format that is needed by the requested node. The Fog server manager will decide whether to store it locally or on the cloud for permanent storage [74]. The massive data size that generated from the IoT sensors can be summarized with fixed time intervals on the Fog and the results that need to be stored for longer periods will be uploaded to the cloud. In addition, data generated by the IoT devices can be faulty or uncomplete, in this case Fog servers must have the ability to regenerate these data and tolerate these errors (virtual sensing).

The following layer is the storage and resource management layer. The storage system in Fog environment is a set of individual virtualized storage components responsible for storing and maintain data availability and usability. Storage virtualization can greatly reduce the cost of the deployed hardware, in addition virtualization can protect from data loss by creating redundant backup versions [79]. On the other side, resource management requires devices capable of scheduling resource allocation and saving energy consumption. Application scheduling and resource allocating is the responsibility of the reliability component. Fog computing have the ability to scale during peak times of demand. Cloud platform can assure horizontal scalability only, while Fog computing assure both horizontal and vertical scalability [80]. The distributed nature of resources arises the problem of resource allocating. The component responsible for resource allocation, de-allocation, and relocation is the resource allocation component. The huge number of fog applications which tries to access the resources at the same time requires efficient access scheduling, this task in carried out by the application scheduling component. Fog systems are very complex designs due to the focus on maintaining the entire IoT entities, Fog nodes, servers, and clouds.

The sixth layer concerned with maintaining security issues, and also concerns with securing fog customers' data. Cloud users request their services directly from the cloud, in contrast, Fog users connect to the Fog system for services. Another responsibility of this layer is the capability of authenticating fog users [81]. To minimize the threat of malicious attack, an encryption component is used on both ends of communication links. Generally, the main method of communication in Fog environment is wireless technologies. Wireless communication has inherited weaknesses like eavesdropping. In this case, it is important to encrypt the user identity and prevent any exploitation outside the node premises.

Finally, the last layer is the application layer. This layer contains all the deployed applications in the environment [82]. The application layer provides a container for solutions with low latency and lower cost.

2.4.3 IoT-Fog Computing Environment

Integrating cloud computing with IoT applications raise various issues. For instance, cloud computing does not support applications acquire real or near real-time requirements like augmented reality, media streaming, and playing online games [83]. In addition, cloud computing does not provide localized services. The concerned issues of cloud computing with IoT were solved using Fog Computing technology by bringing the computation near the end devices. The features held by Fog computing can preserve the security and demands of IoT entities [84]–[86].

2.4.4 Fog Computing Research Challenges

Fog computing is an evolved approach of cloud computing. IoT devises became less and less expensive, furthermore, computations were brought nearer to the end user with the help of the Fog. This new proximity lessens the computations and data roundtrips to the cloud. On the other hand, the nature of Fog computing environment brings challenges in terms of security, network devices, and integration with the Fog.

2.4.4.1 Device and network challenges

Different types of challenges regarding Fog device and network including decentralized framework, networking resources, and device heterogeneity.

Decentralized framework is a feature leads to redundancy in structure, like redundant codes in device on the edge of the network [87], [88]. Therefore, fog computing has to resolve this by minimizing this redundancy in Fog environment. Additionally, in terms of networking resources, the complexity of the Fog environment network is high due to the arbitrary distribution of network entities. Network resources can be allocated via a proper middleware to the demanding application. Furthermore, device heterogeneity, generally, Fog end devices are heterogenous which yield diversity [89]. These are some of the concerns should be considered when developing applications dedicated for the Fog environment. Some computations can be offloaded from the fog to the cloud when they are not time-sensitive.

2.4.4.2 Computational challenges

Fog level computations can be challenging due to different reasons. The main reasons include; differences in computational powers at each network level, resource sharing in distributional networks like Fog ecosystem, the diversity of application requests demands, and also network devices failure, utilization, mobility and compatibility.

Generally, interactions with the cloud servers are inevitable with Fog computing. Yet, the interactions with edge devices must have low latency – for performance considerations, on the other hand, a slower communication is tolerated when the destination is the cloud. The challenge raised is to locate the place of computations, whether it should be committed on the premises fog or in the far cloud.

By default, fog computing is a distributed computation environment. This way, applications run in the fog need to be robust against errors due to imperfections sourced from the fog environment [90], [91]. When edge computes do not have the sufficient computation power, they might reach for those resources in alternative nodes. The scope of computation might be extended to other resources. The challenge here is to share those resources and form a resource pool that accommodate compute request by other applications [92], [93].

Generally, mobile nodes in the Fog need to communicate with each other in an un-interruptible way which requires the fog environment to perform extensive computing [94]. The OpenFog is an N-level environment which mean that large numbers of levels produce delays on the environment. In a mobile Fog environment, the level of delay will be acceptable when the number of levels is thoroughly examined.

Additionally, Devices in the Fog are diverse in type and accessibility; therefore, resources utilization is un-uniform and even random. Fog devices are solely capable of running applications. This challenge is of concern wen adding the probability of failure in Fog devises are high because of the distributed and disperse nature of its environment. As a result, fog devices may become faulty in several occasions, like obsolete service, hardware failures, and software breakdown. Other out-of-control failures include wi-fi connectivity is down and main power malfunctions. Another aspect is that IoT devices are manufactured by variety of companies, having that number of proprietary specifications in hardware as well as software configuration. In addition, security requirements enforce intellectual procedures and devices to work.

Furthermore, request provision handling also raises a challenge. Millions of IoT devices are inside the Fog environment sending request through the network with diverse application types. This diversity represents a challenge to the Fog environment, centralized approaches can be considered to resolve this problem.

2.4.4.3 Security challenges

The diversity of devices in Fog networks make them vulnerable to variety of threats. One type of attack discussed in [95], was the man-in-the-middle attack in the fog computing environment. The aspects of concern in Fog architecture are information and structure. Fog environment extensively use wireless media; therefore, it inherits all the attack threats of this media. The nature of Fog devices makes them vulnerable to physical attacks. The challenge is to operate those devices sufficiently safe on the edge.

2.4.4.4 System management challenges

Challenges on the service level include service-oriented computing, management of resources and cloud and fog integration.

Client services are spread across the cloud and the edge as small services. This disperse of services has its own problem, because it arises the issue of provisioning the service itself for the specific device. Generally, solving this problem demands a suitable architecture can minimize the latency for fog communications.

On the other hand, Fog environment is easily adopted to the lack of resources and frequent failures. A breakdown of fog nodes may have a wide-spread and can lead to a degraded performance. Fog resources are virtualized in the environment. Generally, virtualization arise a number of problems like migration, latency, and initialization, which all need attending for proper management to get better availability during network downtimes.

The fog framework contains both cloud and fog entities. Hence, providing better QoS for different users in terms of their services, while assuring end-to-end communications between the cloud and the fog, is a major issue [96], [97]. In conclusion, end-to-end integration should be dynamically allocated.

2.5 Internet of Things in Fog Environment

Fog computing is placed in the middle between the cloud and end devices. This position brings the fog near the edge of the network. Furthermore, the fog layer can compute processes without frequently routing them to the cloud, which can radically reduce the amount of information transferred to the cloud. The rapidly increasing number of IoT devices implies that a significant amount of data will be transferred throughout the network.

Normally, the lower layer is comprised of huge wirelessly connected peer-to-peer sensors, self-organized, multi-hup network. The entities of this layer are collaborative in collecting and processing information perceived from objects in the network coverage area and routing them to the fog mini-data centers. The network sensors and the nodes in their coverage are normally battery-operated, which have variant time-interval power impulses (periodicity) depending on their nature of operation. Other important metrics that affect nodes operation include the amount of traffic processed, the number of edges a node has (ingress and egress), and external environmental parameters. Thus, energy consumption is a critical factor in the design of such computation-constricted devices.

2.6 Resource Allocation for IoT-Fog Environments

Resource allocation plays a vital role in IoT-Fog environments, directly impacting the quality of service (QoS) experienced by users. It involves effectively managing the distribution of processing, storage, and communication resources among devices and services within the IoT-Fog network. In fog computing, three primary scheduling problems exist: resource allocation, task scheduling, and workflow scheduling.

The main objective of resource allocation is to optimize the allocation of a diverse set of tasks with varying QoS requirements to distributed and heterogeneous fog nodes. The goal is to achieve low latencies, maximize resource utilization, minimize energy consumption, make-span, and cost. As fog devices handle dynamic and diverse tasks from various fog applications (some being latency-sensitive while others are delay-tolerant), these tasks await execution in a ready queue on resource-constrained fog nodes. Hence, finding an efficient and swift sequencing method for these tasks, based on their criticality, is significant to maximize resource utilization and minimize delays, costs, and energy consumption.

The IoT devices generate jobs that can be decomposed into a series of tasks. These tasks can be either independent or dependent. The independent tasks are not affected by the execution order. However, the dependent tasks can execute only after the completion of their parent tasks. Dependent task scheduling, also known as workflow scheduling. The objective of workflow scheduling is to distribute tasks onto heterogeneous fog nodes and decide an execution sequence of all tasks in a workflow according to their dependencies along with minimization of their makespan.

However, resource allocation in IoT-Fog environments is challenging due to several factors, including the dynamic nature of IoT devices, the heterogeneity of fog nodes, and the scalability of the network. Evolutionary algorithms are a promising approach for addressing these challenges in resource allocation. In the following sections, we discuss the challenges of resource allocation in IoT-Fog environments and the adopted methods to solve this problem with an emphasis on evolutionary algorithms.

2.6.1 Challenges in Resource Allocation for IoT-Fog Environments

The key challenges in resource allocation for IoT-Fog environments include (but are not limited to) nodes/clients' mobility, heterogeneity, scalability, and QoS requirements.

First, IoT devices are often mobile and have varying computational capabilities and power constraints. As a result, the availability and demand for resources constantly change, making it challenging to allocate resources effectively. Machine learning and evolutionary algorithms can provide adaptive and dynamic resource allocation techniques to cope with these changes.

Furthermore, Fog nodes in IoT-Fog environments have varying computing and storage capacities and communication bandwidths. In addition, different fog nodes have different QoS requirements and resource constraints. Allocating resources effectively to fog nodes while satisfying their requirements is a challenging task. Deploying machine learning and evolutionary algorithms can optimize the allocation of resources to heterogeneous fog nodes.

Moreover, IoT-Fog networks can scale up to thousands or millions of devices and services. Resource allocation techniques should be scalable and efficient to handle the large volume of devices and services in the network. Using machine learning and evolutionary algorithms can provide scalable and efficient resource allocation techniques.

Finally, IoT-Fog applications have different QoS requirements, such as latency, reliability, and throughput. Resource allocation techniques should be able to allocate resources in a way that meets the QoS requirements of applications. Machine learning and evolutionary algorithms can optimize resource allocation to meet QoS requirements.

2.6.2 Classification of Resource Allocation Approaches in IoT-Fog Environment

The problem of IoT-Fog environment resource allocation is solved using the approaches usually used to solve resource allocation problems. These approaches include traditional approaches, integer linear programming, heuristic methods, fuzzy logic, reinforcement learning, and deep learning approaches.

2.6.2.1 Traditional Algorithms

Conventional algorithms are characterized as static algorithms wherein all the pertinent information regarding tasks and fog resources is pre-known, enabling scheduling decisions. These algorithms are straightforward and uncomplicated, making them easy to comprehend and implement.

First Come First Served (FCFS) is the simplest method where methods are executed according to their time of arrival [98]. There are no priority or time scheduling in the base version of FCFS. **Round Robin** (**RR**) strategy gives a time slot from the CPU time for each job to be processed. The round-robin method also deploys the first-come-first-served approach [98]. when the time window expires, the task is moved to the end of the queue. If the task ended before the end of the assigned time slot, the task is preempted and the processor time is rescheduled. The **Min-Min** algorithm assigns machines to the shortest execution time jobs first [99]. On the other hand, jobs with longer completion times will have longer waiting times. The **Max-Min** algorithm assigns computing resources to the tasks with the longest execution time first, which might cause starvation for tasks with minimum execution time. The previous approaches can be considered as static methods, other approaches introduce priority to the incoming tasks. The Priority approaches sort the arrived tasks according to some attached criteria (mainly, deadline) and prioritizes the assignment of resources accordingly.

2.6.2.2 Integer Linear Programming

ILP is a widely-used mathematical technique where one or more of the decision variables are further constrained to take integer or binary values. The objective function as well as the constraints, must be linear [100]–[103]. **Mixed ILP (MILP)** is frequently employed to resolve optimization challenges encompassing both continuous and discrete variables, as well as nonlinear functions present in either the objective function or the constraints. In MILP some variables are allowed to be non-integer, but the objective functions are linear. This approach is implemented in works like [104]–[107]. **Mixed Integer Non-Linear Programming (MINLP)** is another approach in integer programming, where variables, objective functions and/or constraints can be non-linear. This method was deployed by works like [108]–[111] to optimize their problems.

2.6.2.3 Heuristic Algorithms

Heuristic algorithms are adaptable and appropriate techniques for optimizing scheduling problems, such as resource allocation, with the objective of delivering solutions efficiently within a brief timeframe. Heuristic algorithms do not guarantee an optimal solution but they give a near-optimal solution. Furthermore, heuristics generally have low time complexity [112].

2.6.2.4 Meta-heuristic Algorithms

Over the past few years, meta-heuristic algorithms have experienced a surge in popularity and have become prevalent in solving intricate computational problems. Researchers employ these algorithms to seek optimal or nearly optimal solutions for task allocation in distributed computing environments. The reason for their widespread usage is their ability to yield near-optimal solutions in a reasonable time frame.

Meta-heuristics algorithms include: **Evolutionary Algorithms**: These algorithms imitate the evolutionary processes found in nature to address optimization problems. Examples of evolutionary algorithms with the corresponding original research include GA [113], Genetic Programming (GP) [114], Evolution Strategy (ES) [115], and Biogeography-Based Optimizer (BBO) [116] are examples of evolutionary algorithms. Genetic Algorithm (GA) utilizes chromosomes to represent each individual, which consist of binary-coded genes. The algorithm initiates with a random selection of the initial population, and an objective function is employed to assess the fitness of each chromosome. To generate a new population, mutation and crossover operations are applied to selected chromosomes. These steps repeat till max generations are iterated or the best offspring is found. An implementation of GA is Nondominated Search Genetic Algorithm (NSGA) [117]. A well-known version of NSGA is the NSGA-II [118] which has been deployed in IoT-Fog environments in works like [119]–[122].

Swarm Intelligence (SI): Swarm intelligence is an evolutionary computation-based technique inspired by the collective social behavior of swarms. Collective intelligence is depicted from the interaction between swarm individuals and the interaction of the swarm with its environment. Well-known examples of SI algorithms include Particle Swarm Optimization (PSO) [123], Ant Colony Optimization (ACO) [124], Artificial Bee Colony (ABC) [125], and Fish Swarm Algorithm (FSA) [126].

Swarm-based optimization includes additional **nature-inspired algorithms** like Grey-Wolf Optimization [127], Monkey Search [128], Bee Collecting Pollen Algorithm (BCPA) [129], Cuckoo Search (CS) [130], Dolphin Partner Optimization (DPO) [131], Bat-inspired Algorithm (BA) [132], and Firefly Algorithm (FA) [133] to name a few.

2.6.2.5 Hybrid Heuristic

A hybrid-heuristic algorithm is a combination of two or more heuristic algorithms that leverages the unique strengths of each algorithm while compensating for their individual limitations at each stage. As a result, it produces superior outcomes compared to using a single heuristic alone. Hybrid algorithms may merge a mono-objective algorithm with a population-based algorithm or even combine two heuristic/meta-heuristic algorithms. Hybrid heuristic algorithms were deployed in the field of resource management inside the IoT-Fog environment to enhance the performance metrics of the network. Exemplar hybrid algorithms include a hybrid of modified particle swarm optimization (MPSO) and modified cat swarm optimization (MCSO) [134], Hybrid Flamingo Search with a Genetic Algorithm (HFSGA) [135], Elitism-based GA (EGA) with PSO [136], and fireworks algorithm (FWA) with Heterogeneous Earliest Finish Time (HEFT).

2.6.2.6 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning that has proven effective in addressing the Job-Shop Scheduling Problem (JSSP). RL involves a sequential decision-making process where an agent acts as the decision-maker and learns to optimize its behavior through interactions with the environment [137]. These algorithms are successful due to their ability to handle uncertainty, self-learn from experience, computational efficiency, and adaptability, making them well-suited for scheduling diverse tasks in a fog computing environment [138]. In RL, self-learning agents take actions in response to specific situations to maximize the rewards they receive. By using RL, a system can effectively map situations to appropriate actions. The key concepts in RL include agents, environment, states, actions, and rewards [139].

In recent times, the fusion of deep learning with reinforcement learning (RL) has led to significant advancements in deep RL (DRL) approaches. This powerful combination has demonstrated exceptional performance in complex control domains, underscoring its superior decision-making abilities within complex and unpredictable environments [140].

Table 2.3: Comparison between Optimization Algorithms

Technology	Merits	Demerits
Traditional	• simple and straightforward to implement • minimal computational cost • predictable and consistent in its outcomes	• unsuitable for fog environments characterized by uncertainty and dynamics • lacks adaptability to changing conditions
ILP, MILP	• versatile • comprehensive • widely employed in scheduling tasks	• unsuitable for complex and dynamic fog environments with high dimensions • incapable of managing non-linear effects
Heuristic	• simple and cost-effective to implement • provides solutions within a reasonable timeframe	• prone to being trapped in local optima due to its greedy nature • lacks flexibility and scalability
Meta Heuristic	• capable of finding a nearly optimal solution within a reasonable timeframe • improves resource utilization	• Takes a considerable amount of time to converge • Does not support mobility factor
Hybrid-Heuristic	• Outperforms heuristic algorithms • Yields quick solutions • Demonstrates flexibility	• Identifying the optimal hybrid solution is challenging • Lacks adaptability
Reinforcement Learning	• Simple implementation • Superior performance in uncertain environments	• Challenging to apply in diverse scenarios • Not suitable for complex and dynamic fog computing environments
Deep Reinforcement Learning	• Outperforms basic Reinforcement Learning • Demonstrates strong performance in complex, uncertain, and dynamic fog computing environments	• Exhibits slow convergence when dealing with high-dimensional state-action spaces.

From the previous discussion, we can conclude that traditional and integer linear programming approaches are not fully compatible with the IoT-Fog environment without modification. Despite the simplicity of traditional algorithms, they lack adaptability in the dynamic, and uncertain environment like the IoT-Fog environment. Furthermore, integer linear programming (ILP) algorithms are widely used in scheduling problems (Huang et al., 2022b; Ma et al., 2022b; Mouradian et al., 2019b; G. Peng et al., 2021; Zhou et al., 2022b). ILP algorithms are Non-deterministic Polynomial-time hardness (NP-hard) problems and the complexity of this type of problem grows dramatically with the number of variables (objectives). Hence, when the number of the environment’s variables increases and the nature of objectives is complex, the feasibility of these approaches becomes less and less attractive to researchers. For the stated reasons this work did not implement LP in the proposed solution.

Now, if we compare heuristic algorithms with reinforcement learning (RL) techniques we can notice the following; first, RL typically requires a significant amount of interaction and feedback from the environment to learn optimal policies. In fog computing, obtaining real-time feedback or accurate performance metrics for resource allocation decisions can be challenging. Evolutionary algorithms, on the other hand, can operate without explicit feedback and can be applied in scenarios where limited or no feedback is available.

Next, heuristic algorithms tend to have simpler and more intuitive structures compared to RL algorithms, which often involve complex neural network architectures and learning processes. This simplicity can make heuristic algorithms easier to understand, interpret, and implement in resource allocation scenarios, especially when the problem requirements are well-defined and transparent.
Furthermore, evolutionary algorithms inherently explore the solution space by maintaining a diverse population of candidate solutions. This can be advantageous in fog computing resource allocation, as it allows for broad exploration of different resource allocation configurations. In contrast, RL algorithms often rely on iterative learning through trial and error, which may be time-consuming and may not guarantee comprehensive exploration of the solution space.

Additionally, resource allocation in fog computing often involves optimizing multiple objectives simultaneously, such as minimizing latency, maximizing reliability, and minimizing energy consumption. Evolutionary algorithms excel at handling multi-objective optimization problems and can find a set of diverse solutions that represent tradeoffs between different objectives. RL, on the other hand, may require additional techniques or modifications to handle multi-objective optimization effectively.

In conclusion, we can now state that for the reasons discussed above, evolutionary algorithms are the best approach to solve our problem for IoT-Fog environment resource allocation.

2.7 Optimization

Generally, solving resource allocation in IoT-Fog environment is a challenging problem because it aims to satisfy a number of goals. For instance, when the problem needs to satisfy a minimum use of energy, monetary cost, network resources and computing resources, the solution would be a tradeoff among these targets. From this, an optimization approach can be used to get a near-optimal solution. In practice, optimization problems categorized into single-objective optimization (SOO), and multi-objective optimization (MOO). First, SOO are used when the goal is to optimize a problem with single objective. Next, MOO is more suitable for solving resource allocation problems in IoT-Fog environment due to the multiple goals that the problem inherently has.

In this work we will discuss only MOO problems because it is the method for solving our problem, and the method is similar to SOO and differ only in the number of objectives.

2.7.1 Multi-Objective Optimization Problems

In SOO the result will be a single solution, on the other hand MOO will generate multiple solutions that satisfy the desired goals with different compromises. The mathematical form of a minimization MOO is:

$$minimize\ F(\vec{x}) = f_1(\vec{x}), f_2(\vec{x}), \ldots, f_o(\vec{x}) \tag{2.1}$$

$$subject\ to: g_i(\vec{x}) \geq 0,\ i = 1, 2, .., m \tag{2.2}$$

$$h_i(\vec{x}) = 0, i = 1, 2, \ldots, p \tag{2.3}$$

$$L_i \leq x_i \leq U_i, i = 1, 2, \ldots, n \tag{2.4}$$

Where *n* is the number of variables search space, *o* is the number of objective functions, m is the number of inequality constraints, p is the number of equality constraints, g is the inequality constraints, *h* indicates the equality constraints, [L, *U*] are the boundaries of variables, and i is the variable index.

In SOO solutions can be compared using binary relations like (>, <, and =) because there is only on objective function. For example, in a minimization problem the relation *X*<*Y* (both *X* and *Y* are solutions in the problem) decides if solution *X* has less objective value than *Y*. On the other hand, this approach is not feasible in MOO problems. In this scenario, one solution is considered superior to (dominates) another solution if and only if it achieves equal or superior objective values across all objectives and outperforms the other solution in at least one objective function. This concept is called the Pareto-dominance [146].

The Pareto-optimality of solution X states that there should be no other solution *Y* that is better than (dominant) X, as illustrated in the following relations [147].

$$\begin{gathered}if\ we\ have\ two\ vextors\ \vec{x} = [x_i |\ i = 1, 2, \dots, k], and\ \vec{y} = [y_i |\ i = 1, 2, \dots, k],\\ then\ \vec{x}\ donimnates\ \vec{y}\ (\vec{x} \succ \vec{y}) iff:\\ \forall i \in \{1, 2, \dots, k\}, [f(x_i) \geq f(y_i)] \wedge [\exists i \in \{1, 2, \dots, k\}: f(x_i)]\end{gathered} \quad (2.5)$$

$$\begin{gathered}solution\ x \in X\ is\ called\ Pareto - optimal\ iff:\\ \nexists \vec{y} \in X | F(\vec{y}) \succ F(\vec{x})\end{gathered} \quad (2.6)$$

Another two important definitions related to MOO are the Pareto optimal set and Pareto optimal front. Both concepts are illustrated below.

$$\begin{gathered}The\ Pareto\ set\ is\ the\ set\ of\ all\ Pareto\ optimal\ solutions:\\ P_s \coloneqq \{x, y \in X | \exists F(y) \succ F(x)\}\end{gathered} \quad (2.7)$$

The Pareto optimal front is the set includes all the objective values of the solutions in the Pareto optimal set: (2.8)

$$P_f \coloneqq \{F(x) | x \in P_s\}$$

Now after presenting the essential concepts in MOO, we can illustrate the meta-heuristic algorithms used in the work. First, the bio-inspired algorithm NSGA-II is presented. Next, we present the nature-inspired algorithm MOGWO.

2.8 Non-dominated Sorting Genetic Algorithm II (NSGAII)

The Non-dominated Sorting Genetic Algorithm (NSGA), which was introduced in [117], stands out as one of the most effective evolutionary algorithms (EAs) for multi-objective optimization [148]. NSGA is equipped with a ranking procedure that prioritizes the convergence of solutions, making it capable of discovering optimal or near-optimal solutions. NSGA-II [120], an improved version of NSGA, distinguishes itself through its swift elitist ranking procedure. This means that NSGA-II consistently retains the best solutions (those with higher ranks in the nondominated set) within the most recent population.

The motivation of NSGA-II was to solve the following criticisms of the original work (NSGA):

1. The existing nondominated sorting algorithm used in NSGA has a high computational complexity of $O(MN^3)$ where N represents the population size, and M is the number of objectives. Consequently, when dealing with large population sizes, NSGA becomes computationally expensive due to the intricate nature of the nondominated sorting procedure in each generation.
2. An absence of elitism in NSGA has been highlighted in earlier works like [149]. Introducing elitism to the genetic algorithm (GA) has been shown to significantly enhance performance by preserving valuable solutions once they are discovered.
3. Ensuring diversity in the population to obtain a wide range of equivalent solutions traditionally relied on sharing concepts. However, this approach demands the specification of a sharing parameter (σ_{share}), which can be problematic.

The study reveals that NSGA-II performs better than two other modern (at the time of the study) multi-objective evolutionary algorithms (MOEAs), namely, Pareto-archived evolution strategy (PAES) [150] and strength Pareto EA (SPEA) [151], both in terms of discovering a diverse range of solutions and in approaching the true Pareto-optimal set. The contributions of NSGA-II are: 1) Fast nondominated sorting approach and 2) Diversity preservation.

The fast nondominated sorting approach reduces the computational complexity of the algorithm from $O(MN^3)$ to $O(MN^2)$. For the set of solutions found in the first iteration of the algorithm, two quantities are calculated: i) the domination count n_p, which represents the number of solutions that dominate the investigated solution, and ii) the set of individuals S_p that the solution dominates.

The sharing parameter of the largest distance σ_{share} considered one of the drawbacks of NSGA because it should be tuned manually. On the other hand, NSGA-II uses the value from the crowding distance function to preserve the solutions diversity. This approach eliminates the need for the sharing parameter. To achieve diversity preservation the algorithm introduces two methods; first method is the distance estimation function, and second method is the crowding comparison operator.

The crowding distance of solution i ($i_{distance}$) is calculated by the average side distance of the cuboid formed from its nearest solutions taken from the same front (Figure 2.5) where solid circles represent the solution front, and dashed lines represent the surrounding cuboid of solution i.

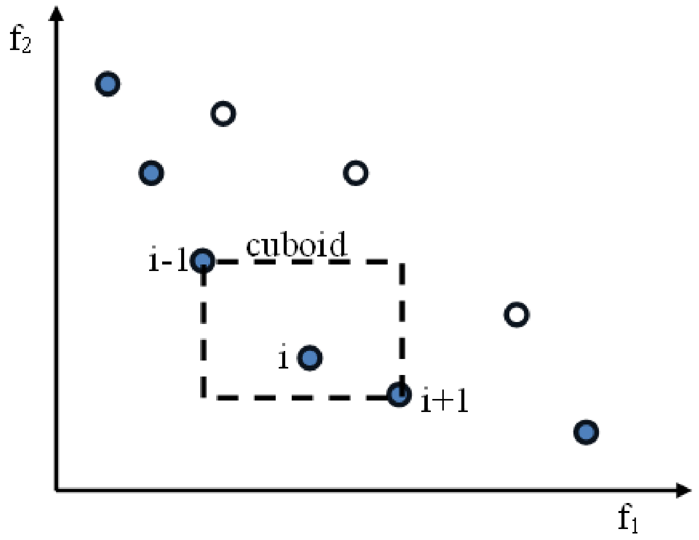

Figure 2.7: Crowding distance in NSGA-II

Algorithm 2.1 shows the method to calculate the crowding distance as proposed in NSGA-II. To ensure that the two solutions that form the edges of the crowding cuboid are always selected, their distance is set empirically to infinity (∞).

```
ALGORITHM 2.1: CrowdingDistance
INPUT: F: the set of fronts in the current generation, O is the set of objective functions
OUTPUT: Crowding distance for each solution
BEGIN
1. SET N to the number of fronts in the current generation
```

```
2. FOR every front $i$ in $F$
  2.1 Initialize distance for all individuals to 0
  2.2 FOR every objective $o$ in $O$
    2.2.1 Sort front $F_i$ on objective $o$ using the Non-Dominated Sort method
    2.2.2 FOR every individual $j$ in front i
      2.2.2.1 IF objective $o$ is the boundary, assign infinite distance to it, i.e., $F_{i,j}^{d} = \infty$
      2.4.3.1 ELSE, $F_{i,j}^{d} = F_{i,j}^{d} + \frac{F_{i,j+1}^{o} - F_{i,j-1}^{o}}{f_{o}^{max} - f_{o}^{min}}$
4. RETURN $F$ with the calculated crowding distances
END
```

The next proposed enhancement is the *crowded-comparison operator* ($\prec_n$) that measures the Pareto front solutions density. This operator provides the required spread of solution along the Pareto front. In the first step of this procedure each solution is given a rank according with its index inside its front after the non-dominated sort. Now each solution will have a rank and a crowding distance. Next, for each two solutions in front F_i, $F_{i,p}$ and $F_{i,q}$ the solution $F_{i,p}$ will be included int next generation if:

1. The rank of $F_{i,p}$ less than the rank of $F_{i,q}$.
2. If they have the same rank the $F_{i,p}$ will be selected if its crowding distance is greater than $F_{i,q}$ $\left(F_{i,p}^{d} > F_{i,q}^{d}\right)$.

Although NSGA-II is a widely used and effective multi-objective optimization algorithm, but it has some limitations. Its drawbacks include high computational complexity, lack of explicit elitism, potential premature convergence, difficulty in handling constraints, and nonuniform distribution of solutions. Additionally, NSGA-II may struggle with problems involving many objectives or complex Pareto fronts. Despite these limitations, NSGA-II remains popular due to its overall effectiveness, and researchers continue to work on improving and adapting the algorithm. Figure 2.5 illustrate the general process of NSGA-II.

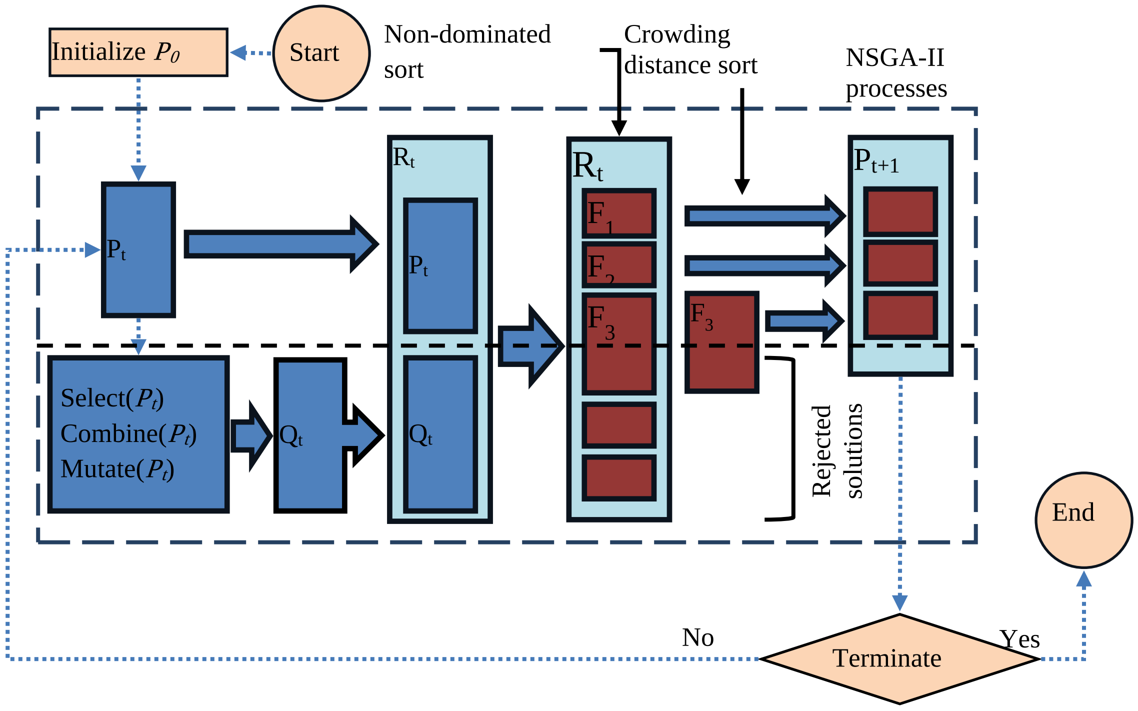

Figure 2.8: NSGA-II Process

2.9 Multi-Objective Grey-Wolf Optimization (MOGWO)

The multi-objective grey-wolf optimization algorithm is an enhancement to the original single-objective GWO [127]. GWO is a nature-inspired optimization algorithm motivated by the social hunting behavior of the Grey-wolf pack. The MOGWO has added a number of contributions to the original GWO including; the nondominated solutions were preserved using an archived solution, and a grid approach was introduced to control the explored space of the archived wolves. Furthermore, the selected three leaders (alpha, beta, and delta) were used to update the archived population.

In MOGWO the alpha leader is the fittest solution so far, then beta and delta wolves are the second and third fittest solution respectively. The omega population is the rest of the candidate solutions. The three leader wolves control the hunting process which comprises the following steps: first, tracking, chasing, and approaching the prey. Next, encircling the pray to kame it stops. Finally, attacking the prey.

The hunting behavior is derived by two variables controls the exploration of the three leader wolves $(\vec{A})$ and $(\vec{C})$ which calculated as follow:

$$\vec{A} = 2.0 \times \vec{a} \times \vec{r}_1 - \vec{a} \tag{2.9}$$

$$\vec{C} = 2.0 \times \vec{r}_2 \tag{2.10}$$

Where a is a vector of values decreasing from 2.0 to 0 during algorithm iterations, and r_1 and r_2 are random number in the range [0,1]. As mentioned before the sigma population follow the three leaders, to update the position of the rest of the pack MOGWO derive the following equations:

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_\alpha - \vec{X}\right| \tag{2.11}$$

$$\vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}\right| \tag{2.12}$$

$$\vec{D}_\delta = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}\right| \tag{2.13}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\vec{D}_\alpha \tag{2.14}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\vec{D}_\beta \tag{2.15}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\vec{D}_\delta \tag{2.16}$$

$$X_{t+1} = \frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3} \tag{2.17}$$

The effect of $\vec{A}$ is that it ensures exploration for the wolves during hunting. When the value of $\vec{A}$ is larger than 1 the wolf will approach the prey (converge), and when it is less than -1 the wolf will withdraw (diverge) from the prey. This effect generates a stochastic behavior for the wolves' hunting process.

The GWO algorithm initiates optimization by generating a random set of solutions as the initial population. Throughout the optimization process, the three best solutions obtained so far are saved and designated as alpha, beta, and delta solutions. The position updating formulas (2.11) to (2.17) are applied to all search agents except alpha, beta, and delta. Concurrently, the parameters "a" and "A" are gradually reduced during iterations. This causes the search agents to diverge from the prey when $|A| > 1$ and converge towards the prey when $|A| < 1$. Ultimately, the position and score of the alpha solution are returned as the best solutions obtained during optimization when a termination condition is met. Figure 2.9 illustrate the general process of the MOGWO algorithm.

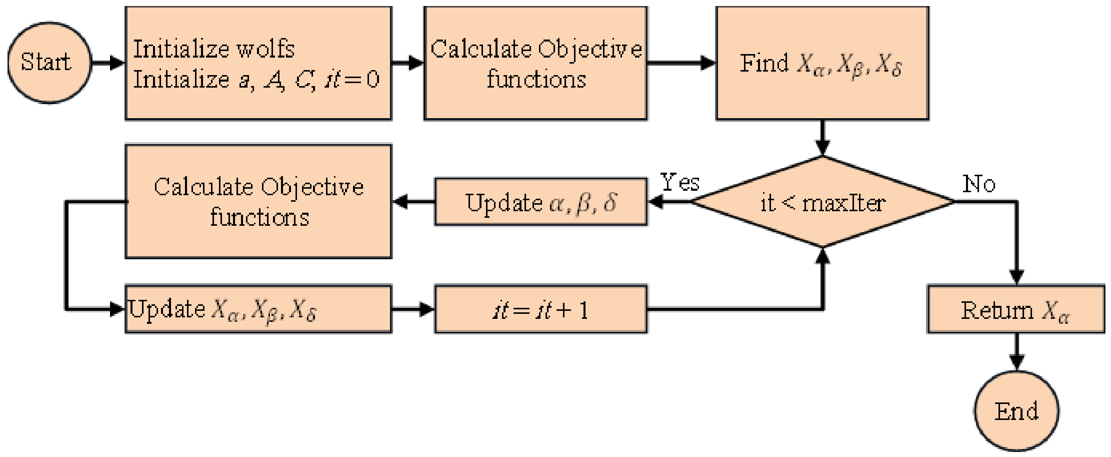
Start
Initialize wolfs
Initialize a, A, C, $it = 0$
Calculate Objective functions
Find $X_\alpha, X_\beta, X_\delta$
$it < maxIter$
Yes
No
Update α, β, δ
Calculate Objective functions
Update $X_\alpha, X_\beta, X_\delta$
$it = it + 1$
Return X_α
End

Figure 2.9: The general process of the MOGWO algorithm.

One of the drawbacks that MOGWO algorithm suffers from is the lack of elitism in its implementation. To overcome this limitation this work introduces elitism to this algorithm to maintain diversity on the generated population and enhance exploration of its individuals.

The next section discus elitism in general and the two implemented elite methods adopted immigration-based elitism and opposition-based learning strategy.

2.10 Elitism in Multi-Objective Evolutionary Algorithms

In general, elitism is the process of inserting new individuals into the population to be used in the next generation [25]. Different strategies were implemented by researchers; the most important strategies are:

1- Finding all the non-dominated solutions in the population and directly insert them into the next generation population.
2- Using a union of the generated population and the non-dominated solutions and select the best individuals for the next generation.
3- Maintaining an external archive for the best solutions so far (non-dominated solutions) and updating this archive each time a new generation is produced.

2.10.1 Adaptive Migration-Based Elitism

This method selects a ratio of the population to replace the worst individuals in the current population. The number of elite populations im is calculated by multiplying the population size *NP* by an adaptive random variable r_{el} as in Eq. 2.18.

$$im = r_{el} \times NP \tag{2.18}$$

The probability of considering an elite population is controlled by an adaptive random variable P_m in the range [-1, 1]. If P_m is greater than a random number *rand* then the elite process will take place, otherwise, the population will stay unchanged [152]. The adaptive process depends on comparing the average fitness of the current population with the average fitness of the elite population. When the elite population has larger average fitness then P_m will be increased by a random number in the range [0, 1] to decrease the probability of inserting new elite individuals in the current population as illustrated in Eq. 2.19. The process of adaptive immigration-based elitism is illustrated in Algorithm 2.2.

$$P_m' = \begin{cases} P_m + C_1, if\ avg(A^{cost}) > avg(P^{cost}) \\ P_m - C_1, otherwise \end{cases} \tag{2.19}$$

This adaptation method reduces the probability of inserting individuals that might guide the optimization process in favor of one objective than another objective. Another advantage of adaptation is the provision of parallelism in selecting the elite population.

ALGORITHM 2.2: Adaptive Immigration-Based Elitism
INPUT: *P* population of the current generation
OTPUT: *E* elite population
BEGIN

```
1. Initialize the method's parameters from the calling algorithm as follows:
  1.1 SET $P_m = rand(-1, 1)$
  1.2 SET $C_1 = rand(0,1)$
  1.3 SET $C_2 = rand(0,1)$
  1.4 SET $E \leftarrow \{\emptyset\}$
  1.5 SET $N$ to the number of individuals in $P$
2. IF $rand$ is less than $P_m$
```

```
  2.1 SET number of immigrated population $im$ to $r_{el} \times N$
  2.2 Select $im$ worst individuals from $P$ and assign it to $E$
  2.3 IF $avg(E^{cost})$ is greater than $avg(P^{cost})$
    2.3.1 Increase $P_m$ by $C_1$
    2.3.2 ELSE, decrease $P_m$ by $C_1$
4. RETURN elite population $E$
END
```

Stagnation mitigation using the adoptive immigration-based elitism is accomplished using the r_{el} adoptive variable. The method followed compares the average crowding distance of the elite population $\left(avrg\left(E^{\|crowding\|}\right)\right)$ with the crowding distance of the immigrated population before transformation $\left(avrg\left(P\{im\}^{\|crowding\|}\right)\right)$. The adopted random variable r_{el} will be updated as illustrated in Eq. 2.20:

$$r_{el} = \begin{cases} r_{el} + (r_{el} \times C_1) \\ r_{el}, otherwise \end{cases} \tag{2.20}$$

When the algorithm detects a stagnation in the generated solutions, it uses the elite population in parallel with MOGWO to inject new solutions to the population. Therefore, the extended generation will provide diversity enhanced individuals that can retain optimality to the next generations.

2.10.2 Opposition-Based Learning Strategy

This approach states that if trying to get an estimate $\tilde{x}$ of a solution x then we might be far from the actual value so with no prior knowledge, we can take the opposite of x [153]. In environment like IoT-Fog and a problem like resource allocation, this assumption is feasible, because if we try a worst placement solution then its opposite might be better and the solution would converge faster.

The method of calculating an opposite of a value is straight forward (Eq. 2.21) and for a vector the method will be calculated for each element (Eq. 2.22).

Let $x \in \Re$ be a real number in the range $x = [a, b]$.
The opposition value $\tilde{x}$ is $\tilde{x} = a + b - x$ (2.21)

Let $P(x_1, x_2, \ldots, x_n$ in an n-dimensional space with $x_1, x_2, \ldots, x_n \in \Re$ and $x_i \in [a_i, b_i]$.
The opposition point $\tilde{P}$ is defined by:
$\tilde{x}_i = a_i + b_i - x_i, for\ i = 1,2, \ldots, n$ (2.22)

Now we can formulate the opposition-based learning as follow: If we have a desired function $f(x)$ with an evaluation function $g(.)$, and x is an element in $f(x)$ with opposite value $\tilde{x}$. Then if $g(f(x)) \geq g(f(\tilde{x}))$ then we continue using x otherwise $\tilde{x}$ is considered.

2.11 The Proposed Optimization Algorithm Evaluation Metrics

The evaluation of the proposed algorithm done with two methods: 1) benchmark test functions, 2) multi-objective evolutionary algorithms performance indicators.

2.11.1 Benchmarking Test Functions

Generally, the benchmark functions provide a tool to measure and compare algorithms performance. The set of functions used in this work is the IEEE CEC09 [154], and the subset of functions implemented are functions from UF1 to UF10. Table 4.1 shows the functions formulae, number of objectives, number of test variables, and search space bounds.

2.11.2 Multi-objective EA Performance Indicators

The following discussion on the performance indicators assumes the existence of two variables A and Z. The first variable A is the objectives values obtained from our algorithm taking the form $A = \{a_1, a_2, \ldots, a_{|A|}\}$. The second variable is Z which is a set of pareto front [11] points and taking the form $Z = \{z_1, z_2, \ldots, z_{|Z|}\}$.

1. The first indicator is **Generational Distance (GD)** [155], this indicator measures the distance from known solution to the Pareto-front reference points, or in another words, compute the average distance from the obtained solution to the reference points, Eq 2.23. The relation below describes how the GD indicator works. Where *d* represents the Euclidian distance (L_2) between point a_i and the nearest point in *Z*, as in Eq. 2.23.

$$GD(A) = \frac{1}{|A|}\left(\sum_{i=1}^{|A|} d_i^p\right)^{1/p} \qquad (2.23)$$

2. The second indicator is the **Generational Distance Plus (GD$^+$)** [155] this indicator is similar to the GD indicator; except they use a modified distance measure d^+. Where d^+ is the minimization distance between the solution point a_i and the reference point z_i, $d^+ = max\{a_i - z_i, 0\}$, Eq 2.24.

$$GD^+(A) = \frac{1}{|A|}\left(\sum_{i=1}^{|A|} {d^+}_i^2\right)^{1/2} \qquad (2.24)$$

3. Another indicator is the **Inverted Generational Distance (IGD)** [156] which measures the inverted distance $(\hat{d})$ i.e., from z_i to the nearest a_i, Eq. 2.25.

$$IGD(A) = \frac{1}{|Z|}\left(\sum_{i=1}^{|Z|} \hat{d}_i^p\right)^{1/p} \qquad (2.25)$$

4. The **Inverted Generational Distance Plus (IGD$^+$)** [156] indicator is another distance related metric. This indicator uses the same modified distance method d^+ used in GD+. Eq. 2.26.

$$IGD^+(A) = \frac{1}{|Z|}\left(\sum_{i=1}^{|Z|} {d^+}_i^2\right)^{1/2} \qquad (2.26)$$

5. The **Hypervolume (HV)** [157] indicator is a measure of solutions diversity (in contrast to other discussed indicators). The HV indicator does not need to know all the Pareto front, only one reference point suffices. The Hypervolume calculates the maximum area/volume between the reference point and the solution points.

6. The Spacing (SP) [158] indicator measures the spreading score of a solution to the approximated Pareto front. For solution *S* this indicator is calculated as follows (Eq. 2.27):

$$SP(S) = \sqrt{\frac{1}{|S|-1}\sum_{i=1}^{|S|}\left(\bar{d} - d_i\right)^2} \tag{2.27}$$

Where d_i is the l_1 distance between a point $s_i \in S$ and the closest point of the Pareto front approximation produced by the same algorithm, and $\bar{d}$ the mean of the d_i.

7. This work considers the **Ratio of Superiority (*RS*)** as another source of algorithms' performance comparison. For all the tables in this section, the Ratio of Superiority (*RS*) for our proposed algorithm verses other algorithms is calculated as in Eq. 2.28:

$$RS = \left(\frac{X_1 - X_2}{X_1}\right) \tag{2.28}$$

The *RS* metric measures the superiority of X_1 over X_2. If *RS* is positive then X_1 is better than X_2, otherwise X_1 is worse than X_2.

Table 2.4: Benchmarking functions used in this work

F	**FORMULA**	**N**	**O**	**BOUNDS**
UF1	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_2 = 1 - \sqrt{x}_1 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ WHERE $J_1 = \{j \mid j\ is\ odd\ and\ 2 \le j \le n\}$ AND $J_2 = \{j \mid j\ is\ even\ and\ 2 \le j \le n\}$.	30	2	$[0,1] \times [-1,1]^{n-1}$
UF2	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2$ $f_2 = 1 - \sqrt{x}_1 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ WHERE J_1 AND J_2 SAME AS UF1 AND $y_j = \begin{cases} x_j - \left[0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] cos\left(6\pi x_1 + \frac{j\pi}{n}\right)\ j \in J_1 \\ x_j - \left[0.3x_1^2 cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\ j \in J_2 \end{cases}$	30	2	$[0,1] \times [-1,1]^{n-1}$
UF3	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = 1 - \sqrt{x}_1 + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ WHERE J_1 AND J_2 SAME AS UF1 AND $y_j = x_j - x_1^{0,5\left(1.0+\frac{3(j-2)}{n-2}\right)}, j = 2, \ldots, n.$	30	2	$[0,1]^n$
UF4	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j)$ $f_2 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j)$ WHERE J_1 AND J_2 SAME AS UF1 AND $y_j = x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \ldots, n$ $h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$	30	2	$[0,1] \times [-2,2]^{n-1}$

UF5	$f_1 = x_1 + \left(\frac{1}{2N} + \varepsilon\right)\lvert sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j)$ $f_2 = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)\lvert sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h(y_j)$ WHERE *J₁* AND *J₂* SAME AS UF1 AND *N*=10, E=0.1 $y_j = x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \dots, n$ $h(t) = 2t^2 - cos(4\pi t) + 1$	30	2	$[0,1] \times [-1,1]^{n-1}$
UF6	$f_1 = x_1 + max\left\{0,2\left(\frac{1}{2N} + \varepsilon\right) sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in j_1} y_1^2 - 2\prod_{j\in j_1} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = 1 - x_1 + max\left\{0,2\left(\frac{1}{2N} + \varepsilon\right) sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ WHERE *J₁* AND *J₂* SAME AS UF1 AND *N*=2, E=0.1 $y_j = x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \dots, n$	30	2	$[0,1] \times [-1,1]^{n-1}$
UF7	$f_1 = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2$ $f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ WHERE *J₁* AND *J₂* SAME AS UF1 AND $y_j = x_j - sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \dots, n$	30	2	$[0,1] \times [-1,1]^{n-1}$
UF8	$f_1 = cos(0.5\pi x_1)cos(0.5\pi x_2) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_2 = cos(0.5\pi x_1)sin(0.5\pi x_2) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_3 = sin(0.5\pi x_1) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ WHERE: $J_1 = \{j \mid 3 \le j \le n, and\ j-1\ is\ a\ multiplication\ of\ 3\}$ $J_2 = \{j \mid 3 \le j \le n, and\ j-2\ is\ a\ multiplication\ of\ 3\}$ $J_3 = \{j \mid 3 \le j \le n, and\ j\ is\ a\ multiplication\ of\ 3\}$	30	3	$[0,1]^2 \times [-2,2]^{n-2}$

UF9	$f_1 = 0.5[max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\} + 2x_1]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_2 = 0.5[max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\} - 2x_1 + 2]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $f_3 = 1 - x_2 + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left[x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ WHERE J1, J2, AND J3 ARE SAME AS UF8 AND E=0.1.	30	3	$[0,1]^2 \times [-2,2]^{n-2}$
UF10	$f_1 = \mathrm{COS}(0.5\pi x_1)cos(0.5\pi x_2) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[4y_j^2 - cos(8\pi y_j) + 1\right]$ $f_2 = \mathrm{COS}(0.5\pi x_1)sin(0.5\pi x_2) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[4y_j^2 - cos(8\pi y_j) + 1\right]$ $f_3 = \mathrm{SIN}(0.5\pi x_1) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left[4y_j^2 - cos(8\pi y_j) + 1\right]$ WHERE J1, J2, AND J3 ARE SAME AS UF8 AND $y_j = x_j - 2x_2 sin\left(2\pi x_1 + \frac{j\pi}{n}\right)$	30	3	$[0,1]^2 \times [-2,2]^{n-2}$

2.12 IoT-Fog Environment Network Design

The second stage in the proposed work is the IoT-Fog environment network design. Network implemented in this research is principally consists of entities and services.

Generally, the network physical topology adopts the layered architecture of fog environment as described in Section 2.4.2. Hence, entities presented as fog and cloud devices (nodes), and clients (Figure 2.10). Furthermore, clients can have sensors and/or actuators installed on them.

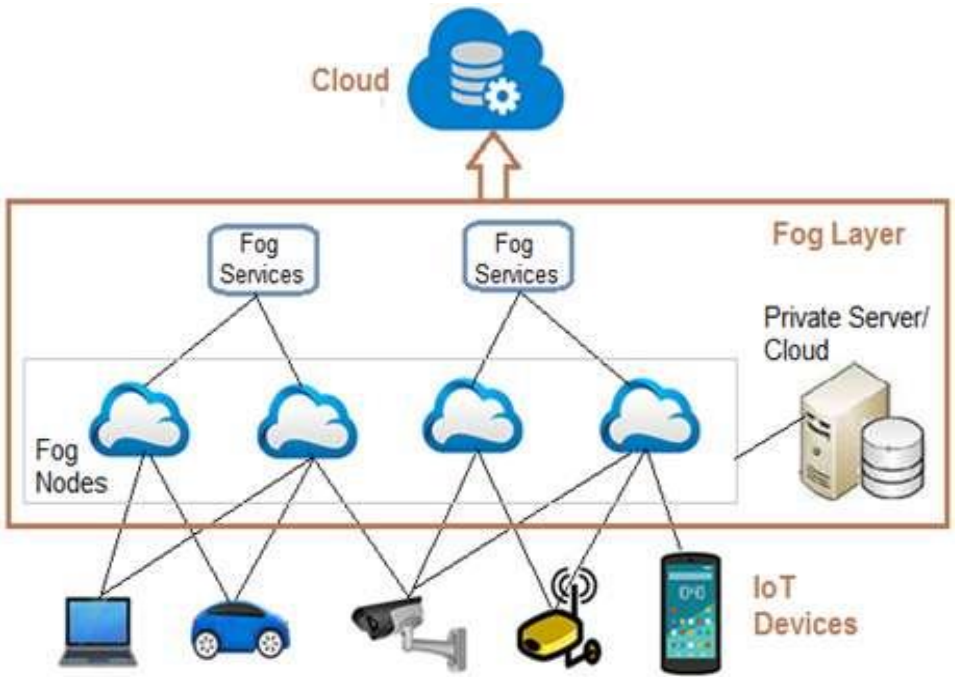

Figure 2.10: IoT-Fog Environment Entities [159]

The IoT-Fog infrastructure devices are interconnected by channels of communication for both cloud and fog nodes [160].

Typically, the edge devices of the network can be mobile (installed inside moving objects such as vehicles, or humans carrying monitoring devices). Following the fog computing paradigm, network links for static devices might be connected to several nearby nodes utilizing wide range of connection media. On the other hand, mobile entities are generally connected to the nearest available stationary node (fog or cloud) via wireless gateway. The logical part of the network is the application(s) installed on the client devices. The method that the application implemented follow the distributed processing paradigm. In consequence, applications are modelled as dependent group of modules (methods) distributed across different nodes as illustrated in Section 2.4. In addition to the aforementioned entities, the architecture assumed the existence of a controller entity that fully aware of all the environment entities and links deployed and their specifications. The controller node represents the network as a directed graph with nodes as vertices and communication links as peer-to-peer edges. Generally, network links do not have to be fully connected, yet there always a link between any two nodes in the network. Meanwhile, the controller (i.e., broker) sees the application as a Directed Acyclic Graph (DAG) with modules as vertices and modules data dependencies as edges, as described in Section 2.12.2. Due to the amount of information available, the controller can deploy the optimization algorithm and obtain a solution on how to map modules on the nodes to run the required application. The next sections describe the main components of the IoT-Fog network.

2.12.1 Network Entities

In principle, the network environment entities comprise of sensors, actuators, and devices (i.e., clients, fog, cloud). An overview of these entities and their collaborations are presented in this section.

2.12.1.1 Network Sensors and Actuators

Network sensors are the IoT part of this research architecture. Sensors are characterized by several specifications like connectivity type, periodicity of messages and the generated messages attributes. For a sensor deployed in the network, it assumed that each sensor has a gateway node that it connected to (i.e., client node). Furthermore, sensors generate sensing data that define the output tuple type to be send to the gateway, the target module, and the transmission distribution rate (messages/time unit). The transmission rate can follow any probability distribution model such as deterministic, normal, or unform to simulate different behaviours of sensing environment. The type of distribution selected will determine the time delay between two generated tuples.

Furthermore, actuator entities are also attached to client devices and have a dedicated connected gateway node. Generally, IoT applications start from a sensor and end with an actuator, which represent an application loop or flow. As soon as a tuple (message) reaches the actuator calculations can be conducted to calculate the end-to-end delay for the corresponding application flow to measure metrics like QoS satisfaction and ultimately validate the resource allocation algorithm performance.

2.12.1.2 Messages (Tuples)

In IoT-Fog environment, messages are the principal element used by the environment entities to communicate between each other. When a network system component (i.e., node or sensor) needs to transmit data to another component (i.e., node or actuator), it formulates a message and transmit it to its gateway if the target is an actuator or to the next node following its processing dependency path. Tuples are tagged by the identifiers of the originated and targeted modules, and the message type. The tuple has also fields that specify the number of resources needed to process the message in terms of Million Instructions (MI), with the length of the attached data (in Bytes) for the network payload. Furthermore, a specialized tuple type is formulated when an application module is need to be migrated to another entity.

2.12.1.3 Devices (Nodes)

Nodes represent the physical network computing devices, which can be categorized as clients, fog, and cloud devices. Generally, computing categories are similar because they all represent physical devices, yet they are different in their computational powers, cost, and energy consumption. Network device (node) computing power is characterized by processing speed (in Instructions Performed per Time unit (IPT)), memory size (in Bytes), and capacity of storage (in Bytes).

Power consumption is modelled by two factors during the operation of the network. First, amount of energy consumed during the busy (or dynamic) time. Second, during idle (or static) time when the node has no task to perform.

Furthermore, nodes are characterized by the cost price of using their resources like CPU, memory, storage, and bandwidth used.

Generally, nodes are responsible of many actions assisting in the process of resource provisioning. When the controller finishes deploying the optimization algorithm it sends a tuple encapsulating the required module to the corresponding nodes to allocate the required resources to modules, these modules will be initialized as virtual machines (VMs) and resources are reserved accordingly. Furthermore, if the node receives a data tuple intended to one of its hosted modules, it will be submitted to that VM for processing. When a tuple received by the node, the node will calculate the service time for the tuple depending on its processing specifications and the VM reserved processing power. The environment nodes maintain a default FIFO queue to schedule arrived tuples. Additionally, nodes responsible of sending any periodic tuple to the next node in the application flow depending on the module dependency. Moreover, during simulation nodes calculate statistics like resources usage costs and energy consumption for the calculation of simulation results. The environment resources usage cost is calculated during resource busy time per amount used per node. For example, to calculate the price of utilizing CPU usage C_n^{Ipt} in node *n* for the time period of Δt with utilization ratio of $u_n(t)$ is illustrated in Equation 2.28.

$$C_n^{Ipt}(\Delta t) = \left(C_n^{Ipt} u_n(t)\right)\Delta t \tag{2.28}$$

In this research, energy consumption is modeled as a linear power consumption model with idle power as the constant factor and the scaling factor as the difference between the busy power and the idle power consumptions. For instance, to calculate power consumption $f_n^{P_w}$ for node *n* with idle power $f_n^{iP_w}$ and busy power $f_n^{bP_w}$ and utilization ratio $u_n(t)$ we use Equation 2.29.

$$f_n^{P_w}(t) = f_n^{iP_w} + \left(f_n^{bP_w} - f_n^{iP_w}\right) \times u_n(t) \tag{2.29}$$

2.12.1.4 Network Controller

The network topology of this research suggests an occurrence of a controller entity that monitors the system components instances. Hence, this controller has the ability to carry out the resource allocation algorithm solution and instruct nodes regarding their rule in the environment. Before the environment initialization, the controller will deploy the proposed optimization algorithm to obtain the initial placement map. Principally, the controller responsible of a number of activities including entities start up, updating network topology, instructing nodes to deploy application modules (VMs), and monitoring the environment.

Upon the initialization of the environment, the controller will attempt to start all the network components and the placement of the application modules on the corresponding nodes as suggested by the resource allocation optimization algorithm. Additionally, depending on the provided solution, the controller generates all the necessary routing tables (i.e., the nodes hop table, the application mapping table, and the modules migration table).

Furthermore, the controller monitors the position of mobile clients to ensure they are connected to the nearest node, this action is regularly performed by scheduling a management event in the controller queue. This behavior ensures the stability of the network topology by creating the required communication links or removing them during handovers or modules migration. Principally, when a mobile client exits the coverage of its computing cluster, a handover action will be performed. During the handover state, the controller will attempt to recompute the allocation algorithm to get a new solution and depending on that solution routing tables will be updated.

Generally, when a node intends to migrate an application module to another node and before sending the VM it informs the controller of this events. Hence, the controller will always be informed of the position of every application module in the environment.

Finally, regarding the simulation environment, after the completion of all events in the simulation the controller sends an end of simulation event to the simulator to terminate the simulation execution. After this the simulator will finalize and save the simulation outcomes.

2.12.1.5 Application Model (Services)

In practice, this research considers an IoT application as a distributed data flow (DDF), where modules reside on different nodes (devices). Hence, the application is represented as a directed graph represents the data dependency across modules.

The application flow in Figure 2.11 illustrates an example application consists of two interdependent modules, T_u and T_v placed on virtual machines VM_1 and VM_n respectively. These virtual machines were instantiated on device $Node_i$ and $Node_j$ as decided by the placement map created by the optimization algorithm.

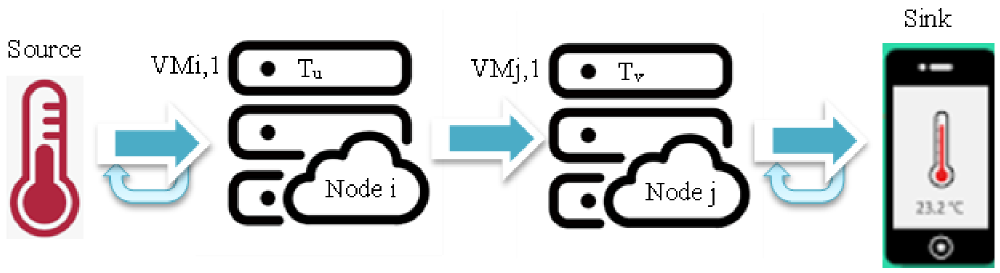

Figure 2.11: An Example Application Flow (loop) with two modules (Services).

The source (normally a sensor) and sink (normally an actuator) belong to client device (like a smart phone or a health monitoring device) that uses that application. The client devices are usually use ad-hoc communication protocols to connect to the network, while network nodes use more persistent methods of communication. In Figure 2.11, the arced arrows represent a periodic application loop that start from a periodic tuple (such as a reading generated every 0.5 second by a temperature sensor) to be processed by the application. the final destination will be an actuator that will take an action depending on the data provided by a tuple send from Tv to the client actuator (ex., an alarm when temperature goes above a threshold).

In this work applications are containers (logical entities) that have the full description of modules requirements and message types that passed through those modules. In addition, an application states the flows (loops) that required to finish a certain task. Figure 2.12 illustrated the logical application modules and messages flow for defining an application “EEG” with one source, one sink, and three modules.

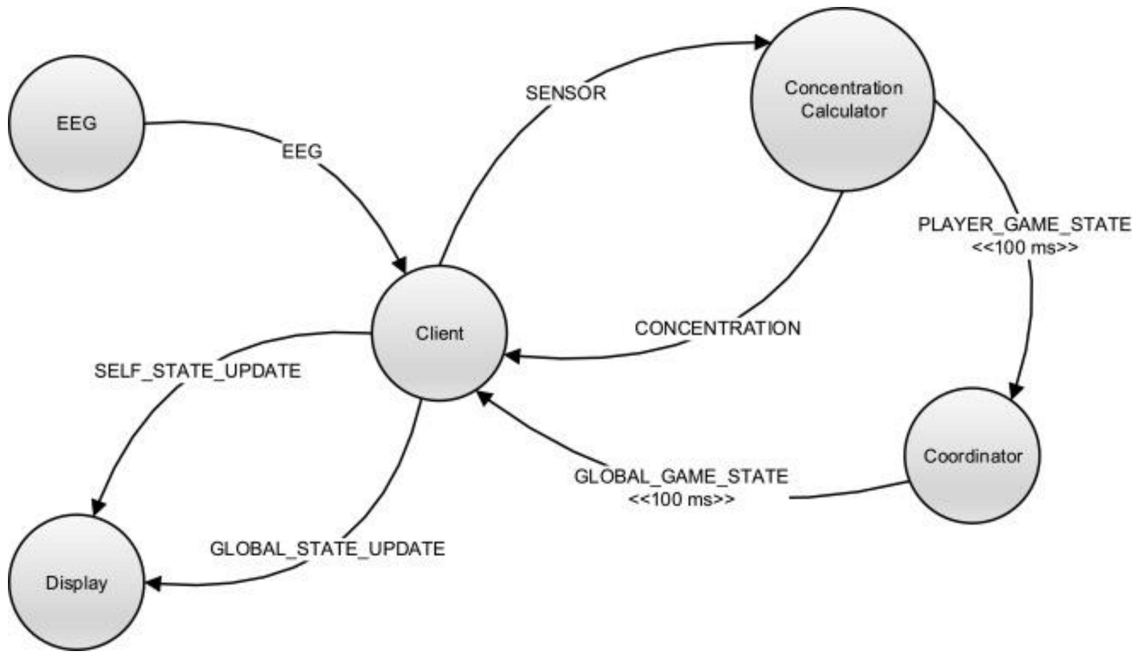

Figure 2.12: Logical flow of modules and messages for the EEG example application.

2.12.2 Time Model for Network Nodes

During the IoT-Fog environment simulation, the simulator provides several timestamps to monitor the performance of the network. These timestamps [161], as illustrated in Figure 2.13 include:

- **Emit Time** (t_{emit}): the time the message sent from the source node.
- **Reception Time** (t_{recep}): the time the message received by the destination node (queued).
- **In Time** (t_{in}): the time the message enters the processing stage.
- **Out Time** (t_{out}): the time the processing stage and transmitted to the next hop in the application loop.

Using these timestamps, the following equations can be derived:

- Waiting Time (t_{wait}): $t_{wait} = t_{in} - t_{recep}$ (2.30.a)
- Service Time (t_{srvc}): $t_{srvc} = t_{out} - t_{in}$ (2.30.b)
- Latency Time (t_{lat}): $t_{lat} = t_{recep} - t_{emit}$ (3.30.c)
- Response Time (t_{resp}): $t_{resp} = t_{out} - t_{recep}$ (2.30.d)
- Total Response Time (t_{total}): $t_{total} = t_{out} - t_{emit}$ (2.30.e)

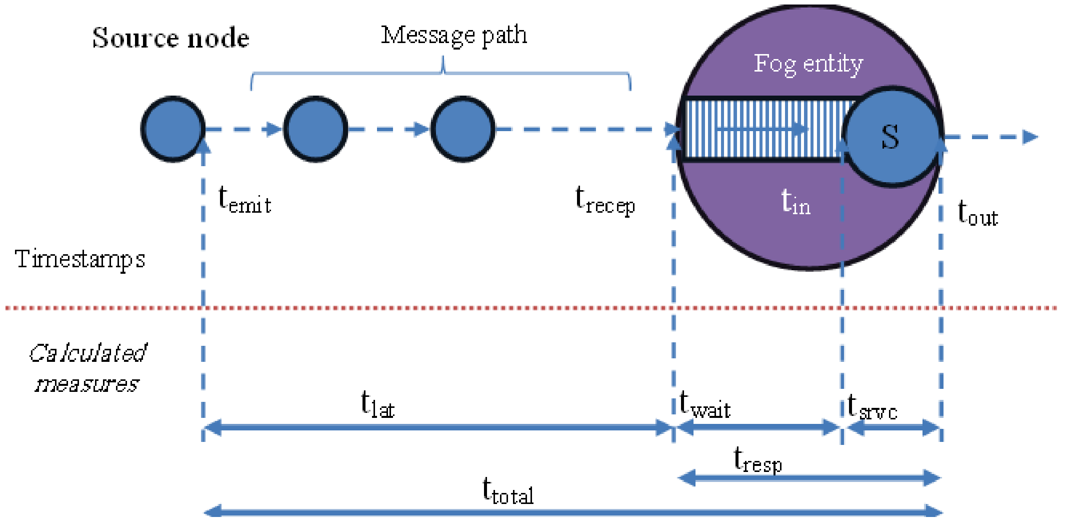

Figure 2.13: Network Node Time Model

2.13 Assumptions

Generally, solving a resource allocation problem in an environment like IoT-Fog is a complex task. To put this problem in the scope of this work, the following assumptions are deployed:

1. The IoT-Fog environment is always connected. Fog computing is an infrastructure network. This work supports static. The assumption here is that the environment elements are always pave a path through the nearest node (fog or cloud) to prevent the probability of an isolated node.
2. The playground of the simulation is a 2D plane, hence nodes are moving in two axes. Furthermore, the simulated area has a full cellular service.
3. Network communication characteristics are presented in terms of transmission latency and network bandwidth.
4. In case of VM migration, the application task will be stopped and migrated to the target node and continue its execution.
5. The resource allocation plan is assumed to start at *t*=0.
6. The IoT applications are executed during the full length of the simulation.
7. In this work, clients would only execute tasks from their IoT application task. Therefore, controllers cannot share and execute tasks from different applications on the client device.

8. The controller executes the optimization algorithm on-demand. When an update interval is set, a time overhead will be introduced to the network performance if the update interval is periodic and small. On the other hand, in case of long update periods, QoS will degrade due to factors like outdated routing paths.

2.14 Problem Formulation and Design

The main goal of the system is to ensure QoS while optimizing the use of the environment resources. These two main goals should be achieved by following the system design objectives: System design variables and Optimization Problem Objectives

2.14.1 System design variables

This section presents the system variables used to define the proposed system. The system design contains the following maps: the placement map (*P*), messages flow map (*R*), and the modules migration map (*V*). The placement map maps each application module (VM) to its hosting node. The message flow represents the communicated messages between any two dependent modules in an application. While the migration map shows the routing map for modules need to be migrated. Table 3.1 represents the variables notion used in this chapter.

The physical topology of the proposed design consists of *N* nodes and *E* edges connecting the nodes. The topology each node $n \in N$ represent a computing element and characterized by its computing power, and the size of their memory and storage, $f_n^{Ipt}, f_n^{Mem}, and\ f_n^{Stor}$ respectively. The power model parameters for nod *n* are characterized by power consumed while node idle, power consumed while node busy, and power consumed by antenna during transmission, $f_n^{iPw}, f_n^{bPw}, and\ f_n^{tx}$ respectively. The node cost is modeled in this work using the fallowing parameters processing cost, memory and storage usage cost, bandwidth usage cost, and the cost of the power consumed, $f_n^{cIpt}, f_n^{cMem}, f_c^{cStor}, f_n^{cBw}, and\ f_n^{cPw}$ respectively.

The communication links are represented by the network graph edges E. Each edge e, where e ∈ E, characterized by link latency, bandwidth, in addition to source and destination nodes that this link represents, $E_e^L, E_e^{Bw}, E_e^S, E_e^D$, respectively.

Additionally, this work considers the IoT applications' tasks perspective. Generally, network clients deploy arbitrary applications to be implemented in the network. These applications vary in the number of modules deployment in the network devices. Principally, there are A applications deployed in the system and each application has M interdependent modules need hosts and processing. A module m demand resources to be executed, the parameters considered here are, processing power, memory and storage size, and deadline time, $M_m^{Ipt}, M_m^{Mem}, M_m^{Stor}, and\ M_m^{MigDl}$ respectively. Basically, for an application $a \in A$ has K application edges, where $K \subseteq E, and\ e_k \in K$. Each application edge e_k demonstrated by its required number of instructions (in Mega instructions) and data size capacity, $e_k^{Mi}, and\ e_k^{Size}$ respectively. In addition, application edges have extended features to define regularity, probability of occurrence, source and target modules, $e_k^{Pe}, e_k^{Pr}, e_k^{S}, and\ e_k^{D}$ respectively. Furthermore, the transmission pattern of the application edges follows either a fractional or burst selectivity with a probability distribution like deterministic, normal or uniform.

Table 2.5: Variables used in the System Modelling

Symbol	Definition
N	Number of environment nodes
A	Number of applications
E	Number of network edges
M	Number of modules
K	Number of modules dependencies
Z	Number of modules pairs with dependencies
Q	Number of application loops
P	Binary matrix [N×M] for modules placement map
R	Binary matrix [Z×E] for tuple routing map
V	Binary matrix [M×E] for module migration map
D	Mapping matrix [N×M] between nodes and possible modules deployment
CP	Current modules placement [NxM]
$\boldsymbol{f^{Fog}}$	A flag to set that a node is in the fog (1) or not (0)
$\boldsymbol{f^{Ipt}}$	Processing power of a node
$\boldsymbol{f^{Mem}}$	Memory size of a node
$\boldsymbol{f^{Stor}}$	Storage size of a node
$\boldsymbol{f^{iPw}}$	Power consumption while node idle
$\boldsymbol{f^{bPw}}$	Power consumption while node busy
$\boldsymbol{f^{tx}}$	Antenna power of a node
$\boldsymbol{C^{Ipt}}$	Price for CPU usage

$\boldsymbol{C^{Mem}}$	Price for memory usage
$\boldsymbol{C^{P}}$	Price of using processing resources (C^{Ipt} and C^{Mem})
$\boldsymbol{C^{Stor}}$	Price for storage usage
$\boldsymbol{C^{Bw}}$	Price of bandwidth usage
$\boldsymbol{C^{Pw}}$	Price of power consumption
$\boldsymbol{E^{S}}$	Network link source node
$\boldsymbol{E^{D}}$	Network link destination node
$\boldsymbol{E^{L}}$	Network link latency
$\boldsymbol{E^{Bw}}$	Network link bandwidth
$\boldsymbol{m^{Ipt}}$	Module demand of processing power
$\boldsymbol{m^{Mem}}$	Module demand of memory
$\boldsymbol{m^{Stor}}$	Module demand of storage
$\boldsymbol{m^{MigDl}}$	Module migration deadline
$\boldsymbol{m^{Bw}}$	Inter-modules bandwidth [M×M]
$\boldsymbol{m^{Mi}}$	Inter-modules program size [M×M]
$\boldsymbol{m^{Size}}$	Inter-modules data size [M×M]
$\boldsymbol{e^{Mi}}$	Tuple program size
$\boldsymbol{e^{Size}}$	Tuple data size
$\boldsymbol{e^{Pe}}$	Link periodicity
$\boldsymbol{e^{Pr}}$	Link selection probability
$\boldsymbol{e^{S}}$	Link source node
$\boldsymbol{e^{D}}$	Link target node
$\boldsymbol{A^{L}}$	Modules list of an application loop [Q×[M×M]
$\boldsymbol{A^{Dl}}$	Application loop deadline
$\boldsymbol{A^{A}}$	Application loop index
$\boldsymbol{A^{Cp}}$	Penalty price for not completing task by its deadline
$\boldsymbol{u^{p}}$	Processing utilization ratio
$\boldsymbol{u^{m}}$	Memory utilization ratio
$\boldsymbol{u^{s}}$	Storage utilization ratio
$\boldsymbol{u^{b}}$	Bandwidth utilization ratio
$\boldsymbol{t^{init}}$	VM setup time

Generally, the information provided by the network links introduce a wealth of knowledge to extract insights on the network communication performance parameters. Basically, these metrics are devoted to an application a loop modules q, where $q \in A_a^{L(q)}$.

For example, Equation 2.31 can be utilized to compute the mean required processing power of each module. This metric helps in assessing the computational load and resource needs of individual modules within the system. Additionally, Equation 2.32 enables the calculation of the bandwidth between modules, providing insights into the communication capabilities and data transfer rates between different components of the system. This information is crucial for optimizing the allocation of resources and ensuring efficient data exchange between modules. In the case of a worst-case scenario, Equation 2.33 allows for the estimation of the quantity of instructions in the queue waiting to be processed by a particular module. This metric helps identify potential bottlenecks or delays in processing, allowing for proactive measures to mitigate performance issues. Furthermore, Equation 2.34 aids in determining the size of data that can be queued for transmission between modules. Understanding the capacity and limitations of the data transmission queue is crucial for managing data flow and preventing congestion or data loss.

$$m_i^{Ipt} = \sum_{k \in K} \frac{e_k^{Pr} e_k^{Mi}}{e_k^{Pe}}, e_k^D = i \tag{2.31}$$

$$m_{i,j}^{Bw} = \sum_{k \in K} \frac{e_k^{Pr} e_k^{Size}}{e_k^{Pe}}, e_k^S = i, e_k^D = j \tag{2.32}$$

$$m_{i,j}^{Mi} = \sum_{k \in K} e_k^{Mi}, e_k^S = i, e_k^D = j \tag{2.33}$$

$$m_{i,j}^{Size} = \sum_{k \in K} e_k^{Size}, e_k^S = i, e_k^D = j \tag{2.34}$$

These metrics enable a comprehensive understanding of the system dynamics and support the optimization of resource allocation to enhance the quality of service in the IoT-Fog environment.

From the environment's controller point of view, two matrices are considered to be the base of knowledge for the placement problem. The first matrix is possible deployment map *D*, where $D_{n,m}$ means that node *n* can host module *m*. This map is important for the controller to find which nodes can host a particular module. The current placement (*CP*) map is used by the controller to find the current allocation of network resources. The allocation rule $CP_{n,m} \in \{0,1\}$ means that module *m* is deployed in node *n* if and only if $CP_{n,m} = 1$. As stated in Section 2.12.1.2, the controller utilizes only part of its resources by ratio of processing power, memory, storage and bandwidth $u^p, u^m, u^s, and\ u^b$ respectively. Besides that, the controller sets a constant time for VM overhead instantiation time t^{init}.

2.14.2 Optimization Problem Objectives

The main aim of this study, as stated in Chapter 1, and Chapter 2, is to enhance the quality-of-service for the proposed system. The maximization of QoS from the client perspective is the provision of a reliable service. On the other hand, this objective can conflict with service provider perspective in terms like power consumption and service cost trade-off. For instance, the controller target to maximize the QoS but at the same time minimize the processing and bandwidth utilization. All related costs parameters to the optimization problem are presented in this Section.

2.14.2.1 QoS Cost Modelling

Generally, the system aims to provide services with sufficient QoS to maximum number of clients. Hence, applications loops should complete their tasks before the deadline time. The system measures the worst-case scenario for all the application loops to ensure that all deadlines are fulfilled. Therefore, the problem can be formulated as a minimization to the number of applications with failed loops. The QoS cost C_{QoS} depends on two delay parameters; the processing delay, L_q^p, and the propagation delay, L_q^t, where q represents an application loop.

Total delay time for an application loop is calculated using the loops delay matrix, A_q^L. Evaluating that an application loop fulfils the deadline constraint should be $L_q^p + L_q^t \leq A_q^D$. The processing time of loop q is the sum of all tuples passing all modules in the loop in the worst-case scenario, see Eq. 2.35. While the propagation delay of loop q is the sum of all propagation delays for tuples transmitted from node to another node in the application loop modules, see Eq. 2.36. The final QoS cost model is presented in Eq. 2.37.

$$L_q^p = \sum_{i\in M}\sum_{j\in M} A_{q,i,j}^L \sum_{n\in N} P_{n,j} \frac{\sum_{i\in M}\sum_{k\in M} m_{i,k}^{Mi} \times P_{n,k}}{\alpha^p \times f_n^{Ipt}} \quad (2.35)$$

$$L_q^t = \sum_{i\in M}\sum_{j\in M} A_{q,i,j}^L \sum_{e\in E} R_{z\prime,e} \left(\frac{\sum_{z\in Z} R_{z,e} \times m_{l_z^S,l_z^D}^{Size}}{\alpha^b \times E_e^{Bw}} + E_e^L\right), l_{z\prime}^S = i, l_{z\prime}^D = j \quad (2.36)$$

$$C_{QoS}(x) = \sum_{a \in A} e_a x_a, e_a = min\left(\sum_{q \in Q} e_q , 1\right), A_q^A = a,$$
$$e_q = \begin{cases} 1, if\ L_q^p + L_q^t > A_q^D \\ 0, otherwise \end{cases}, \forall_q \in [0, Q] \qquad (2.37)$$

2.14.2.2 Energy Consumption Cost Modelling

In this work energy consumption calculated as a linear model taking into account two energy states (i.e., idle and busy). Energy consumption in IoT-Fog environment is a minimization problem. Most part of operational costs goes to energy consumption hence it is important to optimize the resources activities ensuring a trade-off between resource utilization and their power usage. Principally, calculating the energy consumption cost for the system environment C_{pw} involves the power consumed by the compute nodes and the communication power as stated by Eq. 2.38.

$$C_{Pw}(x\} = \left[\sum_{n \in N} f_n^{Fog} \sum_{m \in M} x_n \frac{P_{n,m} m_m^{Ipt}}{f_n^{Ipt} u^p} \left(f^{bPw} - f^{iPw}\right)\right] + \left[\sum_{z \in Z} m_{l_z^S l_z^D}^{Bw} \sum_{e \in E} f_i^{Fog} \left(x_i \frac{R_{z,e}}{E_e^{Bw} u^b}\right) f_i^{tx}\right], i = E_e^S \qquad (2.38)$$

2.14.2.3 Processing Cost Modelling

Calculating the processing cost is importing to measure the utilization of the computing resources to keep their utilization at minimum. The processing cost is considered only fog nodes as any processing on the client-side will not affect the system computing resources. This also applies for the power consumption cost. The calculation for the processing cost is illustrated in Eq. 2.39.

$$C_P = \sum_{n \in N} f_n^{Fog} \sum_{m \in M} x_n \frac{P_{n,m} m_m^{Ipt}}{f_n^{Ipt} u^p} \qquad (2.39)$$

2.14.2.4 Bandwidth Cost Modelling

Another important cost in the system, is the cost of the bandwidth C_{Bw} acquired by models communicating with tuples routing map R. Eq. 2.40 illustrates the method of calculating bandwidth cost.

$$C_{Bw} = \sum_{z\in Z} m^{Bw}_{l^S_z l^D_z} \sum_{e\in E} f_i^{Fog} \left(x_i \frac{R_{z,e}}{E^{Bw}_e u^b}\right), i = E^S_e \tag{2.40}$$

Similar to Eq. 2.38 and 2.39, client nodes do not participate in the calculations of bandwidth cost.

2.14.2.5 Migration Cost Modelling

Modules migration add extra overhead on the system, because this process require bandwidth to complete. Hence, the system will try to use links with low traffic, keeping in mind that the amount of bandwidth that can be used in any link is (1-u^b). The amount a data transferred through traversed links is total size of the VM deploying this module (i.e., memory and storage). Eq. 2.41 shows the module migration cost C_{Mig} calculations.

$$C_{Mig} = \sum_{m\in M}(m^{Mem}_m + m^{Stor}_m) \sum_{e\in E} f_n^{Fog} x_i \frac{V_{m,e}}{E^{Bw}_e(1-u^b)}, i = E^S_e \tag{2.41}$$

Note that the second summation in Equation 3.14 will be considered only if the node is fog compute node and the module has an edge in the migration routing map.

2.14.3 Optimization Problem Constraint

Generally, optimization problems have two parts; the first is the objective functions (cost functions) that defines the functions to be optimized. The second part is the constraints of the cost functions that should be imposed on the variables that control the validation of the whole problem.

2.14.3.1 Mapping Valid Values

In this work there are three maps: placement map, tuples routing map, and modules (VM) migration routing table, P, R, and V respectively. These maps are all binary maps to indicate the fact that the placement is permitted or not. The constraints of valid values for these maps are illustrated in Equations 2.42.a through 2.42.c.

$$P_{i,j} \in \{0,1\}, \forall_i \in [0, N], \forall_j \in [0, M] \quad (2.42.a)$$

$$R_{z,e} \in \{0,1\}, \forall_z \in [0, Z], \forall_e \in [0, E] \quad (2.42.b)$$

$$V_{m,e} \in \{0,1\}, \forall_m \in [0, M], \forall_e \in [0, E] \quad (2.42.c)$$

2.14.3.2 Environment Resources Utilization Constraints

Utilization constraints are used to ensure that the placed module requirements do not exceed the available resources of the hosted node. The resources constraints applied on processing power, memory and storage size, and bandwidth, as shown in Equations 2.43.a through 2.43.d respectively. Additionally, available processing power and available edge bandwidth should be greater than 0 to avoid division by zero, Equations 3.16.e and 3.16.f respectively.

$$P \times m^{Ipt} \leq f^{Ipt} \times u^p \quad (2.43.a)$$

$$P \times m^{Mem} \leq f^{Mem} \times u^m \quad (2.43.b)$$

$$P \times m^{Stor} \leq f^{Stor} \times u^s \quad (2.43.c)$$

$$\sum_{z \in Z} m^{Bw}_{l^S_z l^D_z} \times R_{z,e} \leq E^{Bw}_e \times u^b, \forall_e \in [0, E] \quad (2.43.d)$$

$$f^{Ipt}_n > 0, \forall_n \in [0, N] \quad (2.43.e)$$

$$E^{Bw}_e > 0, \forall_e \in [0, E] \quad (2.43.f)$$

2.14.3.3 Module Deployment Constrains

The possible deployment of an application module is under two roles; first the module should not be permitted to be placed more than once inside a node, Eq. 2.44.a. Second, the module should be accommodated by a legitimate device, Eq. 2.44.b.

$$\sum_{n\in N} P_{n,m} = 1, \forall_m \in [0, M] \tag{2.44.a}$$

$$P \leq D \tag{2.44.b}$$

2.14.3.4 Tuple and Migration Routing Constraints

To check that modules and nodes are placed correctly and registered in the routing tables, the following equations are valid (Equations 2.45.a, and 2.45.b).

$$\sum_{i\in E} R_{z,i} - \sum_{j\in E} R_{z,j} = P_{n,l_z^S} - P_{n,l_z^D}, \forall_z \in [0, z], \forall_n \in [0, N], n = E_i^S, n = E_j^D \tag{2.45.a}$$

$$\sum_{i\in E} V_{m,i} - \sum_{j\in E} V_{m,j} = CP_{n,m} - P_{n,m}, \forall_m \in [0, M], \forall_n \in [0, N], n = E_i^S, n = E_j^D \tag{2.45.b}$$

2.14.3.5 Module Migration Constraint

This constraint checks if the delay taken to migrate validate the preset module migration deadline, Eq. 2.46. It worth notice that migration time is considered only when the algorithm is recomputed, because there would be no migration at first run.

$$L_m^M \leq m_m^{MigDl}, \forall_m \in [0, M] \tag{2.46}$$

2.14.4 The Optimization Problem

From the perspective of the system provider in the IoT-Fog environment, several independent and conflicting objectives arise. Two of these objectives include minimizing power cost, which aims to reduce energy consumption, and minimizing processing cost, which aims to optimize processor utilization. In IoT environments, there exist nodes with low power consumption but limited processing resources. Consequently, the power cost objective seeks to allocate modules to these low-power nodes, while the processing cost objective aims to evenly distribute modules across nodes based on their processing capacities.

Additionally, minimizing processing and bandwidth costs presents another conflict. The processing cost objective strives for a balanced workload distribution among nodes, while the bandwidth cost objective aims to minimize network link usage and achieve a balanced network load. This creates a trade-off between these two costs. Deploying dependent modules (those connected by application edges) within the same node reduces bandwidth usage but increases the node's processing load, and vice versa.

Furthermore, there is a conflicting relationship between QoS and power consumption. These objectives work in opposite directions, as improving QoS typically requires higher power consumption. Consequently, the system provider must find a balance between meeting clients' QoS requirements and minimizing power consumption.

In summary, the system provider faces a multi-objective problem with conflicting objectives, as depicted in Eq. 2.47.

$f_1 = C_{Qos}(x)\ (Eq\ 2.37)$

$f_2 = C_{P_w}(x)\ (Eq. 2.38)$

$f_3 = C_P(x)\ (Eq. 2.39)$

$f_4 = C_B(x)\ (Eq. 2.40)$

$f_5 = C_M(x)\ (Eq. 2.41)$

$$\boldsymbol{Fitness}(\boldsymbol{P},\ \boldsymbol{R},\ \boldsymbol{V}) = Min(f_1, f_2, f_3, f_4, f_5) \tag{2.47}$$

subject to:

Mapping Valid Values (Eq. 2.42)

Environment Resources Utilization (Eq. 2.43)

Module Deployment (Eq. 2.44)

Tuple and Migration Routing (Eq. 2.45)

Module Migration (Eq.2.46)

In this work, the multi-objective optimization problem for resource allocation in the IoT-Fog environment was solved using hybrid bio-inspired evolutionary algorithms. The algorithm considers the optimization of clients' QoS requirement. Also, the minimization of energy consumption taken in to account as an objective for the optimization problem. In addition to adding processing utilization and bandwidth usage requirements to the algorithm objectives. Finally, the optimization of the resources needed for migrating a module is considered as a dynamic objective during the evolution of the environment.

2.15 IoT-Fog Environment Simulation

This work uses the python simulator YAFS to evaluate the proposed algorithm. The YAFS simulator supports mobility of clients' nodes and location-aware communication pattern. This facility enables the simulation of dynamic and mobile environment such as IoT-Fog environment.

2.15.1 Environment Network Configuration

The first step in the simulation is to setup the network definition by preparing JSON files describing the network topology, applications elements, users and initial allocations. The environment configuration elements are illustrated in Figure 2.14.

The network's entities (nodes) and links (edges) properties are not fixed but can be customized according to the simulation environment problem. For example, as this work consider environment variables such as energy consumption and computation prices, the definition of environment entries is customized to include any unimplemented property. The simulator needs these configurations to start the environment evaluation. Furthermore, the environment users are simulated as either sources (sensors) or sinks (actuators) configured in a user definition JSON file. These users are also considered as the environment workload generators.

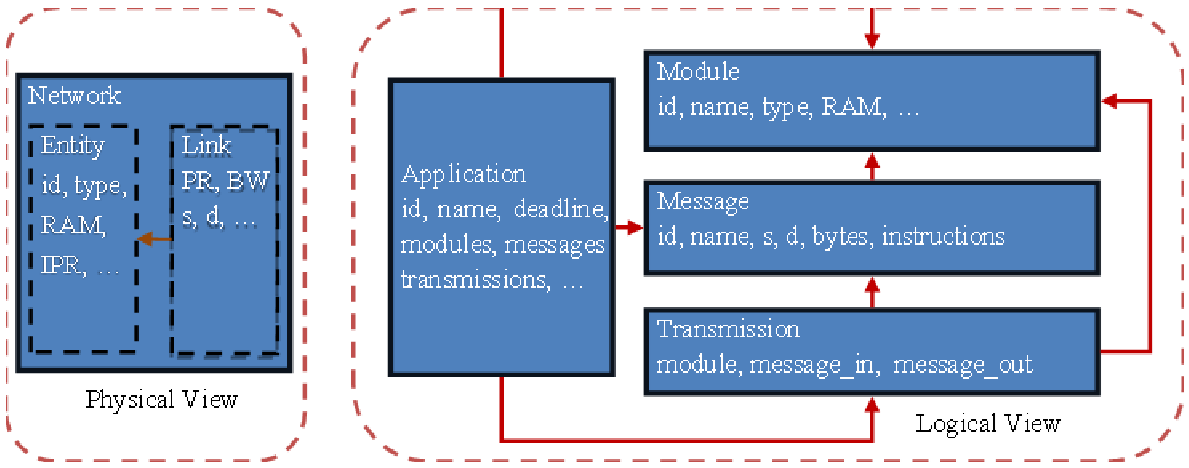

Figure 2.14: Main configurations needed by the simulator.

YAFS simulator supports mobility of clients. To simulate clients' mobility, the simulator should be provided with the mobility plan in the form of coordinates for each mobile client. Many geographic file formats are accepted by the simulator as long as it is supported by the underlying *NetworkX* python library or files can be converted to those formats.

2.15.1.1 Environment Nodes Configuration

The design of environment nodes follows the computational characteristics depicted from the optimization problem formulation stated in Sections 3.6.2 and 3.6.3. Listing 2.1 illustrates the adopted parameters for the node structure.

Listing 2.1: Environment Entity Structure
Entity: {id, type, ipt, ram, stor, ipw, bpw, tx, cipt, cmem, cstor, cpw}

Where '***id***' the node unique id (this can be intrepid as the node IP address inside the physical network),'***type***' the node type like '*cloud'*, '*fog'*, or '*client'*. The entries '***ipt***', '***ram***', '***stor***' represent processing power, memory and storage size respectively. The power consumption mode is represented by the idle and busy power consumption and the transmission power consumption, '***ipw***', '***bpw***', and '***tx***' respectively.

Furthermore, the cost model is parameterized by the cost of processing, memory and storage usage and energy consumed, '***cipt***', '***cmem***', '***cstor***', and '***cpw***' respectively. The Entity structure represents the physical compute device in the simulated environment.

2.15.1.2 Network Links Configuration

The second part of the physical network configuration is the communication link modelling. In this work, network links (i.e., the network graph edges) represent the propagated transmission that encapsulates the transmitted message (tuple).

Network links are characterised (Listing 2.2) by their propagation speed, link bandwidth, source and destination nodes, edge latency, edge periodicity, edge selectivity ration (probability of selection), and cost of transmission, ‘***pr***’, ‘***bw***’, ‘***s***’, ‘***d***’, ‘***el***’, ‘***epe***’, ‘***epr***’, ‘***cbw***’, respectively.

Listing 2.2: Network Link Structure
Link: {pr, bw, s, d, el, epe, epr, cbw}

In the structure above, ‘*pr*’ represent the transmission signal propagation speed which is used in this work to model different types of transmission methods like 3G, 4G, 4G LTE, and other technologies).

2.15.1.3 Application Module Configuration

Modules represents the dependent method in a distributed processing environment and implemented as virtual machines (VM) inside a network node. A module structure is illustrated in Listing 3.3 which comprise of the module unique identifier ‘***id***’ (it can be considered as the application port number as defined by the host node).

Listing 2.3: Application Module Structure
Module: {id, name, type, ipt, ram, stor, migdl}

Other parameters of the Module structure include ‘***name’*** the name of the node to be used when specifying the source or target module of a message. The module type ‘***type’*** parameter shows the functionality of this module, which can be

‘*SOURCE’* means a sensor that can only send message, ‘*SINK’* means an actuator that can only receive messages, and ‘*MODULE’* which represent a computation VM. The parameters ‘***ipt’***,***’mem’***, and ‘***stor’*** represent the required processing power, memory and storage for the module to be allocated when the module executed. The last parameter ‘***migdl’*** represent the migration deadline to be considered when the module needs to be migrated to another node.

2.15.1.4 Application Message Configuration

Principally, messages are the transmitted entities in this work and represent the network traffic unit. A module ‘***s***’ sends a message to the next module ‘***d***’ in its dependency loop logic, as stated in Section 2.12.1.3. Messages are identified by a unique address ‘***id***’ and ‘***name***’, furthermore, they carry data of size ‘***bytes***’ and processing overhead to the target module by ‘***instructions***’. The message data size participates in the calculation of the transmission cost in monetary and power consumption in cooperation with the transition link parameters. The message structure parameters are illustrated in Listing 2.4.

Listing 2.4: Application Message Structure
Message: {id, name, s, d, bytes, instructions}

2.15.1.5 Application Transmission Structure

The Transmission structure represent the pairing between the module and the messages it except and also the messages it produces (injects) to the network traffic. Basically, the Transmission structure represent a state node in an application workflow loop. This structure is formed by the module it represents, the incoming message name, and the produced message name, ‘***module’***, ‘***message_in***’, and ‘***messagge_out***’ respectively, as illustrated in Listing 2.5.

Listing 2.5: Application Transmission Structure
Transmission: {module, message_in, message_out}

2.15.1.6 Environment Application Configuration

Configuring the application structure require stating the modules participating in calculations, the messages transmitted by those modules, and the loops of modules dependencies. Like other structures stated above, an application is identified by its unique identifier ‘***id***’ and ‘***name***’ in addition to the deadline time of its loop’s ‘***deadline***’, Listing 2.6 illustrates the Application structure. The applications’ ‘***id***’ and ‘***name***’ should be unique across the environment. Generally, the environment is represented by physical network components (i.e., Entity and Link) and the logical Application structures. The optimization algorithm task is to place applications’ modules (VMs) in the physical network ensuring efficient performance regarding QoS provided to the clients.

Listing 2.6: Application Structure
Application: {id, name, deadline, modules [Module], messages [Message], transmissions [Transmission]}

The optimization algorithm will generate an allocation map of the available resources represented as an Allocation structure that maps application’s models with the corresponding network entity, as shown in Listing 2.7.

Listing 2.7: Allocation Structure
Allocation: {app, resource, module}

2.15.1.7 Environment Users Configurations

Users are usually configured as sources (generators like sensors) of sinks (consumers like actuators). Hence, this work defines users as sources or sinks (this approach in inherently comply with YAFS definition of users). First, Listing 3.8 define a Source user structure which comprise of the application that user registered (subscribed) with, the message it generates (publishes), the node hosting it, and the mean time period of message generation, ‘***app***’, ‘***message***’, ‘***resource***’, and ‘***lambda***’ respectively. The use of the last parameter ‘***lambda***’ depends on the probability distribution deployed by the user. Listing 3.9 shows the structure of the second type of users; Sink. Messages sinks are usually actuators that receive actuating messages (or the outcome of the application loop) and do not produce any messages. Hence, the structure of Sink user is as follows; the application it registered with, the module that execute that user method and the node that host that module, ‘***app***’, ‘***module***’, and ‘***resource***’ respectively. In many cases the sink user does not has any computational workload on the system.

Listing 2.8: Source User
Source: {app, message, resource, lambda}

Listing 2.9: Sink User
Sink: {app, module, resource}

Figure 2.15 illustrates the structural dependencies of the configurations listed in Section 3.7.1.

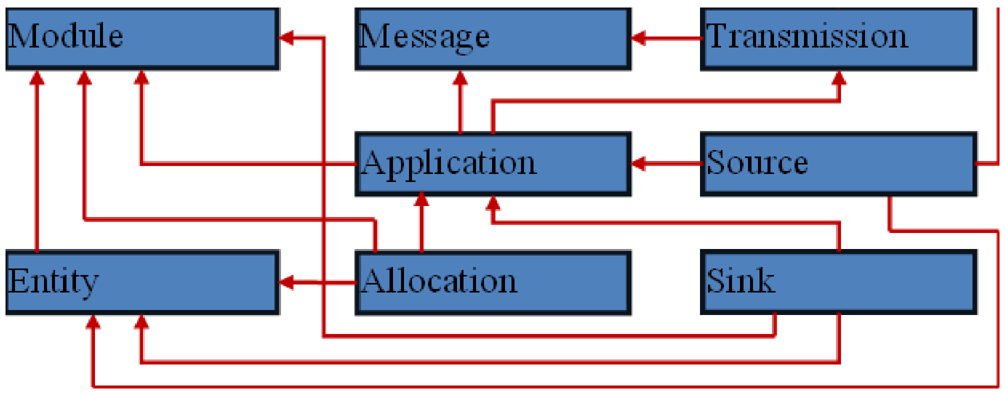

Figure 2.15: Simulation Structures dependencies

2.16 Conclusion

The process of designing the proposed system was detailed in this chapter. The proposed optimization algorithm built by hybridization NSGA-II and MOGWO algorithm and produce a hybrid to test the validity of the proposal. Next, the performance was enhanced by introducing combined migration-based opposition-based learning strategy to produce elite population. In addition, the IoT-Fog environment topology design was detailed to setup the fundamentals base of the simulation experiments. Furthermore, the multi-objective optimization problem was formulated to establish the objective space of the allocation problem for IoT-Fog environment.

CHAPTER Three
The Proposed Algorithm

3.1 Overview

The aim of this thesis is to develop an efficient resource allocation system for fog computing that can preservice the QoS of the IoT-Fog environment. This work can contribute in an open challenge in fog computing i.e., efficient resource allocation, the research can enhance the efficiency and performance of fog computing networks by suggesting better allocations in terms like energy, cost, and processing latency.

This chapter will describe the algorithms and tools used to build the proposed system. First, we describe the optimization problem with its two versions. Next, the build of the deployment IoT-Fog environment was described.

3.2 Proposed System Design

Fog computing is the computing paradigm that brings network devices closer to the user. Hence, applications with low delay tolerance can work more efficiently. On the other hand, the continuously increasing number of IoT devices and the amount of data they generate degrades the network performance. For these reasons, resource allocation algorithms propose the solution to this problem.

The purpose of this thesis is to optimize the resource allocation inside the IoT-Fog environment meanwhile maintaining the quality-of-serves inside the network. Principally, the model structure comprises of two main stages (Figure 3.1). The first stage is the design of the optimization algorithm which is a hybrid method of two well-known algorithms i.e., NSGA-II (Section 2.9) and MOGWO (Section 2.10) as described in Section 3.2.1. The second stage is the IoT-Fog environment network design that the proposed algorithm will be deployed upon as described in Section 3.2.2.

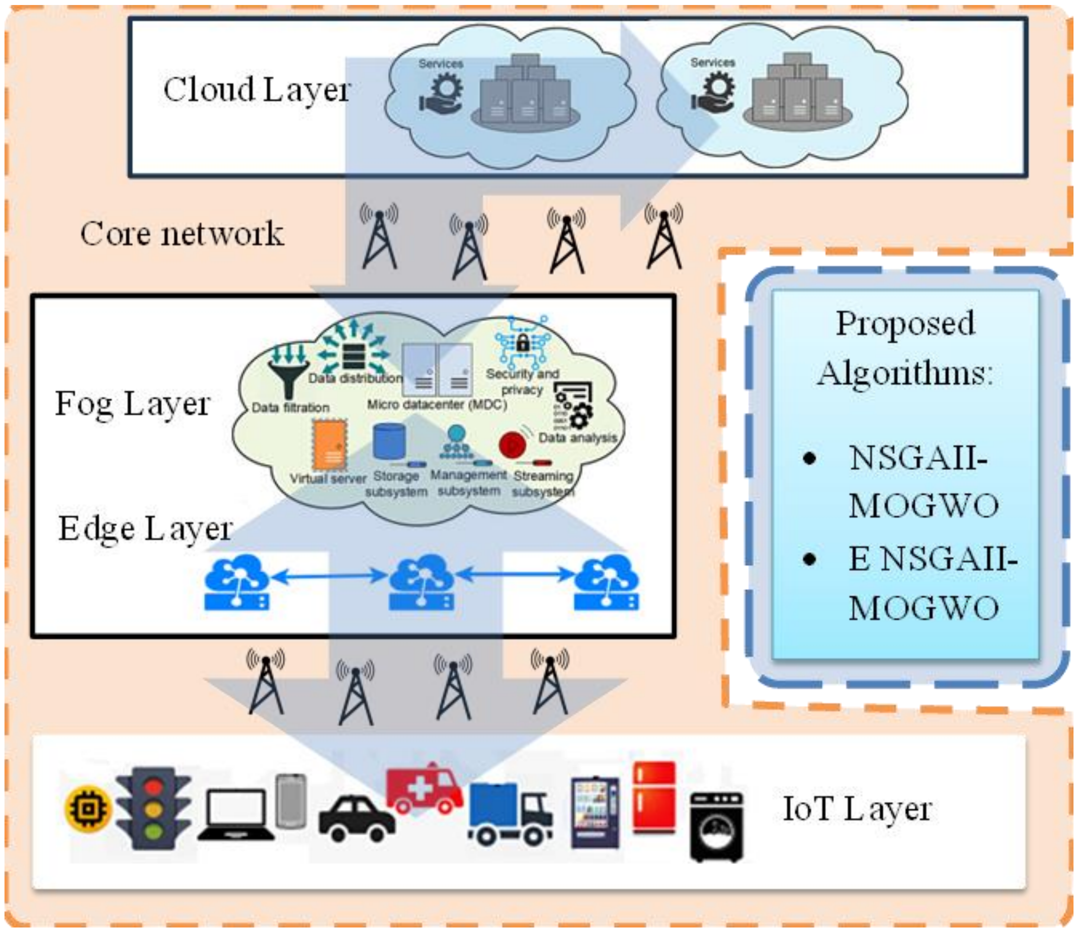

Figure 3.1: Outline of the proposed environment architecture

3.2.1 Proposed Optimization Algorithm Design

The problem of Fog resources allocation is an open issue in the Fog paradigm due to the heterogenic nature of its architecture. Therefore, the main goal of this research is to design an optimization algorithm to provide a near-optimal solution to the placement of the network resources to allocate devices for the IoT applications modules (i.e., VMs). The approach adopted in this research is a hybrid evolutionary algorithm to optimize the process of Fog services provision from and to IoT devices.

In this section, we first introduce a hybrid algorithm (NSGAII-MOGWO), Section 3.2.1.1. Next, the proposed an enhanced elite hybrid method (E NSGAII-MOGWO) presented to solve the problem of IoT-Fog environment resource allocation, Section 3.2.1.2.

3.2.1.1 NSGAII-MOGWO (Hybrid NSGA-II with MOGWO)

Over time, it has become evident that tackling most large-scale global optimization tasks maintain significant challenges. Consequently, researchers have increasingly turned their attention to innovative approaches that integrate algorithmic paradigms from different branches of the field. This blending of algorithms generates what is known as hybrid algorithms. The strategic fusion of diverse algorithms is expected to yield a more flexible and efficient solution approach, particularly suited for addressing large-scale real-world problems. By harnessing the strengths of multiple algorithms, hybridization aims to enhance the problem-solving capabilities and overcome the limitations associated with individual methods [162].

This work proposes a hybrid algorithm that combine two multi-objective well-known algorithms (NSGA-II, and MOGWO). The main skeleton of the algorithm adopted from the NSGA-II algorithm (a fast NSGA approach) and the hybridization algorithm is the MOGWO (a multi-objective version of GWO) to provide diversity to the survival population for the next generation. Algorithm 3.1 illustrate the skeleton for NSGAII-MOGWO.

```
ALGORITHM 3.1: NSGAII-MOGWO Skeleton
INPUT: $N$: population size, $V$ number of variable, $O$ objective function.
OUTPUT: $P$ (the best solution so far).
BEGIN
1. Initialize the population $P$:
  1.1 Generate an initial population of $N$ individuals $P$ with random positions for the search
  agents.
2. Process first population:
  2.1 Calculate the cost values of each individual in the population based on the multiple
  objectives.
  2.2 Calculate the domination state of individuals in $P$
  2.3 Calculate the crowding distance of Pareto fronts in $P$
  2.4 Sort individuals in $P$ according to their crowding distance
3. Transfer individuals in parallel to MOGWO:
  3.1 Select a subset of the best individuals in $P$ from the non-dominated fronts and transfer
  them to MOGWO population $W$.
4. Run MOGWO on the transferred individuals:
  4.1 Initialize $a$, $A$, $C$.
```

```
    4.2 Select $\boldsymbol{W_\alpha}, \boldsymbol{W_\beta}, \boldsymbol{W_\delta}$
    4.3 Calculate $\boldsymbol{X_\alpha}, \boldsymbol{X_\beta}, \boldsymbol{X_\delta}$.
    4.4 WHILE termination condition not met DO
    4.4.1 Update $\boldsymbol{W_\alpha}, \boldsymbol{W_\beta}, \boldsymbol{W_\delta}$
    4.4.2 Update $a$, $A$, $C$.
    4.4.3 Calculate cost for every individual in $W$.
      4.4.4 Update $\boldsymbol{X_\alpha}, \boldsymbol{X_\beta}, \boldsymbol{X_\delta}$.
5. Perform environmental selection:
    5.1 Apply NSGA-II's environmental selection mechanism to select the next generation of
    individuals based on dominance and crowding distance ($Q$).
    5.2 Combine the individuals in parent $P$ and children $Q$ populations of NSGA-II to get $\boldsymbol{P'}$
6. Merge the solutions $\boldsymbol{P'}$ and $W$ to get $R$
7. Perform non-dominated sorting
8. Calculate crowding distance
9. Set P = $\boldsymbol{P'}$
9. IF the termination condition is not satisfied go to step 3
10. Return $P$
**END**
```

In Algorithm 3.1, the entry point of the MOGWO is after grouping Pareto front for solutions and sorting the population individuals by their fitness score in the source NSGA-II. In this work, the size of the wolve pack is controlled by the variable (W_N). Next, the outcome offspring population W from MOGWO is merged with the original population P to form the new population P'. The resulted merged population is then sorted by their crowding distance and dominance. The top N individuals are retained to maintain the size of the population and considered as the population of the next generation. The rest steps are continuing as normal NSGA-II process. The performance of NSGAII-MOGWO algorithm is tested for performance metrics as described in Section 3.2.2. the results of benchmarking tests and simulation performance are presented in Chapter 4. Figure 3.2 illustrate the general flow of the NSGAII-MOGWO algorithm.

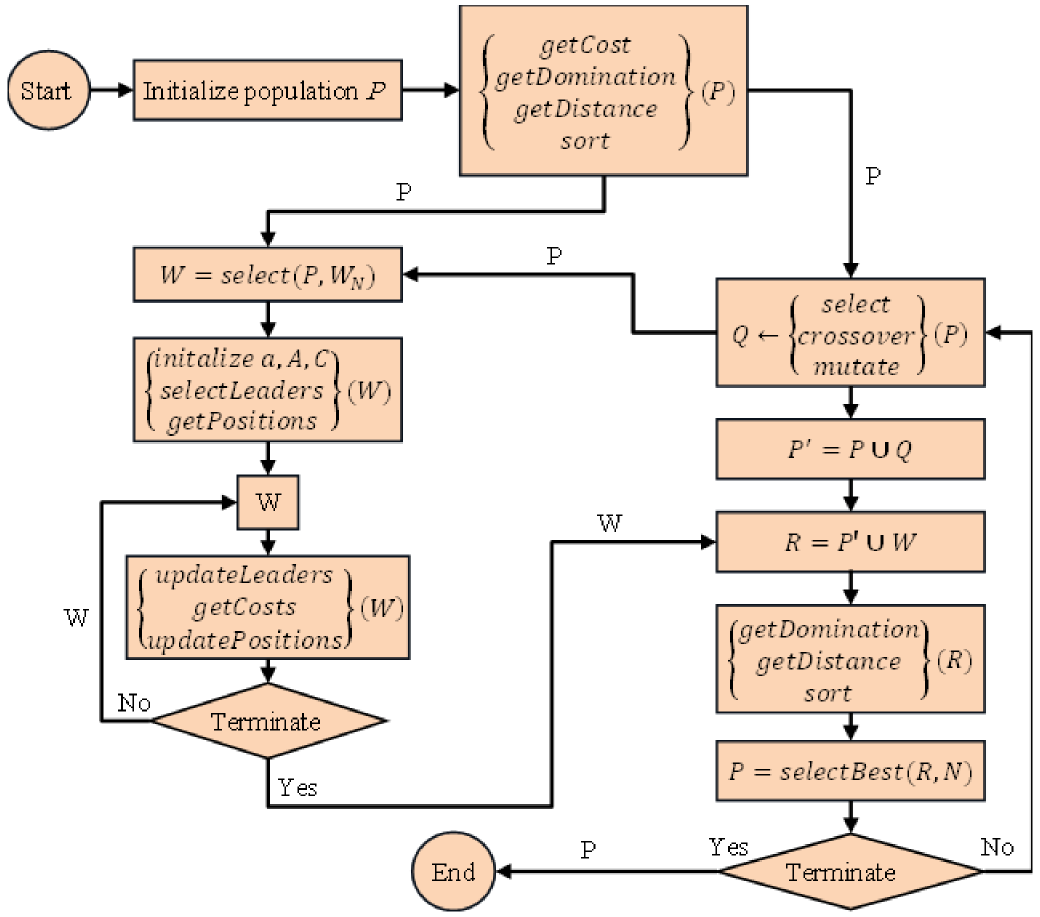
Start
Initialize population P
getCost getDomination getDistance sort (P)
P
P
W = select(P, W_N)
P
select crossover mutate Q ← (P)
initalize a, A, C selectLeaders getPositions (W)
P' = P ∪ Q
W
W
R = P' ∪ W
W
updateLeaders getCosts updatePositions (W)
getDomination getDistance sort (R)
No
Terminate
Yes
P = selectBest(R, N)
End
P
Yes
Terminate
No

Figure 3.2: The general flow of the NSGAII-MOGWO algorithm

3.2.1.2 E NSGAII-MOGWO (NSGAII-MOGWO with Elitism)

One of the contributions of this work is the introduction of a combined elitism method for the MOGWO algorithm. In the base NSGAII-MOGWO algorithm, NSGA-II algorithm considered as the main contributor of the solution space. Meanwhile, the MOGWO is the core for enhancing the quality of the next generation population. From this, the improvement proposed is to introduce adaptive elitism selection to compensate an overcrowded solutions and mitigate stagnation.

The elitism method in this work is an adaptive migration-based elitism with opposition-based learning strategy applied on the migrated (elite) population (both methods and their mathematical models were introduced in Section 2.10). The adaptive immigration-based algorithm can help with the crowding distance by altering the probability of injecting new individuals to the original population. The opposition-based learning strategy generate the elite population be producing sparce individuals from the current solution, hence improving the exploration space of the population. Furthermore, exploitation is maintained by the MOGWO algorithm leaders (α, β, and δ) and also the supplementary ω population.

Algorithm 3.2 details the method of adoptive opposition-based learning strategy proposed in this work. The adoptive random parameter P_{im} in the original work controls the probability of migrating the solution to the next generation. The method uses the traditional genetic methods (selection, crossover, and mutation), this work utilizes the opposition-based learning strategy on the migrated (elite) population. The use of opposition-based is to achieve better objective space saturation which increase the exploration and exploitation for the survival population (offspring). Furthermore, our adoptive method chooses an adoptive number of the worst solutions *im* assigned from a predefined initial ratio r_{el}. The immigration selection ratio is controlled by an external control event of stagnation. This work detects stagnation by the average crowding distance, if the crowding distance decrease, then the immigrated population ratio will increase by a random variable. The probability of executing the stagnation mitigation adaptation process is controlled by another random variable.

```
ALGORITHM 3.2: Combined Elitism
INPUT: P population
OUTPUT: update population P
BEGIN
1. Initialize $P_{im}, r_{el}, C_1, C_2, N$ (number of individuals in P)
2. Calculate cost of all individuals in P
3. IF $rnd$ is grater than $P_{im}$
3.1. Set $im = r_{el} \times N$.
3.2. Select the $im$ worst individuals from the population ($P$) and store them in $X^{im}$.
3.3. For every individual in $X^{im}$
  3.3.1. $k = rnd(0,1)$
  3.3.2. Calculate the opposite position $\tilde{x}_{i,j}^{im}$ using $\tilde{x}_{i,j}^{im} = k \times (a_j^d + b_j^d) - x_{i,j}^{im}$
  3.3.3. IF $\tilde{x}_{i,j}^{im} < a_j$ or $\tilde{x}_{i,j}^{im} > b_j$ then $\tilde{x}_{i,j}^{im} = rnd(a_j, b_j)$
3.4. Calculate cost of $(\tilde{X}^{im})$
3.5. Set $X^{im} = \tilde{X}^{im}$
3.6. IF $avg(f_{im}) > avg(f_{pop})$ then $P_{im} = P_{im} + C_1$
3.7. ELSE, $P_{im} = P_{im} - C_1$
4. Return P
END
```

Principally, the drawbacks of the NSGA-II mentioned in Section 2.8 motivate us to propose the E NSGAII-MOGWO algorithm to combine the strength of NSGA-II in maintaining the Pareto front and MOGWO in enhancing diversity and exploration of the solutions. Primarily, integrating these two algorithms can help to provide a more effective and efficient approach for solving multi-objective optimization problems like resource allocation in IoT-Fog environment.

Algorithm 3.3 outlines the work of the proposed E NSGAII-MOGWO algorithm. In algorithms 3.3, 3.4, and 3.5 the procedure *sort(.)* refers to a group of processes. First the population is sorted using the fast non-dominated sort. Next, finding the crowding distance of the solutions' fronts. Finally, the population is ranked depending on their costs and crowding distance in descendent order.

```
ALGORITHM 3.3: E NSGAII-MOGWO
INPUT: N: population size, V number of variable, O multi-objective functions.
OUTPUT: P (the best solutions so far).
BEGIN
1. Initialization:
  1.1 initialize the algorithm parameters ($P_m$, $r_{el}$, $C_1$, $C_2$)
  1.2 Initialize population P
2. Calculate cost for population P
3. Perform non-dominated sorting and crowding distance for fronts in P
4. Do 4.1 and 4.4 in parallel:
  4.1 Execute Algorithm 3.4 on P and put the result in $\boldsymbol{P'}$
  4.2 Set $W = P$
  4.3 Let E be the set of all dominated individuals in W
  4.4 Execute Algorithm 3.5 on (W, E) and put the result in $\boldsymbol{E'}$
5. Handle algorithms output (i.e., $P', E'$)
  5.1 Combine and sort $P'$ and $E'$ in $P'$
  5.2 Let $P'^{\mu}$ be the average distance of population $P'$
6. Determine potential stagnation:
  6.1 IF $P'^{\mu}$ less than $P^{\mu}$ then $r_{el} = r_{el} + (r_{el} \times C_2)$
  6.2 Set $P^{\mu} = P'^{\mu}$
  6.3 Set $P \leftarrow P'$
7. RETURN P
END
```

In Algorithm 3.3 (the outline of the algorithm is illustrated in Figure 3.3) the process of the E NSGAII-MOGWO uses two branches to generate the candidate solutions. The generated solutions from NSGAII part and E-MOGWO are merged to form a diverse objective space.

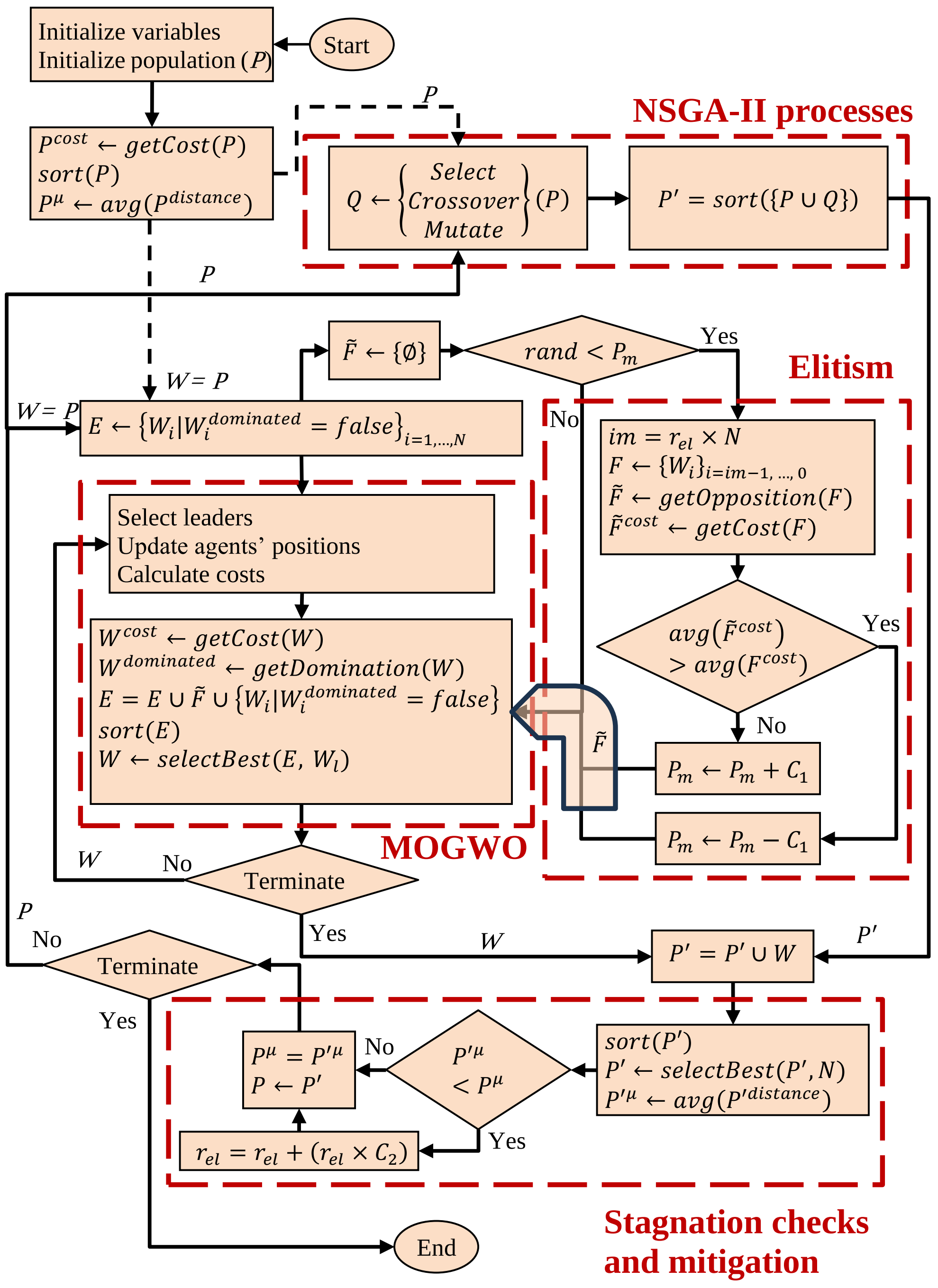
Start
Initialize variables
Initialize population (P)
$P^{cost} \leftarrow getCost(P)$
$sort(P)$
$P^{\mu} \leftarrow avg(P^{distance})$
P
NSGA-II processes
$Q \leftarrow \begin{Bmatrix} Select \\ Crossover \\ Mutate \end{Bmatrix}(P)$
$P' = sort(\{P \cup Q\})$
P
$\tilde{F} \leftarrow \{\emptyset\}$
$rand < P_m$
Yes
No
Elitism
$W = P$
$W = P$
$E \leftarrow \{W_i | W_i^{dominated} = false\}_{i=1,\dots,N}$
$im = r_{el} \times N$
$F \leftarrow \{W_i\}_{i=im-1,\dots,0}$
$\tilde{F} \leftarrow getOpposition(F)$
$\tilde{F}^{cost} \leftarrow getCost(F)$
Select leaders
Update agents' positions
Calculate costs
$avg(\tilde{F}^{cost}) > avg(F^{cost})$
Yes
No
$W^{cost} \leftarrow getCost(W)$
$W^{dominated} \leftarrow getDomination(W)$
$E = E \cup \tilde{F} \cup \{W_i | W_i^{dominated} = false\}$
$sort(E)$
$W \leftarrow selectBest(E, W_l)$
$\tilde{F}$
$P_m \leftarrow P_m + C_1$
$P_m \leftarrow P_m - C_1$
MOGWO
W
No
Terminate
Yes
P
No
Terminate
Yes
W
$P' = P' \cup W$
P'
$P^{\mu} = P'^{\mu}$
$P \leftarrow P'$
No
$P'^{\mu} < P^{\mu}$
$sort(P')$
$P' \leftarrow selectBest(P', N)$
$P'^{\mu} \leftarrow avg(P'^{distance})$
Yes
$r_{el} = r_{el} + (r_{el} \times C_2)$
End
Stagnation checks and mitigation

Figure 3.3: Outline of the E NSGAII-MOGWO Algorithm

The next generation is formulated by executing a fast non-dominated sort, first. Then the crowding distance is calculated and the population is ranked. Finally, the algorithm selects the best N individuals from the population.

The NSGAII branch (illustrated in Algorithm 3.4) takes the first generation as parent to the successive main loop. At the same time, the child population serve as input to the MOGWO and elitism branch (illustrated Algorithm 3.5).

When step 4.1 is finished, two populations are available P', and E' from NSGAII and MOGWO respectively. These two populations are merged in step 4.2 to form the candidate population (P') for the next iteration. The resulted generation is sorted and the domination is calculated to calculate the crowding distance. Next, to preserve the size of the original population, a selection of N best solutions is assigned for the next iteration.

To aid in mitigating solutions stagnation the average crowding distance of P' is calculated (P'^{μ}) and compared with the average crowding distance of the previous iteration (P^{μ}). If $P'^{\mu} < P^{\mu}$ then the immigration ratio r_{el} is updated using the control variable C_2 as in Step 4.3.3.1 of Algorithm 3.3. The algorithm continue execution until the termination condition is met. Finally, the algorithm will return the best solution fund in P.

In Algorithm 3.4, the generated initial population P is used to form the child population Q through the conventional genetic algorithm operations (i.e., selection, crossover, and mutation).

```
**ALGORITHM 3.4**: NSGAII Branch
**INPUT**: P from Algorithm 3.3 step 3
**OUTPUT**: P to be used by Algorithm 3.3 step 5
BEGIN
1. Generate child population Q from selection, crossover, and mutation of population P.
2. Merge parent and children population
    2.1 $R \leftarrow P \cup Q$
3. Determine best individuals found in this iteration
    3.1 Calculate cost and sort population R
    3.2 Select best solutions in R and put them in P
4. RETURN P
**END**
```

The parent and child populations are merged into a new population *R*. A fast non-dominated sort in executed on the merged population (*R*), then the population ranked by their domination and crowding-distance. Next, the *N* best individuals are selected from *R* and assigned to P' and fed to the main loop of Algorithm 3.3. The flow of Algorithm 3.4 is illustrated in Figure 3.4.

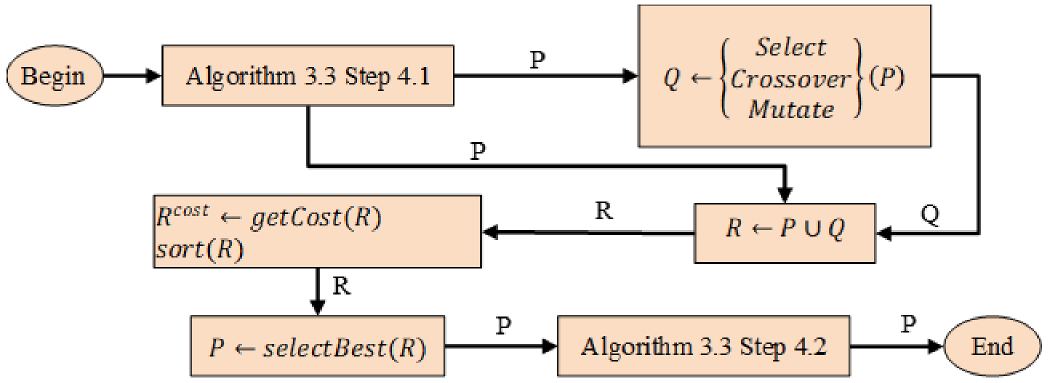
Begin
Algorithm 3.3 Step 4.1
P
$Q \leftarrow \begin{Bmatrix} Select \\ Crossover \\ Mutate \end{Bmatrix}(P)$
P
$R^{cost} \leftarrow getCost(R)$
$sort(R)$
R
$R \leftarrow P \cup Q$
Q
R
$P \leftarrow selectBest(R)$
P
Algorithm 3.3 Step 4.2
P
End

Figure 3.4: The NSGAII branch of E NSGAII-MOGWO Algorithm

The second branch in Algorithm 3.3 is the MOGWO and the combined elitism (Algorithm 3.5). In this branch, population *P* is taken to for two populations, first the wolfs population *W*. The second population is the elite population *E* which is used to keep the best solutions.

The wolfs population enters the main loop of the MOGWO algorithm to optimize the population's solutions and get the best solutions found in this iteration. At the same time the *W* population enters the combined elitism algorithm to construct an elite population through an adoptive approach. The elite mechanism selects *im* worst individuals from *W* using the r_{el} control variable. The elitism branch is considered according to a random adoptive random variable P_m. The value of P_m is controlled using Eq. 2.19. After selecting the population to elite *F*, the method uses the opposition-based learning to transform *F*. The logic of the opposition-based learning in the context of this study, is simulating the question "If this placement is not a candidate, does the opposite placement produce a better solution?"

In Algorithm 3.5, the generated solutions from both branches (i.e., *W*, *F*) are combined along with the non-dominated solutions from *W* to generate the population *E*. This population is sorted with the fast non-dominated sot procedure and only the non-dominated individuals are selected and included in the final population of this algorithm *E*. The workflow of this process is illustrated in Algorithm 3.5 and shown in Figure 3.5.

```
ALGORITHM 3.5: MOGWO and Elitism
INPUT: wItr number of generations, (W, and E) populations from Algorithm 3.3 step 4.1
OUTPUT: E population to Algorithm 3.3 step 4.2
BEGIN
1. Initialize itr=0
2. $W_l$ number of individuals in W
3. Do steps 4 and 5 in parallel
4. MOGWO branch:
  4.1. $a \leftarrow 2 - itr \times \left(\frac{2}{wItr}\right)$
  4.2. For every individual w in W
    4.2.1. Select leaders α, β, γ from W
    4.2.2. Update A, C, D, X using Eq. 2.9 to 2.16 respectively
    4.2.3. $W_w \leftarrow \frac{X_1+X_2+X_3}{3}$ (Eq. 2.17)
  4.3. Calculate cost of population W
  4.4. Determine domination of population W
5. Combined elitism branch:
  5.1 Set $\tilde{F} = \{\emptyset\}$
  5.2 IF $rand$ less than $P_m$
    5.2.1. $im = r_{el} \times N$
    5.2.2. F the top im individuals W
    5.2.3. Calculate the opposition population of F and put it in $\tilde{F}$
    5.2.4. Calculate cost of $\tilde{F}$
    5.2.5. IF $avg(\tilde{F}^{cost})$ greater than $avg(F^{cost})$ then $P_m = P_m + C_1$
    5.2.6. ELSE, $P_m = P_m - C_1$
6. Combine E with $\tilde{F}$ and the non-dominated individuals in W
7. Let W be the best $W_l$ individuals in W
8. If termination condition not met GOTO step 2
9. RETURN W
END
```

A comparable evaluation for the proposed algorithm is presented in Chapter 4 using ten IEEE-CEC benchmark functions. In addition, the proposed algorithm was implemented in simulation scenarios and compared with other algorithm to prove its superiority.

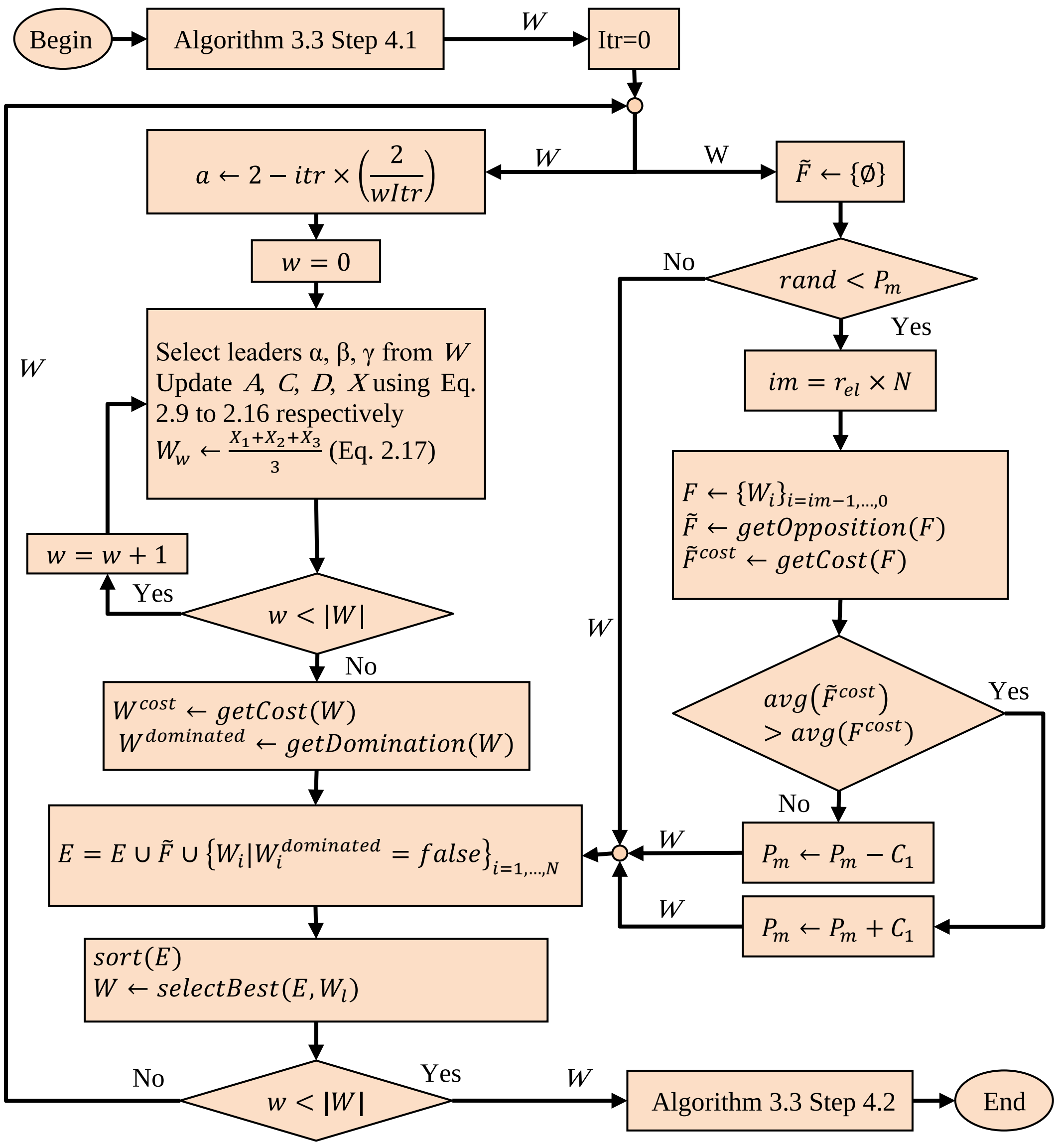
Begin
Algorithm 3.3 Step 4.1
W
Itr=0
$a \leftarrow 2 - itr \times \left(\frac{2}{wItr}\right)$
W
W
$\tilde{F} \leftarrow \{\emptyset\}$
$w = 0$
No
$rand < P_m$
Yes
Select leaders α, β, γ from W
Update A, C, D, X using Eq. 2.9 to 2.16 respectively
$W_w \leftarrow \frac{X_1+X_2+X_3}{3}$ (Eq. 2.17)
W
$im = r_{el} \times N$
$F \leftarrow \{W_i\}_{i=im-1,\ldots,0}$
$\tilde{F} \leftarrow getOpposition(F)$
$\tilde{F}^{cost} \leftarrow getCost(F)$
$w = w + 1$
Yes
$w < |W|$
W
No
$avg(\tilde{F}^{cost}) > avg(F^{cost})$
Yes
$W^{cost} \leftarrow getCost(W)$
$W^{dominated} \leftarrow getDomination(W)$
No
$E = E \cup \tilde{F} \cup \{W_i | W_i^{dominated} = false\}_{i=1,\ldots,N}$
W
$P_m \leftarrow P_m - C_1$
W
$P_m \leftarrow P_m + C_1$
$sort(E)$
$W \leftarrow selectBest(E, W_l)$
No
Yes
W
$w < |W|$
Algorithm 3.3 Step 4.2
End

Figure 3.5: The MOGWO + Elitism branch of E NSGAII-MOGWO Algorithm

3.3 Experiments design

The YAFS [161] simulator contains a number of components to design and execute simulations related to the Fog computing paradigm. In Section 3.3, the main designing components illustrated were the entities and links between entities. In this work, all devices and nodes in all simulation layers are based on the YAFS node structure. Furthermore, all communication links are represented as edges. This analogy came from the architectural design of the simulator which maps the environment as a directed-acyclic graph (DAG) as $G(N, E)$, where N represents the network *nodes* (devices), and E represents the network *edges* (communication links). The nodes and links can be attached with attributes representing parameters like processing power, ram and storage size, link bandwidth, power consumption and so on. As stated in Section 3.2.3.4, the environment controller is the entity that orchestrate and monitor the network performance.

The basic components of the simulation are already explained in Section 3.7.1. The experiments also need to define other functional components that define the environment's population distribution, path selector, and placement algorithm. The experiment setup steps are explained in the next section.

3.3.1 Experiments Setup

The setup procedure is conducted using the following general steps:

Step 1: *Load environment topology configurations*:

To explain this step, Figure 3.6 shows a general example of an IoT-Fog application. In Figure 3.6.a, the application flow represented as modules and their dependences and messages. While Figure 3.6.b illustrates the physical placement of these modules as depicted on the real network layout.

Step 2: *Create the topology graph*:

The topology graph is generated either from the '*networkDefinition.json*' file or from calling the API Application class methods.

Step 3: *Create the application definition*:

The application is the collection of distributed modules (tasks) that that should be allocated and the messages (tuples) that transmitted throughout the environment. Additionally, the application defines the dependencies between its modules (the arrows in Figure 3.6.b).

Step 4: *Define the placement algorithm*:

The YAFS simulator provide predefined placement methods, including placement on network clusters, on edge devices, using custom JSON configuration file, and even no device placement policies. The placement method in this work is an initial placement from custom JSON file, and the proposed algorithm used when there is an update in the simulated environment. Furthermore, YAFS has a "Cloud

Only" policy where all applications are deployed in the cloud only. This policy is useful especially for performance comparisons.

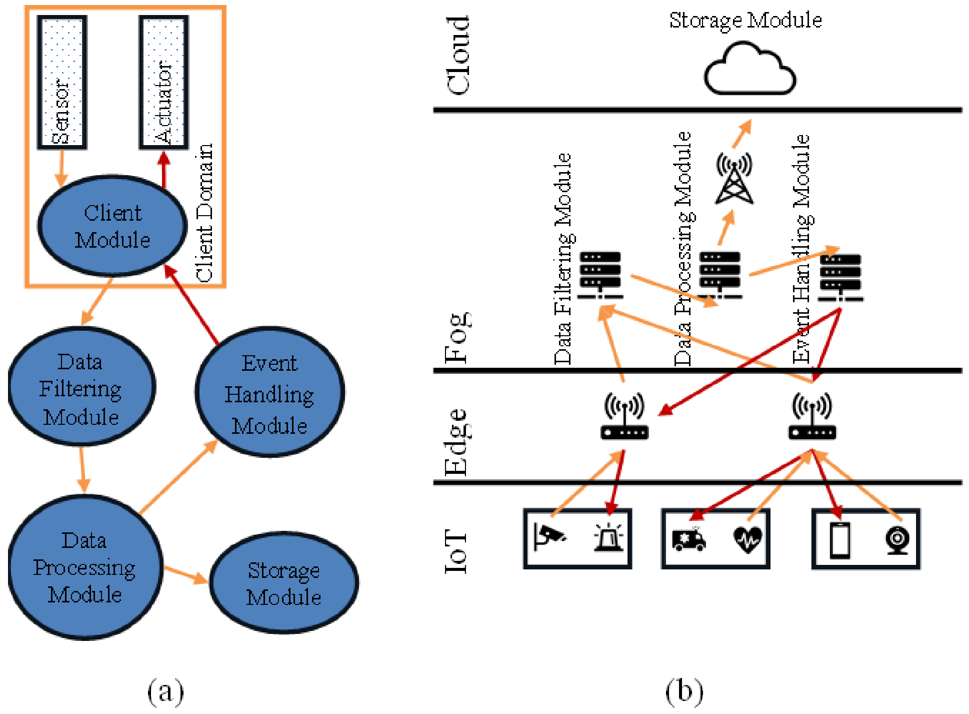

Figure 3.12: Example of application flow (a), and a possible physical deployment (b)

Step 5: *Define the messages routing method*:

The method implemented in this work is client speed-aware routing algorithm. This algorithm is adopted by the simulator to support clients' mobility and handle client handover between environment nodes.

Step 6: *Define the clients population distribution and mobility*:

In this step, the environment clients are defined in terms of number, and distribution probability. The environment clients are considered the main source of network traffic (generators). The clients' mobility is defined by their trajectories which defined in points geo-coordinates (latitude, longitude, elevation). The elevation property, in this work, is always set to xero because the playground of the simulations is 2D plane.

Step 7: *Run the experiment*:

Finally, after setting the environment components, set the number of iterations in each simulation to obtain the average performance of each run. The number of simulation iterations is (30), and the results are stored in the results folder for getting the performance parameters.

Step 8: *Analyze results and draw conclusions*:

After running all the experiments, we analyze the obtained results to examine the performance of the implemented algorithms, as presented in Chapter 4.

CHAPTER Four
Results and Discussions

4.1 Introduction

The proposed optimization system designed in Chapter 3 need to be validated in both stages, first the optimization algorithm validation and then the outcomes from the algorithm simulation in the IoT-Fog network. As mentioned in the previous chapter the designed optimization algorithms are tested with ten multi-objective benchmarking functions. Additionally, the results were validated using five efficiency indicators.

The simulations conducted with the YAFS fog simulator follows the configurations presented in Table 3.3. the analysis of the results from both stages are presented in this chapter.

4.2 Results of Proposed Algorithm Validation

In this section we will present the significant results which acquired by this work. As mentioned in chapter 3, the proposed algorithm was tested on 10 benchmarking functions. Furthermore, the proposed algorithms were validated with performance indicators. The test functions (section 2.12.1) were used to decide and select the algorithm with better performance.

The tests investigate the leader selection method with best performance. The selection criteria metrics considered were GD, GD^+, IGD, IGD^+, and.

The next sections will illustrate the results obtained from the benchmark functions. The related algorithms of this work were NSGA-II, MOGWO, NSGAII-MOGWO, and E NSGAII-MOGWO. The candidate selection methods were Roulette Wheel, Tournament, Stochastic Universal Sampling, Boltzmann, Ranking, Linear Ranking.

For all the presented graphs, the red points in the figures represent the optimal pareto front from the corresponding function. While, the blue points are the solution points obtained from applying the test function on the selection method.

4.2.1 NSGA-II Results

Figures 4.1 to 4.6 shows the results from benchmarking the NSGA-II algorithm using the six candidate selection methods.

The results show that this algorithm proved to converge to the Pareto front for all functions on all selection methods with different efficiency. The most notable behaviour is for functions UF8-UF10 which are 3-objecctive problems, the NSGA-II algorithm did not explore the majority of the objective space. This behaviour is noticed by the crowded cost points in the three graphs.

Figure 4.1 shows the results of the pareto front approximation for the NSGA-II algorithm with the Roulette-Wheel selection method on the benchmarking functions. The performance here indicates an overall good representation for the bi-objective functions. The tri-objective functions show crowding in the suggested solutions due to the local minima problem in NSGA-II

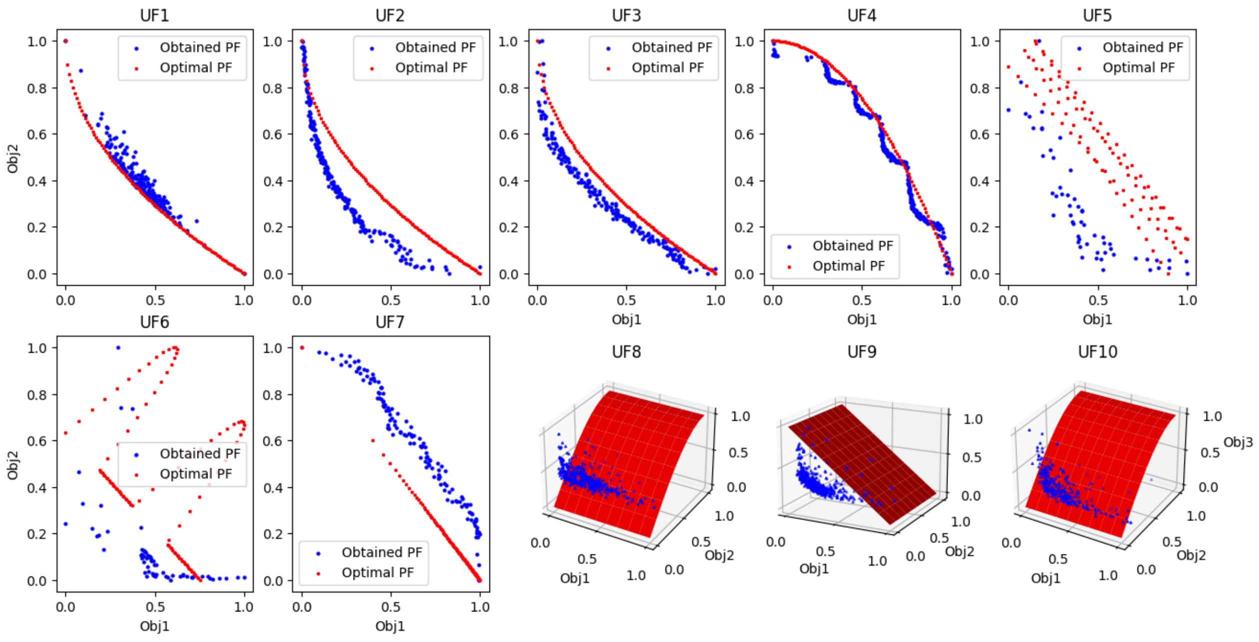

Figure 4.1: NSGA-II with Roulette-Wheel Selection Method

Figure 4.2 shows the results for the Tournament selection method with the ten-benchmarking function. The results show a relatively better performance than Roulette-Wheel except for the tri-objective function with the local-minima problem still exists.

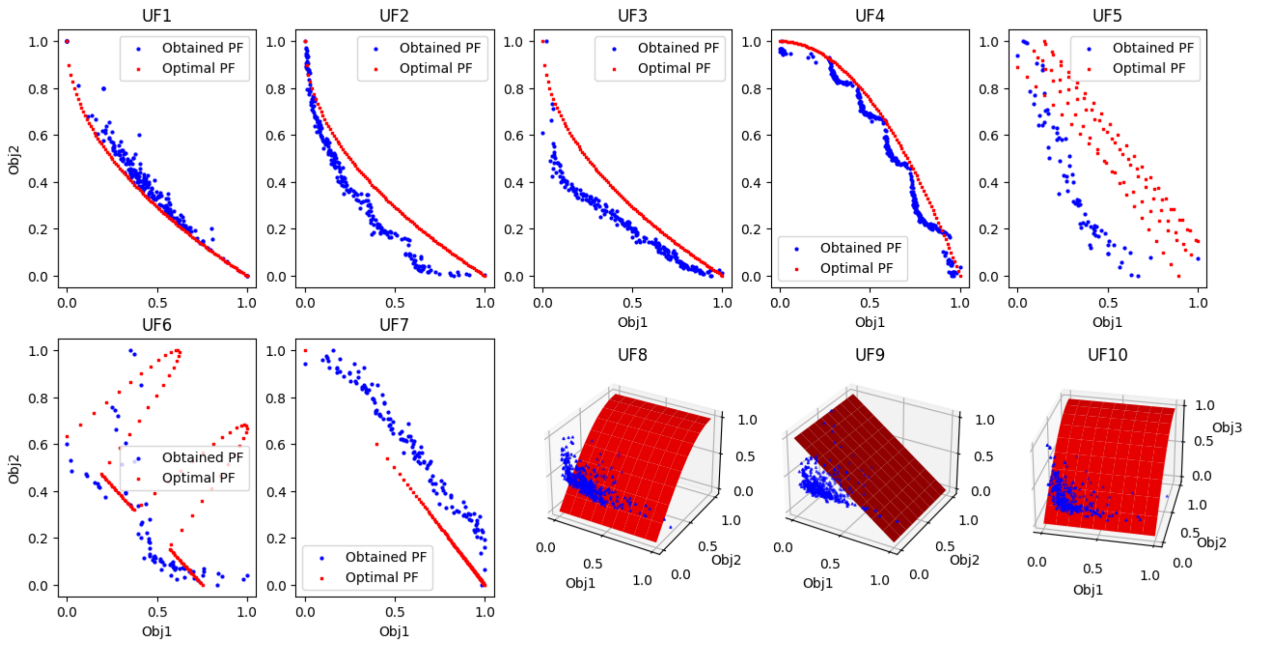

Figure 4.2: NSGA-II with Tournament Selection method

Figure 4.3 shows the results for the Stochastic Universal Sampling selection method on the benchmarking functions. The results illustrate better performance than the Tournament selection method in functions UF3 and UF4, but the problem of local minima is shown for tri-objective functions.

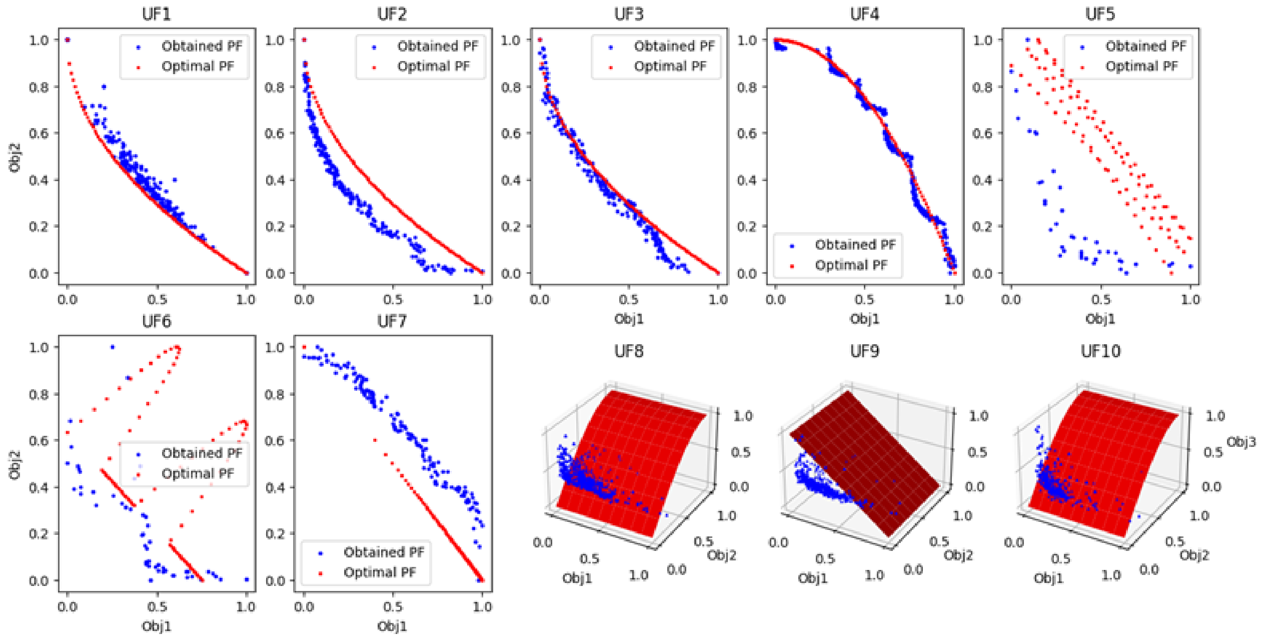

Figure 4.3: NSGA-II with Stochastic Universal Sampling Selection Method

Figure 4.4 shows the results of the Boltzmann selection method with the benchmarking functions. The most notable phenomena in the method are the poor representation for the UF7 bi-objective function and the UF8 tri-objective function. the overall performance here shows a moderated performance comparing with the other considered selectionmethods.

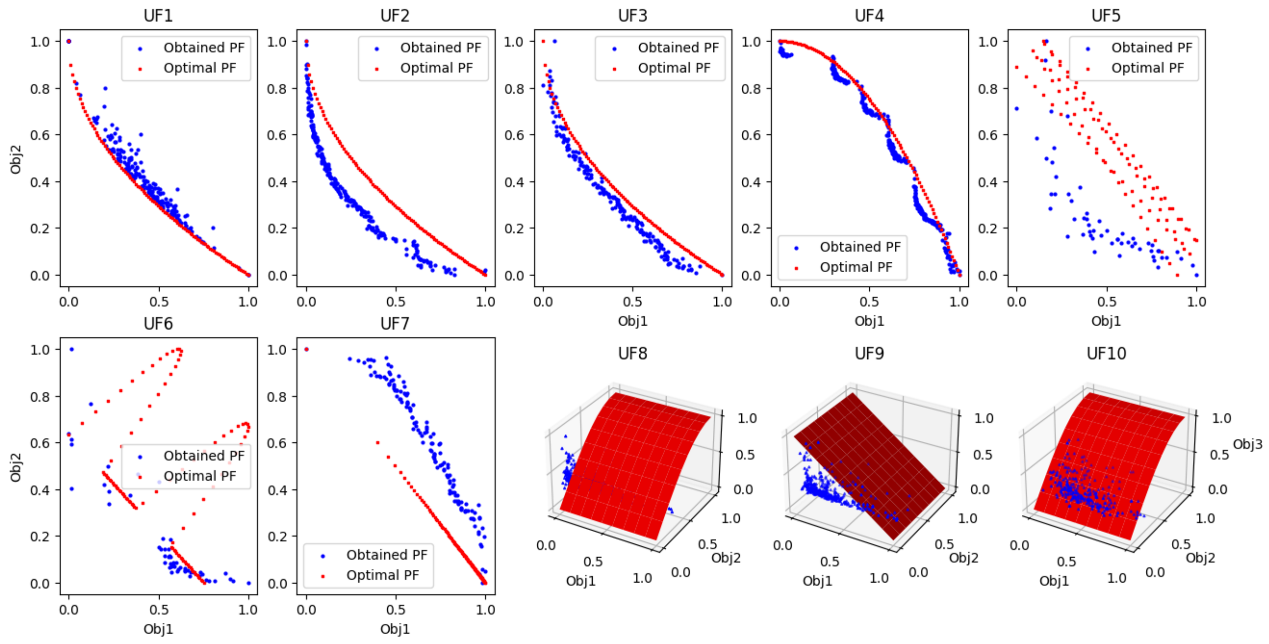

Figure 4.4: NSGA-II with Boltzmann Selection Method

Figure 4.5 shows the results for the Ranking selection method against the benchmarking functions. The results of the bi-objective functions show an average performance comparing with methods like Roulette-Wheel, Tournament, Stochastic Universal Sampling, and Boltzmann method. And better performance than Boltzmann concerning UF8 approximation.

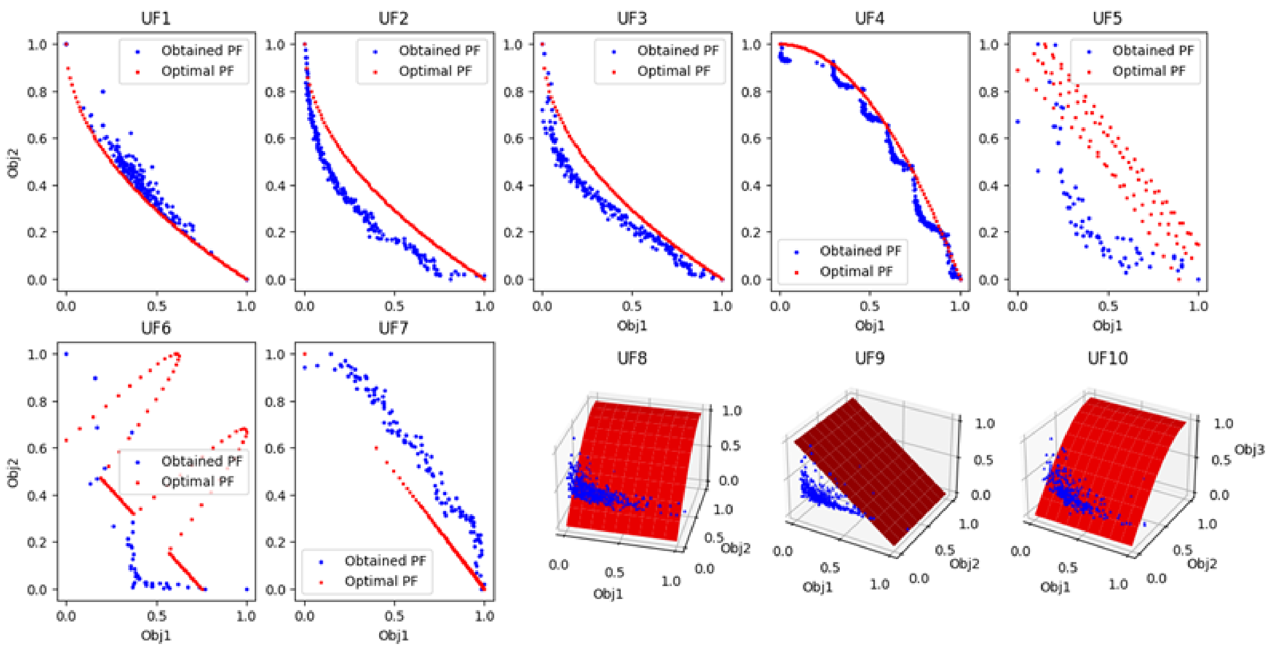

Figure 4.5: NSGA-II with Ranking Selection Method

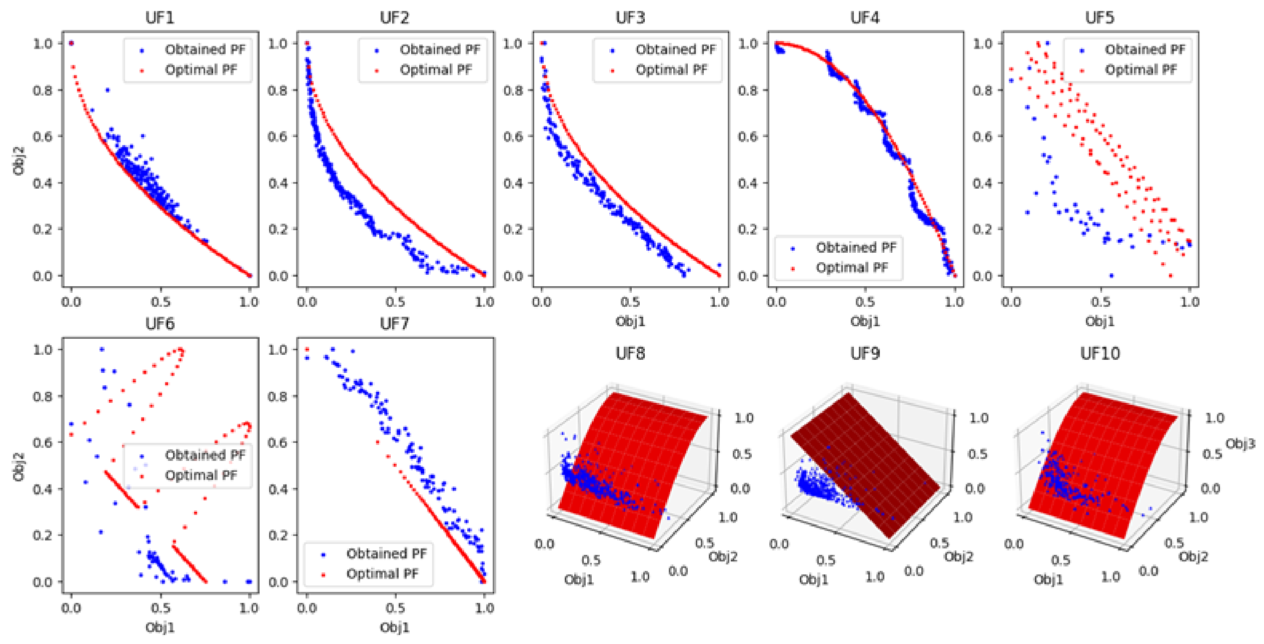

Figure 4.6: NSGA-II with Linear Ranking Selection Method

Furthermore, the figures from bi-objective functions (UF1-UF7) shows the tendency of NSGA-II to preserve the Pareto front behaviour, which is one of the characteristics of this algorithm.

In summary, this algorithm with the generations number given have adequate time to converge in bi-objective functions, but that time was not enough for the method to overcome local-minima (stagnation) for tri-objective functions.

4.2.2 MOGWO Results

The results obtained from the MOGWO algorithm are illustrated in Figures 4.7 to 4.12. The figures show the performance of this algorithm when using different methods of selection.

Figure 4.7 illustrate the results of applying the test function on the roulette wheel selection method. The behaviour of MOGWO shows good representation of the Pareto front for both 2- and 3-objective functions. The most notable behaviour is the perfect approximation of UF7

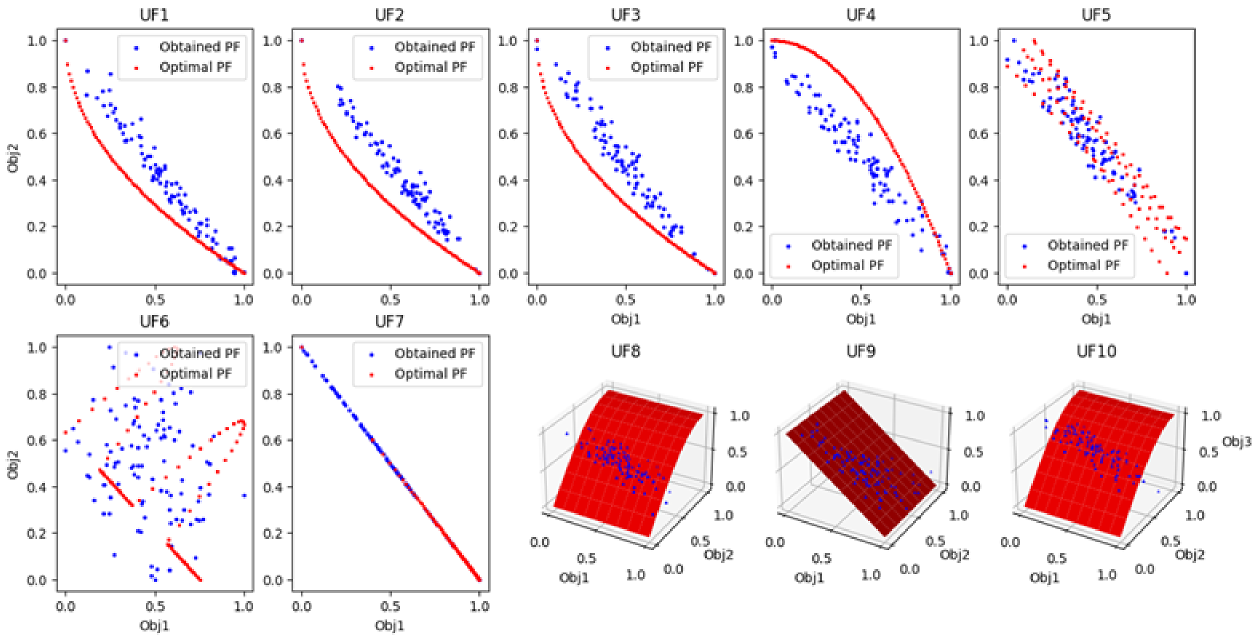

Figure 4.7: MOGWO with Roulette-Wheel Selection Method

In Figure 4.8 an illustration that shows the resulted performance of the MOGWO algorithm of the test function with the tournament selection method. The figure shows almost the same performance obtained. From the roulette wheel method except for UF9 which has. A poor representation of the function’s surface

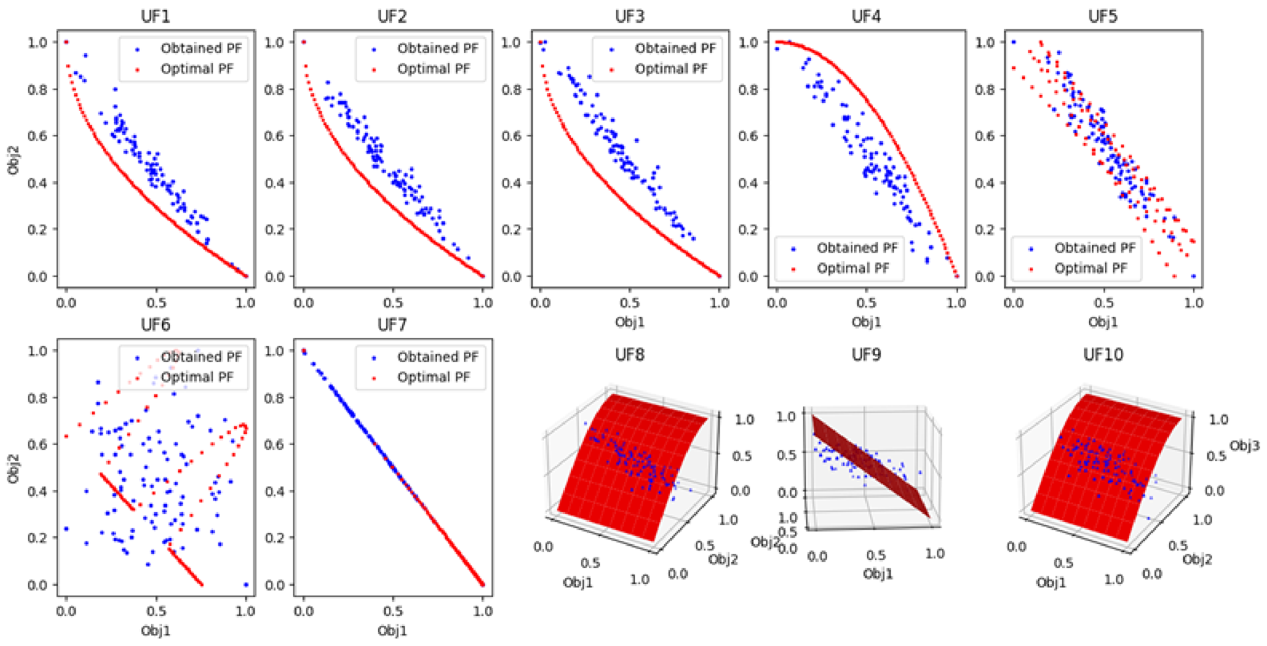

Figure 4.8: MOGWO with Tournament Selection Method

Figure 4.9 illustrates the behaviour of MOGWO algorithm for the test functions with the stochastic universal sampling method. The results show another good representation of the Pareto fronts with exception of UF5 with solutions scattered under the optimal Pareto fronts.

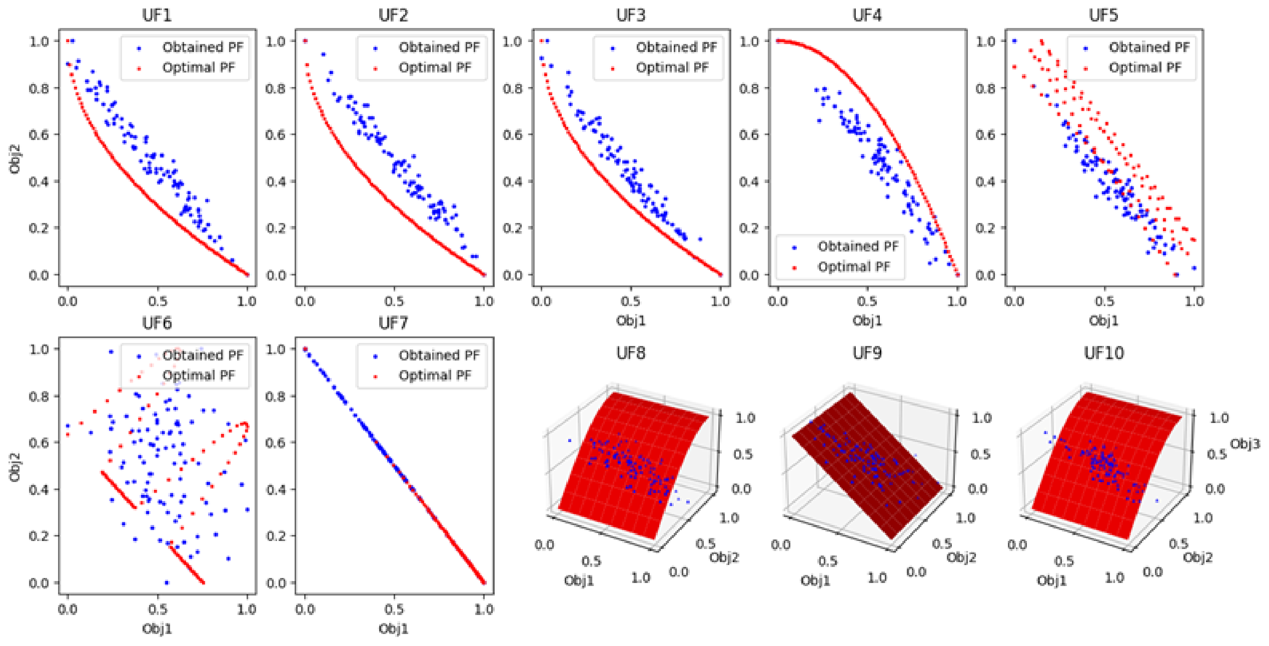

Figure 4.9: MOGWO with Stochastic Universal Sampling Selection Method

Figure 4.10 illustrates the performance of MOGWO algorithm of the test functions with Boltzmann selection method. The performance, in general, similar to SUS method results with slightly better results concerning UF5.

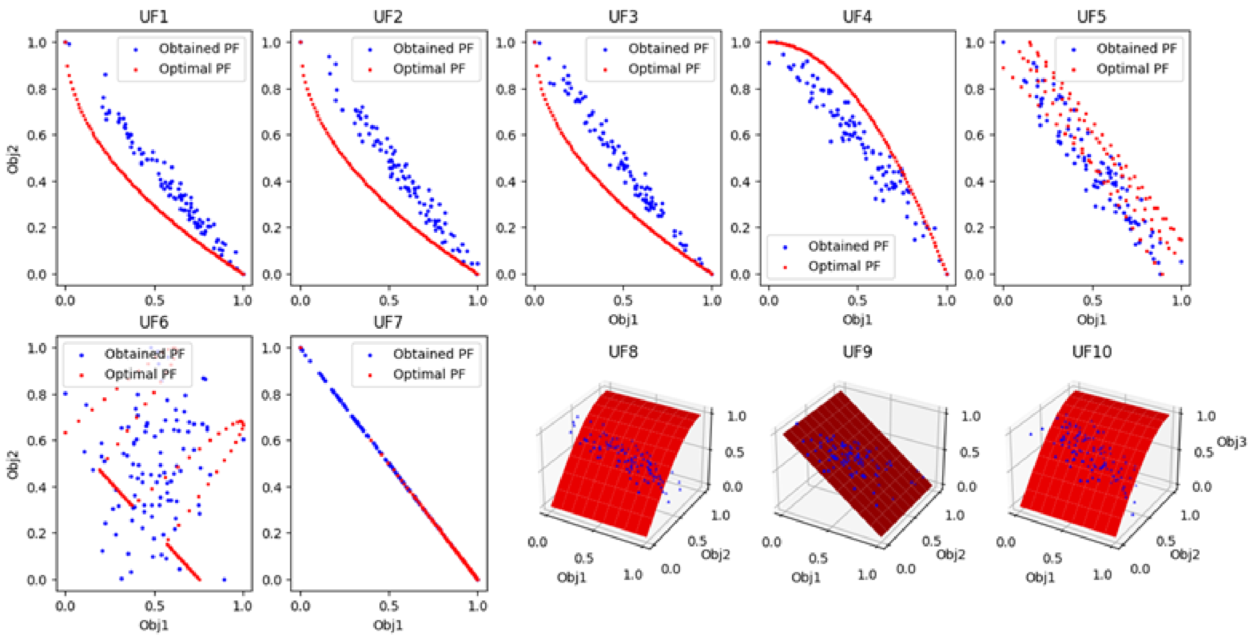

Figure 4.10: MOGWO with Boltzmann Selection Method

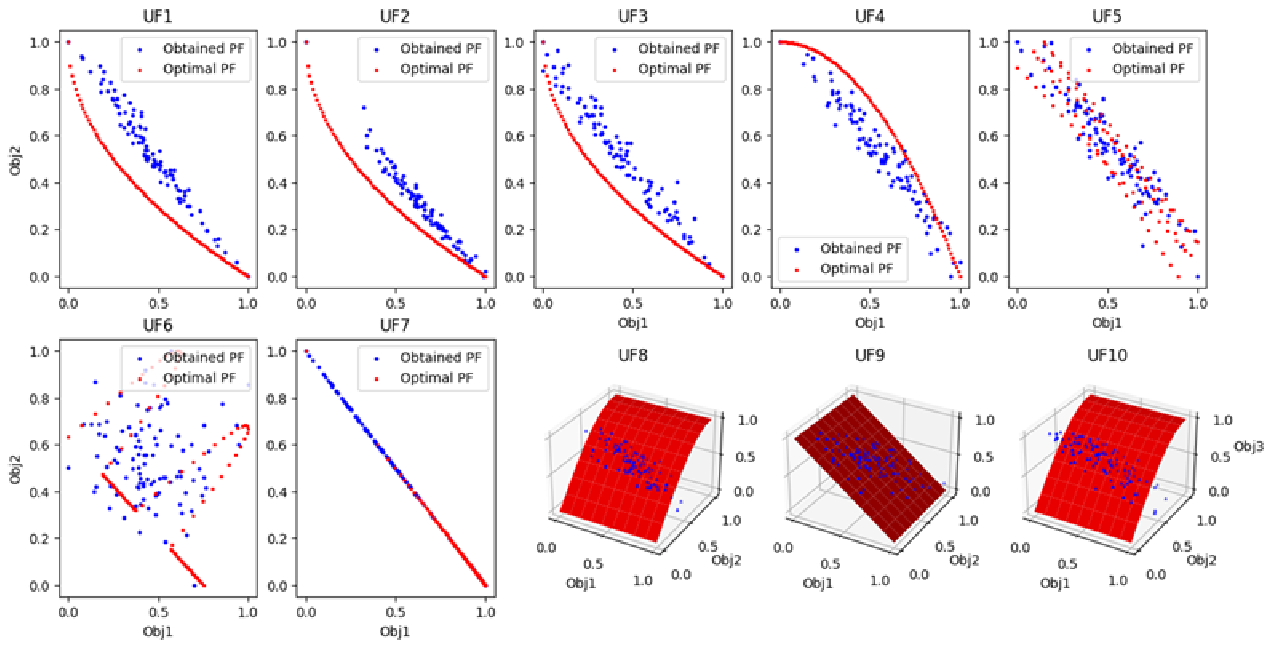

Figure 4.11: MOGWO with Ranking selection

Figure 4.11 illustrate the behavior of the MOGWO algorithm for the test functions with the ranking selection method. The performance of this method is similar to the previous methods except for UF2 where an obvious clustering can be noticed in the generated solutions. This behavior contradicts with the expected diversity in solutions from MOGWO algorithm

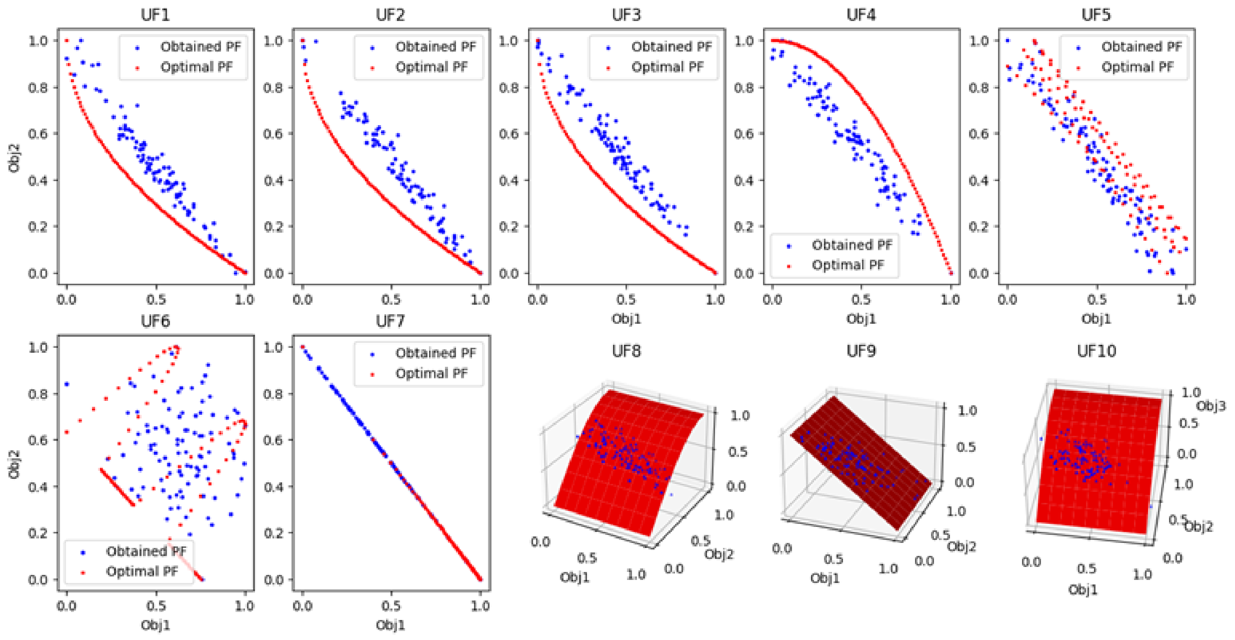

Figure 4.12: MOGWO with Linear Ranking Selection Method

Figure 4.12 illustrate the behaviour of the MOGWO algorithm on the test functions with the linear version of the ranking selection method. The results align with the ranking selection method with better Pareto front approximation for UF2

The above figures show the expected behaviour from MOGWO (a swarm-like algorithm) which represented by the high distribution of solutions cross the objective space of the problem. Furthermore, comparing the behaviour of MOGWO with NSGA-II for functions UF8-10 we can notice the general distribution of the solutions over the objective space surface. From what stated above, the diversity of solutions in MOGWO is better than the diversity of NSGA-II, due to the exploration feature gained from the three leaders (α, β, and δ). All solutions from both algorithms were generated with the same generations number, indicating the MOGWO converge better in problems with higher dimensionality of the objective space. These two distinctive behaviours noticed in both algorithms (the fast convergence in local objective space for NSGA-II, and the ability of MOGWO to expand its objective space and do more exploration) motivate us to combine the two algorithms and propose two hybrid approaches (i.e., NSGAII-MOGWO, and E NSGAII-MOGWO) to benefit from the advantages of algorithms and limit their drawbacks.

4.2.3 NSGAII-MOGWO Results

The implementation of the hybrid NSGAII-MOGWO was initially deployed to study the change in behaviour of the generated solutions when combining both algorithms without any modification.

Figures 4.13 to 4.18 shows the behaviour of NSGAII-MOGWO regarding the different selection methods and the benchmarking functions.

Figure 4.13 shows the results from NSGAII-MOGWO algorithm on the test functions with the roulette wheel selection method. The results indicate a mixed influence from both base algorithms which is clearly noticed in functions UF1, UF3, and UF4.

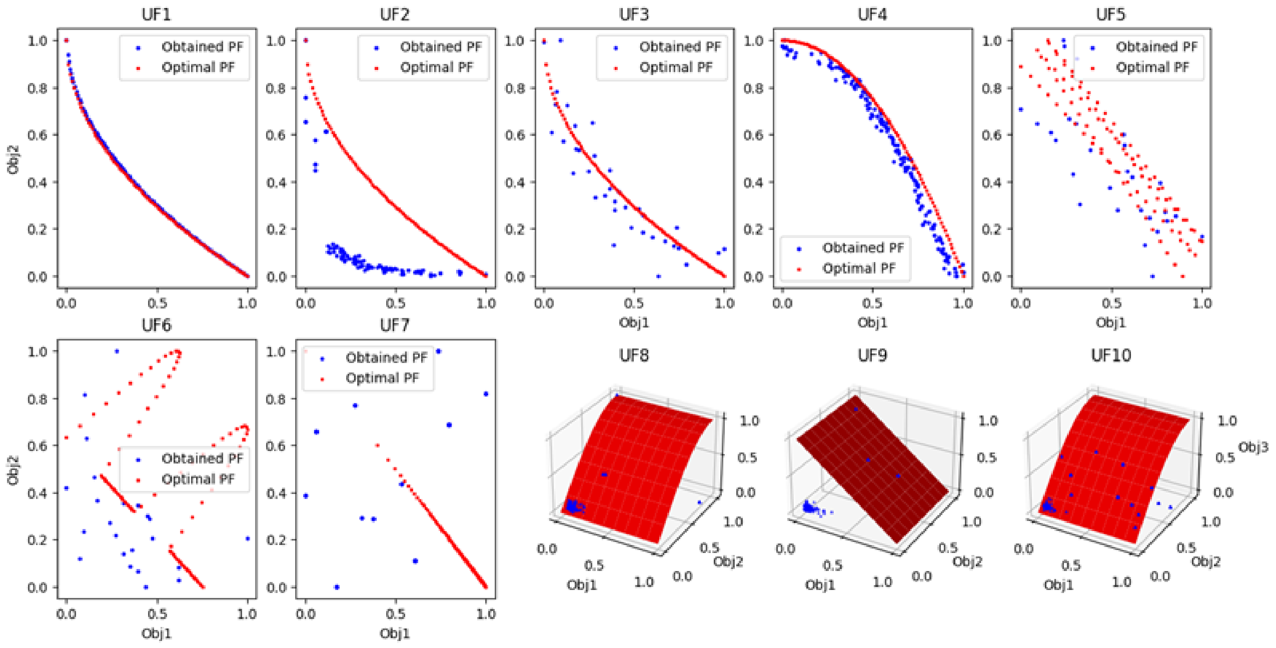

Figure 4.13: NSGAII-MOGWO with Roulette-Wheel Selection Method

Figure 4.14 shows the results from NSGAII-MOGWO algorithm on the test functions with the tournament selection method. The same arguments of Figure 4.13 stand also in this figure. The exception is the bad performance in UF3 and UF5.

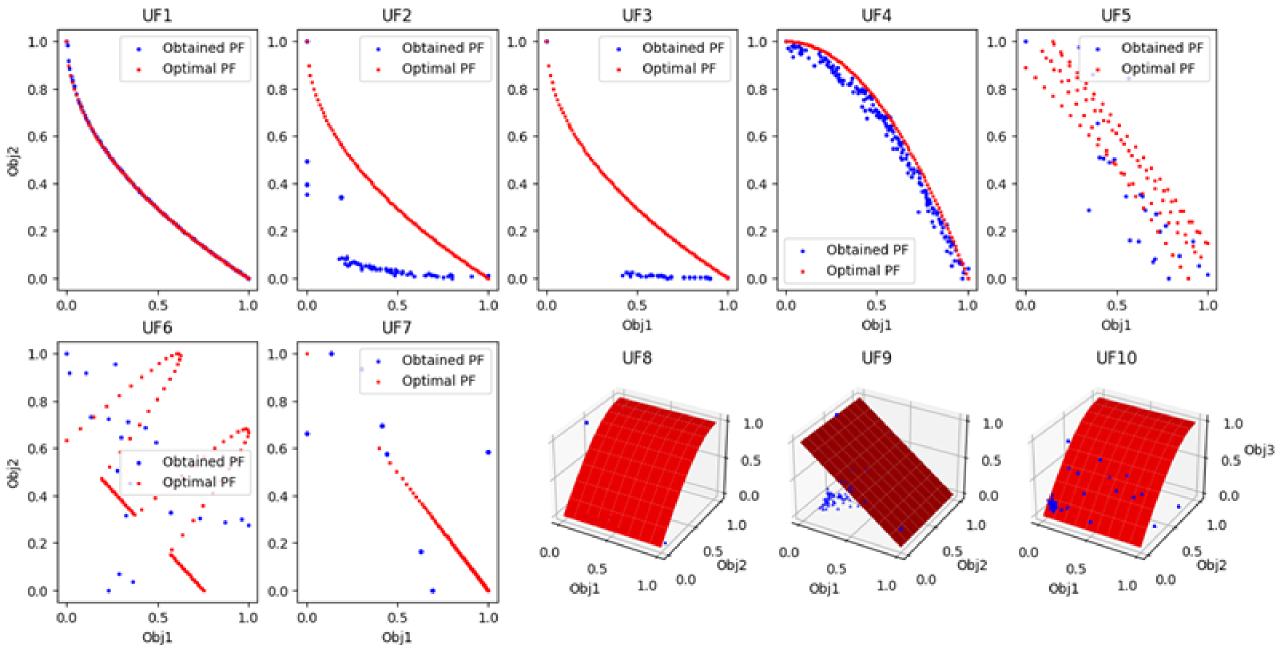

Figure 4.14: NSGAII-MOGWO with Tournament Selection Method

Figure 4.15 shows the results from NSGAII-MOGWO algorithm on the test functions with the stochastic universal sampling selection method. The results presented in this figure flows with the results from previous selection methods.

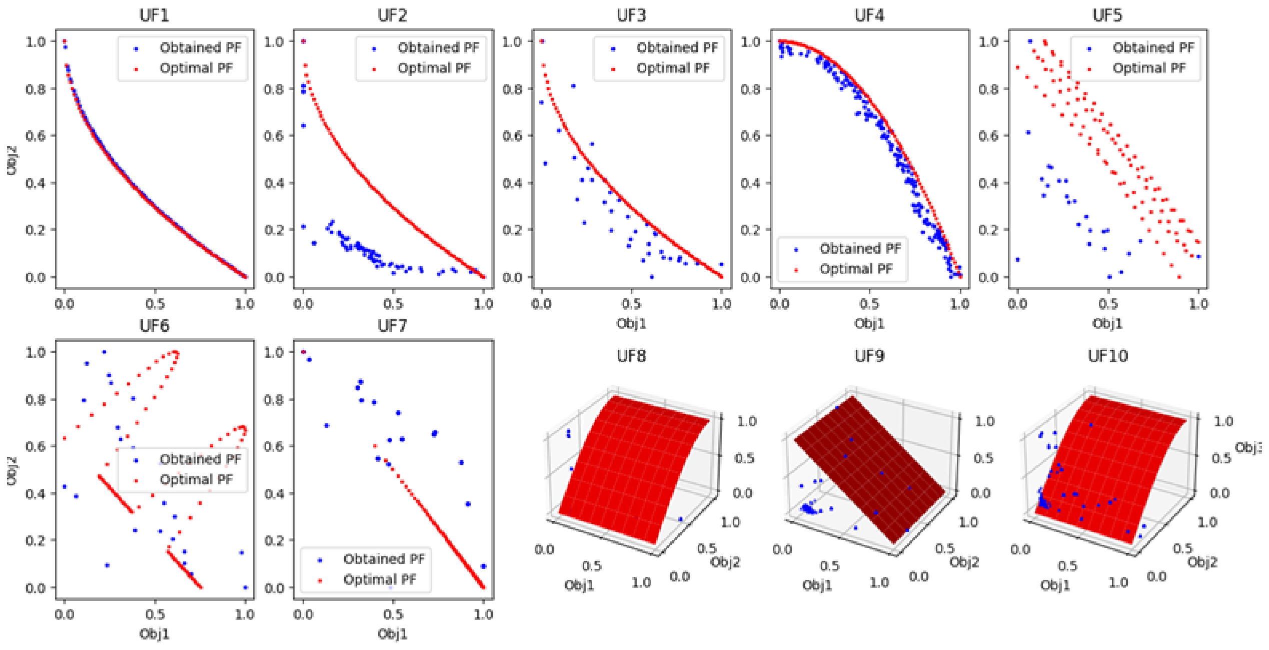

Figure 4.15: NSGAII-MOGWO with Stochastic Universal Sampling Selection Method

Figure 4.16 shows the results from NSGAII-MOGWO algorithm on the test functions with the Boltzmann selection method.

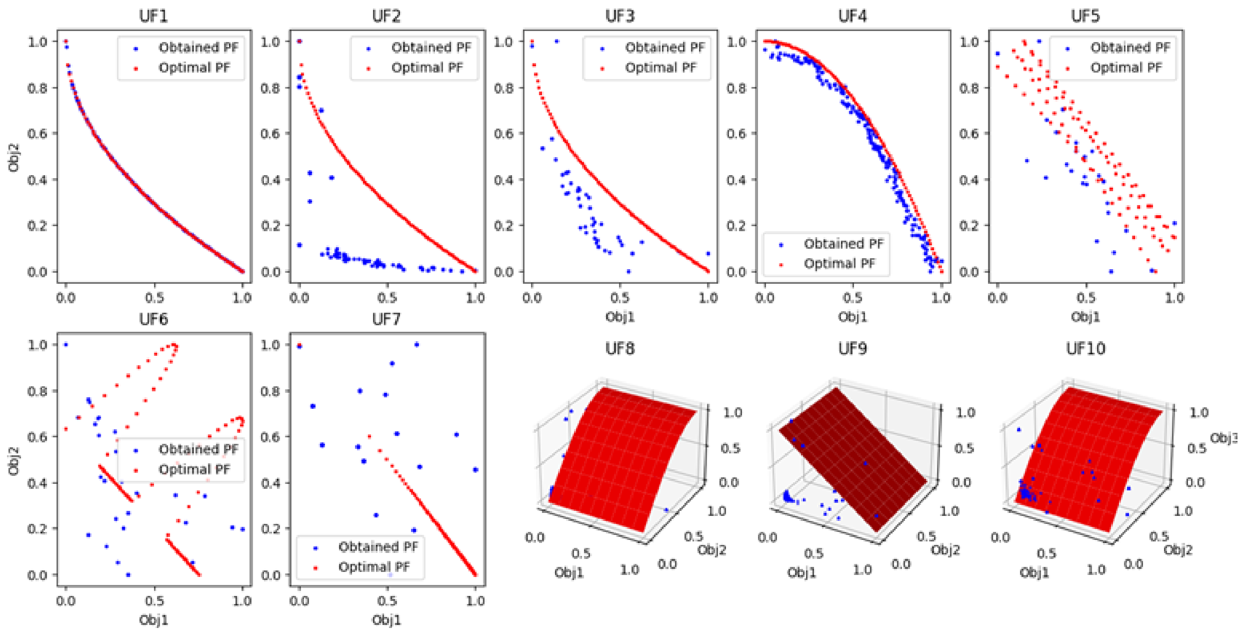

Figure 4.16: NSGAII-MOGWO with Boltzmann Selection Method

Figure 4.17 shows the results from NSGAII-MOGWO algorithm on the test functions with the ranking selection method.

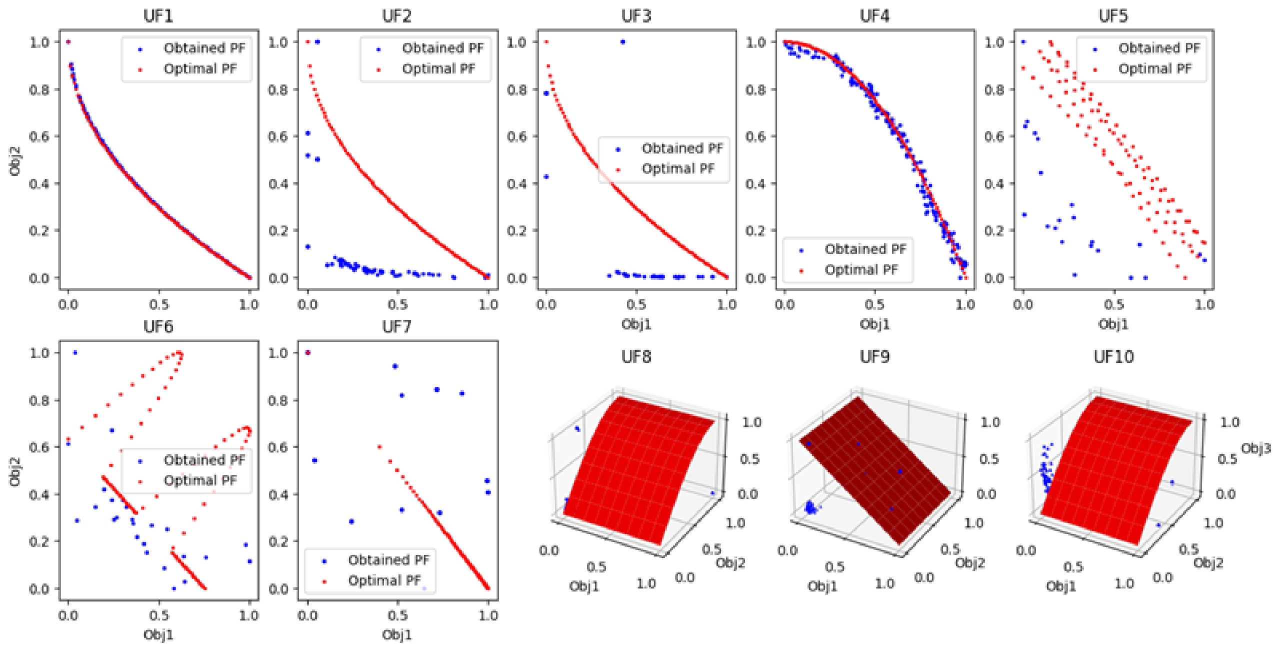

Figure 4.17: NSGAII-MOGWO with Ranking Selection Method

Figure 4.18 shows the results from NSGAII-MOGWO algorithm on the test functions with the Boltzmann selection method.

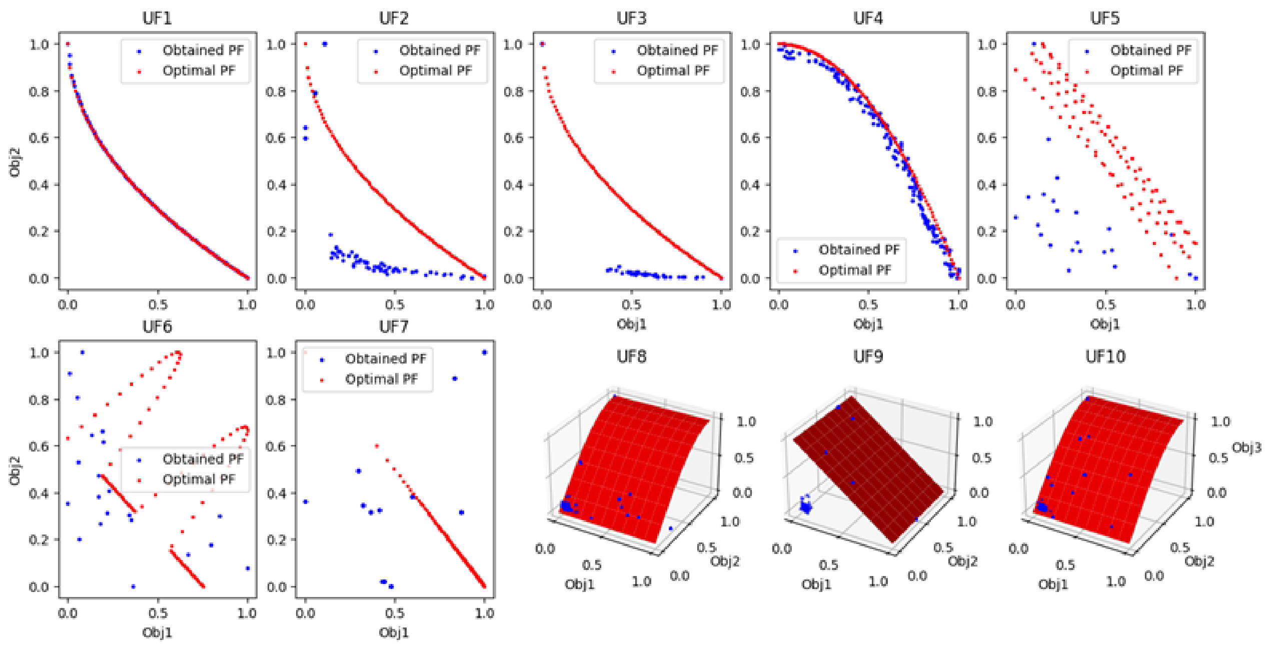

Figure 4.18: NSGAII-MOGWO with Linear Ranking Selection Method

Figure 4.18 shows the results from NSGAII-MOGWO algorithm on the test functions with the linear ranking selection method.

The observed behaviour of solutions inside the objective space indicates the presence of two gravitating poles. On one side, the NSGA-II algorithm tries to pull the solutions towards their local minima and create dense solutions in the objective space. On the other side, the MOGWO has participated in spreading the solutions to saturate the objective space. Although the algorithm maintains the original shape of the Pareto front, the worst performance shown for functions UF2 and UF3. In those two functions, the solutions' crowding distance were significantly smaller in one side (NSGA-II effect) and fewer solutions sparse throughout the objective space (MOGWO effect). Another example of this struggle was noticed in tri-objective functions UF8-10, where the solutions clustered on one side of the objective space surface and a presence of solutions scattered over the rest of the function surface. In summary, the general behaviour of this proposed approach relatively succeeded in adding diversity to the generated solutions through MOGWO.

To conclude, the added diversity from MOGWO was not sufficient to completely impose its exploration feature, thus the second proposed method (E NSGAII-MOGWO) was designed to strengthen the MOGWO to impose more diversity to the generated solutions and mitigate stagnation.

4.2.4 Elite NSGAII-MOGWO Results

This approach was proposed to overcome the limitations indicated in the NSGAII-MOGWO, as detailed in the previous section. The effect of elite population was obvious in the general quality that was superior comparing with the three aforementioned algorithms.

The results shown in Figure 4.19 to 4.24 illustrate the outcomes of the proposed Elite NSGAII-MOGWO with different selection methods and benchmarking functions.

Figure 4.19 shows the results from E NSGAII-MOGWO algorithm on the test functions with the roulette wheel selection method. In all results, the produced solutions are adequately distributed along the search space.

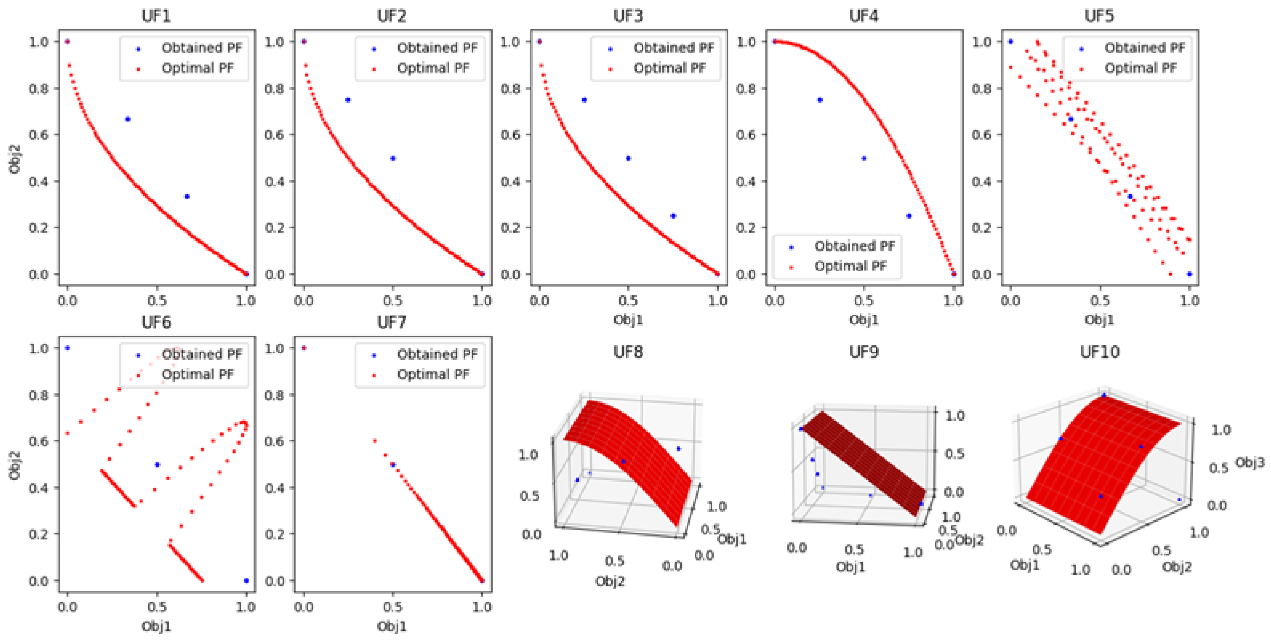

Figure 4.19: E NSGAII-MOGWO with Roulette Wheel Selection Method

Figure 4.20 shows the results from E NSGAII-MOGWO algorithm on the test functions with the tournament selection method.

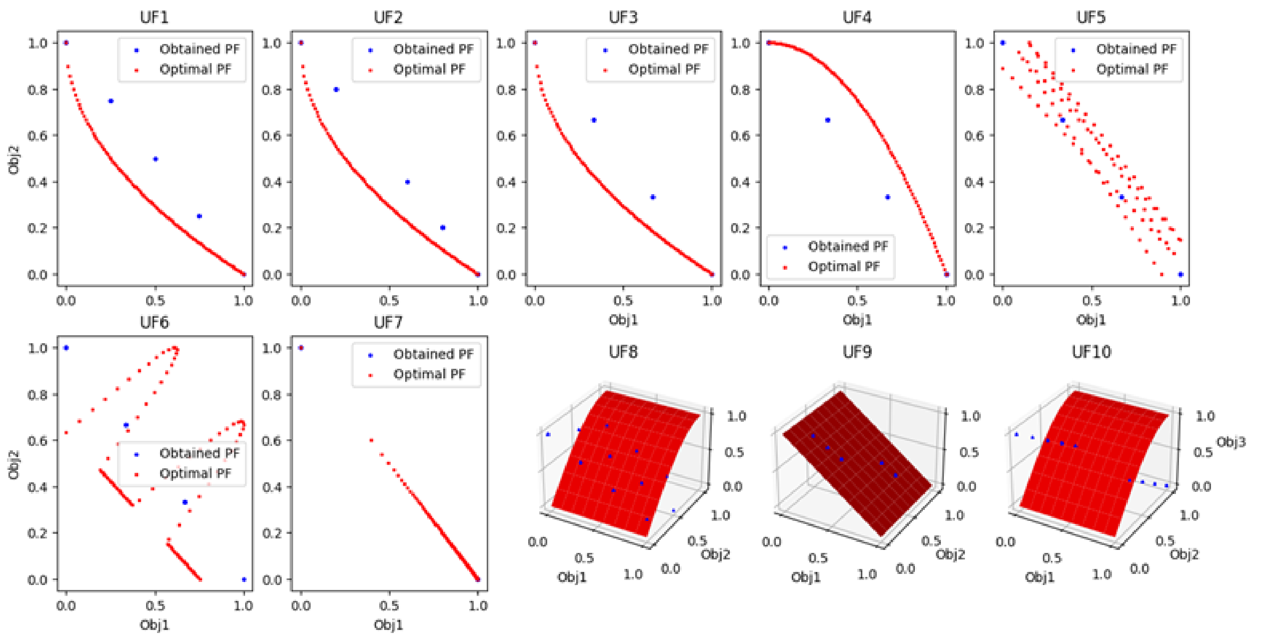

Figure 4.20: E NSGAII-MOGWO with Tournament Selection Method

Figure 4.21 shows the results from E NSGAII-MOGWO algorithm on the test functions with the stochastic universal sampling selection method.

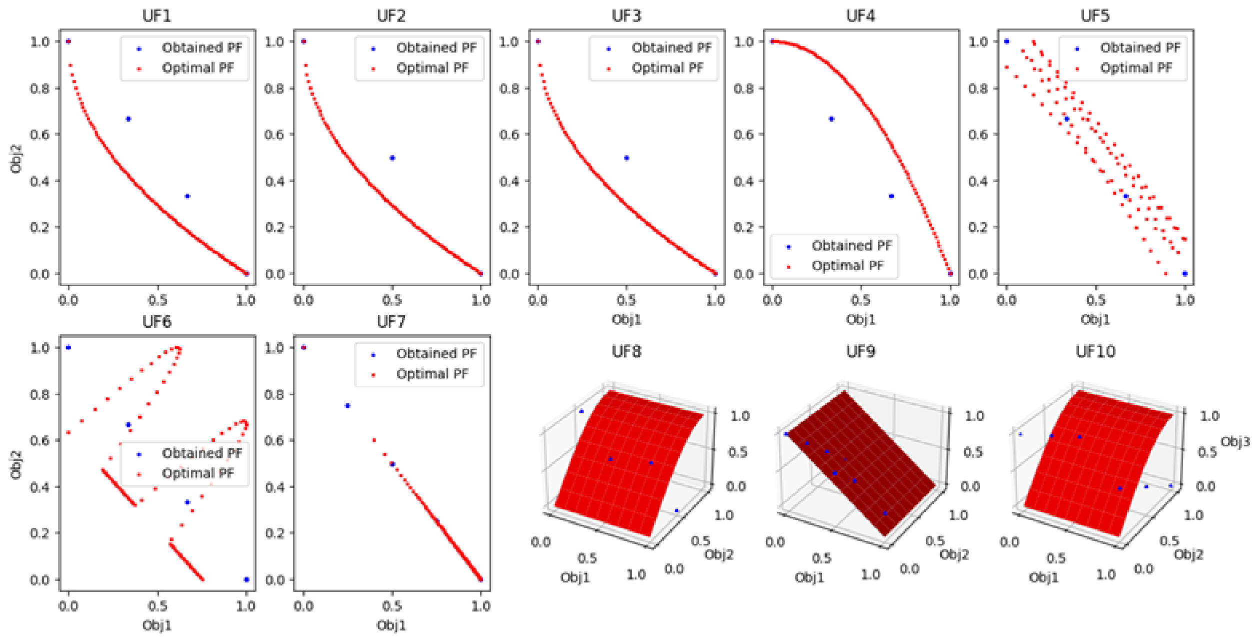

Figure 4.21: E NSGAII-MOGWO with Stochastic Universal Sampling Selection Method

Figure 4.22 shows the results from E NSGAII-MOGWO algorithm on the test functions with the Boltzmann selection method.

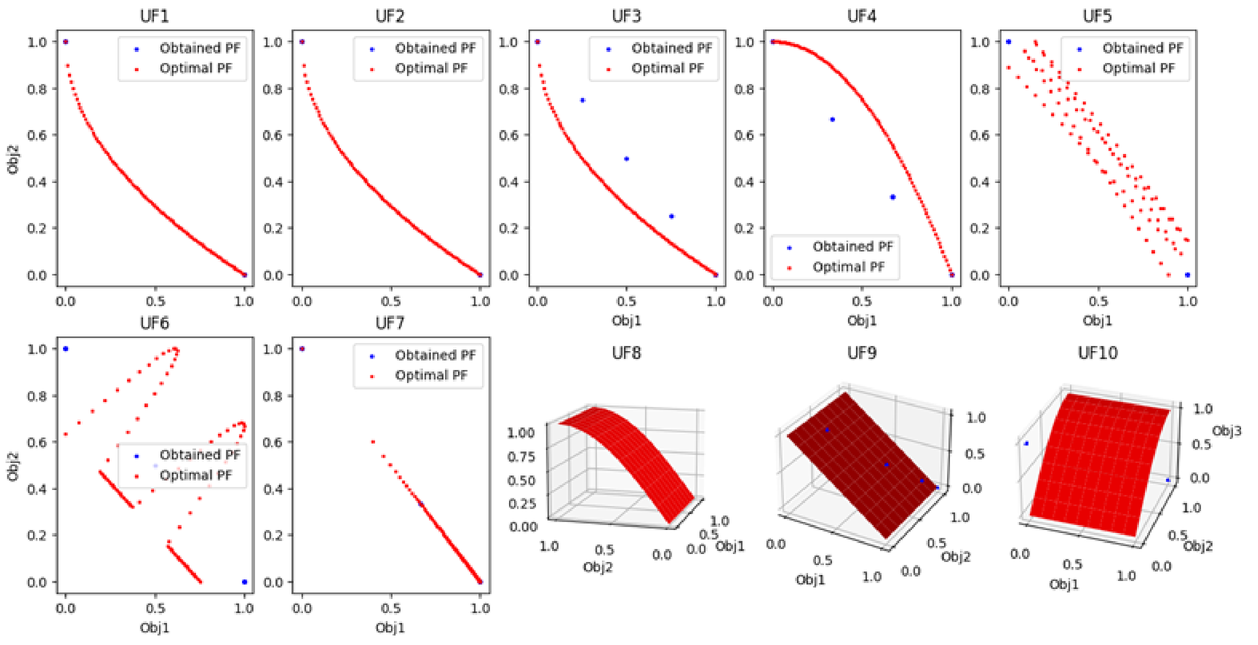

Figure 4.22: E NSGAII-MOGWO with Boltzmann Selection Method

Figure 4.23 shows the results from E NSGAII-MOGWO algorithm on the test functions with the ranking selection method.

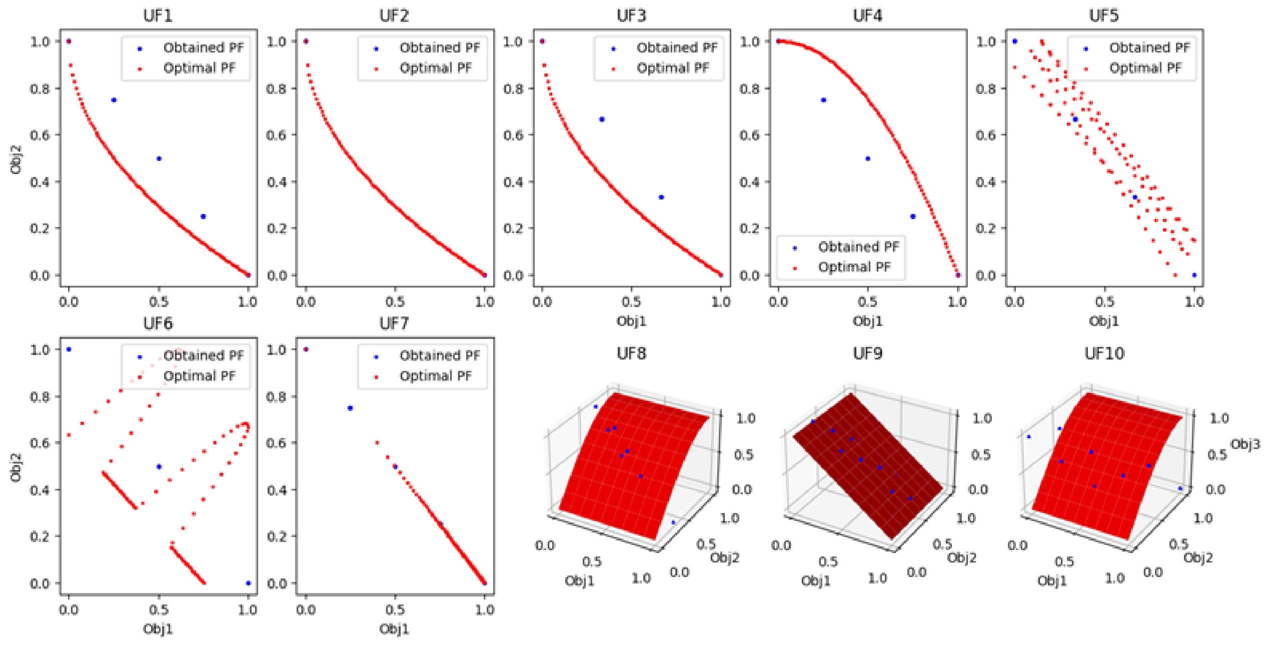

Figure 4.23: E NSGAII-MOGWO with Ranking Selection Method

Figure 4.24 shows the results from E NSGAII-MOGWO algorithm on the test functions with the linear ranking selection method.

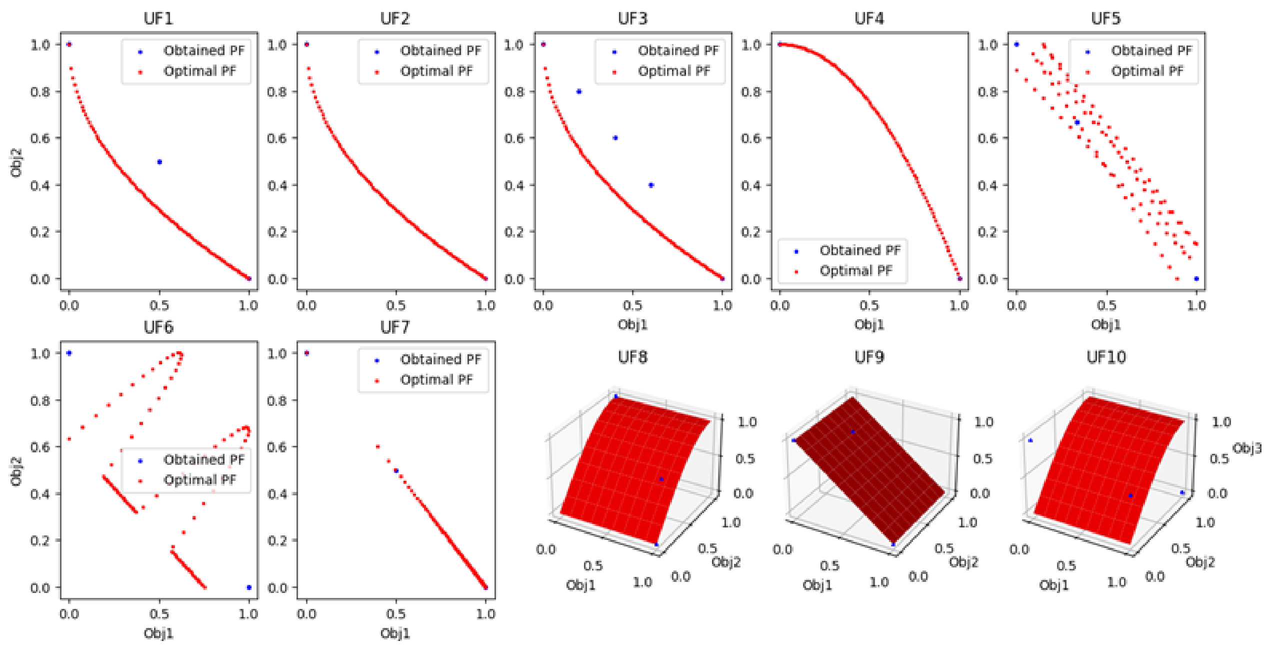

Figure 4.24: E NSGAII-EMOGWO with Linear Ranking Selection Method

The presented results show the ultimate solutions which represented by the leaders' population generated from E NSGAII-MOGWO.

The performance of the proposed algorithm when considering bi-objective functions F1-7 shows a consistence tendency to better representation of the front they formulate. Despite the relative outcomes from the different selection methods, we observe that the upper and lower bounds of the objective space was included in the final population. For UF6, the resulted solutions were parallel to the front with constant distance. From the figures presented above, the E NSGAII-MOGWO succeeded in achieving diversity in the objective space. This conclusion was supported by the performance indicators results in the next section.

The next sections stated the resulted observations for all the candidate selection methods.

4.3 Performance Indicators Comparisons

The tables below show the performance comparisons between the base algorithms (NSGA-II, and MOGWO) and the proposed algorithms (NSGAII-MOGWO and E NSGAII-MOGWO) with different leader selection methods shown in figures 4.1 through 4.24. These comparisons considered in this work as the base to select the final proposed optimization algorithm to solve the IoT-Fog resource allocation problem.

4.3.1 Roulette Wheel Leader selection Method

The results shown in Table 4.1 shows that the final proposed algorithm has better overall performance over other related algorithms. The superiority percentages for the proposed algorithm over other algorithms when using Roulette Wheel leader selection method were (0.7666, 0.6334, and 0.6) for MOGWO, NSGA-II, NSGAII-MOGWO respectively.

Table 4.1: Performance of the Related Algorithms using Roulette Wheel Leader Selection Method

UF	Algorithm	GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0319	0.0318	0.0314	0.0238	0.6134	0.0181
	NSGA-II	0.0078	0.0071	0.0071	0.0069	0.6534	0.0069
	NSGAII-MOGWO	**0.0036**	**0.0022**	**0.0035**	**0.0023**	**0.6610**	0.0072
	E NSGAII-MOGWO	0.0958	0.0951	0.0655	0.0636	0.5485	**0.0290**
UF2	MOGWO	**0.1054**	0.0009	0.0902	**0.0001**	0.8135	0.0143
	NSGA-II	0.2252	**0.0001**	0.1504	0.0000	**0.9140**	0.0151
	NSGAII-MOGWO	0.2197	0.0093	0.1744	0.0000	0.9078	0.0128
	E NSGAII-MOGWO	0.1160	0.1160	**0.0885**	0.0879	0.5030	**0.0404**
UF3	MOGWO	0.0576	0.0005	**0.0374**	**0.0001**	0.7536	0.0096
	NSGA-II	**0.0554**	0.0171	0.0424	0.0007	**0.7672**	0.0349
	NSGAII-MOGWO	0.1465	**0.0001**	0.2303	0.0492	0.6478	**0.0090**
	E NSGAII-MOGWO	0.1190	0.1185	0.0724	0.0712	0.5372	0.0152
UF4	MOGWO	0.0316	0.0028	0.0266	0.0011	0.3710	0.0045
	NSGA-II	0.0364	**0.0003**	0.0250	0.0005	0.4055	0.0103
	NSGAII-MOGWO	**0.0202**	0.0018	**0.0132**	0.0003	0.3748	0.0083
	E NSGAII-MOGWO	0.1335	0.0007	0.0819	**0.0002**	**0.5406**	**0.0165**
UF5	MOGWO	0.1749	0.0008	0.1641	**0.0001**	0.7904	0.0426
	NSGA-II	0.0649	0.0078	0.0746	0.0005	0.6897	0.0543
	NSGAII-MOGWO	0.2312	0.0005	0.2177	0.0007	**0.8793**	**0.0553**
	E NSGAII-MOGWO	**0.0246**	**0.0003**	**0.0423**	0.0128	0.4936	0.0269

UF6	MOGWO	0.1249	0.0003	0.1997	0.0007	0.9191	0.0518
	NSGA-II	0.1354	0.0002	0.1716	0.0004	**0.9288**	**0.1079**
	NSGAII-MOGWO	0.1877	**0.0001**	0.2639	0.0005	0.9068	0.1073
	E NSGAII-MOGWO	**0.0827**	0.0023	**0.0640**	**0.0002**	0.8577	0.0428
UF7	MOGWO	0.1734	0.1726	0.0901	0.0878	0.3213	0.0151
	NSGA-II	0.3515	0.1611	0.2672	0.0723	**0.9332**	0.0000
	NSGAII-MOGWO	0.2316	0.0916	0.1626	0.0436	0.8225	0.0000
	E NSGAII-MOGWO	**0.0661**	**0.0468**	**0.0177**	**0.0125**	0.4863	**0.0212**
UF8	MOGWO	0.4880	**0.0003**	0.2221	0.0159	0.8726	0.0368
	NSGA-II	0.7241	0.0511	0.4877	0.0110	**0.9471**	0.0134
	NSGAII-MOGWO	0.6072	0.0373	0.4147	**0.0090**	0.9201	0.0238
	E NSGAII-MOGWO	**0.1425**	0.0125	**0.1403**	0.0675	0.4884	**0.0692**
UF9	MOGWO	0.2308	0.0800	0.1691	0.0199	0.9017	0.0389
	NSGA-II	0.5113	0.1210	0.6058	**0.0075**	0.9672	0.0199
	NSGAII-MOGWO	0.3882	0.0889	0.3516	0.0097	**0.9696**	0.0228
	E NSGAII-MOGWO	**0.1224**	**0.0673**	**0.1453**	0.1241	0.5324	**0.0589**
UF10	MOGWO	0.5145	**0.0016**	0.2378	0.0157	0.8889	0.0519
	NSGA-II	0.4537	0.0474	0.3463	0.0101	0.3876	0.0166
	NSGAII-MOGWO	0.5524	0.0232	0.3904	**0.0099**	0.9383	0.0122
	E NSGAII-MOGWO	**0.0949**	0.0382	**0.1556**	0.1191	**0.9473**	**0.0684**

Table 4.2 shows a summary of the results from Table 4.1, where E NSGAII-MOGWO scores the first rank comparing with the rest of algorithms. Note that NSGAII-MOGWO ranked second which means that despite the limited contribution of MOGWO it was better than implementing NSGA-II and MOGWO independently.

Table 4.2: Summary of Results from Table 4.1.

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	1	**3**	1	**3**	0	0	0.1333	4
NSGA-II	1	2	0	1	**5**	2	0.1833	3
NSGAII-MOGWO	2	2	2	**3**	3	1	0.2167	2
E NSGAII-MOGWO	**6**	**3**	**7**	**3**	2	**7**	**0.4667**	1

4.3.2 Tournament Leader Selection Method

Table 4.3 below shows another selection method (Tournament) as base of comparison between the selected algorithms. The ratio of superiority for the proposed algorithm were (0.7868, 0.6874, and 0.6563) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. A summary of the results can be found in Table 4.4 with E NSGAII-MOGWO scored first.

Table 4.3: Performance of the Related Algorithms using Tournament Selection method

UF	Algorithm	GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0359	0.0359	0.0195	0.0166	0.6281	0.0153
	NSGA-II	**0.0039**	**0.0028**	**0.0041**	**0.0030**	**0.6601**	0.0061
	NSGAII-MOGWO	0.0083	0.0080	0.0083	0.0079	0.6501	0.0058
	E NSGAII-MOGWO	0.1029	0.1029	0.0620	0.0605	0.5577	**0.0226**
UF2	MOGWO	**0.0746**	0.0008	0.0692	0.0006	0.7849	0.0084
	NSGA-II	0.2175	**0.0001**	0.1616	**0.0003**	**0.9199**	0.0154
	NSGAII-MOGWO	0.1806	0.0002	0.1564	0.0015	0.8510	0.0149
	E NSGAII-MOGWO	0.1086	0.1086	**0.0690**	0.0685	0.5434	**0.0340**
UF3	MOGWO	**0.0881**	0.0002	0.0983	**0.0001**	**0.7982**	0.0219
	NSGA-II	0.1448	**0.0001**	0.2471	0.0715	0.5775	0.0164
	NSGAII-MOGWO	0.1875	0.0004	0.2554	0.0409	0.6869	0.0323
	E NSGAII-MOGWO	0.1347	0.1345	**0.0783**	0.0955	0.5058	**0.0336**
UF4	MOGWO	0.0463	0.0004	0.0327	0.0008	0.3973	0.0056
	NSGA-II	0.0339	0.0002	0.0219	0.0004	0.4054	0.0099
	NSGAII-MOGWO	**0.0250**	0.0011	**0.0155**	0.0006	0.3881	0.0088
	E NSGAII-MOGWO	0.1194	**0.0001**	0.0586	**0.0003**	**0.5145**	**0.0236**
UF5	MOGWO	0.1593	0.0044	0.1706	0.0007	0.7650	0.0335
	NSGA-II	0.0627	0.0122	0.0880	0.0236	0.5655	0.0791
	NSGAII-MOGWO	0.1326	0.0027	0.1100	**0.0001**	**0.8169**	**0.0905**
	E NSGAII-MOGWO	**0.0228**	**0.0025**	**0.0381**	0.0177	0.4621	0.0372
UF6	MOGWO	0.0706	0.0007	0.1463	0.0015	0.8048	0.0276
	NSGA-II	0.1385	0.0009	0.1828	0.0035	0.8080	0.0339
	NSGAII-MOGWO	0.0999	0.0003	0.1191	0.0136	0.7446	**0.0827**
	E NSGAII-MOGWO	**0.0680**	**0.0002**	**0.0632**	**0.0012**	**0.8716**	0.0457
UF7	MOGWO	0.1538	0.1505	0.0774	0.0702	0.3710	0.0164
	NSGA-II	0.2207	0.1491	0.2153	0.0446	0.5888	0.0000
	NSGAII-MOGWO	0.2641	0.1166	0.2572	**0.0001**	**0.8774**	0.0000
	E NSGAII-MOGWO	**0.0960**	**0.0679**	**0.0377**	0.0266	0.4754	**0.0325**

UF8	MOGWO	0.5000	**0.0003**	0.2244	0.0110	0.9023	0.0307
	NSGA-II	0.5853	0.0357	0.4617	0.0132	**0.9589**	0.0318
	NSGAII-MOGWO	0.6041	0.0245	0.3357	**0.0084**	0.9502	**0.0196**
	E NSGAII-MOGWO	**0.1030**	0.0455	**0.1697**	0.1363	0.3995	0.0719
UF9	MOGWO	0.2364	**0.0061**	0.2038	**0.0126**	0.9271	0.0356
	NSGA-II	0.2848	0.0594	0.4338	0.0408	0.8543	0.0584
	NSGAII-MOGWO	0.4631	0.1224	0.5817	0.0144	**0.9382**	0.0237
	E NSGAII-MOGWO	**0.1153**	0.0854	**0.1547**	0.1322	0.4994	**0.0675**
UF10	MOGWO	0.5664	0.0710	0.2494	0.0119	0.9332	0.0486
	NSGA-II	0.4327	0.0361	0.3095	0.0099	0.9358	0.0179
	NSGAII-MOGWO	0.5108	0.0525	0.3345	**0.0095**	**0.9423**	0.0110
	E NSGAII-MOGWO	**0.1134**	**0.0275**	**0.1617**	0.0976	0.4790	**0.0683**

Table 4.4: Summary of Results from Table 4.3.

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	2	2	0	2	1	0	0.1167	4
NSGA-II	1	3	1	2	3	0	0.1667	3
NSGAII-MOGWO	1	0	1	**3**	**4**	3	0.2000	2
E NSGAII-MOGWO	**6**	**5**	**8**	**3**	2	**7**	**0.5167**	1

4.3.3 Ranking Leader Selection Method

The ranking selection method guides the process of choosing individuals with high fitness. Table 4.5 below shows the performance of the algorithms and additionally the percentage and ranking of the algorithms. Generally, the proposed algorithm records better performance than other algorithms with superiority percentage of (0.3477, 0.4782, and 0.5651) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. The summary of results can be found in Table 4.6 which shows that the proposed algorithm scored the first place.

Table 4.5: Performance of the Related Algorithms using Ranking Selection Method

UF		GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0378	0.0378	0.0288	0.0230	0.6146	0.0136
	NSGA-II	0.0052	0.0048	0.0053	0.0048	0.6563	0.0044
	NSGAII-MOGWO	**0.0040**	**0.0025**	**0.0039**	**0.0026**	**0.6607**	0.0066
	E NSGAII-MOGWO	0.1380	0.1379	0.0891	0.0889	0.5139	**0.0207**

UF2	MOGWO	0.0901	0.0003	0.0698	0.0002	0.7915	0.0073
	NSGA-II	0.2503	0.0035	0.1753	**0.0001**	**0.9719**	0.0095
	NSGAII-MOGWO	0.2066	**0.0004**	0.1527	0.0005	0.9463	0.0282
	E NSGAII-MOGWO	**0.0842**	0.0842	**0.0576**	0.0865	0.4851	**0.0605**
UF3	MOGWO	0.0647	**0.0002**	**0.0404**	**0.0001**	0.7650	0.0110
	NSGA-II	0.1996	0.0664	0.1890	0.0002	**0.8481**	0.0142
	NSGAII-MOGWO	0.0753	0.0125	0.0535	0.0050	0.8021	**0.0320**
	E NSGAII-MOGWO	**0.0107**	0.1072	0.0661	0.0652	0.5600	0.0208
UF4	MOGWO	0.0379	0.0067	0.0300	0.0002	0.3842	0.0055
	NSGA-II	**0.0165**	0.0049	**0.0118**	0.0002	0.3624	0.0075
	NSGAII-MOGWO	0.0228	**0.0010**	0.0156	**0.0001**	0.3875	0.0094
	E NSGAII-MOGWO	0.0859	0.0012	0.0456	0.0006	**0.4680**	**0.0236**
UF5	MOGWO	0.1680	0.0007	0.1502	0.0003	0.7932	0.0372
	NSGA-II	0.2609	**0.0001**	0.2417	**0.0001**	**0.9330**	**0.0782**
	NSGAII-MOGWO	0.0899	0.0047	0.1086	0.0005	0.6977	0.0701
	E NSGAII-MOGWO	**0.0270**	0.0042	**0.0324**	0.0077	0.5164	0.0307
UF6	MOGWO	0.1427	**0.0002**	0.2042	0.0024	0.7834	0.0687
	NSGA-II	0.1045	0.0004	0.1787	0.0003	0.8452	0.0933
	NSGAII-MOGWO	0.1401	0.0003	0.1878	**0.0001**	**0.9669**	**0.1211**
	E NSGAII-MOGWO	**0.0617**	0.0027	**0.0460**	0.0080	0.7800	0.0496
UF7	MOGWO	0.1582	0.1560	0.0665	0.0624	0.3575	**0.0150**
	NSGA-II	0.2848	0.1809	0.2165	**0.0001**	**0.7363**	0.0000
	NSGAII-MOGWO	0.1863	0.0934	0.1500	0.0971	0.5007	0.0000
	E NSGAII-MOGWO	**0.0798**	**0.0565**	**0.0245**	0.0173	0.4818	0.0135
UF8	MOGWO	0.4696	**0.0006**	0.2452	0.0133	0.8710	0.0331
	NSGA-II	0.6752	0.0491	0.4283	**0.0071**	**0.9540**	0.0200
	NSGAII-MOGWO	0.7979	0.0494	0.6878	0.0098	0.9476	0.0184
	E NSGAII-MOGWO	**0.1082**	0.0181	**0.1693**	0.1109	0.4412	**0.0640**
UF9	MOGWO	0.2814	**0.0010**	0.1953	0.0119	0.9425	0.0418
	NSGA-II	0.4450	0.0837	0.3514	0.0077	0.9653	0.0557
	NSGAII-MOGWO	0.4782	0.0868	0.5421	**0.0046**	**0.9815**	**0.0130**
	E NSGAII-MOGWO	**0.1562**	0.1311	**0.1873**	0.1737	0.4686	0.0617
UF10	MOGWO	0.4902	**0.0013**	0.2498	0.0197	0.8676	0.0506
	NSGA-II	0.4129	0.0341	0.3990	0.0327	0.7315	0.0291
	NSGAII-MOGWO	0.6122	0.0440	0.4146	**0.0082**	**0.9433**	0.0168
	E NSGAII-MOGWO	**0.1023**	0.0270	**0.1190**	0.0717	0.4663	**0.0629**

Table 4.6: Summary of Results from Table 4.5.

The Ranking selection method comply with the method of leader selection where the leader solutions are always at the head of the population.

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	0	**5**	1	1	0	**6**	0.2167	3
NSGA-II	1	1	1	3	**5**	1	0.2000	4
NSGAII-MOGWO	1	3	1	**5**	4	1	0.2500	2
E NSGAII-MOGWO	**8**	1	**7**	0	1	2	**0.3167**	1

4.3.4 Linear Ranking Leader Selection Method

Linear ranking adds linear probability to the rank of individuals to leverage exploration in evolutionary algorithms. Therefore, considered as a selection method to be considered in this work. The resulted performance of the suggested algorithms using linear ranking are illustrated in Table 4.7. From the results summary in Table 4.8, the percentage of superiority were as follow (0.3333, 0.75, and 0.4168) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. This test indicates another superiority of the proposed algorithm over the other algorithm which ranked first. The second place was taken also by MOGWO because this method is an alteration of the Ranking method and the previous argument in Section 4.3.3 stand approved.

Table 4.7: Performance of the Related Algorithms using Linear Ranking Selection Method

UF	Algorithm	GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0381	0.0381	0.0322	0.0265	0.6041	0.0193
	NSGA-II	0.0043	0.0031	**0.0039**	**0.0029**	0.6597	0.0064
	NSGAII-MOGWO	**0.0040**	**0.0028**	0.0043	0.0030	**0.6603**	0.0058
	E NSGAII-MOGWO	0.1087	0.1081	0.0675	0.0650	0.5563	**0.0213**
UF2	MOGWO	**0.0971**	0.0003	0.0752	0.0004	0.8035	0.0076
	NSGA-II	0.2019	0.0083	0.1654	0.0005	0.8920	0.0168
	NSGAII-MOGWO	0.2145	**0.0001**	0.1821	**0.0001**	**0.9591**	0.0143
	E NSGAII-MOGWO	0.1226	0.1224	**0.0847**	0.0844	0.5168	**0.0176**
UF3	MOGWO	**0.0589**	0.0004	**0.0464**	**0.0001**	**0.7570**	0.0157
	NSGA-II	0.1644	0.0002	0.2397	0.0552	0.6281	0.0170
	NSGAII-MOGWO	0.2061	**0.0001**	0.2464	0.0459	0.6663	0.0186
	E NSGAII-MOGWO	0.1261	0.1254	0.0871	0.0860	0.5271	**0.0349**

UF4	MOGWO	**0.0245**	0.0031	0.0276	0.0032	0.3544	0.0046
	NSGA-II	0.0246	0.0013	**0.0156**	0.0007	0.3817	0.0082
	NSGAII-MOGWO	0.0325	0.0003	0.0208	**0.0001**	0.3981	0.0085
	E NSGAII-MOGWO	0.1287	**0.0001**	0.0752	0.0025	**0.5107**	**0.0364**
UF5	MOGWO	0.1645	0.0010	0.1546	0.0002	0.8142	0.0454
	NSGA-II	0.2710	0.0004	0.1881	0.0003	**0.9188**	0.0972
	NSGAII-MOGWO	0.2494	0.0043	0.2148	**0.0001**	0.8775	**0.1081**
	E NSGAII-MOGWO	**0.0296**	**0.0002**	**0.0394**	0.0013	0.5816	0.0233
UF6	MOGWO	0.1184	0.0004	0.1865	0.0005	0.8591	0.0547
	NSGA-II	0.1331	**0.0001**	0.1793	**0.0001**	**0.9170**	0.0738
	NSGAII-MOGWO	0.1101	0.0007	0.1851	0.0005	0.8836	**0.0971**
	E NSGAII-MOGWO	**0.0790**	0.0094	**0.0562**	0.0346	0.6300	0.0457
UF7	MOGWO	0.1230	0.1183	0.0483	0.0451	0.4040	0.0173
	NSGA-II	0.3249	0.0928	0.2048	0.0002	**0.8467**	0.0000
	NSGAII-MOGWO	0.3012	0.1541	0.1720	**0.0001**	0.7987	**0.0211**
	E NSGAII-MOGWO	**0.0735**	**0.0520**	**0.0198**	0.0140	0.4855	0.0192
UF8	MOGWO	0.4783	**0.0007**	0.2150	0.0132	0.8845	0.0374
	NSGA-II	0.5858	0.0181	0.4531	**0.0067**	0.9508	0.0224
	NSGAII-MOGWO	0.7048	0.0510	0.6078	0.0066	**0.9696**	0.0347
	E NSGAII-MOGWO	**0.0918**	0.0303	**0.1488**	0.1128	0.4105	**0.0535**
UF9	MOGWO	0.2599	**0.0012**	0.2223	0.0127	0.9255	0.0512
	NSGA-II	0.4413	0.1175	0.4347	**0.0093**	**0.9585**	0.0161
	NSGAII-MOGWO	0.3407	0.0992	0.4071	0.0174	0.9224	0.0388
	E NSGAII-MOGWO	**0.1136**	0.0627	**0.1441**	0.1202	0.5164	**0.0573**
UF10	MOGWO	0.4721	0.0011	0.2304	0.0251	0.8363	0.0498
	NSGA-II	0.5938	0.0386	0.4099	0.0075	**0.9630**	**0.0113**
	NSGAII-MOGWO	0.5140	0.0363	0.3632	0.0116	0.9205	0.0150
	E NSGAII-MOGWO	**0.1488**	**0.0004**	**0.1629**	**0.0074**	0.4712	0.0707

Table 4.8: Summary of Results from Table 4.7.

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	3	2	1	1	1	0	0.1333	3
NSGA-II	0	1	2	**4**	**5**	1	0.2167	4
NSGAII-MOGWO	1	3	0	**4**	3	3	0.2333	2
E NSGAII-MOGWO	**6**	**4**	**7**	1	1	**6**	**0.4167**	1

4.3.5 Boltzmann Leader Selection Method

Generally, the Boltzmann method consider fitness as energy and the probability of selection is assigned depending on the fitness value. The results of using the Boltzmann selection method are shown in Table 4.9. In the summary of results illustrated in Table 4.10, the performance of the proposed algorithm against the related algorithms were (0.52, 0.64, and 0.4401) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. The E NSGAII-MOGWO has additional first rank in this test. Additionally, it can be seen that NSGAII-MOGWO regained the second place as Boltzmann selection method does not depend on ranking like the Ranking method or probabilistic ranking like Linear Ranking method.

Table 4.9: Performance of the Related Algorithms using Boltzmann Selection Method

UF	Algorithm	GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0329	0.0329	0.0205	0.0172	0.6305	**0.0166**
	NSGA-II	**0.0035**	**0.0022**	**0.0035**	**0.0024**	**0.6607**	0.0043
	NSGAII-MOGWO	0.0061	0.0056	0.0058	0.0055	0.6551	0.0057
	E NSGAII-MOGWO	0.1028	0.1028	0.0804	0.0800	0.5139	0.0163
UF2	MOGWO	0.1102	0.0001	0.0898	0.0004	0.8136	0.0064
	NSGA-II	0.1966	0.0012	0.1299	0.0002	**0.9647**	0.0147
	NSGAII-MOGWO	0.2244	**0.0007**	0.1645	**0.0001**	0.9419	0.0230
	E NSGAII-MOGWO	**0.1096**	0.1295	**0.0892**	0.0889	0.5046	**0.0259**
UF3	MOGWO	**0.0496**	0.0008	**0.0352**	**0.0001**	0.7404	0.0167
	NSGA-II	0.1361	0.0050	0.0913	0.0014	**0.8169**	0.0458
	NSGAII-MOGWO	0.1345	**0.0001**	0.1733	0.0197	0.7323	**0.0098**
	E NSGAII-MOGWO	0.1150	0.1149	0.0773	0.0770	0.5315	0.0157
UF4	MOGWO	0.0350	0.0008	0.0315	**0.0005**	0.3761	0.0049
	NSGA-II	0.0340	0.0003	0.0231	0.0009	0.3992	0.0085
	NSGAII-MOGWO	**0.0332**	0.0003	**0.0223**	0.0007	0.3970	0.0084
	E NSGAII-MOGWO	0.0921	**0.0002**	0.0483	0.0006	**0.4841**	**0.0213**
UF5	MOGWO	0.1545	0.0003	0.1439	0.0003	0.7913	0.0349
	NSGA-II	0.0646	0.0041	0.0867	0.0029	0.6671	0.0848
	NSGAII-MOGWO	0.2450	0.0018	0.2259	**0.0001**	**0.9158**	**0.1277**
	E NSGAII-MOGWO	**0.0302**	**0.0001**	**0.0402**	0.0031	0.5721	0.0312
UF6	MOGWO	0.0884	0.0003	0.1795	0.0012	0.7955	0.0639
	NSGA-II	0.1092	0.0007	0.1552	0.0008	0.8635	0.0755
	NSGAII-MOGWO	0.1365	**0.0001**	0.1892	**0.0004**	**0.8854**	**0.1130**

	E NSGAII-MOGWO	**0.0798**	0.0040	**0.0520**	0.0015	0.8496	0.0433
UF7	MOGWO	0.2042	0.2039	0.0900	0.0824	0.2814	**0.0253**
	NSGA-II	0.2116	0.1135	0.2312	0.0005	0.6956	0.0000
	NSGAII-MOGWO	0.3262	0.1624	0.4934	**0.0001**	**0.9116**	0.0000
	E NSGAII-MOGWO	**0.0662**	**0.0468**	**0.0139**	0.0098	0.4880	0.0139
UF8	MOGWO	0.4960	**0.0009**	0.2589	0.0125	0.8871	0.0344
	NSGA-II	0.5265	0.0232	0.3590	0.0050	0.9704	0.0128
	NSGAII-MOGWO	0.5901	0.0087	0.3909	**0.0031**	**0.9880**	0.0101
	E NSGAII-MOGWO	**0.1059**	0.0378	**0.1535**	0.1062	0.4286	**0.0677**
UF9	MOGWO	0.2455	**0.0051**	0.1610	0.0145	0.9253	0.0391
	NSGA-II	0.3453	0.0104	0.2888	**0.0059**	**0.9770**	0.0111
	NSGAII-MOGWO	0.2583	0.0324	0.2164	0.0098	0.9663	0.0232
	E NSGAII-MOGWO	**0.1149**	0.0797	**0.1490**	0.1212	0.5205	**0.0717**
UF10	MOGWO	0.4468	**0.0023**	0.2493	0.0185	0.8750	**0.0534**
	NSGA-II	**0.4576**	0.0118	0.2707	**0.0096**	**0.9537**	0.0280
	NSGAII-MOGWO	0.4396	0.0189	0.2931	0.0097	0.9294	0.0227
	E NSGAII-MOGWO	0.1145	0.0388	**0.1298**	0.0674	0.4750	0.0524

Table 4.10: Summary of Results from Table 4.9.

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	1	**3**	1	2	0	3	0.1667	4
NSGA-II	2	1	1	3	**5**	0	0.2000	3
NSGAII-MOGWO	1	**3**	1	**5**	4	3	0.2833	2
E NSGAII-MOGWO	**6**	**3**	**7**	0	1	**4**	**0.3500**	1

4.3.6 Stochastic Universal Sampling Leader Selection Method

Finally, a stochastic method was incorporated as a method to select the survival individuals based on their fitness while maintaining diversity in population. The results of implementing this selection method is illustrated in Table 4.11 with results summary in Table 4.12 shows the percentage and ranking of the related algorithms. The superiority percentage of the proposed algorithm were (0.1906, 0.6191, and 0.3334) for MOGWO, NSGA-II, NSGAII-MOGWO respectively. The E NSGAII-MOGWO ranked also first in this test indicating its superiority in all selection methods. The MOGWO scored second in this test due to the uniform distribution of samples that cover search space bounds of the solutions which means more exploration, hence MOGWO adopts well with this selection method.

Table 4.11: Performance of the Related Algorithms using Stochastic Universal Sampling Selection Method

UF	Algorithm	GD	GD+	IGD	IGD+	HV	S
UF1	MOGWO	0.0365	0.0364	0.0297	0.0236	**0.6131**	0.0118
	NSGA-II	0.0060	0.0055	0.0058	0.0054	0.6549	**0.0049**
	NSGAII-MOGWO	**0.0037**	**0.0023**	**0.0038**	**0.0025**	0.6611	0.0053
	E NSGAII-MOGWO	0.1092	0.1091	0.0662	0.0654	0.5562	0.0223
UF2	MOGWO	**0.0906**	**0.0001**	**0.0692**	0.0002	0.7954	0.0103
	NSGA-II	0.2010	0.0002	0.1510	**0.0001**	**0.9336**	0.0122
	NSGAII-MOGWO	0.1993	0.0094	0.1469	0.0004	0.9045	0.0128
	E NSGAII-MOGWO	0.1348	0.1348	0.0941	0.0939	0.5063	**0.0233**
UF3	MOGWO	**0.0300**	0.0029	**0.0259**	0.0003	0.7244	0.0141
	NSGA-II	0.0738	0.0068	0.0433	**0.0001**	**0.8191**	**0.0635**
	NSGAII-MOGWO	0.1390	**0.0001**	0.1812	0.0035	0.8139	0.0124
	E NSGAII-MOGWO	0.0900	0.0897	0.0606	0.0574	0.5656	0.0284
UF4	MOGWO	0.0205	0.0060	0.0204	0.0043	0.3449	0.0044
	NSGA-II	0.0343	0.0002	0.0209	**0.0001**	0.4034	0.0083
	NSGAII-MOGWO	0.0210	0.0017	**0.0126**	0.0003	0.3792	**0.0084**
	E NSGAII-MOGWO	**0.0104**	**0.0001**	0.0771	0.0026	**0.4905**	0.0446
UF5	MOGWO	0.2371	0.0009	0.2245	0.0002	0.8440	0.0439
	NSGA-II	0.2334	0.0005	0.2156	**0.0001**	**0.9629**	**0.1177**
	NSGAII-MOGWO	0.1514	0.0003	0.0776	0.0027	0.7894	0.0679
	E NSGAII-MOGWO	**0.0360**	**0.0001**	**0.0743**	0.0035	0.5765	0.0356
UF6	MOGWO	0.1023	0.0004	0.1748	**0.0001**	0.8491	0.0462
	NSGA-II	0.1085	0.0008	0.1623	0.0041	0.8438	0.0604
	NSGAII-MOGWO	0.1552	0.0007	0.2119	0.0012	**0.9506**	**0.0945**
	E NSGAII-MOGWO	**0.0829**	**0.0002**	**0.0577**	0.0026	0.7685	0.0498
UF7	MOGWO	0.1739	0.1694	0.0951	0.0836	**0.3491**	**0.0152**
	NSGA-II	0.1831	0.1623	0.1293	0.0002	0.6381	0.0000
	NSGAII-MOGWO	0.2371	0.1415	0.1370	**0.0001**	0.8453	0.0000
	E NSGAII-MOGWO	**0.0560**	**0.0396**	**0.0105**	0.0074	0.4894	0.0145
UF8	MOGWO	0.5113	**0.0004**	0.2389	0.0116	0.8911	0.0346
	NSGA-II	0.5489	0.0213	0.3637	**0.0062**	0.9655	0.0218
	NSGAII-MOGWO	0.6077	0.0314	0.3300	0.0063	**0.9733**	**0.0159**
	E NSGAII-MOGWO	**0.1313**	0.0239	**0.1432**	0.0714	0.4900	0.0605
UF9	MOGWO	0.2292	**0.0026**	**0.1516**	0.0151	0.9201	0.0392
	NSGA-II	0.3487	0.0406	0.3647	**0.0065**	**0.9817**	0.0129
	NSGAII-MOGWO	0.3756	0.0682	0.3146	0.0092	0.9723	0.0190

	E NSGAII-MOGWO	**0.1383**	0.1114	0.1968	0.1757	0.4624	**0.0701**
UF10	MOGWO	0.5400	0.0013	0.2496	0.0165	0.8989	0.0542
	NSGA-II	0.4313	0.0117	0.2809	**0.0082**	**0.9484**	0.0102
	NSGAII-MOGWO	0.4499	0.0434	0.3886	0.0115	0.9366	0.0220
	E NSGAII-MOGWO	**0.1406**	**0.0090**	**0.1466**	0.0754	0.4854	**0.0638**

Table 4.12: Summary of Results from Table 4.11

Algorithm	GD	GD+	IGD	IGD+	HV	S	Avg.	Rank
MOGWO	2	3	3	1	2	1	0.2000	2
NSGA-II	0	0	0	**7**	**5**	**3**	0.2500	3
NSGAII-MOGWO	1	2	2	2	2	**3**	0.2000	2
E NSGAII-MOGWO	**7**	**5**	**5**	0	1	**3**	**0.3500**	1

After completing all the test, the proposed algorithm (E NSGAII-MOGWO) was the superior algorithm over the other three algorithms. Now, we select the selection method that the proposed algorithm had the highest average performance over the other compared algorithms. Table 4.13 shows the results obtained from tests presented in section 4.3.1 through 4.3.6. The average performance column in Table 4.13 indicates that the best selection method to deploy with this algorithm is the Tournament selection method.

Table 4.13: Performance Comparison between Selection Methods for each algorithm.

Method	**MOGWO**	**NSGA-II**	**NSGAII-MOGWO**	**Average Absolute Performance**
Roulette Wheel	76.66%	63.34%	60.00%	66.66%
Tournament	78.68%	68.78%	65.63%	71.03%
Ranking	34.77%	47.82%	56.51%	46.37%
Linear Ranking	33.33%	75.00%	41.67%	50.00%
Boltzmann	52.00%	64.00%	44.00%	53.33%
Stochastic Universal Sampling	19.06%	61.00%	33.34%	37.80%

4.3.7 Comparing E NSGAII-MOGWO with Related Works

The performance of the proposed E NSGAII-MOGWO algorithm was compared against six multi-objective optimization algorithms (DGEA [28], ARMOEA [12], AGEMOEA [14], CMOPSO [13], HDMOPSO [29]). The above selected works utilize the same benchmarking function utilized in this work. The performance indicators used are IGD and HV, as mentioned earlier IGD to measure the average minimum distance to the optimal Pareto front, and HV to measure the obtained solutions' diversity. Tables 4.14 And 4.15 illustrate the resulted measures for IGD and HV respectively. All results were obtained using Python 3.10 scripts on i7 PC with 1.80GHz (8 logical processors) and 16 GB RAM.

Table 4.14: IGD indicator values for the ten benchmark functions (UF1-UF10)

Function	**DGEA**	**ARMOEA**	**AGEMOEA**	**CMOPSO**	**HDMOPSO**	**E NSGAII-MOGWO**
UF1	0.6229	0.1169	0.1120	0.0894	0.1028	**0.0620**
UF2	0.1703	0.0788	0.0705	0.0628	**0.0427**	0.0690
UF3	0.5740	0.4333	0.4287	0.3793	0.2261	**0.0783**
UF4	0.1223	0.0818	0.0819	0.1141	**0.0562**	0.0586
UF5	2.9718	0.6769	0.7250	0.8685	0.4720	**0.0381**
UF6	2.5695	0.5098	0.5259	0.3923	0.5770	**0.0632**
UF7	0.7351	0.2165	0.1656	0.1598	0.0547	**0.0377**
UF8	0.7439	0.3433	0.3565	0.6347	0.2989	**0.1697**
UF9	0.7665	0.4557	0.4864	0.9199	**0.1384**	0.1547
UF10	4.6292	1.0200	1.1947	4.3317	0.5134	**0.1617**

The results for IGD indicator are the average value of 30 runs of the E NSGAII-MOGWO using the 10 benchmarking functions. The entries with bold font-face in Table 4.? are the ones with best average value for each function. The UF1-10 are complex and hard to converge problems, the overall performance was 70% better than other compared approaches. For bi-objective function (UF1-7) the hit score was 5/7 better and 2/3 for the more complex tri-objective problems (UF8-10).

Although, HDMOPSO was better in two of the bi-objective benchmarking functions (UF2 and UF4) but the difference was (-0.0263 and -0.0023) respectively, and for tri-objective function was (-0.0163). On the other hand, the proposed algorithm outperforms HDMOPSO on the remaining functions (UF1, 3, 5, 6, 7, 8, 10) by (0.0408, 0.1478, 0.4339, 0.5138, 0.017, 0.1292, and 0.3517) respectively. The results show that the proposed algorithm can simulate the optimal Pareto front more efficiently than the opponent evolutionary and swarm algorithms. Thus, the proposed algorithm shows better convergence to the optimal Pareto front than the other compared algorithms, because of using MOGWO and elitism to have better convergence and overcome the limitation of NSGA-II of fast convergence as proved from the obtained results.

Table 4.15: HV indicator values for the ten benchmark functions (UF1-UF10)

Function	**DGEA**	**ARMOEA**	**AGEMOEA**	**CMOPSO**	**HDMOPSO**	**E NSGAII-MOGWO**
UF1	0.0961	0.5583	0.5582	0.5815	**0.5835**	0.5577
UF2	0.5107	0.6311	0.6362	0.6447	**0.6734**	0.5434
UF3	0.1180	0.2064	0.2132	0.2764	0.4430	**0.5058**
UF4	0.2771	0.3321	0.3337	0.2863	0.3697	**0.5145**
UF5	0.0000	0.0303	0.0258	0.0149	0.0197	**0.4621**
UF6	0.0000	0.0481	0.0377	0.1498	0.0131	**0.8716**
UF7	0.0184	0.3626	0.3874	0.4192	**0.5064**	0.4754
UF8	0.0216	0.3004	0.2923	0.0094	0.3529	**0.3995**
UF9	0.0656	0.3146	0.2950	0.0126	**0.6343**	0.4994
UF10	0.0000	0.0000	0.0000	0.0000	0.3191	**0.4790**

Additionally, the results obtained from the HV indicator clearly shows that the proposed algorithm successfully enhance the diversity of the objective space while searching for the optimal solution. Generally, the Hyper-volume (HV) indicator is used the diversity of the obtained solution across the objective space, which is a maximizing problem. The results obtained from this indicator shows better diversity performance in 6/10 of the used problems with (4/7) hit score for bi-objective functions and (2/3) for tri-objective functions. The HDMOPSO has better performance in function (UF1, 2, and 7) with differences (-0.0258, -0.13, and -0.031) respectively. While in the tri-objective functions the difference was -0.1349 for UF9. The proposed algorithm has better performance than all other algorithms in functions (UF3, 4, 5, 6, 8, 10) with differences of (0.0628, 0.1448, 0.4424, 0.8585, 0.0466, and 0.1599) respectively against the HDMOSPO. The better diversity in the optimal solution of the proposed algorithm came from the inherited advantage of using MOGWO and the deployed strategy of mitigation of stagnation from the adaptive opposition-based learning in changing the elite population to compensate with the current crowding distance of the intermediate solution during algorithm iterations.

In conclusion, the presented results proved that the proposed algorithm succeeded to overcome the fast convergence and local optima traps usually accompany the evolutionary optimization algorithms when used in complex and challenging problems as IoT-Fog resources allocation.

The next section presents the scenarios conducted to apply the proposed algorithm in the IoT-Fog environment and the results obtained from these experiments.

4.4 IoT-Environment Experiments Results

The applicability of the proposed system was experimented with the python fog simulator YAFS. The experiments follow the configuration parameters described in Table 4.16. The experiment deployed various number of applications running in the IoT-Fog environment. Furthermore, those applications use different number of dependent modules. The experiments also implement varying number of IoT clients, sensors, and sinks.

Table 4.16: Simulation Experiments Configuration Variables

#	Variable	Value(s)
Simulation Environment Parameters		
1	Simulation Time	10000s
2	Number of Apps	1-20
3	Number of Services	1-20 Service/App
4	Number of Fog Nodes	100 – 200
5	Number of Sensors	100-300
6	Number of actuators	100-300
7	Number of clients	200-400
8	Clients' mobility	Directional
9	Number of clouds	1
Cloud Parameters		
10	RAM Size	40GB
11	CPU	4.4G IPT
12	Bandwidth	1Gbips

13	Idle energy consumption	16*83.25 mW
14	Busy energy consumption	16*103 mW
15	Processing cost	0.00005 $G
16	Ram usage cost	0.016 $G
17	BW cost	0.0005 $G
18	Link latency	10ms
Fog Parameters		
19	RAM Size	4GB
20	CPU	Uniform (1.0-2.8) G IPT
21	Bandwidth	100Mbips
22	Idle energy consumption	83.4333 mW
23	Busy energy consumption	107.339 mW
24	Processing cost	0.00015 $G
25	Ram usage cost	0.000005 $G
26	BW cost	0.0001 $G
27	Link latency	5ms
Client parameters		
28	RAM Size	1GB
29	CPU	1G IPT
30	Bandwidth	Uniform (100-10000) kbips
31	Idle energy consumption	82.44 mW
32	Busy energy consumption	87.53 mW
33	Processing cost	0
34	Ram usage cost	0
35	BW cost	0
36	Link latency	5ms

4.4.1 Network Time Analysis

4.4.2 Network Latency Time

Figure 4.25 below shows the performance of the proposed algorithm in terms of latency time of the deployed application. The first entry for both figures is when the application’s modules are all placed on the cloud, which considered as the reference point for the algorithm performance calculations.

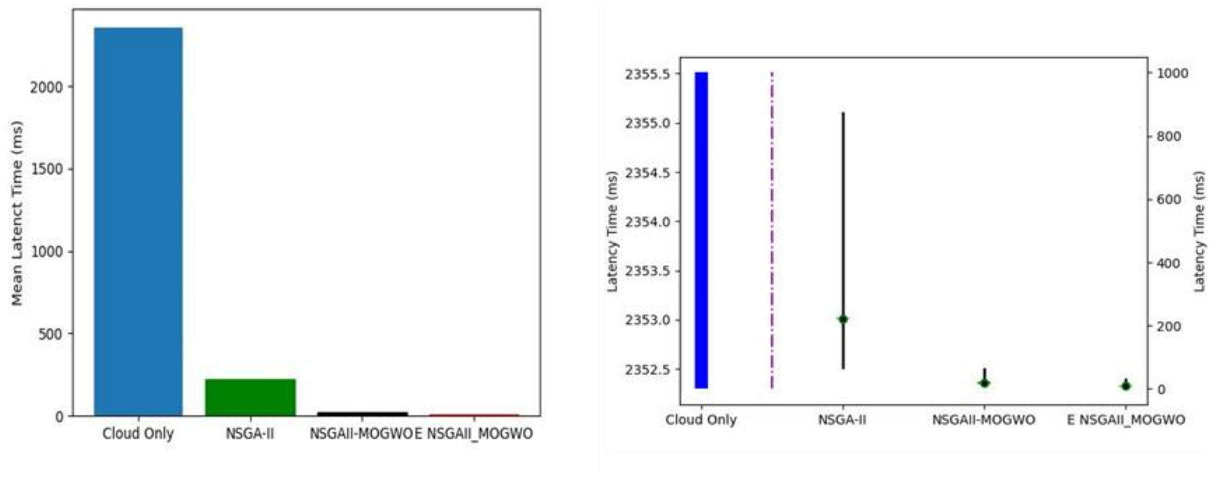

Figure 4.25: Network Mean Latency Time (ms)

The figure shows the differences in latency time for services traversing throughout the environment. The base value of comparison is the values obtained from deploying all services on the cloud (worst case scenario). From the figures, it is clear that deploying applications in Fog nodes can significantly reduce the latency of service responses. Furthermore, utilizing evolutionary algorithms can further reduce latency time. The mean latency time obtained from the cloud was about (2354(±1.6067) ms), when comparing this value with latencies (222(±1.2749) ms, 19(±1.6559) ms, 9(±1.3475) ms) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively, we notice that the reduction is in terms of magnitude. The percentage of reduction was (10.6%, 124.5%, 255.6%) and the ratio of the obtained latency to the latency from cloud-only strategy was (0.0943, 0.008, 0.0039) for the deployed algorithms respectively.

4.4.2.1 Request Waiting Time

The values obtained for the requests mean waiting time and the statistics for the obtained data as box-plot are illustrated in Figure 4.26.

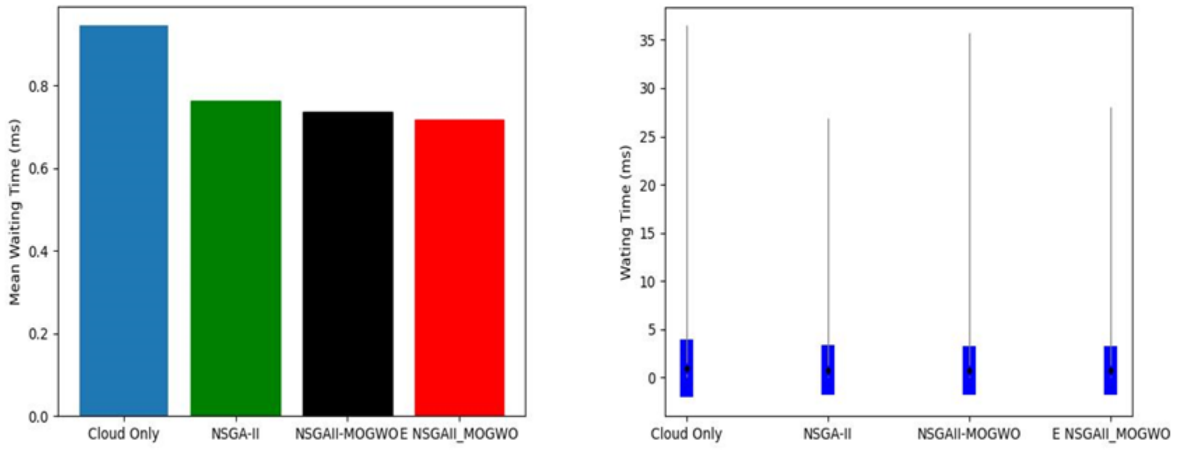

Figure 4.26: Request Mean Waiting Time (ms)

The significantly of the waiting times gives us a view for the network behavior during saturating times. Generally, the illustrated values show less mean waiting time for requests passing through an environment managed by the E NSGAII-MOGWO algorithm. The mean waiting time values are (0.9454(±3.0114), 0.7640(±2.5729), 0.7354(±2.5242), 0.7188(±2.4900)) for cloud-only, NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The percentage of difference between cloud placement and the algorithms placement are (19.18%, 22.21%, 23.97%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. This percentage concludes that another benefit from the proposed algorithm is the reduction in request waiting times.

4.4.2.2 Message Response Time and Total Message Response Time

The message response time is the time from the message entering the queue until it transmitted to the next hop, this calculation concerns only compute nodes not the intermediate routing path nodes. Figure 4.27 illustrates the response times obtained from simulations and representing the nodes mean response time and their summary statistics. On the other hand, the total message response time is from the message been transmitted from the previous hop till it sent out to the next hop. The total message response time is illustrated in Figure 4.28.

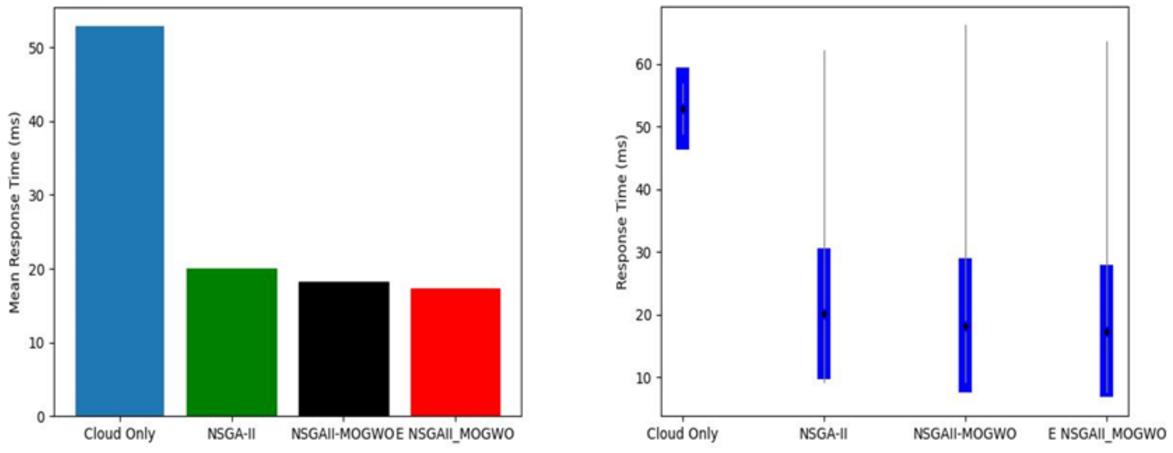

Figure 2.27: Nodes Mean Response times (ms)

The outcomes for the message mean response time were (52.8782(±6.5379), 20.0441 (±10.4649), 18.1964(±10.7007), 17.3433(±10.5819)) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The percentage of difference between the only-cloud placement strategy gained from the evaluated placement algorithms were (-62.09%, -65.59%, -67.20%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively.

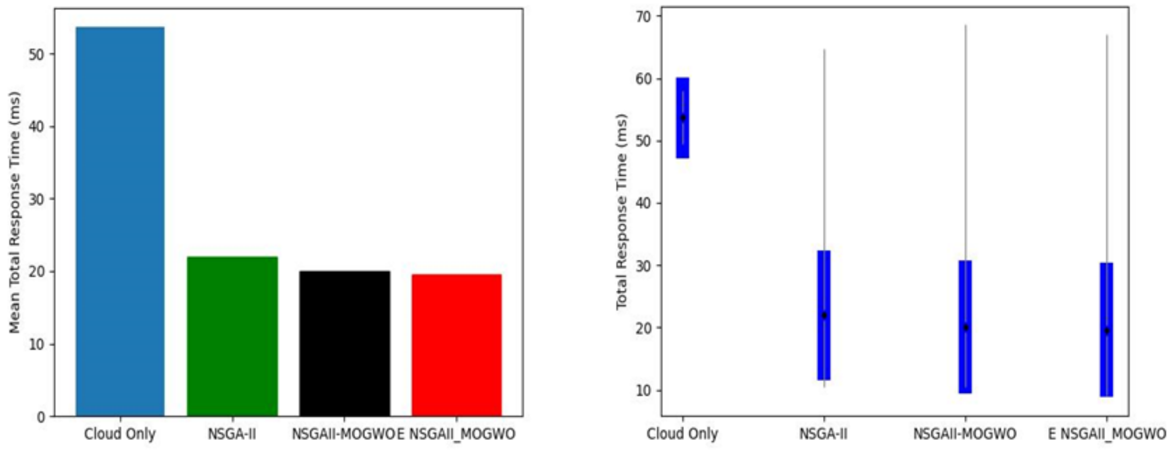

Figure 4.28: Total Message Response Time (ms)

On the other hand, the outcomes for the message mean total response time were (53.6509(±6.5276), 21.9350(±10.4369), 20.0385(±10.7183), 19.5623(±10.7812)) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The percentage of difference between the only-cloud placement strategy gained from the evaluated placement algorithms were (-59.12%, -62.65%, -63.54%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively.

Both figures demonstrate the effect of the waiting time on the performance of the network in terms of the time takes to be completely processed (response time) and the time takes the next hop to receive the resultant response. The proposed algorithm scores a reduction of response time of (-67.20% and -63.54%) for message response time and total message response time respectively. The gained performance is due to the semi-optimal placement of resources provided by the proposed algorithm.

4.4.2.3 Application Loops Latency Time

Figure 4.29 illustrates the effect of application loop latency time with the number of dependent modules. Generally, the figure indicates a proportional relation between number of engaged modules (loop length) and latency time as the number of computations increases also the time required for a message to traverse the whole modules path. The cloud only placement strategy has very high latency time comparing with other implemented algorithms. On the far end, the proposed algorithm scored the least latencies, as detailed below.

The plots presented in Figure 4.29 clearly shows two groups of behavior, the first one is a steep increase in loop latency time with number of modules which represented by NSGA-II, NSGAII-MOGWO and cloud only placement, despite the difference in scale. The other trend is a steady moderate increase in loop latency time represented by the proposed algorithm. Figure 4.29 shows a box-plot for the four approaches which shows that the proposed algorithm has minimum average and standard deviations comparing to other compared approaches. The obtained results from the application loop latency time were (68189.4849(±53493.8027), 6235.4165(±4681.6239), 5909.3633(±4197.7612), 3159.4258(±2028.0579)) for Cloud-Only, NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The percentage of difference between cloud placement and other implemented algorithms were (-90.87, -91.33%, -95.37%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively.

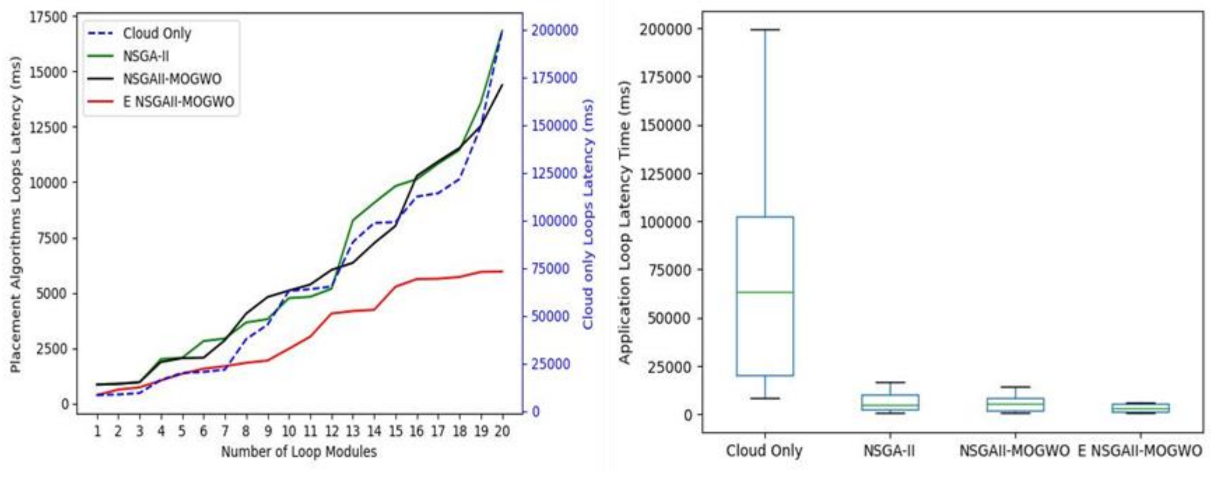

Figure 4.29: Application Loops Latency Time (ms)

4.4.3 CPU and Memory Usage Cost

One of the objectives of this paper is to reduce resources utilization. This analysis checks the usage of nodes CPU and RAM among the placed modules. Figure 4.30 shows the CPU and RAM utilizations obtained from the comparison algorithms in addition to the Cloud-only placement. The mean values obtained from the placement methods were (12023.1975(±15773.3071), 134.5020(±185.2745), 86.8183(±33.1803), 25.7522(±32.4814)) for Cloud-Only, NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The percentage of difference comparing with cloud-only (-98.88%, -99.28%, -99.79%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The performance metrics above are other indictors of the E NSGAII-MOGWO better placement strategy concerning processing costs and also scalability with number of applications deployed in the IoT-Fog environment.

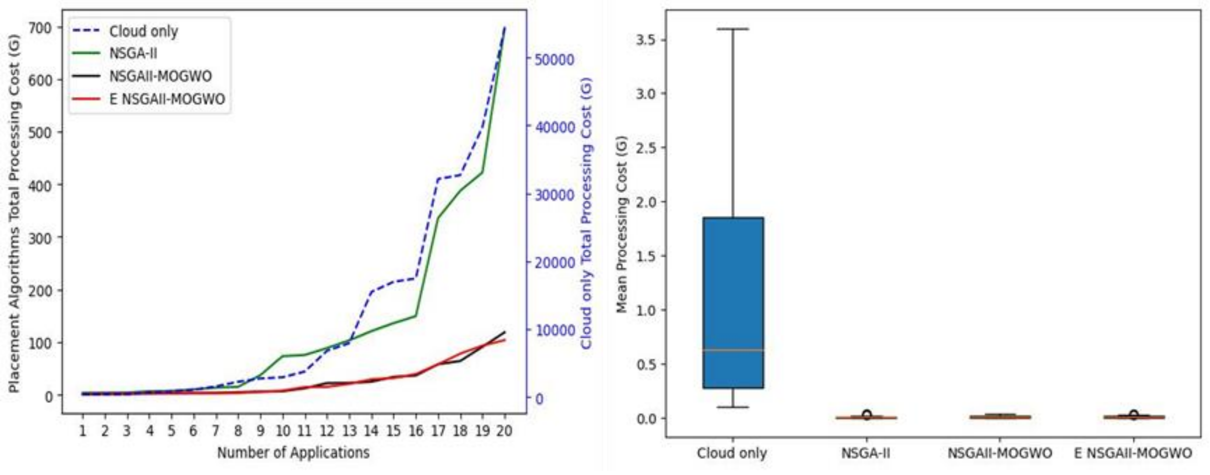

Figure 4.30: Total Processing Cost ($G)

4.4.4 Network Energy Consumption Analysis

Figure 4.31 shows the network energy consumption in Mega Joules/second. The mean energy consumption outcomes from the deployed algorithms were grouped as nodes idle energy, nodes busy energy, and the sum of idle and busy energy. For the mean idle energy consumption, we get (21.3577MJ/s, 2.3909MJ/s, 0.6315MJ/s, 0.4779Mj/s) for Cloud-only, NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. Furthermore, the ratio of differences between Cloud-only and other approaches were (-88.81%, -97.04%, -97.76%) for NSGA-II, NSGAII-MOGWO, and E NSGAII-MOGWO respectively. The results indicate that proposed algorithm succeeded in preserving energy consumptions by 97.76% than deploying applications only in the cloud.

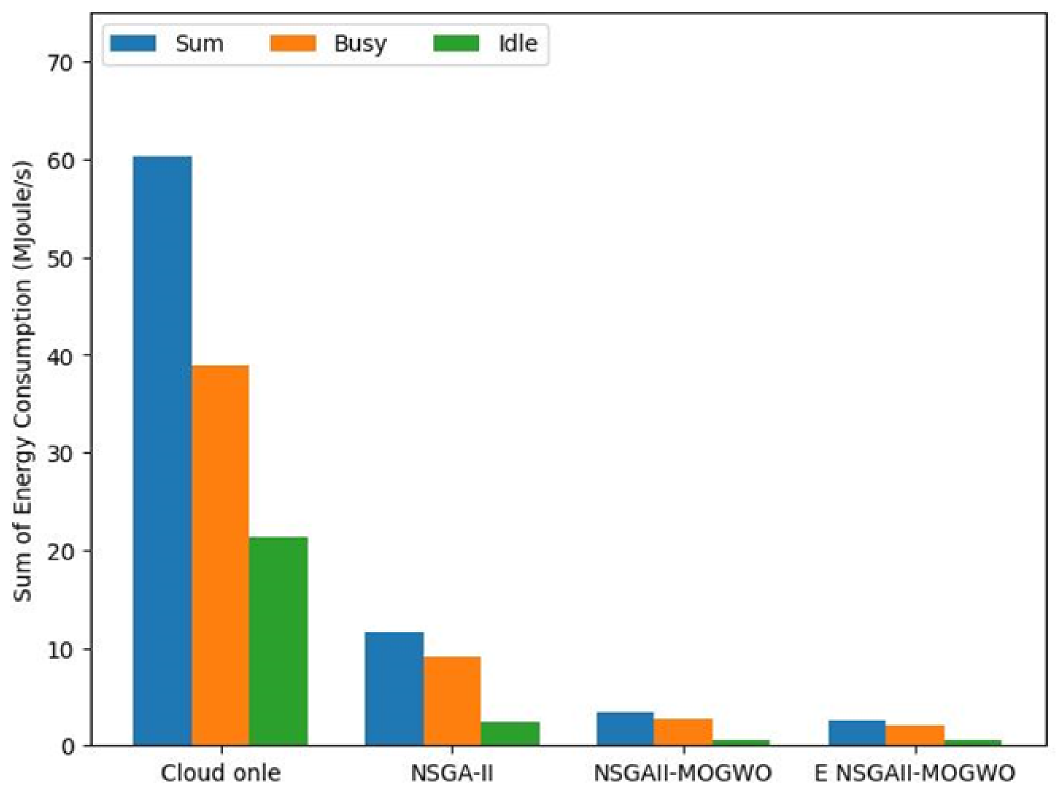

Figure 4.31: Network Energy Consumption

4.5 Conclusions

In this chapter we present the results obtained from testing the proposed algorithm. The tests were conducted with two goals; first to decide which selection method is better suited for the algorithm. Next, to observe the behavior of the algorithm when tested to optimize complex and hard to converge benchmarking functions from the CEC09 set. The outcomes show that E NSGAII-MOGWO algorithm has superior performance and the applicable selection method is the Tournament selection method.

Furthermore, the proposed algorithm tested on a simulated Io-Fog environment with different settings. All the simulation experiments prove that E NSGAII-MOGWO enhance the performance of the environment and ultimately the QoS of the IoT-Fog environment.

CHAPTER Five
Conclusions and Future Works

5.1 Conclusions

The main goal of this work was to design an optimization algorithm to enhance the QoS in the IoT-Fog environment. The proposed algorithm succeeded in minimizing the environment's costs such as energy consumption, the monetary cost of using the network resources, the total utilization of network workload, and the application's total latency. The work was implemented using two well-known meta-heuristic algorithms NSGA-II and MOGWO to design the final hybrid algorithm E NSGAII-MOGWO. In this work, elitism was introduced to the MOGWO part of the proposed algorithm to enhance the diversity of the solutions generated by NSGA-II. From the conducted simulations, this work proved that this algorithm is applicable for complex problems like resource allocation in the IoT-Fog environment. This work achieved the maximizing of QoS by minimizing the given objectives to reach a resource placement and distribution near-optimal.

The proposed algorithm was validated using multi-objective evolutionary algorithms' benchmarking and performance indicators. Furthermore, the proposed algorithm was compared to the related papers and proved its superiority over the implemented algorithms.

The most important conclusions that arise during the path of this work design and implementation include:

1. When building a hybrid algorithm from only the base algorithms, the performance of the resulting hybridization improved. On the other hand, the drawbacks of the base algorithms might be inherently persistent. This was seen in the NSGAII-MOGWO implementation with the poor maintenance of the Pareto front when experimented with the benchmarking functions.
2. The MOGWO algorithm has the potential for further improvements to be used in solving resource allocation problems. This can be seen in the case of adding elitism to enhance the performance of NSGAII-MOGWO.
3. Using opposition-based elitism with MOGWO add the ability to generate better and more diverse solutions that cover a larger area in the objective space

of the problem. This intuitively flows from that if this placement is not efficient then the opposite placement might speed up convergence to the optimal solution.

4. The proposed E NSGAII-MOGWO overperforms the NSGAII-MOGWO when adding an adaptive method to alter the number of migrated individuals to overcome the stagnation (local minima) of the population.
5. NSGA-II algorithm generates a set of solutions in the Pareto front (Pareto set). If this behaviour is conserved in the proposed algorithm, then there would be a need for further methods to select the best solution. But the use of the MOGWO algorithm in the core of the algorithm relaxes that issue by yielding only the alpha solution.

5.2 Future Works

This work contributes to the field of resource allocation in the IoT-Fog environment. There are some ideas to be as future works.

1. The implemented are both evolutionary algorithms, and other types of recently trended approaches like machine learning.
2. Studying the effect of implementing the proposed algorithm in other environments like VANET.
3. Implement quality-of-experience (QoE) metrics like reliability and availability in the objective functions.

References

[1] R. Yezdani and S. M. K. Quadri, “Power and Performance Issues and Management Approaches in Cloud Computing,” in *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, IEEE, Dec. 2022, pp. 2112–2120. doi: 10.1109/ICAC3N56670.2022.10074073.

[2] M. Sumathi, N. Vijayaraj, S. P. Raja, and M. Rajkamal, “HHO-ACO hybridized load balancing technique in cloud computing,” *International Journal of Information Technology*, vol. 15, no. 3, pp. 1357–1365, Mar. 2023, doi: 10.1007/s41870-023-01159-0.

[3] A. Rani, V. Prakash, and M. Darbari, “Fog Computing Paradigm with Internet of Things to Solve Challenges of Cloud with IoT,” 2022, pp. 72–84. doi: 10.1007/978-3-031-23724-9_7.

[4] R. Das and M. M. Inuwa, “A review on fog computing: Issues, characteristics, challenges, and potential applications,” *Telematics and Informatics Reports*, vol. 10, p. 100049, Jun. 2023, doi: 10.1016/j.teler.2023.100049.

[5] W. Bai and Y. Wang, “Jointly Optimize Partial Computation Offloading and Resource Allocation in Cloud-Fog Cooperative Networks,” *Electronics (Basel)*, vol. 12, no. 15, p. 3224, Jul. 2023, doi: 10.3390/electronics12153224.

[6] A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, “Fog computing for next-generation Internet of Things: Fundamental, state-of-the-art and research challenges,” *Comput Sci Rev*, vol. 48, p. 100549, May 2023, doi: 10.1016/j.cosrev.2023.100549.

[7] H. Tran-Dang and D.-S. Kim, “A Survey on Matching Theory for Distributed Computation Offloading in IoT-Fog-Cloud Systems: Perspectives and Open Issues,” *IEEE Access*, vol. 10, pp. 118353–118369, 2022, doi: 10.1109/ACCESS.2022.3219427.

[8] S. Tang, “Performance Modeling and Optimization for a Fog-Based IoT Platform,” *IoT*, vol. 4, no. 2, pp. 183–201, Jun. 2023, doi: 10.3390/iot4020010.

[9] N. Morkevičius, A. Liutkevičius, and A. Venčkauskas, “Multi-Objective Path Optimization in Fog Architectures Using the Particle Swarm Optimization Approach,” *Sensors*, vol. 23, no. 6, p. 3110, Mar. 2023, doi: 10.3390/s23063110.

[10] H. Suleiman, “A Cost-Aware Framework for QoS-Based and Energy-Efficient Scheduling in Cloud–Fog Computing,” *Future Internet*, vol. 14, no. 11, p. 333, Nov. 2022, doi: 10.3390/fi14110333.

[11] S. Petchrompo, D. W. Coit, A. Brintrup, A. Wannakrairot, and A. K. Parlikad, “A review of Pareto pruning methods for multi-objective optimization,” *Comput Ind Eng*, vol. 167, p. 108022, May 2022, doi: 10.1016/j.cie.2022.108022.

[12] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, “An Indicator-Based Multiobjective Evolutionary Algorithm With Reference Point Adaptation for Better Versatility,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609–622, Aug. 2018, doi: 10.1109/TEVC.2017.2749619.

[13] X. Zhang, X. Zheng, R. Cheng, J. Qiu, and Y. Jin, “A competitive mechanism based multi-objective particle swarm optimizer with fast convergence,” *Inf Sci (N Y)*, vol. 427, pp. 63–76, Feb. 2018, doi: 10.1016/j.ins.2017.10.037.

[14] A. Panichella, “An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA: ACM, Jul. 2019, pp. 595–603. doi: 10.1145/3321707.3321839.

[15] R. Xu *et al.*, “Improved Particle Swarm Optimization Based Workflow Scheduling in Cloud-Fog Environment,” 2019, pp. 337–347. doi: 10.1007/978-3-030-11641-5_27.

[16] Y. Zhuang and H. Zhou, “A Hyper-Heuristic Resource Allocation Algorithm for Fog Computing,” in *Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence*, New York, NY, USA: ACM, May 2020, pp. 194–199. doi: 10.1145/3390557.3394321.

[17] A. Anu and A. Singhrova, “PRIORITIZED GA-PSO ALGORITHM FOR EFFICIENT RESOURCE ALLOCATION IN FOG COMPUTING,” *Indian Journal of Computer Science and Engineering*, vol. 11, no. 6, pp. 907–916, Dec. 2020, doi: 10.21817/indjcse/2020/v11i6/201106205.

[18] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, and Y. Hu, “A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services,” *IEEE Access*, vol. 8, pp. 65085–65095, 2020, doi: 10.1109/ACCESS.2020.2983742.

[19] Y. Zhang, W. Zhang, K. Peng, D. Yan, and Q. Wu, “A novel edge server selection method based on combined genetic algorithm and simulated

annealing algorithm," *Automatika*, vol. 62, no. 1, pp. 32–43, Jan. 2021, doi: 10.1080/00051144.2020.1837499.

[20] N. Potu, C. Jatoth, and P. Parvataneni, "Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments," *Concurr Comput*, vol. 33, no. 23, Dec. 2021, doi: 10.1002/cpe.6163.

[21] B. V. Natesha and R. M. R. Guddeti, "Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," *Journal of Network and Computer Applications*, vol. 178, p. 102972, Mar. 2021, doi: 10.1016/j.jnca.2020.102972.

[22] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. Ryan, and K.-K. R. Choo, "An Automated Task Scheduling Model Using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2294–2308, Oct. 2022, doi: 10.1109/TCC.2020.3032386.

[23] R. Sing, S. K. Bhoi, N. Panigrahi, K. S. Sahoo, N. Jhanjhi, and M. A. AlZain, "A Whale Optimization Algorithm Based Resource Allocation Scheme for Cloud-Fog Based IoT Applications," *Electronics (Basel)*, vol. 11, no. 19, p. 3207, Oct. 2022, doi: 10.3390/electronics11193207.

[24] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "An Evolutionary Multi-objective Optimization Technique to Deploy the IoT Services in Fog-enabled Networks: An Autonomous Approach," *Applied Artificial Intelligence*, vol. 36, no. 1, Dec. 2022, doi: 10.1080/08839514.2021.2008149.

[25] Y. Ramzanpoor, M. Hosseini Shirvani, and M. Golsorkhtabaramiri, "Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure," *Complex & Intelligent Systems*, vol. 8, no. 1, pp. 361–392, Feb. 2022, doi: 10.1007/s40747-021-00368-z.

[26] S. Harika and B. C. Krishna, "Multi-Objective Optimization-Oriented Resource Allocation in the Fog Environment," *International Journal of Information Technology and Web Engineering*, vol. 17, no. 1, pp. 1–25, Apr. 2022, doi: 10.4018/IJITWE.297969.

[27] M. Iyapparaja, N. Khalaf Alshammari, M. Sathish Kumar, S. Siva Rama Krishnan, and C. Lal Chowdhary, "Efficient Resource Allocation in Fog Computing Using QTCS Model," *Computers, Materials & Continua*, vol. 70, no. 2, pp. 2225–2239, 2022, doi: 10.32604/cmc.2022.015707.

[28] C. He, R. Cheng, and D. Yazdani, “Adaptive Offspring Generation for Evolutionary Large-Scale Multiobjective Optimization,” *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 2, pp. 786–798, Feb. 2022, doi: 10.1109/TSMC.2020.3003926.

[29] X. Shu, Y. Liu, Q. Zhang, and M. Yang, “A Novel Multiobjective Particle Swarm Optimization Combining Hypercube and Distance,” *Sci Program*, vol. 2022, pp. 1–21, Apr. 2022, doi: 10.1155/2022/9448419.

[30] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, “Multi-Objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing,” *IEEE Access*, vol. 11, pp. 20635–20646, 2023, doi: 10.1109/ACCESS.2023.3241240.

[31] F. A. Saif, R. Latip, Z. M. Hanapi, M. A. Alrshah, and S. Kamarudin, “Workload Allocation Toward Energy Consumption-Delay Trade-Off in Cloud-Fog Computing Using Multi-Objective NPSO Algorithm,” *IEEE Access*, vol. 11, pp. 45393–45404, 2023, doi: 10.1109/ACCESS.2023.3266822.

[32] H. Reffad, A. Alti, and A. Almuhirat, “A Dynamic Adaptive Bio-Inspired Multi-Agent System for Healthcare Task Deployment,” *Engineering, Technology & Applied Science Research*, vol. 13, no. 1, pp. 10192–10198, Feb. 2023, doi: 10.48084/etasr.5570.

[33] A. Mohammadzadeh, M. Akbari Zarkesh, P. Haji Shahmohamd, J. Akhavan, and A. Chhabra, “Energy-aware workflow scheduling in fog computing using a hybrid chaotic algorithm,” *J Supercomput*, May 2023, doi: 10.1007/s11227-023-05330-z.

[34] S. M. Rajagopal, M. Supriya, and R. Buyya, “Resource Provisioning Using Meta-Heuristic Methods for IoT Microservices With Mobility Management,” *IEEE Access*, vol. 11, pp. 60915–60938, 2023, doi: 10.1109/ACCESS.2023.3281348.

[35] Y. Ramzanpoor, M. Hosseini Shirvani, and M. Golsorkhtabaramiri, “Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure,” *Complex & Intelligent Systems*, vol. 8, no. 1, pp. 361–392, Feb. 2022, doi: 10.1007/s40747-021-00368-z.

[36] S. M. Rajagopal, M. Supriya, and R. Buyya, “Resource Provisioning using Meta-heuristic Methods for IoT Microservices with Mobility Management,” *IEEE Access*, pp. 1–1, 2023, doi: 10.1109/ACCESS.2023.3281348.

[37] D.-N. Le, B. Seth, and S. Dalal, “A Hybrid Approach of Secret Sharingwith Fragmentation and Encryptionin Cloud Environment for SecuringOutsourced Medical Database:A Revolutionary Approach,” *Journal of Cyber Security and Mobility*, vol. 7, no. 4, pp. 379–408, 2018, doi: 10.13052/jcsm2245-1439.742.

[38] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, “Challenges and Opportunities in Edge Computing,” Sep. 2016.

[39] J. Hanen, Z. Kechaou, and M. Ben Ayed, “An enhanced healthcare system in mobile cloud computing environment,” *Vietnam Journal of Computer Science*, vol. 3, no. 4, pp. 267–277, Nov. 2016, doi: 10.1007/s40595-016-0076-y.

[40] S. Lu, J. Wu, J. Shi, P. Lu, J. Fang, and H. Liu, “A Dynamic Service Placement Based on Deep Reinforcement Learning in Mobile Edge Computing,” *Network*, vol. 2, no. 1, pp. 106–122, Feb. 2022, doi: 10.3390/network2010008.

[41] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, “Fog computing: from architecture to edge computing and big data processing,” *J Supercomput*, vol. 75, no. 4, pp. 2070–2105, Apr. 2019, doi: 10.1007/s11227-018-2701-2.

[42] R. Singh, S. V. Akram, A. Gehlot, D. Buddhi, N. Priyadarshi, and B. Twala, “Energy System 4.0: Digitalization of the Energy Sector with Inclination towards Sustainability,” *Sensors*, vol. 22, no. 17, p. 6619, Sep. 2022, doi: 10.3390/s22176619.

[43] S. v. Margariti, V. v. Dimakopoulos, and G. Tsoumanis, “Modeling and Simulation Tools for Fog Computing—A Comprehensive Survey from a Cost Perspective,” *Future Internet*, vol. 12, no. 5, p. 89, May 2020, doi: 10.3390/fi12050089.

[44] D. Aishwarya and R. I. Minu, “Real-Time Surveillance Video Analytics: A Survey on the Computing Infrastructures,” 2023, pp. 249–259. doi: 10.1007/978-981-19-5292-0_23.

[45] R. Guo, Y. Liu, Z. Lin, Y. Zheng, W. Zhang, and J. Yang, “Optimal Design of Resource Discovery-Allocation- Transmission-Offloading Strategy in Mobile Edge Computing,” in *2022 International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM)*, IEEE, Aug. 2022, pp. 6–12. doi: 10.1109/MLCCIM55934.2022.00010.

[46] A. Kapoor, “Assessment Of Intrusion Monitoring Methods Using Digital Forensics In A Cloud Infrastructure,” *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4267941.

[47] A. Ometov, O. L. Molua, M. Komarov, and J. Nurmi, “A Survey of Security in Cloud, Edge, and Fog Computing,” *Sensors*, vol. 22, no. 3, p. 927, Jan. 2022, doi: 10.3390/s22030927.

[48] Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo, and F. Li, “Mobility Support for Fog Computing: An SDN Approach,” *IEEE Communications Magazine*, vol. 56, no. 5, pp. 53–59, May 2018, doi: 10.1109/MCOM.2018.1700908.

[49] T. Nguyen Gia, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fog Computing Approach for Mobility Support in Internet-of-Things Systems,” *IEEE Access*, vol. 6, pp. 36064–36082, 2018, doi: 10.1109/ACCESS.2018.2848119.

[50] Y. Gong, “Improving Mobile Network Performance with Mobile Edge Computing,” in *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, IEEE, Nov. 2020, pp. 660–663. doi: 10.1109/ICIBA50161.2020.9277200.

[51] J. Lee, A. Mtibaa, and S. Mastorakis, “A Case for Compute Reuse in Future Edge Systems: An Empirical Study,” in *2019 IEEE Globecom Workshops (GC Wkshps)*, IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/GCWkshps45667.2019.9024587.

[52] H. Roger, S. Bhowmik, and T. Linn, “A Framework for Decentralized Parallel Complex Event Processing on Heterogeneous Infrastructures,” in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2021, pp. 190–196. doi: 10.1109/BigData52589.2021.9671516.

[53] P. G. Vinueza Naranjo, E. Baccarelli, and M. Scarpiniti, “Design and energy-efficient resource management of virtualized networked Fog architectures for the real-time support of IoT applications,” *J Supercomput*, vol. 74, no. 6, pp. 2470–2507, Jun. 2018, doi: 10.1007/s11227-018-2274-0.

[54] A. A.-N. Patwary *et al.*, “Authentication, Access Control, Privacy, Threats and Trust Management Towards Securing Fog Computing Environments: A Review,” Feb. 2020.

[55] A. A.-N. Patwary *et al.*, “Towards Secure Fog Computing: A Survey on Trust Management, Privacy, Authentication, Threats and Access Control,” *Electronics (Basel)*, vol. 10, no. 10, p. 1171, May 2021, doi: 10.3390/electronics10101171.

[56] V. K. M and L. C, “Cyber-Resilient Privacy Preservation and Secure Billing Approach for Smart Energy Metering Devices,” *International Journal of*

Engineering Trends and Technology, vol. 70, no. 9, pp. 337–345, Oct. 2022, doi: 10.14445/22315381/IJETT-V70I9P233.

[57] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms," *ACM Comput Surv*, vol. 52, no. 6, pp. 1–36, Nov. 2020, doi: 10.1145/3362031.

[58] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms," *ACM Comput Surv*, vol. 52, no. 6, pp. 1–36, Nov. 2020, doi: 10.1145/3362031.

[59] X. Liu and J. Liu, "A truthful online mechanism for virtual machine provisioning and allocation in clouds," *Cluster Comput*, vol. 25, no. 2, pp. 1095–1109, Apr. 2022, doi: 10.1007/s10586-021-03499-7.

[60] Y. Ren, Y. Liu, S. Ji, A. K. Sangaiah, and J. Wang, "Incentive Mechanism of Data Storage Based on Blockchain for Wireless Sensor Networks," *Mobile Information Systems*, vol. 2018, pp. 1–10, Aug. 2018, doi: 10.1155/2018/6874158.

[61] M. A. Alawami, A. R. Vinay, and H. Kim, "LocID: A Secure and Usable Location-Based Smartphone Unlocking Scheme Using Wi-Fi Signals and Light Intensity," *IEEE Internet Things J*, vol. 9, no. 23, pp. 24357–24372, Dec. 2022, doi: 10.1109/JIOT.2022.3189358.

[62] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms," *ACM Comput Surv*, vol. 52, no. 6, pp. 1–36, Nov. 2020, doi: 10.1145/3362031.

[63] D. Aishwarya and R. I. Minu, "Real-Time Surveillance Video Analytics: A Survey on the Computing Infrastructures," 2023, pp. 249–259. doi: 10.1007/978-981-19-5292-0_23.

[64] Y. Ren, Y. Liu, S. Ji, A. K. Sangaiah, and J. Wang, "Incentive Mechanism of Data Storage Based on Blockchain for Wireless Sensor Networks," *Mobile Information Systems*, vol. 2018, pp. 1–10, Aug. 2018, doi: 10.1155/2018/6874158.

[65] L. Ceragioli, L. Galletta, P. Degano, and D. Basin, "IFCIL: An Information Flow Configuration Language for SELinux," in *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, IEEE, Aug. 2022, pp. 243–259. doi: 10.1109/CSF54842.2022.9919690.

[66] R. Hanumantharaju *et al.*, "Secured Fog-Based System for Smart Healthcare Application," 2022, pp. 185–197. doi: 10.1007/978-981-19-5037-7_12.

[67] P. Nikolaou *et al.*, “Total Cost of Ownership Perspective of Cloud vs Edge Deployments of IoT Applications,” in *Computing at the EDGE*, Cham: Springer International Publishing, 2022, pp. 141–161. doi: 10.1007/978-3-030-74536-3_6.

[68] A. Alghamdi, A. Alzahrani, and V. Thayananthan, “EXECUTION TIME AND POWER CONSUMPTION OPTIMIZATION in FOG COMPUTING ENVIRONMENT,” *International Journal of Computer Science and Network Security*, vol. 21, no. 1, pp. 137–142, 2021.

[69] V. Kumar, A. A. Laghari, S. Karim, M. Shakir, and A. Anwar Brohi, “Comparison of Fog Computing &amp; Cloud Computing,” *International Journal of Mathematical Sciences and Computing*, vol. 5, no. 1, pp. 31–41, Jan. 2019, doi: 10.5815/ijmsc.2019.01.03.

[70] H. Atlam, R. Walters, and G. Wills, “Fog Computing and the Internet of Things: A Review,” *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 10, Apr. 2018, doi: 10.3390/bdcc2020010.

[71] H. K. Apat, B. Sahoo, and S. Mohanty, “A Quality of Service(QoS) aware Fog Computing model for intelligent (IoT) applications,” in *2021 19th OITS International Conference on Information Technology (OCIT)*, IEEE, Dec. 2021, pp. 267–272. doi: 10.1109/OCIT53463.2021.00061.

[72] M. Faraji-Mehmandar, S. Jabbehdari, and H. Haj Seyyed Javadi, “A proactive fog service provisioning framework for Internet of Things applications: An autonomic approach,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 11, Nov. 2021, doi: 10.1002/ett.4342.

[73] S. R. Hassan, I. Ahmad, A. U. Rehman, S. Hussen, and H. Hamam, “Design of Resource-Aware Load Allocation for Heterogeneous Fog Computing Environments,” *Wirel Commun Mob Comput*, vol. 2022, pp. 1–11, Jun. 2022, doi: 10.1155/2022/3543640.

[74] D. Lan, Y. Liu, A. Taherkordi, F. Eliassen, S. Delbruel, and L. Lei, “A federated fog-cloud framework for data processing and orchestration,” in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, New York, NY, USA: ACM, Mar. 2021, pp. 729–736. doi: 10.1145/3412841.3444962.

[75] S. K. Biswash, “Device and network driven cellular networks architecture and mobility management technique for fog computing-based mobile

communication system," *Journal of Network and Computer Applications*, vol. 200, p. 103317, Apr. 2022, doi: 10.1016/j.jnca.2021.103317.

[76] G. Cui, W. Zhang, Y. Xiao, L. Yao, and Z. Fang, "Cooperative Perception Technology of Autonomous Driving in the Internet of Vehicles Environment: A Review," *Sensors*, vol. 22, no. 15, p. 5535, Jul. 2022, doi: 10.3390/s22155535.

[77] H. Atlam, R. Walters, and G. Wills, "Fog Computing and the Internet of Things: A Review," *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 10, Apr. 2018, doi: 10.3390/bdcc2020010.

[78] S. R. Hassan, I. Ahmad, A. U. Rehman, S. Hussen, and H. Hamam, "Design of Resource-Aware Load Allocation for Heterogeneous Fog Computing Environments," *Wirel Commun Mob Comput*, vol. 2022, pp. 1–11, Jun. 2022, doi: 10.1155/2022/3543640.

[79] S. H. and N. V., "A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges," *ICT Express*, vol. 7, no. 2, pp. 162–176, Jun. 2021, doi: 10.1016/j.icte.2021.05.004.

[80] S. P. Singh, R. Kumar, A. Sharma, J. H. Abawajy, and R. Kaur, "Energy efficient load balancing hybrid priority assigned laxity algorithm in fog computing," *Cluster Comput*, vol. 25, no. 5, pp. 3325–3342, Oct. 2022, doi: 10.1007/s10586-022-03554-x.

[81] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments," *J Parallel Distrib Comput*, vol. 132, pp. 274–283, Oct. 2019, doi: 10.1016/j.jpdc.2018.07.003.

[82] A. R. Hameed, S. ul Islam, I. Ahmad, and K. Munir, "Energy- and performance-aware load-balancing in vehicular fog computing," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100454, Jun. 2021, doi: 10.1016/j.suscom.2020.100454.

[83] B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, "Orchestration in Fog Computing: A Comprehensive Survey," *ACM Comput Surv*, vol. 55, no. 2, pp. 1–34, Mar. 2023, doi: 10.1145/3486221.

[84] A. A. Alli and M. M. Alam, "The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications," *Internet of Things*, vol. 9, p. 100177, Mar. 2020, doi: 10.1016/j.iot.2020.100177.

[85] H. Chegini, R. K. Naha, A. Mahanti, and P. Thulasiraman, “Process Automation in an IoT–Fog–Cloud Ecosystem: A Survey and Taxonomy,” *IoT*, vol. 2, no. 1, pp. 92–118, Feb. 2021, doi: 10.3390/iot2010006.

[86] A. Alkhresheh, K. Elgazzar, and H. S. Hassanein, “DACIoT: Dynamic Access Control Framework for IoT Deployments,” *IEEE Internet Things J*, vol. 7, no. 12, pp. 11401–11419, Dec. 2020, doi: 10.1109/JIOT.2020.3002709.

[87] G. Javadzadeh and A. M. Rahmani, “Fog Computing Applications in Smart Cities: A Systematic Survey,” *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, Feb. 2020, doi: 10.1007/s11276-019-02208-y.

[88] H. Rashid Abdulqadir *et al.*, “A Study of Moving from Cloud Computing to Fog Computing,” *Qubahan Academic Journal*, vol. 1, no. 2, pp. 60–70, Apr. 2021, doi: 10.48161/qaj.v1n2a49.

[89] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, and O. Alfandi, “Fog Computing Framework for Internet of Things Applications,” in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*, IEEE, Sep. 2018, pp. 71–77. doi: 10.1109/DeSE.2018.00017.

[90] J. Yue and M. Xiao, “Coding for Distributed Fog Computing in Internet of Mobile Things,” *IEEE Trans Mob Comput*, vol. 20, no. 4, pp. 1337–1350, Apr. 2021, doi: 10.1109/TMC.2019.2963668.

[91] H. M. Fard, R. Prodan, and F. Wolf, “A Container-Driven Approach for Resource Provisioning in Edge-Fog Cloud,” 2020, pp. 59–76. doi: 10.1007/978-3-030-58628-7_5.

[92] H. Zahmatkesh and F. Al-Turjman, “Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview,” *Sustain Cities Soc*, vol. 59, p. 102139, Aug. 2020, doi: 10.1016/j.scs.2020.102139.

[93] Q. Xu and J. Zhang, “piFogBed: A Fog Computing Testbed Based on Raspberry Pi,” in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, IEEE, Oct. 2019, pp. 1–8. doi: 10.1109/IPCCC47392.2019.8958741.

[94] G. Katsikogiannis, D. Kallergis, Z. Garofalaki, S. Mitropoulos, and C. Douligeris, “A policy-aware Service Oriented Architecture for secure machine-to-machine communications,” *Ad Hoc Networks*, vol. 80, pp. 70–80, Nov. 2018, doi: 10.1016/j.adhoc.2018.06.003.

[95] D. Puthal, S. P. Mohanty, S. A. Bhavake, G. Morgan, and R. Ranjan, "Fog Computing Security Challenges and Future Directions [Energy and Security]," *IEEE Consumer Electronics Magazine*, vol. 8, no. 3, pp. 92–96, May 2019, doi: 10.1109/MCE.2019.2893674.

[96] L. Peng, A. R. Dhaini, and P.-H. Ho, "Toward integrated Cloud–Fog networks for efficient IoT provisioning: Key challenges and solutions," *Future Generation Computer Systems*, vol. 88, pp. 606–613, Nov. 2018, doi: 10.1016/j.future.2018.05.015.

[97] A. Molina Zarca, M. Bagaa, J. Bernal Bernabe, T. Taleb, and A. F. Skarmeta, "Semantic-Aware Security Orchestration in SDN/NFV-Enabled IoT Systems," *Sensors*, vol. 20, no. 13, p. 3622, Jun. 2020, doi: 10.3390/s20133622.

[98] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurr Comput*, vol. 32, no. 7, Apr. 2020, doi: 10.1002/cpe.5581.

[99] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, and M. Naeem, "Min-Min Scheduling Algorithm for Efficient Resource Distribution Using Cloud and Fog in Smart Buildings," 2019, pp. 15–27. doi: 10.1007/978-3-030-02613-4_2.

[100] Z. Ma *et al.*, "Towards Revenue-Driven Multi-User Online Task Offloading in Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1185–1198, May 2022, doi: 10.1109/TPDS.2021.3105325.

[101] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1130–1143, May 2019, doi: 10.1109/JSAC.2019.2906790.

[102] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "ULOOF: A User Level Online Offloading Framework for Mobile Edge Computing," *IEEE Trans Mob Comput*, vol. 17, no. 11, pp. 2660–2674, Nov. 2018, doi: 10.1109/TMC.2018.2815015.

[103] L. Wang, L. Jiao, T. He, J. Li, and H. Bal, "Service Placement for Collaborative Edge Applications," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 34–47, Feb. 2021, doi: 10.1109/TNET.2020.3025985.

[104] X. Xu *et al.*, “A computation offloading method over big data for IoT-enabled cloud-edge computing,” *Future Generation Computer Systems*, vol. 95, pp. 522–533, Jun. 2019, doi: 10.1016/j.future.2018.12.055.

[105] G. Peng, H. Wu, H. Wu, and K. Wolter, “Constrained Multiobjective Optimization for IoT-Enabled Computation Offloading in Collaborative Edge and Cloud Computing,” *IEEE Internet Things J*, vol. 8, no. 17, pp. 13723–13736, Sep. 2021, doi: 10.1109/JIOT.2021.3067732.

[106] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, “Efficient Dependent Task Offloading for Multiple Applications in MEC-Cloud System,” *IEEE Trans Mob Comput*, vol. 22, no. 4, pp. 2147–2162, Apr. 2023, doi: 10.1109/TMC.2021.3119200.

[107] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, “Distributed Deep Learning-based Offloading for Mobile Edge Computing Networks,” *Mobile Networks and Applications*, vol. 27, no. 3, pp. 1123–1130, Jun. 2022, doi: 10.1007/s11036-018-1177-x.

[108] R. Zhou *et al.*, “Online Task Offloading for 5G Small Cell Networks,” *IEEE Trans Mob Comput*, vol. 21, no. 6, pp. 2103–2115, Jun. 2022, doi: 10.1109/TMC.2020.3036390.

[109] B. Yang, X. Cao, J. Bassey, X. Li, T. Kroecker, and L. Qian, “Computation Offloading in Multi-Access Edge Computing Networks: A Multi-Task Learning Approach,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, IEEE, May 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761212.

[110] E. F. Maleki, L. Mashayekhy, and S. M. Nabavinejad, “Mobility-Aware Computation Offloading in Edge Computing Using Machine Learning,” *IEEE Trans Mob Comput*, vol. 22, no. 1, pp. 328–340, Jan. 2023, doi: 10.1109/TMC.2021.3085527.

[111] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, “Multiagent DDPG-Based Joint Task Partitioning and Power Control in Fog Computing Networks,” *IEEE Internet Things J*, vol. 9, no. 1, pp. 104–116, Jan. 2022, doi: 10.1109/JIOT.2021.3091508.

[112] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, “Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions,” *ACM Comput Surv*, vol. 54, no. 11s, pp. 1–38, Jan. 2022, doi: 10.1145/3513002.

[113] J. H. Holland, “Genetic Algorithms,” *Sci Am*, vol. 267, no. 1, pp. 66–73, 1992, [Online]. Available: http://www.jstor.org/stable/24939139

[114] J. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 33, no. 1. 1992.

[115] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies – A comprehensive introduction,” *Nat Comput*, vol. 1, no. 1, pp. 3–52, 2002, doi: 10.1023/A:1015059928466.

[116] D. Simon, “Biogeography-Based Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, Dec. 2008, doi: 10.1109/TEVC.2008.919004.

[117] N. Srinivas and K. Deb, “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms,” *Evol Comput*, vol. 2, no. 3, pp. 221–248, Sep. 1994, doi: 10.1162/evco.1994.2.3.221.

[118] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[119] T. Mekki, R. Jmal, L. Chaari, I. Jabri, and A. Rachedi, “Vehicular Fog Resource Allocation Scheme: A Multi-Objective Optimization based Approach,” in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, Jan. 2020, pp. 1–6. doi: 10.1109/CCNC46108.2020.9045361.

[120] Y. Sun, F. Lin, and H. Xu, “Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II,” *Wirel Pers Commun*, vol. 102, no. 2, pp. 1369–1385, Sep. 2018, doi: 10.1007/s11277-017-5200-5.

[121] A. J. Miriam, R. Saminathan, and S. Chakaravarthi, “Non-dominated Sorting Genetic Algorithm (NSGA-III) for effective resource allocation in cloud,” *Evol Intell*, vol. 14, no. 2, pp. 759–765, Jun. 2021, doi: 10.1007/s12065-020-00436-2.

[122] M. A. B. Al-Tarawneh, “Bi-objective optimization of application placement in fog computing environments,” *J Ambient Intell Humaniz Comput*, vol. 13, no. 1, pp. 445–468, Jan. 2022, doi: 10.1007/s12652-021-02910-w.

[123] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, IEEE, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[124] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Comput Intell Mag*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[125] D. Karaboga and B. Basturk, “Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems,” in *Foundations of Fuzzy Logic and Soft Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 789–798. doi: 10.1007/978-3-540-72950-1_77.

[126] X. L. Li, Z. J. Shao, and J. X. Qian, “Optimizing method based on autonomous animats: Fish-swarm Algorithm,” *Xitong Gongcheng Lilun yu Shijian/System Engineering Theory and Practice*, vol. 22, no. 11, 2002.

[127] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[128] A. Mucherino, O. Seref, O. Seref, O. E. Kundakcioglu, and P. Pardalos, “Monkey search: a novel metaheuristic search for global optimization,” in *AIP Conference Proceedings*, AIP, 2007, pp. 162–173. doi: 10.1063/1.2817338.

[129] X. Lu and Y. Zhou, “A Novel Global Convergence Algorithm: Bee Collecting Pollen Algorithm,” 2008, pp. 518–525. doi: 10.1007/978-3-540-85984-0_62.

[130] X.-S. Yang and Suash Deb, “Cuckoo Search via L&#x00E9;vy flights,” in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, IEEE, 2009, pp. 210–214. doi: 10.1109/NABIC.2009.5393690.

[131] Y. Shiqin, J. Jianjun, and Y. Guangxing, “A Dolphin Partner Optimization,” in *2009 WRI Global Congress on Intelligent Systems*, IEEE, 2009, pp. 124–128. doi: 10.1109/GCIS.2009.464.

[132] X.-S. Yang, “A New Metaheuristic Bat-Inspired Algorithm,” 2010, pp. 65–74. doi: 10.1007/978-3-642-12538-6_6.

[133] X. S. Yang, “Firefly algorithm, stochastic test functions and design optimisation,” *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, p. 78, 2010, doi: 10.1504/IJBIC.2010.032124.

[134] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, “A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource

Management in Fog Computing," *IEEE Access*, vol. 7, pp. 115760–115773, 2019, doi: 10.1109/ACCESS.2019.2924958.

[135] S. M. Hussain and G. R. Begh, "Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment," *J Comput Sci*, vol. 64, p. 101828, Oct. 2022, doi: 10.1016/j.jocs.2022.101828.

[136] B. V. Natesha and R. M. R. Guddeti, "Meta-heuristic Based Hybrid Service Placement Strategies for Two-Level Fog Computing Architecture," *Journal of Network and Systems Management*, vol. 30, no. 3, p. 47, Jul. 2022, doi: 10.1007/s10922-022-09660-w.

[137] X. Liu, Z. Qin, and Y. Gao, "Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, IEEE, May 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761385.

[138] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *Journal of Communications and Networks*, vol. 24, no. 1, pp. 83–98, Feb. 2022, doi: 10.23919/JCN.2021.000041.

[139] S.-S. Lee and S. Lee, "Resource Allocation for Vehicular Fog Computing Using Reinforcement Learning Combined With Heuristic Information," *IEEE Internet Things J*, vol. 7, no. 10, pp. 10450–10464, Oct. 2020, doi: 10.1109/JIOT.2020.2996213.

[140] H. Che, Z. Bai, R. Zuo, and H. Li, "A Deep Reinforcement Learning Approach to the Optimization of Data Center Task Scheduling," *Complexity*, vol. 2020, pp. 1–12, Aug. 2020, doi: 10.1155/2020/3046769.

[141] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed Deep Learning-based Offloading for Mobile Edge Computing Networks," *Mobile Networks and Applications*, vol. 27, no. 3, pp. 1123–1130, Jun. 2022, doi: 10.1007/s11036-018-1177-x.

[142] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1130–1143, May 2019, doi: 10.1109/JSAC.2019.2906790.

[143] R. Zhou *et al.*, “Online Task Offloading for 5G Small Cell Networks,” *IEEE Trans Mob Comput*, vol. 21, no. 6, pp. 2103–2115, Jun. 2022, doi: 10.1109/TMC.2020.3036390.

[144] G. Peng, H. Wu, H. Wu, and K. Wolter, “Constrained Multiobjective Optimization for IoT-Enabled Computation Offloading in Collaborative Edge and Cloud Computing,” *IEEE Internet Things J*, vol. 8, no. 17, pp. 13723–13736, Sep. 2021, doi: 10.1109/JIOT.2021.3067732.

[145] Z. Ma *et al.*, “Towards Revenue-Driven Multi-User Online Task Offloading in Edge Computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1185–1198, May 2022, doi: 10.1109/TPDS.2021.3105325.

[146] C. A. Coello Coello, “Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored,” *Front Comput Sci China*, vol. 3, no. 1, pp. 18–30, Mar. 2009, doi: 10.1007/s11704-009-0005-7.

[147] P. Ngatchou, A. Zarei, and A. El-Sharkawi, “Pareto Multi Objective Optimization,” in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, IEEE, pp. 84–91. doi: 10.1109/ISAP.2005.1599245.

[148] R. Mohanty, S. Suman, and S. K. Das, “Modeling the Axial Capacity of Bored Piles Using Multi-Objective Feature Selection, Functional Network and Multivariate Adaptive Regression Spline,” in *Handbook of Neural Computation*, Elsevier, 2017, pp. 295–309. doi: 10.1016/B978-0-12-811318-9.00016-8.

[149] E. Zitzler, K. Deb, and L. Thiele, “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,” *Evol Comput*, vol. 8, no. 2, pp. 173–195, Jun. 2000, doi: 10.1162/106365600568202.

[150] J. Knowles and D. Corne, “The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation,” in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, IEEE, 1999, pp. 98–105. doi: 10.1109/CEC.1999.781913.

[151] E. Zitzler, “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications,” *Ph.D. Thesis*, no. 30, 1999.

[152] S. Yang, “Genetic Algorithms with Elitism-Based Immigrants for Changing Optimization Problems,” in *Applications of Evolutinary Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 627–636. doi: 10.1007/978-3-540-71805-5_69.

[153] H. R. Tizhoosh, “Opposition-Based Learning: A New Scheme for Machine Intelligence,” in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06),* IEEE, 2005, pp. 695–701. doi: 10.1109/CIMCA.2005.1631345.

[154] W. K. Mashwani and A. Salhi, “Multiobjective evolutionary algorithm based on multimethod with dynamic resources allocation,” *Applied Soft Computing Journal*, vol. 39, 2016, doi: 10.1016/j.asoc.2015.08.059.

[155] Y. Liu, J. Wei, X. Li, and M. Li, “Generational Distance Indicator-Based Evolutionary Algorithm with an Improved Niching Method for Many-Objective Optimization Problems,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2916634.

[156] X. Cai, Y. Xiao, M. Li, H. Hu, H. Ishibuchi, and X. Li, “A Grid-Based Inverted Generational Distance for Multi/Many-Objective Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, 2021, doi: 10.1109/TEVC.2020.2991040.

[157] A. P. Guerreiro, C. M. Fonseca, and L. Paquete, “The Hypervolume Indicator: Computational Problems and Algorithms,” *ACM Computing Surveys*, vol. 54, no. 6. 2021. doi: 10.1145/3453474.

[158] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon, “Performance indicators in multiobjective optimization,” *Eur J Oper Res*, vol. 292, no. 2, pp. 397–422, Jul. 2021, doi: 10.1016/j.ejor.2020.11.016.

[159] H. Atlam, R. Walters, and G. Wills, “Fog Computing and the Internet of Things: A Review,” *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 10, Apr. 2018, doi: 10.3390/bdcc2020010.

[160] A. Yousefpour *et al.*, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, vol. 98, pp. 289–330, Sep. 2019, doi: 10.1016/j.sysarc.2019.02.009.

[161] I. Lera, C. Guerrero, and C. Juiz, “YAFS: A Simulator for IoT Scenarios in Fog Computing,” *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[162] P. Sanju, “Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks,” *Journal of Engineering Research*, p. 100122, Jun. 2023.

الملخص

يعد تخصيص الموارد للتطبيقات مشكلة صعبة وحساسة في البيئة الضبابية وانترنت الاشياء. عادة مايكون تخصيص الموارد بصورة فعالة على العقد الضبابية مفيدا لمستخدمي التطبيقات لحصولهمعلى جودة خدمة عالية

تتبنت هذه الدراسة خوارزمية هجينة لمعالجة مشكلة تخصيص الموارد. تتكون الخوارزمية الهجينة من خوارزميتين اساسيتين مشهورتين وهما NSGA-II و .MOGWO مرت الخوارزمية المقترحة بدورين، الاول هو خوارزمية NSGAII-MOGW وهو عبارة عن هجين بمعالجة متوازية للخوارزميتين الاساسية. الطور الثاني هو خوارزمية هجينة محسنة E NSGAII-MOGWO باضافة طريقة نخبوية مركبة لتحسين الخوارزمية المقترحة الاولى تم اختبار اداء الخوارزميتين المقترحتين باستخدام عشرة دوال اختبار وستة طرق اختيار وهي (Roulette Wheel, Tournament, Stochastic Universal Sampling, Boltzmann, Ranking, and Linear Ranking).

اظهرت المقارنة تفوقا لخوارزمية E NSGAII-MOGWO عند اسخدام طريقة بنسبة منافسة 53.33% ومحصلة افضلية 0.7868، 0.6874، و 0.6563 على الخوارزميات MOGWO, NSGA-II, NSGAII-MOGWOعلى الرتيب. وهذا يشير الى صلاحية هذه الخوارزمية لحل مشكلات معقدة مثل تخصيص الموارد.

اضافة الى هذا، تمت مقارنة خوارزمية E NSGAII-MOGWO مع خمسة خوارزميات امثلية متعددة الاهداف. اضهرت النتائج تفوقا بمقدار 70% بالنسبة لمؤشرIGD ، و 60% بالنسبة لمؤشر Hypervolumeلدوال الاختبار العشرة.

تم استخدام بيئة محاكاة للتأكد من ملائمة الحلول التي تولدها الخوارزمية. عند اجراء المحاكاة استطاعت الخوارزمية المقترحة من تقليل وقت الانتضار، وقت الاستجابة، والوقت اللازم لاكمال مسار برمجي بنسب 23.97%، 67.2%، 95.27% على الترتيب.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

تحسين تخصيص الموارد لتحسين جودة الخدمة في بيئة إنترنت الأشياء و الحوسبة الضبابية

اطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات -جامعة بابل كجزء من متطلبات

نيل درجة الدكتوراه في تكنولوجيا المعلومات- شبكات المعلومات

من قبل

بلاسم علاوي حسين

بإشراف
أ.د. سكينة حسن هاشم

1444 هـ **2023 م**