Multitiered Worker-Oriented Resource Allocation at the Extreme Edge

by

Marah De'bas

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
September 2022

Dedication

FOR THE SAKE OF ALLAH

to my parents, and beloved family

Abstract

Democratizing Edge Computing (EC) by leveraging the prolific yet underutilized computational resources of IoT devices, also referred to as Extreme Edge Devices (EEDs), has gained significant momentum lately. In such edge computing paradigms, fair resource allocation is a major concern. However, fairness is typically considered from the requester's perspective, whereas fairness for workers is mostly overlooked. In this thesis, we propose the Multitiered Worker-oriented Resource Allocation (MWORA) framework. MWORA aims to minimize the drop rate and task response delay while enabling fair resource allocation that maintains a specific satisfactory profit for workers. Such a satisfactory profit is maintained to prevent workers from leaving the system and ensure their recurrent subscription to the service provider. MWORA also accounts for the fact that EEDs are user-owned devices and are thus subject to a dynamic user access behavior, which can affect the level of computational resources endowed by workers. MWORA considers this by enabling multitiered computational resources to be granted by each worker depending on the price of the allocated task. MWORA formulates the resource allocation problem as an Integer Linear Program (ILP). We also propose the MWORA-Weighted Sum (MWORA-WS) scheme to derive an analytical solution using the Karush–Kuhn–Tucker (KKT) conditions and Lagrangian analysis. Extensive simulations show that MWORA outperforms prominent

resource allocation schemes in terms of the average response delay, service capacity, worker satisfaction ratio, and fairness. In addition, it is shown that MWORA-WS closely approaches the optimal solution rendered by MWORA.

Co-Authorship

Journal Articles

- Marah De'bas, Sara A. Elsayed, and Hossam S. Hassanein, "Worker-Oriented Fair Resource Allocation in Extreme Edge Computing," 2022, (Journal paper in preparation)

Conference Publications

- Marah De'bas, Sara A. Elsayed, and Hossam S. Hassanein, "Multitiered Worker-Oriented Resource Allocation at the Extreme Edge," 2022 IEEE Global Communications Conference (GLOBECOM), Rio de Janeiro, Brazil, 2022, (Accepted)

Acknowledgments

I am sincerely grateful to Prof. Hossam Hassanein for supporting and guiding me throughout my MSc journey with your knowledge, patience, and kindness and for always caring about my well-being. Thank you, Prof. Hassanein. I am heartily thankful to you for everything you have done for me.

I would also like to express my gratitude and appreciation to Dr. Sara Elsayed, my friend, and my mentor. Thank you for your guidance and help in exploring my research area, and all our fruitful discussions. Thank you for the time you spent reviewing my work and helping me in writing my first paper and this thesis.

I'm thankful to Mrs. Basia Palmer for her continuous support during the editing stages.

I am incredibly grateful to all my colleagues and friends in the Telecommunications Research Lab (TRL) for the fruitful meetings.

I am heartily thankful to my best flatmates ever Maram & Alaa. Maram, thank you for helping me with my move to Kingston since day one; thank you for all the fruitful discussions we had together. Alaa, I can't express how I'm thankful for all the times we spent together, our outings, the pranks we did to each other, and our late night discussions; for our laughs, I'm genuinely grateful for our friendship.

Also, I'm heartfully thankful to Rawan for all her support and guidance before I

moved to Canada. I'm also thankful for her support throughout my journey. Thank you for all our discussions related to my work and even on a personal level; thank you for your continuous support. I'm also heartfully thankful for the cutest couple I have ever met Safa & Eslam. I'm grateful for our friendship.

To my friends here in Kingston (Shaza, Heba, Nagib, Safa, Eslam, Rawan, Alaa, Eman, Mariam, Sharief, Qamar, Ataa, Anas, Mahmoud, and Yazan) who were a second family for me, I'm thankful to all of you for the fun, enjoyment, and trips we went on together. I'm grateful for all our night gatherings, to our discussions, and for all the laughs that we had to gather and for all our memorable times.

And on a personal level, I'm heartfully thankful to my family; without my family's support and encouragement, I would not have had the tenacity to stay on this path.

Statement Of Originality

I hereby certify that all the work presented in this thesis is original, and all notions and/or techniques attributed to others have been fully acknowledged in accordance with the proper referencing practices.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CC	Cloud Computing
D2D	Device-to-Device
EC	Edge Computing
EED	Extreme Edge Devices
ILP	Integer Linear Program
IoT	Internet of Things
KKT	Karush–Kuhn–Tucker
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
ML	Machine learning
MO	Multi-Objective
MWORA	Multitiered Worker-Oriented Resource Allocation
MWORA-WS	Multitiered Worker-Oriented Resource Allocation - Weighted Sum
QoE	Quality of Experience
QoS	Quality of Service

ROFA	Requesters-Oriented Fair Allocation
ROFA-MC	Requesters-Oriented Fair Allocation - Multiple Capacity
ROFA-SC	Requesters-Oriented Fair Allocation - Single Capacity
SP	Service Provider
WS	Weighted-Sum

List of Symbols

t_i	Task i
w_j	Worker j
γ_{ij}	Response delay of task t_i on worker w_j
s_j	Satisfactory reward of worker w_j
π_j	Actual reward obtained by worker w_j
μ_j	Onboard CPU capacitance of worker w_j
$c_j^{\max}$	Maximum CPU clock speed of worker w_j
δ	Maximum load capacity percentage
α_j	Minimum load capacity percentage of worker w_j
c_{ij}	Worker w_j CPU clock speed levels
$\delta c_j^{\max}$	Maximum load capacity of worker w_j
$\alpha_j \delta c_j^{\max}$	Minimum load capacity of worker w_j
Ω_{jH}	High profit threshold of worker w_j

Ω_{jL}	Low profit threshold of worker w_j
ψ	Profit Percentage
L_i	Amount of data in bits for task t_i
R_{ij}	Data rate link between w_j and requester
q_i	Average Task computation intensity for task t_i
d_{ij}	Distance between worker w_j and task t_i
v	Propagation speed
ϵ_i	task t_i deadline
β	Service provider budget
τ_j	Price of one CPU cycle defined by worker w_j
p_{ij}	Cost of executing task t_i on worker w_j

Chapter 1

Introduction

1.1 Overview and Motivation

With the proliferation of the Internet of Things (IoT), it is estimated that 23.3 billion IoT devices will be connected to the Internet by 2025 [1]. This unprecedented surge is expected to impose severe demands on computing resources to meet the rigorous Quality of Service (QoS) requirements associated with latency-sensitive and data-intensive IoT applications [2]. Satisfying such requirements can be challenging in cloud computing, due to the substantial amount of data that must be fully transmitted to distant data centers, which increases delay and creates a heavy traffic load at backhaul links [3].

Edge Computing (EC) has paved the way toward mitigating the aforementioned issues by moving the computing service closer to end users [2]. However, most EC platforms and models rely on infrastructure-based edge nodes exclusively controlled by cloud service providers and/or network operators [4]. Breaking this monopoly by harvesting plentiful yet underutilized computational resources of Extreme Edge

Devices (EEDs) [1] [5], such as tablets, smartphones, and autonomous vehicles, can democratize the edge and enable more players to develop and manage their own edge cloud. Democratizing the edge can foster a new tech market that is people-retained, democratically controlled, accessible and/or rewarding to all. In addition, parallel processing at EEDs can bring the computing service closer to end-users, which can significantly reduce the delay [6]. Given its significant impact, resource allocation is crucial in such computing paradigms [6].

Aside from load balancing approaches [7], most resource allocation schemes in EC have considered fairness from the requesters' perspective while overlooking worker fairness. In EED-enabled computing environments, overlooking workers fairness regarding achieving a satisfactory profit can cause workers to leave the system, avoid recurrent subscription to the service, and join other competing service providers (e.g. service facilitators acting as mediators between requesters and workers [8]). In the long run, this can affect the QoS, since the available resources might not be able to cope with the load imposed by incoming requests.

Another important aspect that is typically overlooked in EED-enabled computing environments is the fact that EEDs are user-owned devices, which makes them subject to a dynamic user access behavior. In particular, at any point in time, users can stream a video, play a video game, or run any intensive application on their own devices. This can profoundly impact the level of available computational resources that such devices are willing to offer for offloaded tasks. In other words, EEDs may not be available at the moment the service provider solicits their resources or may not be willing to forego all of their computational resources. Thus, it is imperative to design resource allocation schemes that account for the dynamic user access behavior

[1] In this thesis, we use the terms "Workers" and "EEDs" interchangeably.

exhibited by EEDs.

1.2 Objectives and Contributions

Our objectives can be summarized as follows:

1. Achieving edge democratization that leverages the prolific yet underutilized computational resources of EEDs (i.e., workers) while enabling resource allocation that ensures worker-oriented fairness and preserves workers participation in the service.

2. Making resource allocation decisions that maximize the QoS provided to requesters by minimizing the drop rate and task response delay.

3. Accounting for the dynamic user access behavior exhibited by workers and its impact on the level of computational resources that they may be willing to offer.

4. Ensuring attainable solutions.

5. Studying the long-term impact of workers' departure from the system on the QoS provided to requesters.

The contributions of this thesis are as follows:

1. *Multitiered Worker-oriented Resource Allocation (MWORA)*: MWORA addresses Objectives 1, 2 and 3. It formulates the resource allocation problem as an Integer Linear Program (ILP) and is solved using the Gurobi optimizer [9]. MWORA aims to minimize the drop rate and task response delay, while sustaining a certain level of satisfactory profit demanded by each worker in order to ensure workers fairness and their continuous participation in the service. This

is while abiding by a certain budget specified by the service provider and a certain deadline indicated by each requester. In addition, MWORA considers the impact of the dynamic user access behavior on the level of computational resources provided by workers. It does so by considering that this level can differ based on the cost of compromising the worker's convenience. Take, for instance, the scenario in which a user is streaming a video on their device. In this scenario, the worker (user's device) could be willing to forego the video by pausing and devoting their maximum computational capability to an offloaded task if it is worth a high financial reward. Alternatively, if the reward is moderate, the worker could continue streaming the video and devote less computational capability to the offloaded task. Note that the worker here is sacrificing a smaller portion of their capabilities to preserve some of its own convenience. If the task's reward is too low, the worker could refrain from giving up any of their resources. Thus, in contrast to existing schemes, MWORA fosters granting multitiered computational capabilities by each worker based on the financial reward associated with the offloaded task. To the best of our knowledge, MWORA is the first scheme that solicits multitiered computational capabilities and aims to achieve fair resource allocation for workers by ensuring that each worker receives a certain demanded reward.

2. *MWORA-Weighted Sum (MWORA-WS)*: MWORA-WS addresses Objective 4. It provides an analytical solution using the Karush–Kuhn–Tucker (KKT) conditions [10] and Lagrangian analysis [11]. MWORA-WS offers an upper-bound solution to the resource allocation problem provided by MWORA, since the closed-form solution yielded by the latter is not attainable, due to the fact

that the decision variable is expressed in terms of other unknown multipliers. Extensive simulations have shown that MWORA-WS closely approaches the optimal solution, yielding a significantly small gap compared to MWORA.

3. *Extensive Simulations over Rounds of Operation*: This addresses Objective 5. In order to capture the impact of maintaining workers participation in the service on the QoS, we conduct extensive simulations over multiple rounds of operation, where the satisfaction of each worker is assessed at the end of each round to determine if it leaves the system or continues offering its computational resources to the service provider. The following rounds then proceed with only the workers that have been satisfied. Performance evaluation shows that MWORA yields significant improvements in terms of various QoS metrics compared to prominent resource allocation schemes that do not consider worker's fairness and satisfaction.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a literature review of edge computing paradigms, as well as resource allocation schemes in edge computing. Chapter 3 presents MWORA, the underlying system model, problem formulation, and a detailed discussion of the performance evaluation compared to other prominent resource allocation schemes. Chapter 4 introduces MWORA-WS the underlying problem relaxation, analytical solution, and performance evaluation compared to the optimal solution yielded by MWORA. Finally, Chapter 5 concludes our work and outlines future research directions.

Chapter 2

Edge Computing Paradigms and Resource Allocation Schemes

This chapter provides an overview of the most prevalent computer paradigms. Starting with Cloud Computing [12], Fog Computing [13], Mobile Cloud Computing [14], Multi-access Edge Computing [15, 16], and Mist Computing (referred to as extreme edge computing) [5].

2.1 Edge Computing Paradigms

Considering the rapid growth in data usage, it becomes challenging for individuals and organizations to maintain all their vital information and services. That is mainly why Cloud Computing (CC), Mobile Cloud Computing (MCC), and Mobile Edge Computing (MEC) paradigms have become instrumental to many application areas.

CC services have shown many advantages and benefits in the last decade: high-performing storage and ample processing capabilities at reduced costs. Nevertheless, cloud computing suffers several significant disadvantages, such as high latency, downtime and service outages, and security vulnerability [14].

It is estimated that 23.3 billion IoT devices will be connected to the Internet by 2025 [1]. This growth will ultimately lead to surging demands in data storage and computing resources, along with networking infrastructure to accommodate users' requirements in terms of (QoS), and quality of experience (QoE). The existing conventional cloud models are unable to support the demands of the diversified applications and services that need to develop, process, and store these traffic trends [17]. Hence, research efforts were pushed toward proposing a new model of computing called fog computing [18].

The central concept behind fog computing was inspired by the natural phenomenon of fog, where a layer of moist air much closer to the earth's surface than clouds. This concept is translated into bringing cloud capabilities closer to the users by processing data at the network's edge [18]. With a similar analogy, edge computing and its related paradigms, MCC, MEC, and extreme edge computing, were introduced. These paradigms are collectively referred to as the as next-generation of cloud computing [19], where data is processed directly on devices rather than being sent to other nodes or data centers [6].

The edge infrastructure is anticipated to receive a cumulative capital expenditure of up to $800B, according to [20]. Furthermore, according to [21], the edge computing market is anticipated to grow from $36B in 2021 to $87B by 2026. Such investment shows the importance of the edge computing market.

Edge computing's main objectives are to minimize the communication latency, and reduce the volume of data transferred to the cloud, therefore alleviating pressure off of the bandwidth. These objectives ensure that information, particularly in real-time applications, do not suffer from latency problems and cause performance

degradation. However, scheduling tasks to the limited resources at the edge remains a significant challenge. Thus, different scheduling/resource allocation techniques were proposed in the literature. In all instances, maintaining fairness is one of the key considerations for real/critical time tasks and task scheduling. In this chapter, we review the latest proposed scheduling/resource allocation algorithms that achieve fairness between users while scheduling independent tasks on a single-edge server. However, before discussing the related work that focuses on fair resource allocation, we will provide an overview of resource allocation in edge computing.

2.2 Resource Allocation in Edge Computing

Resource Allocation refers to the collection of actions and techniques the participants(users/edge nodes/service providers) use to efficiently assign the task (request/data provided by the users) to the available resource (could be any computing/storage resources) to execute and complete the assigned task and for the participants to achieve their objectives depending on the availability of those resources. Different actions can be taken to achieve the desired objectives by the participant, such as computation offloading actions, which decide where to offload the task to the edge node or cloud and how to offload it [22].

There are two classifications for computation offloading: direction offloading or granularity offloading [23]. Edge computing architecture consists of three layers: the device layer, the edge layer, and the cloud layer. Direction offloading is used to offload between different layers or within the layer itself. For example, vertical offloading occurs when the a device offloads to edge layer, or an edge server offloads to the cloud layer. Alternatively, horizontal offloading occurs within the layer itself. For example,

offloading from one device to another, or from one edge server to another [23]. On the other hand, granularity offloading is categorized into full offloading (also called binary offloading), where the tasks are fully offloaded to another entity for computation, and partial offloading, where the task is partitioned, some partitions are processed locally, and some are offloaded [23, 24].

In addition, resource allocation actions are needed to allocate the tasks to the available resource [25]. Resource provision is needed for service placement decisions [26]. However, resource scheduling has different methodologies depending on the control architecture, which can be either centralized or distributed, which will be further discussed in the following Section 2.3.

2.3 Resource Allocation Schemes based on Control Architecture

The resource allocation process in edge computing is managed with either a centralized approach with one central entity, or a distributed approach with multiple entities. The following sub-sections, 2.3.1 and 2.3.2, will describe centralized and distributed control approaches, respectively.

2.3.1 Centralized-based

In the centralized based, a central entity handles resource allocation by collecting all the necessary information about task requests and resource availabilities, and notifies the involved node devices of the resource allocation decisions. Several benefits are associated with the centralized-based control architecture approach:

1. The environment is predictable and stable most of the time.

2. It is easy to implement a centralized control architecture.

3. It can easily find the optimal solution.

Different approaches for resource allocation are used in centralized architecture, such as Convex Optimization, Heuristic Algorithm, and Machine learning (ML) [23].

In convex optimization an optimization model to maximize/minimize or both under certain constraints is developed to help in resource allocation and offloading decisions. Convex optimization techniques are widely used and can quickly obtain a sub-optimal result. Studies in [27, 28] used the convex optimization technique to help the computation offloading decisions while minimizing the response time.

The Heuristic Algorithm is one of the most popular techniques for solving NP-hard problems using simple heuristics or meta-heuristic algorithms such as greedy-based or genetic-based algorithms. Heuristic algorithms are efficient but are not guaranteed to find the optimal solution. A study in [29] used a heuristic algorithm to solve the placement problem while minimizing the service cost.

The above methods cannot quickly adapt to the dynamic access behavior of the users; therefore, they cannot achieve optimal resource allocation decisions. Hence, Machine learning included as a resource allocation techniques for centralized methods since it has strong parallel processing and learning capabilities. However, the drawback of it requires a large amount of data and parameters, and it is considered a black-box technique since the learning process cannot be observed. Therefore, the study in [30] used deep supervised learning techniques to minimize the overhead computation offloading decisions.

2.3.2 Distributed-based

In a distributed environment, each node makes allocation decisions individually. Therefore, compared to the centralized based, the distributed based is more flexible and has a lower complexity since the optimization problem can be divided into several sub-problems.

Different methods for resource allocation are used in a distributed architecture, such as Game theory, Matching theory, Auction, and Federated learning [23].

The fundamental concept of a distributed approach based on game theory is to treat each user as a player. The optimal response option is made collaboratively or non-collaboratively by the players. It is simple and easy to implement. However, it may not obtain the global minimum solution. A study in [31] used the Stackelberg game to maximize the revenue of the edge server.

The matching theory is mainly used in studying the relationship between users and service providers. It has a tremendous advantage particularly in highly dense networks but can be used only for binary offloading decisions. Authors in [32] proposed an algorithm to minimize overhead computations.

In auctions, edge nodes act as sellers, and the requesters act as bidders trying to achieve a trade-off between the requesters and the nodes. It is practical in real-world scenarios, but the solution may not be optimal since it will always need a third party for auction management. The study in [33] used the Auction technique to provide resource-sharing solutions in MEC.

Federated learning is part of machine learning that enables the training of dynamic resource scheduling algorithms on numerous distributed edge devices without

exchanging data. For example, a [34] used federated learning for computation offloading decisions.

2.4 Resource Allocation Schemes based on Performance Indicators

In this section, we survey the latest resource scheduling algorithms by categorizing them by their objectives. Primarily, existing works have consider one or more of the following factors as their objective(s):

- Deadline: Maximum time allowed to complete a request.
- Cost: A cost that a requester pays to a service to execute computation, communications or data storage requests over a specified time span.
- Energy: The amount of consumed energy by a resource to fulfill a request.
- Latency: The time it takes for a request to be sent to a receiver and processed. Simply, latency is the combination of execution and response time.
- Scalability: A system's capability to sustain its efficiency as the number of service requests or resources increases.
- Reliability: A system's capability to complete its required job in a specified time under defined constraints.
- Security: Ensuring that safe techniques are used to protect the data.

These are the most commonly used measures in assessing any scheduling algorithm or comparing it with an existing proposed algorithm. The study in [35] provides a detailed description of these measures. According to their analysis, 23% of categorized

resource allocation schemes aim to improve the response delay; as in [36], while 5% aim to improve the execution delay. The remaining 72% are distributed among the remaining factors, including the recruitment cost [37].

Resource allocation is a technique used to allocate computing resources to various processes, threads, data flows, and programs/applications [38]. Fair resource allocation is required to balance the load on the system, ensure fair distribution of resources, and give priority according to the set of defined rules in the implemented scheduling algorithm while allocating the available resources [7, 38].

In general, fair resource allocation strives to guarantee the capability of a system to serve all requests and achieve specific QoS measures [7]. Aside from load balancing schemes [7], existing research efforts have mostly considered fair resource allocation from the requester's perspective.

In the following subsections, we will discuss related research efforts that attempt to allocate resources fairly while considering different objectives, such as minimizing delay, minimizing energy consumption, and maximizing profit.

2.4.1 Delay-based

In [39], the authors aim to minimize the maximum server delay in a multi-user and multi-server system in MEC by fairly selecting the appropriate server on which the requester can offload tasks, which minimizes the overall delay of the system. In [40], a multi-objective optimization problem formulation aims to minimize the delay, energy consumption, and cost paid to the MEC service provider. A study on minimizing delay and energy consumption is also proposed in [41] by partitioning the task and offloading it to multiple devices in MEC. The work in [42] partitions the task into

portions and aims to minimize the sum difference between the actual delay taken and the desired delay of each portion. In [43], the authors present a resource allocation scheme that maximizes the number of tasks executed within their allowed deadline while simultaneously ensuring fairness by prioritizing the tasks and maintaining high network stability.

2.4.2 Energy-based

A study in [44] aims to minimize the energy consumption used by mobile devices under a particular average delay constraint. Similarly, a study in [45] aims to minimize the weighted sum of energy consumption while abiding by a certain average delay constraint. In [46], the authors aim to minimize the delay while jointly abiding by the task deadline. In [47], the authors aim to minimize the average energy consumption needed to execute all tasks in a device-to-device (D2D) system. Similarly, a study in [48] aims to minimize the energy consumption of mobile devices in D2D systems using 5G cellular networks by using Lyapunov optimization [49].

2.4.3 Profit-based

A study in [50] aims to maximize the profit of edge devices to get a high QoS using auction-based computation offloading. Similarly, a study in [51] that uses an auction-based mechanism aims to motivate edge devices to provide a service by maximizing their profit through conducting several rounds of auctions. In [52], the authors aim to maximize the profit while abiding by the task deadline using a single objective optimization model. Similarly, [53] aims to maximize service providers' profit using a heuristic-based genetic algorithm.

Chapter 3

Multitiered Worker-Oriented Resource Allocation (MWORA)

In this chapter, we present MWORA, its underlying system model, problem formulation, and performance evaluation. In MWORA, the system consists of user-owned devices operating as workers, a set of requesters with tasks that need to be offloaded for execution, and a Service Provider (SP) functioning as the centralized entity responsible for making resource allocation decisions in the system. The SP has a coverage area, a set of workers operating within this coverage area, and a set of tasks received from the requesters. The SP is responsible for allocating the tasks to the participating workers under certain constraints and limitations.

3.1 System Model

Given a set of workers W = $\{w_1, w_2, \ldots w_m\}$, a set of requesters U = $\{u_1, u_2, \ldots u_k\}$, and a set of tasks T = $\{t_1, t_2, \ldots t_n\}$ at any given time. Each task t_i has a certain computation workload or intensity q_i (in CPU cycles/bit) and involves a certain amount

of data in bits, denoted L_i, and should be executed within a specified deadline, denoted ϵ_i. The distance between requester u_i and worker w_j is denoted d_{ij}, R_{ij} denotes the data rate of the link between u_i and w_j, and the propagation speed is denoted v. SP has a specific budget β that is used to recruit the available workers. This budget should not be exceeded.

Each worker w_j specifies a certain price τ_j that it should be paid per executing one CPU cycle. Also, each worker w_j specifies a minimum profit it should gain to be deemed satisfied (i.e., a satisfactory reward), denoted s_j. This satisfactory reward acts as an indicator of a worker's satisfaction by SP. One of the SP's primary functions when assigning tasks to workers is to assign one or more tasks to a worker who is capable of completing each of the assigned tasks within the task deadline ϵ_i. At the same time, the assigned tasks must render the required satisfactory profit s_j of each worker without exceeding the budget β.

The cost of executing each task t_i on worker w_j is denoted p_{ij} and is given by Eq. 3.1.

$$p_{ij} = \tau_j q_i L_i \tag{3.1}$$

Each worker w_j has a maximum CPU clock speed denoted $c_j^{\max}$ (in CPU cycles/sec). Each worker w_j is willing to dedicate one of three possible levels of its $c_j^{\max}$ according to a high and low-profit threshold that each worker specifies, denoted Ω_{jH} and Ω_{jL}, respectively. The CPU cycle frequency that each worker w_j is willing to dedicate to execute task t_i is denoted c_{ij}, and is given by Eq.3.2, where $0 \leq \delta \leq 1$.

$$c_{ij} = \begin{cases} \delta c_j^{\max} & \text{if } p_{ij} \geq \Omega_{jH} \\ \alpha_j \delta c_j^{\max} & \text{if } \Omega_{jL} \leq p_{ij} < \Omega_{jH} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Note that $\delta c_j^{\max}$ is the maximum load capacity that can be purchased from worker w_j that ensures that it is not completely overwhelmed. The capacity level $\alpha_j \delta c_j^{\max}$ reduces the capacity that can be purchased from w_j by a defined factor α_j , where $0 \leq \alpha_j \leq 1$. If the price of executing task t_i on worker w_j, p_{ij}, does not correspond to any of the preceding levels, the worker refrains from devoting any capacity level to task t_i (the task t_i will be added to a new set named Ni) ; therefore, Ω_{jL} of w_j acts as a cut-off point below which the worker does not perform the task.

The response delay associated with executing task t_i on worker w_j, denoted γ_{ij}, is composed of the execution delay, transmission delay, and propagation delay, and is given by Eq. 3.3.

$$\gamma_{ij} = \frac{q_i L_i}{c_{ij}} + \frac{L_i}{R_{ij}} + \frac{d_{ij}}{v} \tag{3.3}$$

Figure 3.1 summarizes the main workflow for MWORA system model and the needed information that needs to be exchanged between the workers, requesters, and the tasks.

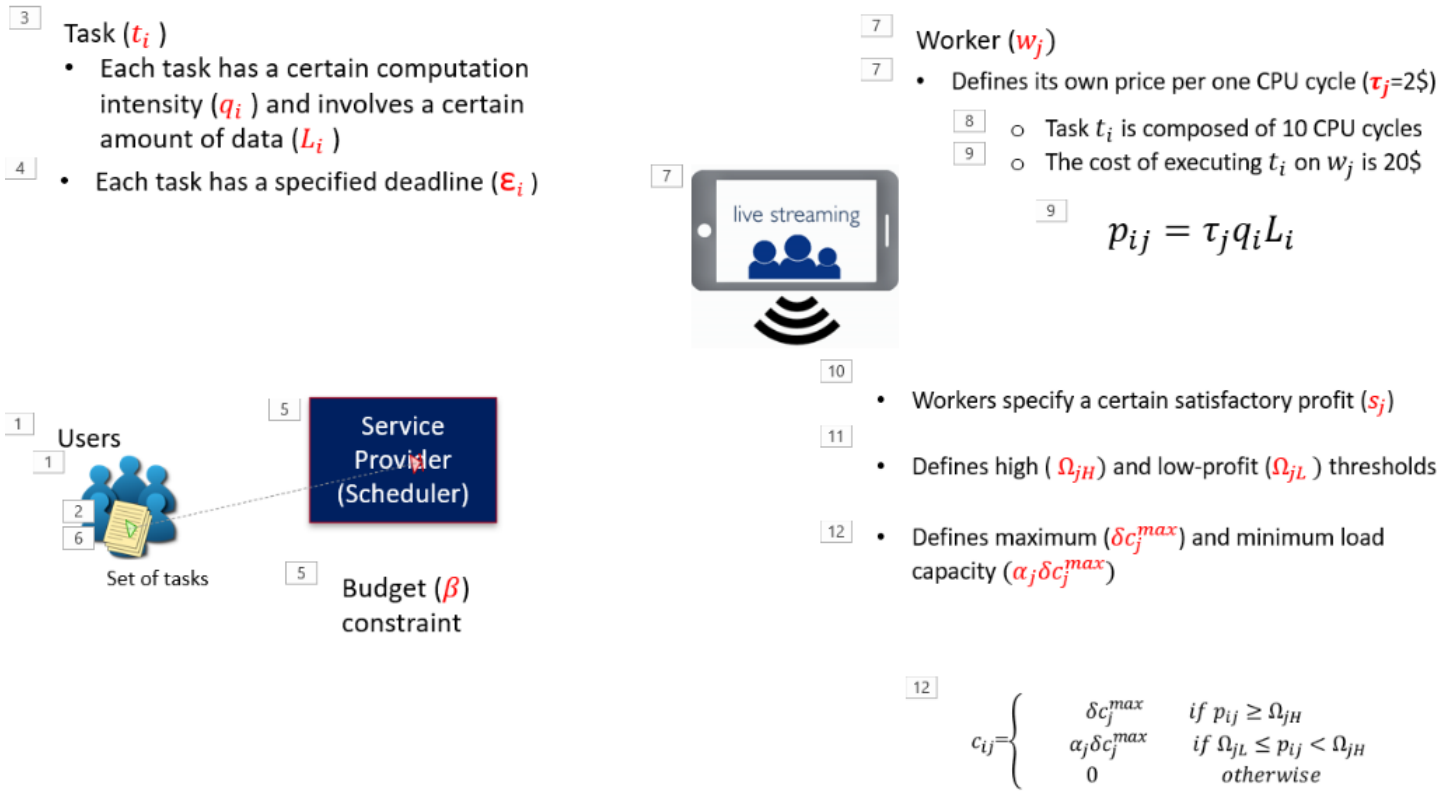


Figure 3.1: MWORA System Model: A summarized workflow

3.2 Problem Formulation

The objective is to minimize the drop rate (i.e., maximizing the service capacity: the number of tasks executed) and minimize the total response delay. We formulate the problem as an ILP multi-objective problem, where the binary decision variable x_{ij} is set to 1 if task t_i is assigned to w_j, and 0 otherwise, as given by Eq. 3.4.

$$x_{ij} = \begin{cases} 1 & \text{if } t_i \text{ is assigned to } w_j \\ 0 & \text{Otherwise} \end{cases} \tag{3.4}$$

The problem formulation is shown below:

$$\min_{x_{ij}}(|T| - \sum_{j \in W} \sum_{i \in T} x_{ij}) \tag{3.5a}$$

and

$$\min_{x_{ij}} \sum_{j \in W} \sum_{i \in T} x_{ij} \gamma_{ij} \tag{3.5b}$$

subject to:

$$\sum_{j \in W} x_{ij} \leq 1 \quad \forall i \in T \tag{3.5c}$$

$$\sum_{j \in W} \sum_{i \in T} x_{ij} p_{ij} \leq \beta \tag{3.5d}$$

$$\sum_{j \in N_i} x_{ij} = 0 \quad \forall j \in W \tag{3.5e}$$

$$\sum_{i \in T} x_{ij} c_{ij} \leq \delta c_j^{\max} \quad \forall j \in W \tag{3.5f}$$

$$\sum_{j \in W} \sum_{i \in T} x_{ij} \gamma_{ij} \leq \epsilon_i \quad \forall i \in T \tag{3.5g}$$

$$\sum_{i \in T} x_{ij} p_{ij} \geq \psi s_j \quad \forall j \in W_j \tag{3.5h}$$

The objectives in Eq. (3.5a) and Eq. (3.5b) are subject to the constraints (3.5c)-(3.5h). Constraint (3.5c) specifies that each task must be assigned to only one worker, thus ensuring that each task is assigned only once. Constraint (3.5d) ensures that the total reward gained by all workers from the tasks assigned to them does not exceed the budget specified by the SP. Constraint (3.5e) ensures that no task is assigned to a worker from the set N_i, which is the set of tasks with which the worker has a zero c_{ij} . Constraint (3.5f) ensures that for each worker, the CPU cycle frequency used to execute all the tasks assigned to it does not exceed the maximum load capacity that can be purchased from the worker. Constraint (3.5g) ensures that when assigning a task to a worker, the task is executed within the task deadline ϵ_i. Constraint (3.5h) ensures that for each worker, the total reward gained from all the tasks assigned to it is no less than a certain threshold.

3.3 Performance Evaluation

This section evaluates the performance of MWORA compared to a baseline resource allocation approach, referred to as Requesters-oriented Fair Allocation (ROFA). ROFA is a representative of existing resource allocation schemes that focus on fairness in terms of the QoS provided to requesters without considering the satisfaction of workers [39]. Note that ROFA does not support multitiered computational capabilities. In order to show the separate effect of workers satisfaction and the multitiered computational capabilities fostered by MWORA, we implement ROFA twice, once with multitiered computational capabilities and once where each worker grants only a single level of computational capabilities. The former approach is referred to as ROFA-MC, and the latter is referred to as ROFA-SC.

In order to show the effect of satisfying workers, we implement MWORA, ROFA-MC, and ROFA-SC over 10 rounds, where only the satisfied workers remain in the system for the next round to execute the incoming tasks. To demonstrate this effect, we show the results of each scheme for the first and last rounds, referred to as MWORA-F and MWORA-L, ROFA-MC-F, ROFA-MC-L, and ROFA-SC-F and ROFA-SC-L, respectively.

We use the following performance metrics:

1. **Average response delay** is the average time taken to offload and execute the tasks (i.e., the average γ_{ij}, where γ_{ij} is given by Eq. 3.3).

2. **Service capacity ratio** is the ratio of the number of successfully executed tasks to the total number of tasks. The service capacity ratio reflects how MWORA is minimizing the drop rate, since the latter is the inverse of the former.

3. **Workers satisfaction ratio** is the ratio of the number of workers that acquire their desired satisfactory reward s_j to the total number of workers. The worker's satisfaction ratio highlights the novel perspective of MWORA.

4. **Workers fairness**, which is calculated using Jain's fairness index [54], denoted F, and is given by Eq. 3.6, where π_j denotes the actual reward obtained by worker w_j, and F ranges from 0 (best case scenario) to 1 (worst case scenario). The worker's fairness reflects how fair MWORA is from the worker's perspective.

$$F = \frac{\left(\sum_{j=1}^{n} \pi_j - s_j\right)^2}{n \cdot \sum_{j=1}^{n} \left(\pi_j^2 - s_j^2\right)} \tag{3.6}$$

5. **Total energy consumption of workers**, which is the sum of the energy consumed by each worker due to task execution, as given by Eq. 3.7 [55], where μ_j is the onboard CPU capacitance of worker w_j, and $\lambda = 2$. The total energy consumption of workers reflects the cost paid for minimizing the drop rate and satisfying the workers.

$$\sum_{j \in W} \sum_{i \in T} x_{ij} \mu_j q_i L_i c_{ij}^{\lambda - 1} \tag{3.7}$$

3.3.1 Simulation Setup

MWORA, ROFA-MC, and ROFA-SC are all implemented using Gurobi optimizer [9], and since our problem is multi-objective problem Eq. 3.5a is set to have the first priority and the second is for Eq. 3.5b. Simulations are performed over a $500m \times 500m$ area, where all requesters and workers are uniformly distributed. The total number of tasks unless otherwise specified, is set to 250, and unless otherwise

specified, the number of available workers is set to 100. The data size of tasks is uniformly distributed in the range of [5, 10] bits. Unless otherwise specified, Unless otherwise specified, the average task computational intensity q is set to 20. The propagation speed v is set to 120 m/s and the data rate is uniformly distributed in the range of [15, 30] bits/sec. The maximum computation capability of workers $c_j^{\max}$ is set in the range of [300, 800] CPU cycles/sec. The price per one CPU cycle that is specified by workers ranges between [0.1, 0.4]. The factor δ is set to 0.9, whereas α_j ranges between [0.1, 0.4]. The satisfactory profit of workers ranges between [100, 250], and Ω_{jH} and Ω_{jL} are set in the range of [90, 250] and [5, 89], respectively. The budget is set to 2500. The value of ψ is set to 80%. The deadline of tasks ranges between [0.5, 10] seconds. The onboard capacitance of all workers μ_j is set to 10^{-11} , and λ is set to 2.

Simulations are conducted over 10 rounds to show the effect of worker satisfaction. Each round is a new time interval during which the SP assigns incoming tasks to operating workers. The workers that are unsatisfied with their reward leave the system, so the number of operating workers can decrease between successive rounds. To demonstrate this effect, we present the first and last rounds of each scheme.

3.3.2 Results and Analysis

We evaluate the performance of MWORA, ROFA-MC, and ROFA-SC over varying average task computation intensity q, varying number of tasks, varying budget, and varying number of workers. We conduct four experiments to evaluate the performance of MWORA. All experiments are repeated 50 times for each instance, and simulation results are presented at a confidence level=95%. Note that the rendered confidence

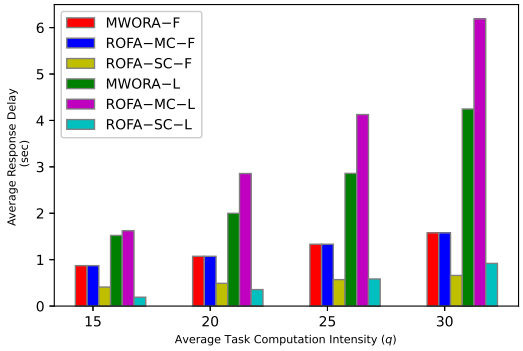
intervals are shown in Figure 3.3(a). Since the confidence intervals are negligible, they are not explicitly depicted in the other figures for clarity of presentation. In the legend notation of each experiment, the letter F after the scheme name indicates the first round, and L indicates the last round.

3.3.2.1 The Impact of Task Intensity

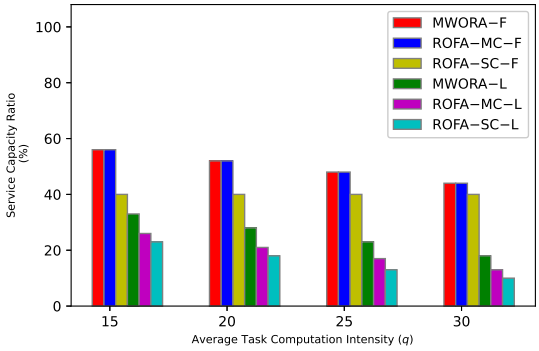
In this experiment, we assess the performance of MWORA, ROFA-MC, and ROFA-SC over varying average task computation intensity q. Below is a detailed discussion of the performance results in terms of the average response delay, service capacity, workers satisfaction, fairness, and energy consumption. Note that the performance results of MWORA, ROFA-MC, and ROFA-SC are depicted in the first round (MWORA-F, ROFA-MC-F, and ROFA-SC-F) and the last round (MWORA-L, ROFA-MC-L, and ROFA-SC-L).

1. Average response delay

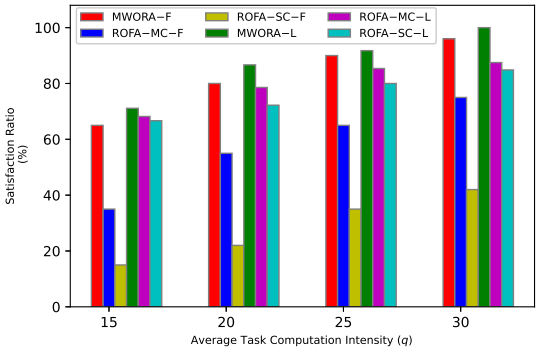
Figure 3.2(a) shows the effect of varying the average computational intensity q on the average response delay of MWORA, ROFA-MC, and ROFA-SC. Note that the average response delay in all schemes increases as q increases. This is due to the increase in the number of CPU cycles that need to be executed, which increases the execution delay. In addition, MWORA, ROFA-MC, and ROFA-SC exhibit an increase in the average response delay between the first and last rounds. This is because in the last round, fewer workers remain in the service, since all unsatisfied workers that do not receive their demanded reward unsubscribe from the service, leaving fewer workers to handle the incoming task requests and thus incurring higher



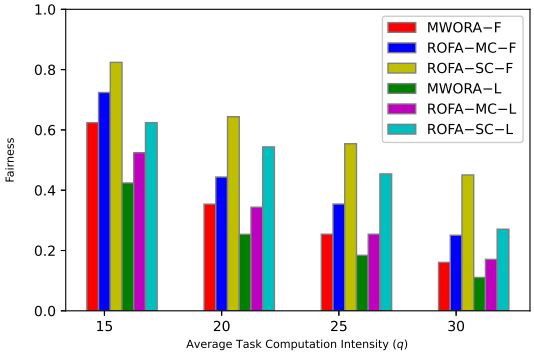
(a) Average response delay



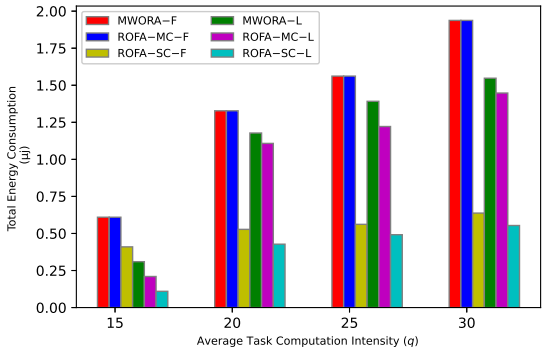
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 3.2: Performance results of MWORA, ROFA-MC, and ROFA-SC over varying task computation intensity q

response delay.

As shown in Figure 3.2(a), MWORA-F and ROFA-MC-F yield the same average response delay. This can be attributed to the fact that workers in MWORA and ROFA-MC dedicate the same amount of computational capability and make the same resource allocation decisions during the first round, whereas the effect of workers satisfaction exhibited by MWORA starts to manifest in the next rounds. In contrast, ROFA-SC-F renders the lowest average response delay, yielding up to a 58% improvement compared to MWORA-F, and ROFA-SC-F, respectively. This is since in ROFA-SC, it is assumed that all workers are exploited at their maximum computational capability. Note that in ROFA-SC, all workers have a single level rather than multitiered computational capabilities. Consequently, workers are assigned a single task at a time, and each worker executes the assigned task using its maximum computational capability, thus reducing the delay.

Figure 3.2(a) also depicts the performance of MWORA-L, ROFA-MC-L, and ROFA-SC-L, which represent the schemes in the last round of operation. Note that MWORA-L outperforms ROFA-MC-L, with a significant delay reduction of up to 31%. This can be attributed to the fact that MWORA manages to retain the service of a higher number of workers in the long run. This is since, as opposed to ROFA-MC, MWORA accounts for the workers' satisfaction. In contrast, ROFA-SC-L yields the lowest average delay, with an improvement of 88% and 77% compared to ROFA-MC-L and MWORA-L, respectively. This is due to the same reasons mentioned above. However, as illustrated later, this comes at the expense of service capacity, which also reflects the percentage of requests that are executed within the deadline. Thus, in terms of satisfying more delay-sensitive tasks, MWORA yields the best results among

all schemes.

2. Service capacity ratio

We conduct the same experiment to assess the service capacity ratio of MWORA, ROFA-MC, and ROFA-SC over varying q. As depicted in Figure 3.2(b), the service capacity decreases in both the first and last rounds of all schemes as q increases. This is because as q increases, the price of executing the tasks increases, which limits the number of tasks that can be executed within the budget limit. Note that MWORA-F and ROFA-MC-F yield the same service capacity. This is since workers offer the same computational capability for task execution during the first round. In addition, MWORA-F significantly outperforms ROFA-SC-F, with an up to 28% improvement in service capacity. The reason is that workers in MWORA have multitiered rather than single computational capabilities, which increases the number of tasks that they can perform at a time. In the last round, MWORA outperforms both ROFA-MC and ROFA-SC, where MWORA-L yields improvements of up to 38% and 44% compared to ROFA-MC-L and ROFA-SC-L, respectively. This can be attributed to the fact that MWORA accounts for workers' satisfaction, which increases the number of satisfied workers, thus increasing the number of available workers that are willing to execute more tasks in the long run.

3. Workers satisfaction ratio

Figure 3.2(c) depicts the workers' satisfaction ratio of MWORA, ROFA-MC, and ROFA-SC over varying q. Note that the workers' satisfaction ratio increases as q increases in both the first and last rounds in all schemes. This is because tasks with higher computational intensity cost higher than those with lower intensity. Thus,

workers gain larger rewards when performing tasks with higher computational intensity, which increases the chance of their satisfaction. In the first round, MWORA renders superior performance compared to ROFA-MC and ROFA-SC, where MWORA-F yields an improvement of up to 72% and 85% compared to ROFA-MC-F and ROFA-SC-F, respectively. A similar pattern is also witnessed in the last round, where MWORA-L yields an improvement of up to 14% and 16% compared to ROFA-MC-L and ROFA-SC-L, respectively. This is since, as opposed to ROFA-MC and ROFA-SC, MWORA accounts for the satisfaction of workers and strives to grant them a certain satisfactory reward. Note that ROFA-SC yields the lowest satisfaction ratio compared to MWORA and ROFA-MC, since workers in ROFA-SC offer a single level rather than multitiered computational capabilities. This single level endowment limits the number of tasks that can be executed, which reduces the rewards gained by workers.

4. Workers fairness

Figure 3.2(d) shows the workers fairness index of MWORA, ROFA-MC, and ROFA-SC over varying q. As previously mentioned, a lower fairness index (Eq. 3.6) implies better fairness. Note that the fairness index decreases as q increases in both the first and last rounds of all schemes. This is because the higher the value of q, the higher the reward granted to workers, which increases the chance that workers either receive their satisfactory reward or come close to receiving it, thus ensuring a fair distribution of the available budget. It can be observed that MWORA significantly outperforms ROFA-MC and ROFA-SC in both the first and last rounds. In particular,

MWORA-F yields an improvement of up to 35% and 118% compared to ROFA-MC-F, and ROFA-SC-F, respectively, whereas MWORA-L yields an improvement of up to 34% and 114% compared to ROFA-MC-L, and ROFA-SC-L, respectively. This is since MWORA accounts for workers fairness and strives to reach the satisfactory reward demanded by each worker. It is worth noting that the leverage gained by MWORA is more evident compared to ROFA-SC due to the multitiered nature of MWORA, which enables more tasks to be allocated to workers, thus increasing the chance of workers-oriented fairness.

5. Total energy consumption

Figure 3.2(e) shows the impact of varying q on the total energy consumption of workers in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as q increases, the total energy consumption increases in both the first and last rounds of all schemes. This is because as q increases, the intensity of tasks increases, which causes the workers to expend more energy during their execution. MWORA-F and ROFA-MC-F render the same level of energy consumption, since both schemes allocate the same computational capability levels in the first round. In the last round, MWORA-L yields up to a 91% higher energy consumption than ROFA-MC-L due to the fact that MWORA-L performs more tasks, since it retains more workers in the last round than ROFA-MC-L. In contrast, ROFA-SC yields the lowest energy consumption in both the first and last rounds. In particular, ROFA-SC-F renders an improvement of up to 87% to MWORA-F and ROFA-MC-F, respectively, whereas ROFA-SC-L renders an improvement of up to 91% and 87% compared to MWORA-L and ROFA-MC-L, respectively. This is since ROFA-SC uses a single rather than multitiered computational capabilities, which forces each worker to perform a single task

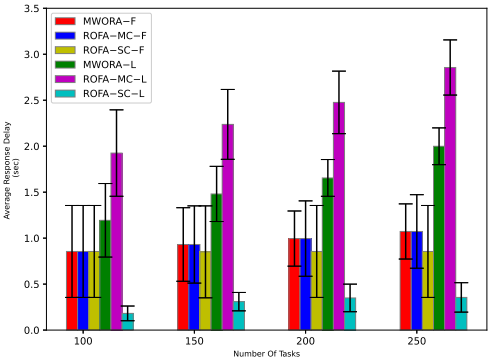
at a time, thus reducing the energy consumption. Consequently, it is evident that the leverage gained by MWORA in terms of service capacity, workers satisfaction, and fairness comes at the expense of energy consumption.

3.3.2.2 The Impact of the Number of Tasks

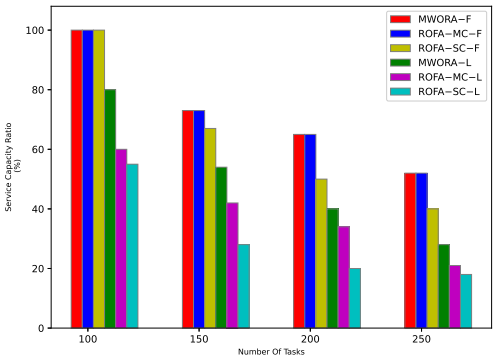
In this experiment, we assess the performance of MWORA, ROFA-MC and ROFA-SC over varying number of tasks. Below is an analysis of the yielded results in terms of the performance metrics.

1. Average response delay and service capacity ratio

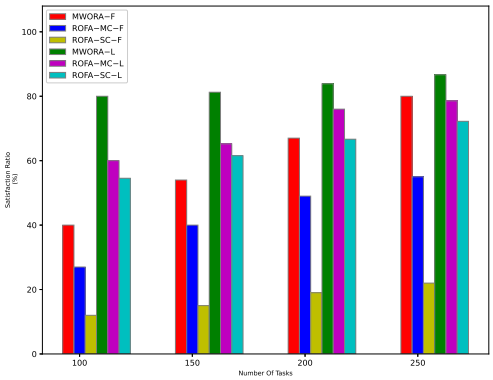
Figure 3.3(a) and Figure 3.3(b) show the effect of varying the number of tasks on the average response delay and the service capacity ratio, respectively in MWORA, ROFA-MC, and ROFA-SC. As the number of tasks increases, the average response delay increases and the service capacity ratio decreases in both the first and last rounds of MWORA and ROFA-MC. This is because the increase in the number of tasks that need to be executed prompts the need to solicit the lower rather than the higher level of the computational capabilities of workers so as to enable multiple tasks rather than a single task to be executed on workers. As a result, the service capacity ratio decreases since the chance of executing tasks within the task deadline becomes lower, since some workers are soliciting the lower computational capability. In addition, the execution delay increases, which in turn increases the average response delay. In contrast, ROFA-SC-F renders the same average response delay as the number of tasks increases. This is since ROFA-SC-F always solicits a single rather than multitiered computational capabilities, thus enabling only one task to be executed at a time using the maximum available computational capability of



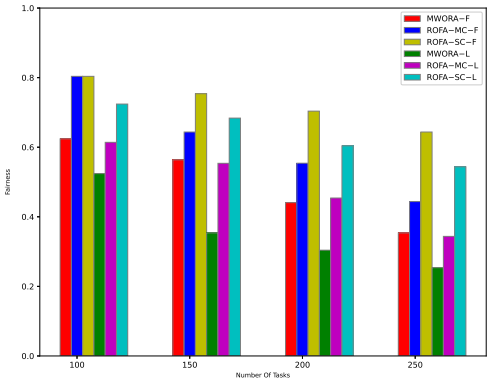
(a) Average response delay



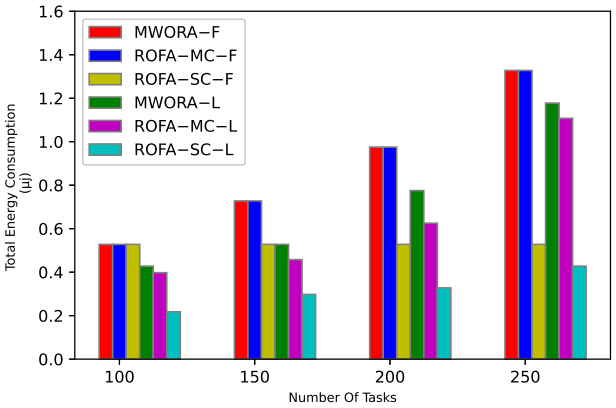
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 3.3: Performance results of MWORA, ROFA-MC, and ROFA-SC over varying number of tasks

workers. Hence, ROFA-SC-F yields the lowest average response delay, with an improvement of up to 20% compared to MWORA-F and ROFA-MC-F, respectively. In addition, MWORA-F renders a higher service capacity ratio, with an improvement of up to 23% compared to ROFA-SC-F, whereas MWORA-L yields the highest service capacity ratio, with an increase of up to 33% and 52% compared to ROFA-MC-L and ROFA-SC-L, respectively. Note that the service capacity ratio of ROFA-SC-F decreases as the number of tasks increases. This is since the number of executed tasks remains the same, while the total number of tasks increases, which leads to a reduction in their ratio.

As shown in Figure 3.3(a) and Figure 3.3(b), MWORA-F and ROFA-MC-F yield the same average response delay and service capacity ratio. This is since the effect of workers' satisfaction that MWORA fosters does not manifest in the first round. In contrast, it can be observed that in the last round, MWORA significantly outperforms ROFA-MC, where MWORA-L yields a reduction in delay of up to 30% and an increase in the service capacity ratio of up to 33% compared to ROFA-MC-L. This can be attributed to the fact that MWORA-L succeeds in keeping more workers in the system, since, as opposed to ROFA-MC-L and ROFA-SC-L, it accounts for their satisfaction. Note that the higher the number of workers available in the system, the higher the chance of executing more tasks and the higher the chance of selecting workers with higher computational capabilities to execute the tasks, thus reducing the delay. On the other hand, it can be observed that ROFA-SC-L yields the lowest average response delay, with an improvement of up to 77% and 90% compared to MWORA-L and ROFA-MC-L, respectively. This is due to the fact that ROFA-SC uses a single level of the computational capability of workers, which is the maximum

level.

3. Workers satisfaction ratio and fairness

Figure 3.3(c) and Figure 3.3(d) demonstrate the satisfaction ratio and fairness of workers, respectively over varying number of tasks in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the number of tasks increases, the satisfaction ratio and fairness in both the first and last rounds of all schemes improve. This is because the higher the number of available tasks, the higher the compensation that workers can receive since they can execute more tasks, which increases the chance of gaining their satisfactory reward or closely approaching it, thus ensuring a high satisfaction ratio and an increased chance of a fair distribution of the rewards. It can be observed that MWORA yields the best satisfaction ratio and fairness compared to ROFA-MC and ROFA-SC in both the first and last rounds. In particular, MWORA-F renders an improvement of up to 48% and 70%, as well as 22% and 61% compared to ROFA-MC-F and ROFA-SC-F in terms of the satisfaction ratio and fairness, respectively. In addition, MWORA-L renders an improvement of up to 33% and 41%, as well as 36% and 87% compared to ROFA-MC-L and ROFA-SC-L in terms of the satisfaction ratio and fairness, respectively. This is due to the same previously mentioned reasons.

5. Total energy consumption

Figure 3.3(e) illustrates the impact of the number of tasks on the total energy consumption of workers in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the number of tasks increases, the total energy consumption increases. This is because the higher the number of available tasks, the higher the number of executed tasks by workers, which increases the total energy consumption. Note that MWORA

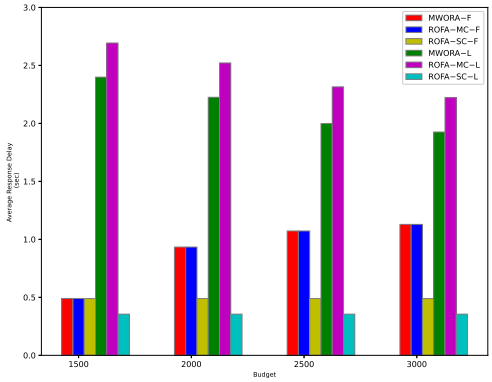
yields the highest energy consumption compared to ROFA-MC and ROFA-SC, since it executes a significantly larger number of tasks. This is due to the same aforementioned reasons. It is also worth noting that ROFA-SC renders the lowest energy consumption, since it executes the lowest number of tasks. In particular, ROFA-SC-F renders an improvement of up to 60% compared to MWORA-F and ROFA-MC-F, respectively, and ROFA-SC-L renders an improvement of up to 56% and 47% compared to MWORA-L and ROFA-MC-L, respectively.

3.3.2.3 The Impact of Budget

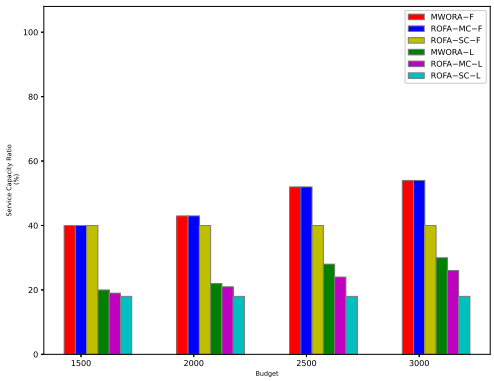
In this experiment, we evaluate the performance of MWORA, ROFA-MC and ROFA-SC over varying budget β. Below is an analysis of the yielded results in terms of the performance metrics.

1. Average Response Delay and Service Capacity Ratio

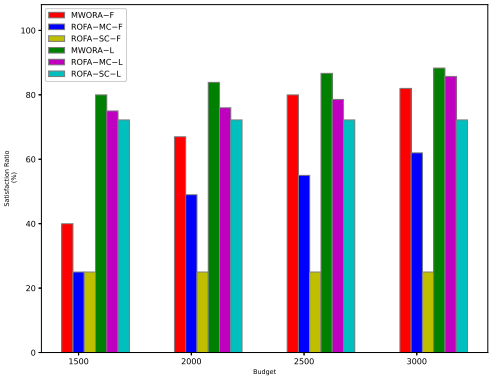
As shown in Figure 3.4(a) and Figure 3.4(b) depict the impact of varying the budget on the average response delay and service capacity ratio, respectively, in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the budget increases, the average response delay and service capacity ratio increase in both rounds of MWORA and ROFA-MC. This is because as the budget increases, it becomes more affordable to pay more workers to execute a larger number of tasks. This increases the chance of soliciting the lower capability level of workers to execute more tasks, while at the same time affording the higher cost of more capable workers. This increases the execution delay, thus the response delay, as well as the percentage of tasks that can be executed within the deadline with respect to the total number of tasks (i.e.,



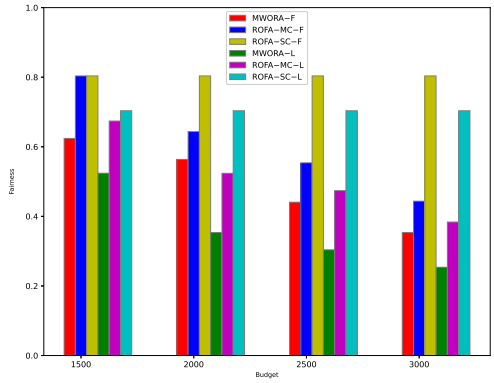
(a) Average response delay



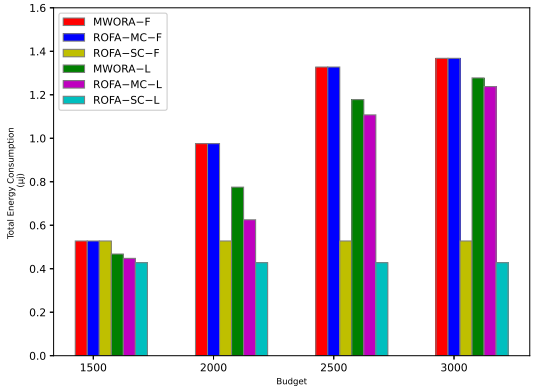
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 3.4: Performance results of MWORA, ROFA-MC, and ROFA-SC over varying budget

service capacity ratio). In contrast, varying the budget has no impact on ROFA-SC in terms of the average response delay and service capacity ratio. This is since, all workers in ROFA-SC can only provide a single-level of their computational capability regardless of the available budget, thus executing a single task at a time using the maximum computational capability of workers. Thus, ROFA-SC-F yields the lowest average response delay, with an improvement of up to 56% compared to MWORA-F and ROFA-MC-F, respectively. In addition, MWORA-F renders a higher service capacity ratio, with an improvement of up to 25% compared to ROFA-SC-F, whereas MWORA-L yields the highest service capacity ratio, with an increase of up to 20% and 35% compared to ROFA-MC-L and ROFA-SC-L, respectively.

As shown in Figure 3.4(a) and Figure 3.4(b), MWORA-F and ROFA-MC-F yield the same average response delay and service capacity ratio. This is since the effect of workers' satisfaction that MWORA fosters does not manifest in the first round. In contrast, it can be observed that in the last round, MWORA significantly outperforms ROFA-MC, where MWORA-L yields a reduction in delay of up to 14% compared to ROFA-MC-L. This can be attributed to the fact that MWORA-L succeeds in keeping more workers in the system, particularly as the budget increases, since as opposed to ROFA-MC-L and ROFA-SC-L, MWORA-L accounts for their satisfaction. Note that the higher the number of available workers in the system, the higher the chance of executing more tasks and the higher the chance of selecting workers with higher computational capabilities to execute the tasks, thus reducing the delay and increasing the service capacity ratio. On the other hand, it can be observed that ROFA-SC-L yields the lowest average response delay, with an improvement of up to 80% and 86% compared to MWORA-L and ROFA-SC-L, respectively. This is due to the same

reasons mentioned above.

3. Workers Satisfaction Ratio and Fairness

Figure 3.4(c) and Figure 3.4(d) demonstrate the satisfaction ratio and fairness of workers, respectively, over varying budget in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the budget increases, the satisfaction ratio and fairness in both the first and last round of MWORA and ROFA-MC improve. This is because the higher the budget, the higher the reward that workers receive, which increases the chance of gaining their satisfactory reward or closely approaching it, thus ensuring a high satisfaction ratio and an increased chance of a fair distribution of the rewards. In contrast, increasing the budget has no impact on the satisfaction ratio and fairness in ROFA-SC, since all workers can only execute a single task at a time, and thus the number of executed tasks remains the same regardless of the budget. It can be observed that MWORA yields the best satisfaction ratio and fairness compared to ROFA-MC and ROFA-SC in both the first and last rounds. In particular, MWORA-F renders an improvement of up to 60% and 69%, as well as 21% and 65% compared to ROFA-MC-F and ROFA-SC-F in terms of the satisfaction ratio and fairness, respectively. In addition, MWORA-L renders an improvement of up to 10% and 18%, as well as 33% and 76% compared to ROFA-MC-L and ROFA-SC-L in terms of the satisfaction ratio and fairness, respectively. This is due to the same previously mentioned reasons.

5. Total energy consumption

Figure 3.4(e) illustrates the impact of the budget on the total energy consumption of workers in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the

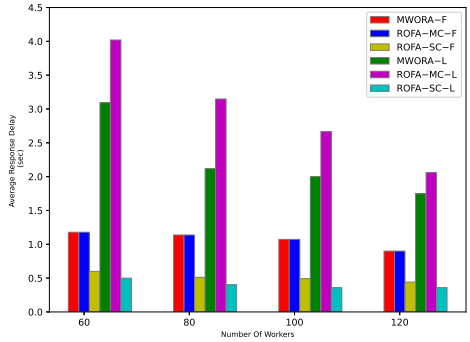
budget increases, the total energy consumption increases in MWORA and ROFA-MC. This is because the higher the budget, the higher the number of executed tasks by workers, which increases the total energy consumption. In contrast, the budget has no impact on the total energy consumption in both the first and last rounds of ROFA-SC. This is since it has no impact on the number of executed tasks. Note that MWORA yields the highest energy consumption compared to ROFA-MC and ROFA-SC, since it executes a significantly larger number of tasks. This is due to the same aforementioned reasons. It is also worth noting that ROFA-SC renders the lowest energy consumption, since it executes the lowest number of tasks. In particular, ROFA-SC-F renders an improvement of up to 60% compared to MWORA-F and ROFA-MC-F, respectively, and ROFA-SC-L renders an improvement of up to 62% and 58% compared to MWORA-L and ROFA-MC-L, respectively.

3.3.2.4 The Impact of the Number of Workers

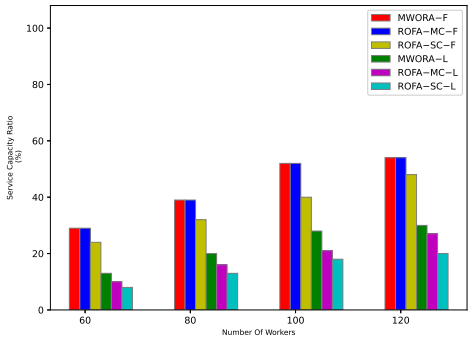
Finally, we will demonstrate the performance results of MWORA, ROFA-MC and ROFA-SC over a varying number of workers. Below is an analysis of the yielded results in terms of the performance metrics.

1. Average Response Delay and Service Capacity Ratio

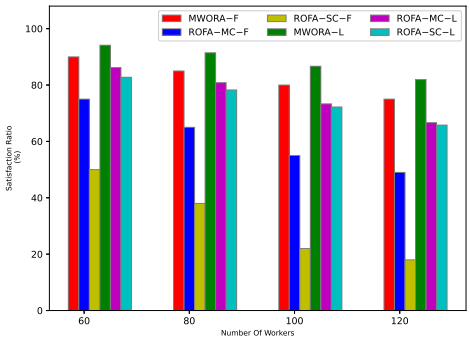
As shown in Figure 3.5(a) and Figure 3.5(b) depict the impact of varying the number of workers on the average response delay and service capacity ratio, respectively, in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the number of workers increases, the average response delay and service capacity ratio increase in both rounds of MWORA and ROFA-MC. This is because as the number of workers



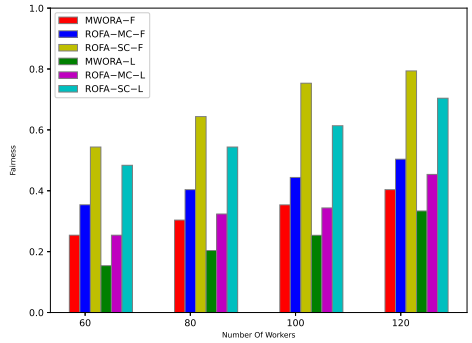
(a) Average response delay



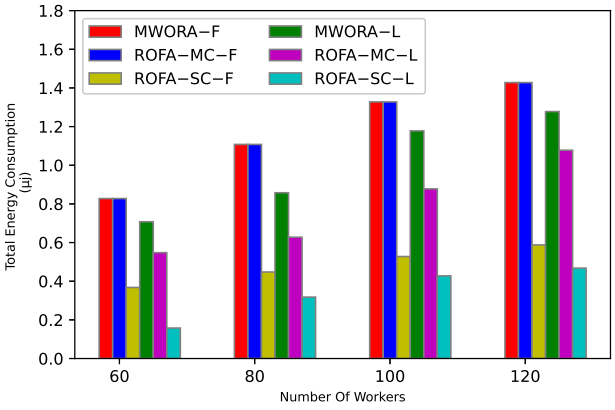
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 3.5: Performance results of MWORA, ROFA-MC, and ROFA-SC over varying number of workers

increases, more workers are available to execute a larger number of tasks. This increases the chance of soliciting the lower capability level of workers to execute more tasks, while at the same time affording the higher cost of more capable workers. This increases the execution delay, thus the response delay, as well as the percentage of tasks that can be executed within the deadline with respect to the total number of tasks (i.e., service capacity ratio). In contrast, ROFA-SC can only provide a single-level of their computational capability, thus executing a single task at a time using the maximum computational capability of workers, therefore, the number of executed tasks is equal to the number of workers. Thus, ROFA-SC-F yields the lowest average response delay, with an improvement of up to 53% compared to MWORA-F and ROFA-MC-F, respectively. In addition, MWORA-F renders a higher service capacity ratio, with an improvement of up to 23% compared to ROFA-SC-F, whereas MWORA-L yields the highest service capacity ratio, with an increase of up to 33% and 38% compared to ROFA-MC-L and ROFA-SC-L, respectively.

As shown in Figure 3.5(a) and Figure 3.5(b), MWORA-F and ROFA-MC-F yield the same average response delay and service capacity ratio. This is since the effect of workers' satisfaction that MWORA fosters does not manifest in the first round. In contrast, it can be observed that in the last round, MWORA significantly outperforms ROFA-MC, where MWORA-L yields a reduction in delay of up to 25% compared to ROFA-MC-L. This can be attributed to the fact that MWORA-L succeeds in keeping more workers in the system, particularly as the number of workers increases, since, as opposed to ROFA-MC-L and ROFA-SC-L, MWORA-L accounts for their satisfaction. Note that the higher the number of available workers in the system, the higher the chance of executing more tasks and the higher the chance of selecting workers with

higher computational capabilities to execute the tasks, thus reducing the delay and increasing the service capacity ratio. On the other hand, it can be observed that ROFA-SC-L yields the lowest average response delay, with an improvement of up to 75% and 82% compared to MWORA-L and ROFA-SC-L, respectively. This is due to the same reasons mentioned above.

3. Workers satisfaction ratio and fairness

Figure 3.5(c) and Figure 3.5(d) demonstrate the satisfaction ratio and fairness of workers, respectively, over a varying number of workers in MWORA, ROFA-MC, and ROFA-SC. As the number of workers increases, the satisfaction ratio and fairness in all schemes' first and last rounds decreases. This is because ensuring a high satisfaction ratio and a fair distribution of the rewards is harder on the SP with the increase in the number of workers as it results in higher competition. It can be observed that MWORA yields the best satisfaction ratio and fairness compared to ROFA-MC and ROFA-SC in both the first and last rounds. In particular, MWORA-F renders an improvement of up to 53% and 76%, as well as 28% and 78%, compared to ROFA-MC-F and ROFA-SC-F in terms of the satisfaction ratio and fairness, respectively. In addition, MWORA-L renders an improvement of up to 22% and 19%, as well as 28% and 67%, compared to ROFA-MC-L and ROFA-SC-L in terms of the satisfaction ratio and fairness, respectively. This is due to the same previously mentioned reasons.

5. Total energy consumption

Figure 3.5(e) illustrates the impact of the number of workers on the total energy consumption of workers in MWORA, ROFA-MC, and ROFA-SC. It can be observed that as the number of workers increases, the total energy consumption increases. This is because the higher the number of available workers, the higher the number of

executed tasks by workers, which increases the total energy consumption. Note that MWORA yields the highest energy consumption compared to ROFA-MC and ROFA-SC, since it executes a significantly larger number of tasks. This is due to the same aforementioned reasons. It is also worth noting that ROFA-SC renders the lowest energy consumption, since it executes the lowest number of tasks. In particular, ROFA-SC-F renders an improvement of up to 59% compared to MWORA-F and ROFA-MC-F, respectively, and ROFA-SC-L renders an improvement of up to 76% and 71% compared to MWORA-L and ROFA-MC-L, respectively.

Chapter 4

MWORA-Weighted Sum (MWORA-WS)

4.1 Problem Relaxation

We propose the MWORA-Weighted Sum (MWORA-WS) scheme a relaxed version of MWORA to derive an analytical solution as demonstrated in Section 4.2 using the Karush–Kuhn–Tucker (KKT) conditions and Lagrangian analysis. MWORA-WS compares the performance of the relaxed scheme that is constrained by a weight that sums up to 1 to MWORA.

The objective of MWORA-WS is a weighted sum of minimizing the drop rate while minimizing the total response delay. The problem is formulated as a weighted sum single-objective problem with the same binary decision variable as in MWORA. x_{ij} is set to 1 if the task t_i is assigned to w_j, or, x_{ij} is set to otherwise, according to Eq. 3.4. MWORA-WS will use the same system model and simulation setup for MWORA (described in Section 3.1 and 3.3.1)

The problem formulation is shown below:

$$\min_{x_{ij}} \hat{k} \left(|T| - \sum_{j \in W} \sum_{i \in T} x_{ij} \right) + \hat{u} \left(\sum_{j \in W} \sum_{i \in T} x_{ij} \gamma_{ij} \right) \tag{4.1}$$

4.2 Analytical Solution

MWORA-WS problem formulation shown in (4.1) is formulated as an ILP problem that has a single linear objective function and a binary decision variable, which is x_{ij}. The inequality constraints in (3.5c), (3.5d), (3.5f), (3.5g), (3.5h) are all linear constraints. In order to analytically solve the problem, we simplify it by relaxing the constraints. We first examine the problem's convexity in (4.1) by analyzing the inequality and equality constraints. The only non-convex constraint is the discrete constraint (3.5e). By removing the discrete constraint (3.5e) and replacing it with the continuous constraint as shown (4.2a) and (4.2b), thus all inequality constraints become convex.

$$\sum_{j \in N_i} x_{ij} > 0 \quad \forall i \in T \tag{4.2a}$$

$$\sum_{j \in N_i} x_{ij} \leq 0 \quad \forall i \in T \tag{4.2b}$$

After removing the discrete constraint in (4.1), Now the relaxed problem in (4.1) is convex, and the analytical solution can be attainable. Note that closed-form solutions for ILPs are not possible in general. Consequently, using Lagrangian multipliers [11] and Karush–Kuhn–Tucker (KKT) conditions [10], we aim to obtain a upper bound for the optimal objective value of x_{ij}.

The Lagrangian function of the relaxed program shown in (4.1) is given by Eq.(A.8) in Appendix A. The Lagrangian multipliers associated with the objective function of the relaxed program in (4.1), and its constraints are represented by the following vectors $\hat{\boldsymbol{k}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}^{(1)}, \hat{\boldsymbol{c}}^{(2)}, \hat{\boldsymbol{d}}, \hat{\boldsymbol{e}},$ and $\hat{\boldsymbol{f}}$, respectively. The upper bound optimal value

of x_{ij}, which is responsible for assigning task t_i to worker w_j, is shown in (4.3) :

$$x_{ij} = \begin{cases} 1 - \sum_{k \neq j} x_{ik} & \text{when } \hat{a} > 0 \\ \frac{\beta - \sum_{k \neq i} \sum_{L \neq j} x_{kL} p_{kL}}{p_{ij}} & \text{when } \hat{b} > 0 \\ \sum_{k \neq j} x_{ik} & \text{when } \hat{c}^{(1)} > 0 \\ \sum_{k \neq j} x_{ik} & \text{when } \hat{c}^{(2)} > 0 \\ \frac{\delta c_j^{\max} - \sum_{k \neq j} x_{ik} c_{ik}}{c_{ij}} & \text{when } \hat{d} > 0 \\ \frac{\epsilon_i - \sum_{k \neq i} \sum_{L \neq j} x_{kL} \gamma_{kL}}{\gamma_{ij}} & \text{when } \hat{e} > 0 \\ \frac{\psi s_j - \sum_{k \neq i} x_{kj} P_{kj}}{P_{ij}} & \text{when } \hat{f} > 0 \\ \frac{\hat{a} + \hat{b}\beta + \delta c_j^{max} + \hat{e}\epsilon_i + \hat{f}\psi s_j}{\hat{k} - \hat{u}\gamma_{ij}} & \end{cases} \tag{4.3}$$

According to the bounds obtained in (4.3), the closed-form solution is not attainable because the decision variable x_{ij} is expressed in terms of other unknown multipliers. The proof of (4.3) and the complete analytical solution can be found in Appendix A.

Moreover, we have implemented MWORA-WS using Gurobi optimizer [9] and compared it with MWORA, and the results will be shown in the following Section 4.3.

4.3 Performance Evaluation

MWORA-WS will use the same five performance metrics used for MWORA (defined in Section 3.3), and the same simulation setup and system parameters used in MWORA (mentioned in Section 3.3.1). Table 4.1 presents a summary of the simulation parameters. Unless otherwise specified.

Table 4.1: Simulation Parameters of MWORA and MWORA-WS

Parameter	Value
`number of tasks`	250
`computation intensity of tasks on average` (q_i)	20 cycles/bit
`data size of task` (L_i)	[5, 10] bits
`number of workers`	100
`price per one CPU cycle` (τ_j)	[0.1, 0.4]
`computation capability of workers` (c_{ij})	[300, 800] cycles/bit
`budget` (β)	2500

In MWORA-WS, we define the weights $\hat{k}$, and $\hat{u}$ supplied to the objective function. After conducting several experiments varying the weights $\hat{k}$, and $\hat{u}$ to sum to 1, and determining which weights for $\hat{k}$ and $\hat{u}$ perform the best based on the five performance metrics; when $\hat{k}$ was set to 0.90, and $\hat{u}$ was set to 0.10, MWORA-WS showed the best results regarding, service capacity ratio, and workers' satisfaction ratio, workers fairness.

4.3.1 Results and Analysis

To compare the performance of MWORA-WS to MWORA, we conducted four experiments; first, we varied the average task computation intensity; next, we varied the number of tasks, the budget, and lastly, the number of workers. In the experiments legend notation, MWORA will be referenced as MO (Multi-Objective), and MWORA-WS will be referenced as WS (Weighted Sum), also, in the notation, F after the scheme name indicates the first round, and L indicates the last round.

4.3.1.1 The Impact of Task Intensity

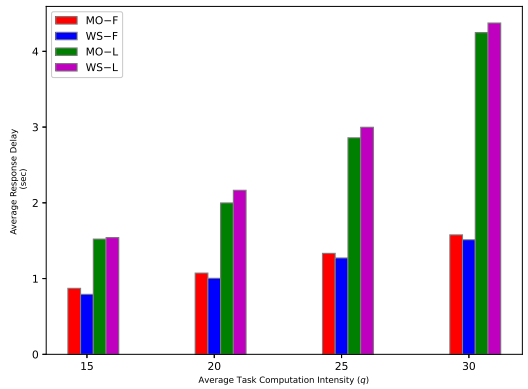
Performance results of MWORA versus MWORA-WS over varying computation intensity q; all parameters were the same as listed in Table 4.1.

1. Average response delay

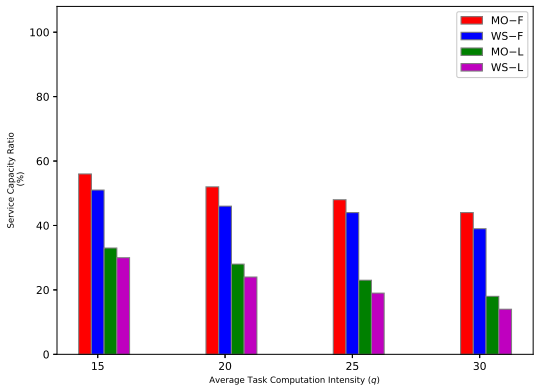
Figure 4.1(a) shows the effect of varying the average computational intensity q on the average response delay of MWORA and MWORA-WS. Note that workers in MWORA and MWORA-WS dedicate the same amount of computational capability and make the same resource allocation decisions during the first round. Both schemes need to satisfy the workers, the difference being that in MWORA-WS, the objective function is constrained by weights.

As depicted in Figure 4.1(a), MWORA and MWORA-WS show an upward trend as the average computational intensity q increases in the first and last rounds. The increasing value of q increases the average response delay since it accounts for the time needed to execute a task. MWORA shows a higher average response delay than MWORA-WS in all experiments in the first round when varying the values of q. MWORA is optimized and not weighted with specific weights as MWORA-WS. As a result, MWORA executes more tasks; as shown in Figure 4.1(b), the total delay and total executed tasks are higher for MWORA than MWORA-WS. Therefore, the average response delay for MWORA has a slight increase compared to MWORA-WS. MWORA has up to a 10% increase in an average response delay.

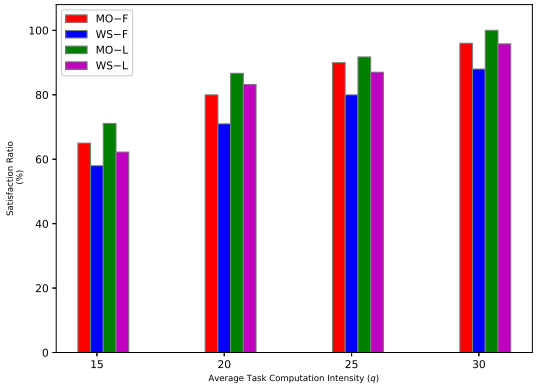
On the other hand, as seen in Figure 4.1(a), MWORA consistently outperforms MWORA-WS in the last round. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 7% in the last round, respectively. MWORA-WS has a higher average response delay than MWORA in



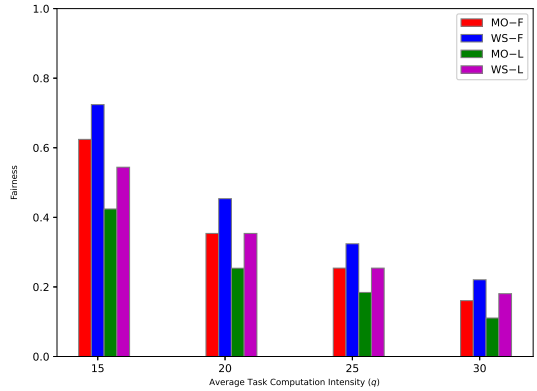
(a) Average response delay



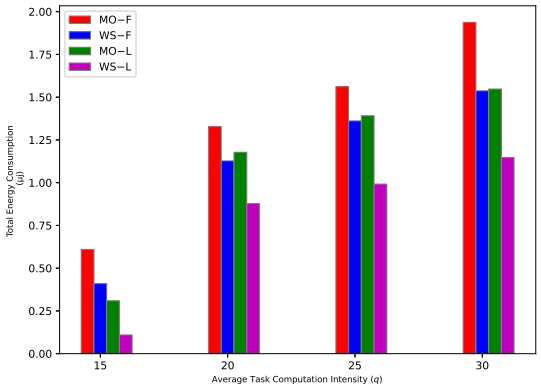
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 4.1: Performance results of MWORA and MWORA-WS over varying task computation intensity q

Note: In this set of figures MWORA is referenced as MO (Multi-Objective), and MWORA-WS is referenced as WS (Weighted-Sum), F is the first round, and L is the Last round.

the last round. Even though the workers dedicate the same computational capability, MWORA will retain a higher number of workers subscribed with the SP in the long run and, unlike MWORA-WS, is constrained by weights that limit the remaining number of subscribed workers. Unsatisfied workers who unsubscribe from SP are usually workers with higher capabilities. As a result, the average response delay in MWORA-WS in the last round is higher since the remaining workers, compared to MWORA, have lower computational capabilities to execute tasks.

2. Service capacity ratio

Figure 4.1(b) assess the service capacity ratio of MWORA and MWORA-WS, As depicted in Figure 4.1(b) the service capacity decreases in both the first and last rounds for both schemes as q increases, because when the computational intensity increases. As a result, the price of executing the tasks increases, which limits the number of tasks that can be executed within the SP budget limit.

MWORA outperforms MWORA-WS in both the first and last rounds since weights constrain MWORA-WS. Therefore the amount of executed tasks is limited due to the weights supplied; as a result, MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 28% in the last round, respectively. MWORA-WS shows a higher gap percentage difference in the last round because a higher number of workers in MWORA continue to subscribe with SP compared to MWORA-WS. As a result, MWORA is executing more tasks.

3. Workers satisfaction ratio

Figure 4.1(c) depicts the workers' satisfaction ratio of MWORA and MWORA-WS over varying q. Note that the workers' satisfaction ratio increases as q increases in

both schemes' first and last rounds. This is because the price increases for tasks with higher computational intensity. Hence, workers are rewarded more when performing tasks with higher computational intensity, and their satisfaction levels consequently increase for both MWORA and MWORA-WS in the first and last rounds.

As depicted, MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 15% in the last round, respectively. MWORA offers superior performance compared to MWORA-WS because MWORA, in each round, retains more workers operating with SP than MWORA-WS.

4. Workers fairness

Figure 4.1(d) shows the workers fairness index of MWORA, and MWORA-WS over varying q. Recall that a lower fairness index implies better fairness in fairness Eq. 3.6. As depicted in Figure 4.1(d), MWORA surpasses MWORA-WS in the achieved fairness in the first and last rounds, for varying levels of q, since MWORA-WS is constrained with weights. As a result, workers in MWORA-WS are not executing the number of tasks needed to render the satisfactory profit needed by workers compared to MWORA workers. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 27%, in the first round and up to a 38% in the last round, respectively.

5. Total energy consumption

Figure 4.1(e) shows the impact of varying q on the total energy consumption of workers in MWORA, and MWORA-WS. As shown in Figure 4.1(b) The energy consumption increases because the intensity of tasks increases, which causes the workers to expend more energy during their execution. MWORA-WS shows a lower energy

consumption for both the first and last rounds, when compared to MWORA since workers in MWORA-WS are executing a fewer tasks. For both schemes, the last round shows lower energy consumption than the first round since fewer workers are available. Also, in the first and last round, MWORA performs more tasks than MWORA-WS, as shown in Figure 4.1(b), hence, exhibiting higher energy consumption than MWORA-WS. as a result in the first and last round, MWORA yields up to a 46% and 55% higher energy consumption than MWORA-WS, respectively.

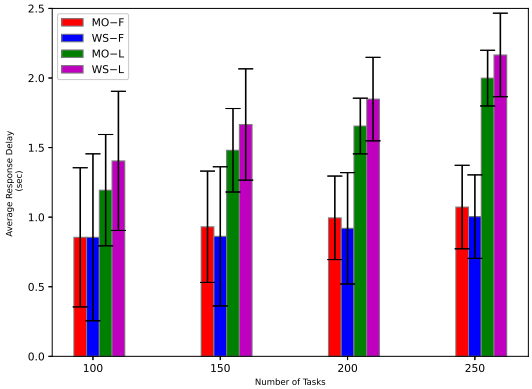
4.3.1.2 The Impact of the Number of Tasks

Performance results of MWORA versus MWORA-WS when varying the number of tasks. 4.1.

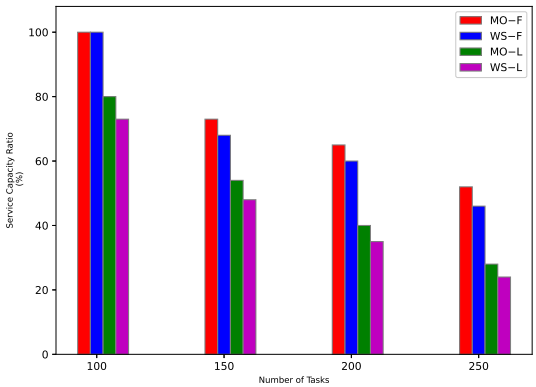
1. Average response delay

As shown in Figure 4.2(a), the first round, both schemes have identical values for the average response delay when the number of tasks was set to 100, this is because the number of tasks equals the number of workers. Each worker only takes one task to gain profit. When we increase the number of tasks in the next experiment, we will start seeing the difference between the MWORA and MWORA-WS.

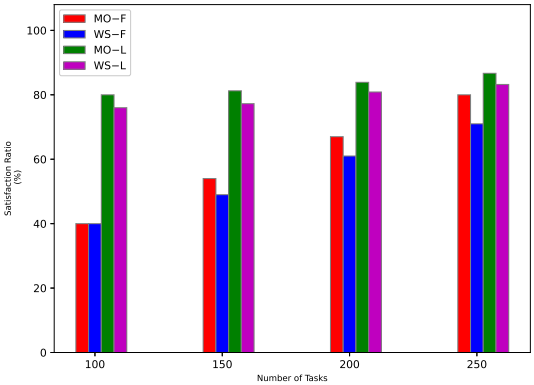
For the first round, with the increase in the number of tasks, the average response delay for both schemes is increased because more tasks are executed. MWORA in the first round has a higher average delay than MWORA-WS, this is because MWORA executes more tasks, as shown in Figure 4.2(b). Because of that, the total delay and the total executed tasks is higher for MWORA than MWORA-WS. This is similar to what was shown in the previous experiment when varying average task intensity. Therefore, the average response delay for MWORA has a slight increase compared to



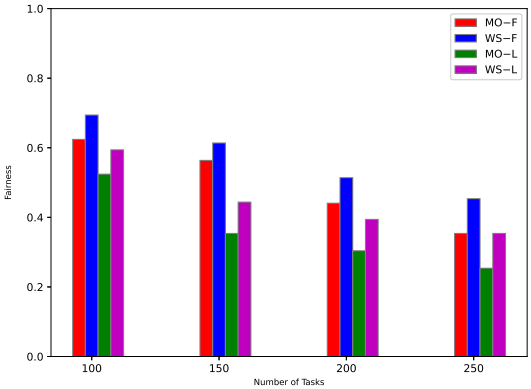
(a) Average response delay



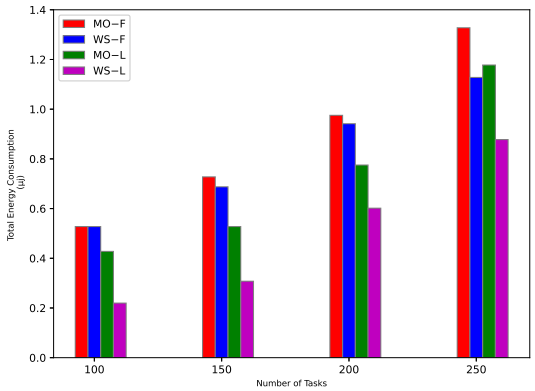
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 4.2: Performance results of MWORA and MWORA-WS over varying number of tasks

Note: In this set of figures MWORA is referenced as MO (Multi-Objective), and MWORA-WS is referenced as WS (Weighted-Sum), F is the first round, and L is the Last round.

MWORA-WS. MWORA has up to a 8% increase in average response delay.

However, as shown in Figure 4.2(a), MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a gap of up to 15% in the last round. The average response delay in MWORA-WS is greater than in MWORA, this is because, in the long run, MWORA will retain more workers with higher capabilities subscribed to the SP compared to MWORA-WS. As a result, MWORA consistently outperforms MWORA-WS in the last round.

2. Service capacity ratio

As demonstrated in Figure 4.2(b). Similar to Figure 4.2(a), in the first round, both schemes had identical values when the number of tasks was set to 100 this is because each task is assigned to one worker and the number of tasks equals the number of the workers. With the increase in the number of tasks in the next experiment, we will see MWORA executing more tasks compared to MWORA-WS since in MWORA-WS, the amount of executed tasks is limited due to the sum of weights supplied. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 16% in the last round.

However, as a service capacity ratio percentage, it is decreasing when compared to the preceding experiment; this is because, as we mentioned in Section 3.3 in the second point of the performance metrics, service capacity is calculated based on the number of successfully executed tasks divided by a total number of tasks. Therefore, as a ratio, the percentage decreases when the number of tasks increases.

3. Workers satisfaction ratio

We evaluate the performance of MWORA and MWORA-WS in terms of workers' satisfaction ratio in Figure 4.2(c). The workers' satisfaction ratio increases as the

number of tasks increase for both MWORA and MWORA-WS in the same round, this is because more tasks are available for workers to be executed. Therefore, workers are compensated more when they perform more tasks, and their levels of satisfaction rise accordingly. It bears repeating that the workers' satisfaction ratio is calculated by considering the remaining number of workers in the system at each round. Consequently, in the subsequent round, the satisfaction ratio of workers will be higher than in the preceding round.

In addition, As depicted, MWORA offers superior performance compared to MWORA-WS. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 6% in the last round, and this is because MWORA in each round retains more workers operating with the SP than MWORA-WS and executing more tasks compared to MWORA-WS as shown in Figure 4.2(b).

4. Workers fairness

As seen in Figure 4.2(d) MWORA is superior to MWORA-WS in terms of the fairness achieved in the first and last rounds, for varying numbers of tasks, because MWORA-WS is constrained by weights, which prevents its workers from performing enough tasks to generate as much profit as their MWORA counterparts. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 22%, in the first round and up to a 29% in the last round. Remember that in fairness Eq. 3.6, the lower the value, the fairer the system.

5. Total energy consumption

We study the impact of the total energy consumed by the workers in Figure 4.2(e).

MWORA and MWORA-WS show a higher energy consumption than in the preceding experiment when increasing the number of tasks as both schemes are executing more tasks, as shown in Figure 4.2(b). However, MWORA-WS shows lower energy consumption for both the first and last rounds since workers in MWORA-WS are executing a lower number of tasks, as shown in Figure 4.2(b) compared to MWORA. Except for the first round when the number of tasks was set to 100. MWORA and MWORA-WS show identical energy consumption since they execute the same number of tasks.

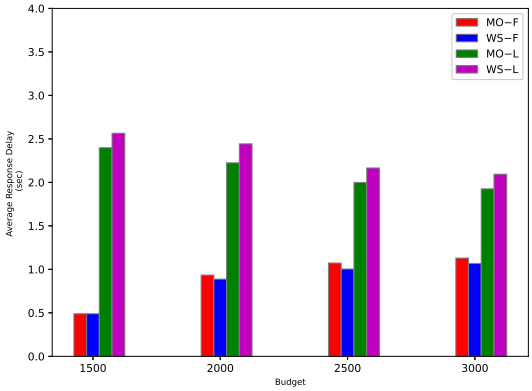
Moreover, the last round shows lower energy consumption for both schemes compared to the first round since fewer workers are available for both schemes than in the first round. Also, in the first and last rounds, MWORA performs more tasks than MWORA-WS; as a result, MWORA exhibits a higher energy consumption compared to MWORA-WS. MWORA yields up to a 17% and 34% higher energy consumption compared to MWORA-WS in the first and last rounds, respectively.

4.3.1.3 The Impact of Budget

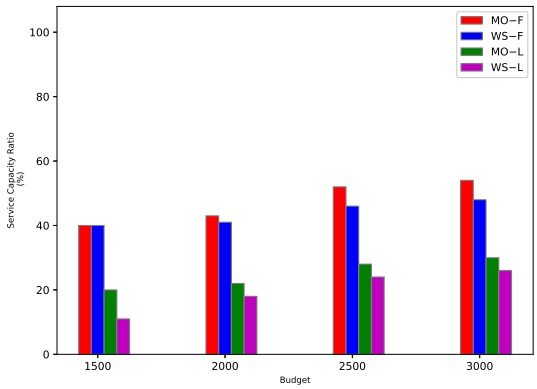
Performance results of MWORA versus MWORA-WS when varying the budget β that the SP uses to recruit available workers.

1. Average response delay

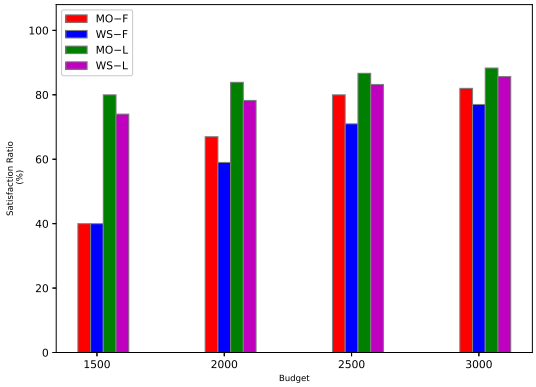
As shown in Figure 4.3(a), the first round, both schemes have identical values for the average response delay when the budget was set to 1500 since this budget was enough to execute 100 tasks, as shown in Figure 4.3(b) only 40% of the tasks were executed from a total of 250 tasks, this is because the number of tasks equals the number of workers. In the final round MWORA has a higher average delay since



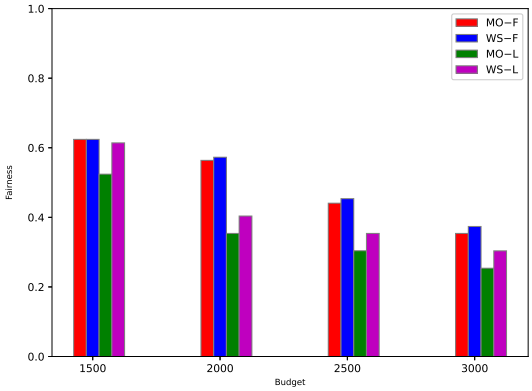
(a) Average response delay



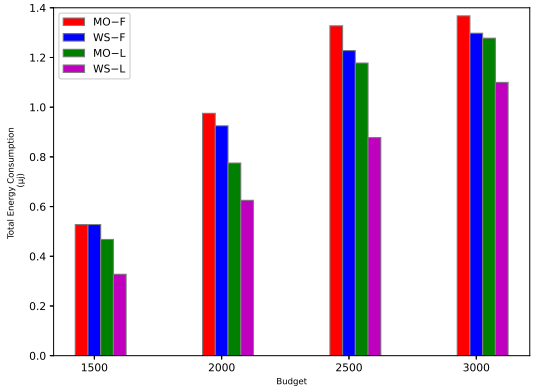
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 4.3: Performance results of MWORA and MWORA-WS over varying budget

Note: In this set of figures MWORA is referenced as MO (Multi-Objective), and MWORA-WS is referenced as WS (Weighted-Sum), F is the first round, and L is the Last round.

MWORA executes more tasks, as shown in Figure 4.3(b). Consequently, the total delay of MWORA and the total executed tasks is higher for MWORA than MWORA-WS, similar to the behavior in the previous experiments when the number of tasks and task intensity were varied. Therefore, the average response delay for MWORA increased up to 6% compared to MWORA-WS.

However, Figure 4.3(a) shows that MWORA consistently outperforms MWORA-WS in the last round. For the last round, the average response delay in MWORA-WS is higher than in MWORA. This is because, in the long run, MWORA will retain more workers with higher capabilities subscribed to the SP than MWORA-WS. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to a 9% in the last round.

2. Service capacity ratio

As depicted in Figure 4.3(b). Similar to Figure 4.3(a), in the first round, both schemes had identical values when the budget was set to 1500. This amount was enough to execute 40% (100) of the available tasks. Given there were 100 workers, each worker was assigned one task. With the increase in the budget in the next experiment, we will see MWORA is executing more tasks compared to MWORA-WS since in MWORA-WS amount of executed tasks is limited due to the sum of weights supplied.

MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 22% in the last round. In contrast, MWORA surpasses MWORA-WS in the last round since MWORA continues to operate with the SP with more workers; consequently, more tasks are executed.

3. Workers satisfaction ratio

We evaluate the performance of MWORA and MWORA-WS in terms of workers' satisfaction ratio in Figure 4.3(c). The workers' satisfaction ratio increases as the budget increase for both MWORA and MWORA-WS in the same round, and this is because more budget is available for workers to execute more tasks. Therefore, workers are compensated more when they perform more tasks, and their levels of satisfaction rise accordingly.

In addition, as depicted in Figure 4.3(c), MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 13%, in the first round and up to a 8% in the last round, and this is because MWORA in each round retains more workers operating with SP than MWORA-WS and executing more tasks compared to MWORA-WS as shown in Figure 4.3(b). It bears repeating that the workers' satisfaction ratio is calculated by considering the remaining number of workers in the system at each round. Consequently, in the subsequent round, the satisfaction ratio of workers will be higher than in the preceding round of the same scheme.

4. Workers fairness

As seen in Figure 4.3(d), which shows that MWORA is superior to MWORA-WS in terms of the fairness achieved in the first and last rounds when varying budget because MWORA-WS is constrained by weights, which prevents its workers from performing sufficient tasks to generate as much profit as their MWORA counterparts. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 5%, in the first round and up to a 16% in the last round. We observe that the gap between the first and last rounds of each scheme decreases as

the budget increases because, as mentioned before, more workers are subscribed to the SP for both schemes in the last round.

Moreover, in the last round MWORA and MWORA-WS display identical values in the first round when the budget is set to 1,500, as shown in Figure 4.3(b), and this is because both schemes execute the same number of tasks.

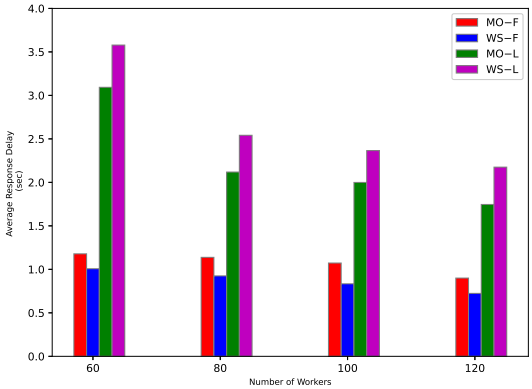
5. Total energy consumption

We study the impact of the total energy consumed by the workers. As shown in Figure 4.3(e), MWORA and MWORA-WS consume more energy as the budget increases because they are executing more tasks. As shown in Figure 4.3(b), however, MWORA-WS has lower energy consumption for both the first and last rounds because workers in MWORA-WS are executing fewer tasks. In the first experiment, when the budget was only 1500, MWORA and MWORA-WS executed the same number of tasks. As a result that their energy consumption is identical.

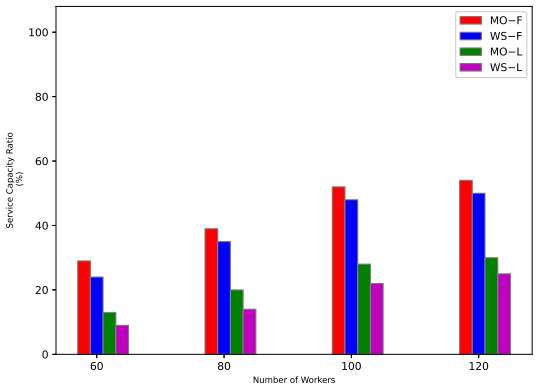
Moreover, for both schemes, energy consumption is lower in the last round compared to the first round, as there are fewer available workers in the last round, so executing a fewer number of tasks. MWORA exhibits higher energy consumption than MWORA-WS because it executes more tasks in the last round compared to MWORA-WS, as shown in Figure 4.3(b). MWORA yields up to a 18% and 34% higher energy consumption compared to MWORA-WS in the first and last round, respectively.

4.3.1.4 The Impact of the Number of Workers

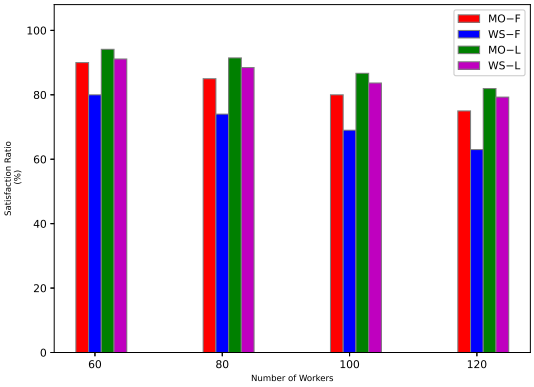
Finally, Performance results of MWORA versus MWORA-WS when varying the number of workers.



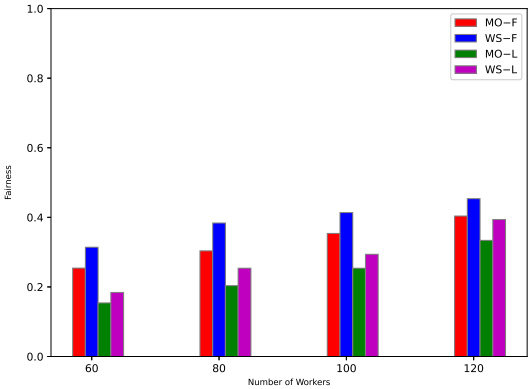
(a) Average response delay



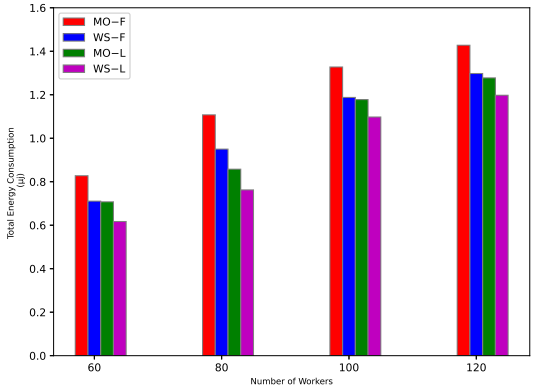
(b) Service capacity ratio



(c) Workers satisfaction ratio



(d) Workers fairness



(e) Total energy consumption of workers

Figure 4.4: Performance results of MWORA and MWORA-WS over varying number of workers

Note: In this set of figures MWORA is referenced as MO (Multi-Objective), and MWORA-WS is referenced as WS (Weighted-Sum), F is the first round, and L is the Last round.

1. Average response delay

As shown in Figure 4.4(a), for the first round, with the increase in the number of workers, the average response delay for both schemes is decreased as more tasks are executed since more workers are available consequently, the total delay increases as the number of executed tasks increases, and the delay decreases.

However, in the final round MWORA has a higher average delay than MWORA-WS, and this is because MWORA executes more tasks, as shown in Figure 4.4(b). Because of that, the total delay of MWORA and the total executed tasks is higher for MWORA than MWORA-WS in the same experiment. Therefore, the average response delay for MWORA has a slight increase compared to MWORA-WS. MWORA has up to a 16% increase in the average response delay.

Moreover, as shown in Figure 4.4(a), MWORA consistently outperforms MWORA-WS in the last round. For the last round results, the average response delay in MWORA-WS is greater than in MWORA, this is because, in the long run, MWORA will retain more workers with higher capabilities subscribed with the SP compared to MWORA-WS. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to a 20% in the last round.

2. Service capacity ratio

As illustrated in Figure 4.4(b). When increasing the number of workers, the service capacity ratio increases compared to the preceding experiment since more workers are available, therefore, more tasks are executed from the total available tasks. We will see MWORA executing more tasks than MWORA-WS in each experiment since the MWORA-WS amount of executed tasks is limited due to the sum of weights supplied. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding

a small gap of up to 20%, in the first round and up to a 44% in the last round.

3. Workers satisfaction ratio

We evaluate the performance of MWORA and MWORA-WS in terms of workers' satisfaction ratio in Figure 4.4(c). The workers' satisfaction ratio decreases as the number of workers increases for both MWORA and MWORA-WS in the same round, this is because the SP cannot easily satisfy all the workers from the preceding experiment in addition to the new workers in the current experiment. It bears repeating that the workers' satisfaction ratio is calculated by considering the remaining number of workers in the system at each round. Consequently, in the subsequent round, the satisfaction ratio of workers will be higher than in the preceding round.

As depicted in Figure 4.4(c), MWORA offers superior performance in the first round compared to MWORA-WS. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 19%, in the first round and up to a 4% in the last round compared to MWORA-WS this is because MWORA in each round retains more workers operating with the SP than MWORA-WS and is executing more tasks compared to MWORA-WS as shown in Figure 4.4(b).

4. Workers fairness

As seen in Figure 4.4(d) which shows that MWORA is superior to MWORA-WS in terms of the fairness achieved in the first and last rounds, when varying the numbers of workers, because MWORA-WS is constrained by weights, which prevents its workers from performing enough tasks to generate as much profit as their MWORA counterparts. But as the number of workers increases, we observe an increase in the fairness value (recall that, according to the Jain fairness equation, the lower the value, the fairer the system). Indicating that the SP was not sufficiently fair to the

workers since the SP budget is insufficient to satisfy all workers. MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 21%, in the first round and up to a 19% in the last round.

5. Total energy consumption

The impact of the total energy consumed by the workers is illustrated in Figure 4.4(e). MWORA and MWORA-WS show a higher energy consumption than in the preceding experiment when increasing the number of workers as both MWORA and MWORA-WS are executing more tasks, as shown in Figure 4.4(b). However, MWORA-WS shows lower energy consumption for both the first and last rounds since workers in MWORA-WS are executing a less tasks, compared to MWORA as shown in Figure 4.4(b).

In addition, the last round of both schemes exhibits lower energy consumption than the first round of each scheme, as fewer workers are available in the final round of each scheme than in the first round. Furthermore, MWORA performs more tasks than MWORA-WS in the first and last rounds. As a result, MWORA has a higher energy consumption than MWORA-WS. MWORA yields up to a 16% and 13% higher energy consumption compared to MWORA-WS in the first and last round, respectively.

Chapter 5

Conclusion and Future Work

5.1 Summary and Conclusion

This thesis proposes the Multitiered Worker-oriented Resource Allocation (MWORA) scheme. MWORA utilizes the abundant yet underutilized computational resources of EEDs. In contrast to existing baseline resource allocation schemes, MWORA considers that EEDs are user-owned devices and are subject to dynamic user access behavior, which can impact their computational capabilities and introduce a human factor related to preserving the user's Quality of Service (QoS). To address this issue, MWORA fosters multitiered computational capabilities granted by workers, compliant with what the user is willing to sacrifice based on the corresponding reward of the offloaded task. In addition, MWORA achieves fair resource allocation from the worker's perspective.

Maintaining the demanded profit encourages workers to remain in the system and recurrently donate their computational services, which can significantly improve the QoS overall. Extensive simulations have shown that MWORA achieves significant improvements compared to other baseline resource allocation schemes regarding average

response delay, service capacity, worker satisfaction ratio, and fairness.

Moreover, we have compared MWORA, the multi-objective scheme to MWORA-Weighted Sum (MWORA-WS). Simulation results have shown that in the long term, MWORA-WS closely approaches the optimal solution rendered by MWORA, yielding a small gap of up to 7%, 16%, 8%, 16%, and 13% in terms of average response delay, service capacity ratio, worker satisfaction ratio, fairness, and total energy consumption, respectively.

5.2 Future Work

Future work could include implementing MWORA in a distributed architecture instead of a centralized architecture and using game theory to solve the resource allocation problem. In game theory, each entity is treated as a player who would allocate the resources. In addition, since MWORA uses binary offloading, and matching theory works with binary offloading, using matching theory to study the relation between users and service providers would be novel research in resource allocation. In addition, a challenge that will face MWORA in implementing it in practice can we trust all the information provided by the workers regarding their computational capabilities. To address this challenge, many research efforts are investigating benchmark techniques to asses the worker's current state and capabilities.

References

[1] GSMA, "The mobile economy 2022." Accessed on 2022, August. [Online]. Available: https://www.gsma.com/mobileeconomy/wp-content/uploads/2022/02/280222-The-Mobile-Economy-2022.pdf

[2] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[3] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng, "Mobile cloud computing: Challenges and future research directions," *Journal of Network and Computer Applications*, vol. 115, pp. 70–85, 2018.

[4] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat, "Democratizing the network edge," *ACM SIGCOMM Computer Communication Review*, vol. 49, pp. 31–36, 2019.

[5] J. Portilla, G. Mujica, J.-S. Lee, and T. Riesgo, "The extreme edge at the bottom of the internet of things: A review," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3179–3190, 2019.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] M. Kaur and R. Aron, "A systematic study of load balancing approaches in the fog computing environment," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 9202–9247, 2021.

[8] KDS, "Kings distributed systems." Accessed on 2022, August. [Online]. Available: https://kingsds.network/

[9] Gurobi, "Gurobi optimizer reference manual." Accessed on 2022, August. [Online]. Available: https://www.gurobi.com/

[10] S. I. Gass and M. C. Fu, Eds., *Karush-Kuhn-Tucker (KKT) Conditions*. Boston, MA: Springer US, 2013, pp. 833–834. [Online]. Available: https://doi.org/10.1007/978-1-4419-1153-7_200359

[11] F. Antonio, "About lagrangian methods in integer optimization," *Annals of Operations Research*, vol. 139, pp. 163–, 10 2005.

[12] M. Ahmed, A. Sina, R. Chowdhury, and M. Ahmed, "An advanced survey on cloud computing and state-of-the-art research issues," *IJCSI International Journal of Computer Science*, 2012.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 13–16. [Online]. Available: https://doi.org/10.1145/2342509.2342513

[14] Ruay-Shiung-Chang, J. Gao, V. Gruhn, J. He, G. Roussos, and W. Tsai, "Mobile cloud computing research - issues, challenges and needs," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE Computer Society, 2013, pp. 442–453.

[15] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166 079–166 108, 2019.

[16] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[17] Cisco, "Fog computing and the internet of things: Extend the cloud to where the things are, white paper, cisco, san jose, ca," Cisco, Tech. Rep., 2015. [Online]. Available: https://www.cisco.com/c/dam/enus/solutions/trends/iot/docs/computing-overview.pdf

[18] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[19] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17302224

[20] T. L. Foundation, "State of the edge report." Accessed on 2022, August. [Online]. Available: https://stateoftheedge.com/reports/state-of-the-edge-report-2021/

[21] MarketsAndMarkets, "Edge computing market by component, application, organization size, vertical and region - global forecast to 2025." Accessed on 2022, August. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/edge-computing-market-133384090.html

[22] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, "A survey of computation offloading in edge computing," in *2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2020, pp. 1–6.

[23] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications surveys and tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[24] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[25] Z. Lin, J. Liu, J. Xiao, and S. Zi, "A survey: Resource allocation technology based on edge computing in iiot," in *International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, 2020, pp. 1–5.

[26] H. Zhu and C. Huang, "Cost-efficient vnf placement strategy for iot networks with availability assurance," in *IEEE Vehicular Technology Conference*, vol. 2017-, 2018, pp. 1–5.

[27] Y. Deng, Z. Chen, X. Yao, S. Hassan, and A. M. A. Ibrahim, "Parallel offloading in green and sustainable mobile edge computing for delay-constrained iot system," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 202–12 214, 2019.

[28] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[29] J. Meng, C. Zeng, H. Tan, Z. Li, B. Li, and X.-Y. Li, "Joint heterogeneous server placement and application configuration in edge computing," in *IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 488–497.

[30] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–6.

[31] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 420–423, 2018.

[32] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4832–4841, 2020.

[33] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju, "Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing," *IEEE Transactions on Services Computing*, vol. 9, no. 6, pp. 895–909, 2016.

[34] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

[35] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service-aware approaches in fog computing," *International Journal of Communication Systems*, vol. 33, no. 8, p. e4340, 2020, e4340 IJCS-19-0006.R2.

[36] R. F. ElKhatib, S. A. ElSayed, N. Zorba, and H. S. Hassanein, "Optimal proactive resource allocation at the extreme edge," *IEEE International Conference on Communications (ICC)*, 2022.

[37] I. M. Amer and S. Sorour, "Cost-based compute cluster formation in edge computing," *IEEE International Conference on Communications (ICC)*, 2022.

[38] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, 2019.

[39] S. Xiao, C. Liu, K. Li, and K. Li, "System delay optimization for mobile edge computing," *Future Generation Computer Systems*, vol. 109, pp. 17–28, 2020.

[40] P. Wang, K. Li, B. Xiao, and K. Li, "Multi-objective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing," *IEEE internet of things journal*, pp. 1–1, 2021.

[41] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 2992–3005, 2021.

[42] S. Zarandi and H. Tabassum, "Delay minimization in sliced multi-cell mobile edge computing (mec) systems," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1964–1968, 2021.

[43] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency," *IEEE Communications Letters*, vol. 24, no. 2, pp. 307–311, 2020.

[44] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2013, pp. 26–30.

[45] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[46] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[47] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[48] Q. Jia, R. Xie, T. Qinqin, X. Li, T. Huang, J. Liu, and Y. Liu, "Energy-efficient computation offloading in 5g cellular networks with edge computingand d2d communications," *IET Communications*, vol. 13, 05 2019.

[49] B. Polyak and P. Shcherbakov, "Lyapunov functions: An optimization theory perspective, *this work was supported by the russian scientific foundation, project no. 16-11-10015," *IFAC-PapersOnLine*, vol. 50, pp. 7456–7461, 2017.

[50] R. Wang, C. Zang, P. He, Y. Cui, and D. Wu, "Auction-based profit maximization offloading in mobile edge computing," *Digital Communications and Networks*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822000475

[51] Q. Wang, S. Guo, Y. Wang, and Y. Yang, "Incentive mechanism for edge cloud profit maximization in mobile edge computing," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[52] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, 2021.

[53] S. Singh and D. Ho Kim, "Profit optimization for mobile edge computing using genetic algorithm," in *2021 IEEE Region 10 Symposium (TENSYMP)*, 2021, pp. 1–6.

[54] R. Jain, D. M. Chiu, and H. WR, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *CoRR*, vol. cs.NI/9809099, 1998.

[55] U. Yaqub and S. Sorour, "Multi-objective resource optimization for hierarchical mobile edge computing," in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018.

Appendix A

KKT and Lagrangian Analysis of the ILP Problem

Applying KKT conditions to the constraints in (3.5c), (3.5d), (3.5f), (3.5g), (3.5h), (4.2a), (4.2b) and solving each one with respect to x_{ij} results in having the following set of equations:

$$\hat{a}\left(\sum_{j\in\omega} x_{ij} - 1\right) = 0 \quad \forall i \in T \text{ and } \forall j \in W \tag{A.1a}$$

$$\hat{a}\sum x_{ij} = \hat{a} \tag{A.1b}$$

$$x_{ij} = 1 - \sum_{k\neq j} x_{ik} \quad \text{when } \hat{a} > 0 \tag{A.1c}$$

$$\hat{b}\left(\sum_{j\in\omega}\sum_{i\in T} x_{ij}p_{ij} - \beta\right) = 0 \quad \forall i \in T \text{ and } \forall j \in W \tag{A.2a}$$

$$\hat{b}\sum_{j\in\omega}\sum_{i\in T} x_{ij}p_{ij} = \hat{b}\beta \tag{A.2b}$$

$$x_{ij} = \frac{\beta - \sum_{k\neq i}\sum_{L\neq j} x_{kL}p_{kL}}{p_{ij}} \quad \text{when } \hat{b} > 0 \tag{A.2c}$$

$$\hat{c}^{(1)}\left(\sum_{j\in N_i} x_{ij} = 0\right) \quad \forall i \in T \tag{A.3a}$$

$$x_{ij} = \sum_{k\neq j} x_{ik} \quad \text{when } \hat{c}^{(1)} > 0 \tag{A.3b}$$

$$\hat{c}^{(2)}\left(\sum_{j\in N_i} x_{ij} = 0\right) \quad \forall i \in T \tag{A.4a}$$

$$x_{ij} = \sum_{k\neq j} x_{ik} \quad \text{when } \hat{c}^{(2)} > 0 \tag{A.4b}$$

$$\hat{d}\left(\sum_{i\in T} x_{ij}c_{ij} - \delta c_j^{\max}\right) = 0 \quad \forall j \in W \tag{A.5a}$$

$$\hat{d}\left(\sum_{i\in T} x_{ij}c_{ij}\right) = \hat{d}\delta c_j^{\max} \tag{A.5b}$$

$$x_{ij} = \frac{\delta c_j^{\max} - \sum_{k\neq j} x_{ik}c_{ik}}{c_{ij}} \quad \text{when } \hat{d} > 0 \tag{A.5c}$$

$$\hat{e}\left(\sum_{j\in\omega}\sum_{i\in T} x_{ij}\gamma_{ij} - \epsilon_i\right) = 0 \tag{A.6a}$$

$$\hat{e}\sum_{j\in\omega}\sum_{i\in T} x_{ij}\gamma_{ij} = \hat{e}\epsilon_i \tag{A.6b}$$

$$x_{ij} = \frac{\epsilon_i - \sum_{k\neq i}\sum_{L\neq j} x_{kL}\gamma_{kL}}{\gamma_{ij}} \quad \text{when } \hat{e} > 0 \tag{A.6c}$$

$$\hat{f}\left(\psi s_j - \sum_{i\in T} x_{ij}p_{ij}\right) = 0 \quad \forall j \in W \tag{A.7a}$$

$$\hat{f}\sum_{i\in T} x_{ij}p_{ij} = \hat{f}\psi s_j \tag{A.7b}$$

$$x_{ij} = \frac{\psi s_j - \sum_{k\neq i} x_{kj}P_{kj}}{P_{ij}} \quad \text{when } \hat{f} > 0 \tag{A.7c}$$

The Lagrangian function associated with the relaxed program in 4.1 is given by the following Eq.(A.8):

$$\begin{aligned}
&\mathcal{L}(\hat{k},\hat{u},\hat{a},\hat{b},\hat{c}^{(1)},\hat{c}^{(2)},\hat{d},\hat{e},\hat{f}) = \\
&\hat{k}\left(|T| - \sum_{j\in W\in}\sum_{i\in T} x_{ij}\right) + \hat{u}\left(\sum_{j\in W\in}\sum_{i\in T} x_{ij}\gamma_{ij}\right) + \\
&\hat{a}\left(\sum_{j\in W} x_{ij} - 1\right) + \\
&\hat{b}\left(\sum_{j\in W}\sum_{i\in T} x_{ij}p_{ij} - \beta\right) + \\
&\hat{c}^{(1)}\left(\sum_{j\in N_i} x_{ij}\right) + \hat{c}^{(2)}\left(\sum_{j\in N_i} x_{ij}\right) + \\
&\hat{d}\left(\sum_{i\in T} x_{ij}c_{ij} - \delta c_j^{\max}\right) + \\
&\hat{e}\left(\sum_{j\in W}\sum_{i\in T} x_{ij}\gamma_{ij} - \epsilon_i\right) + \\
&\hat{f}\left(\psi s_j - \sum_{i\in T} x_{ij}P_{ij}\right)
\end{aligned} \tag{A.8}$$

Solving the Lagrangian function in (A.8) to get the gradient of $\mathcal{L}$ with respect to x_{ij} at the optimal point :

$$\frac{\Delta \mathcal{L}}{\Delta x_{ij}} = -\hat{k} + \hat{u}\gamma_{ij} + \hat{a} + \hat{b}P_{ij} + \hat{c}^{(1)} + \hat{c}^{(2)} + \hat{d}c_{ij} + \hat{e}\gamma_{ij} - \hat{f}P_{ij} \tag{A.9}$$

Multiplying both sides of Eq.(A.9) by x_{ij} we get:

$$\begin{aligned} -\hat{k}x_{ij} + \hat{u}x_{ij}\gamma_{ij} + \hat{a}x_{ij} + \hat{b}x_{ij}p_{ij} + \\ \hat{c}^{(1)}x_{ij} + \hat{c}^{(2)}x_{ij} + \hat{d}x_{ij}c_{ij} + \hat{e}x_{ij}\gamma_{ij} - \hat{f}x_{ij}P_{ij} = 0 \end{aligned} \tag{A.10}$$

using Eqs.(A.1b),(A.2b),(A.3b),(A.4b),(A.5b),(A.6b),(A.7b) in Eq.(A.10) we get:

$$-\hat{k}x_{ij} + \hat{u}x_{ij}\gamma_{ij} + \hat{a} + \hat{b}\beta + \delta c_j^{\max} + \hat{e}\epsilon_i + \hat{f}\psi s_j = 0 \tag{A.11}$$

solving Eq.(A.11) with respect to x_{ij} we get the following Eq.(A.12) which shows the upper bound of x_{ij}.

$$x_{ij} = \frac{\hat{a} + \hat{b}\beta + \delta c_j^{\max} + \hat{e}\epsilon_i + \hat{f}\psi s_j}{\hat{k} - \hat{u}\gamma_{ij}} \tag{A.12}$$

upper bound values for x_{ij} shown in Eq.(4.3) are obtained from Eqs.(A.1c),(A.2c),(A.3b),(A.4b),(A.5c),(A.6c),(A.7c) and (A.12)