TETRA: Time- and Energy-Aware TOPSIS-based Resource Allocation

Sri Vibhu Paruchuri

A thesis
submitted in partial fulfillment
of the requirements for the degree of

Master of Science

University of Washington

2024

Committee:
Eyhab Al-Masri
Hossam Fattah
Wei Cheng

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

TETRA: Time- and Energy-Aware TOPSIS-based Resource Allocation

Sri Vibhu Paruchuri

Chair of the Supervisory Committee:

Eyhab Al-Masri

School of Engineering & Technology

With the exponential growth of IoT devices, there has been an increasing demand for distributed computing paradigms such as edge computing and fog computing to address the limitations of cloud computing. Resource scheduling is a critical aspect across the different layers, as it ensures that the available resources are efficiently utilized and allocated to different tasks. Most of the existing resource scheduling algorithms for fog computing environments focus primarily on performance metrics such as makespan, resource utilization, and cost separately. However, there is a need for dynamic multi-objective optimization techniques that can be energy-aware while not compromising on makespan. In this thesis, we introduce a novel resource scheduling algorithm for fog computing environments that optimizes time and energy consumption, which ensures higher performance and lower data center costs. The algorithm considers all the available Virtual Machines (VM) in the fog computing environment. Then, it uses the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), which is a multi-criteria decision analysis (MCDA) method, to identify the optimal resources. Our algorithm considers multiple computational parameters such as Million Instructions Per Second (MIPS), the number of processing cores, and thermal design power (TDP) to rank available resources. We conducted a series of experiments, and our algorithm achieves a multi-objective optimization for scheduling IoT tasks on higher-ranked resources resulting in a 7%, 19% and 25% optimization rates in makespan over Best-Fit, Greedy and First-Fit algorithms respectively. In addition, the optimizations in energy consumption over the Best-Fit, Greedy and First-Fit algorithms from our experiments were 1%, 41% and 27%, respectively.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IoT	Internet of Things
MCDA	Multi-Criteria Decision Analysis
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
TETRA	A ***T***ime and ***E****nergy-aware* ***T****OPSIS-based* ***R***esource scheduling ***A***lgorithm
MIPS	Million instructions per second
CPU	Central Processing Unit
TDP	Thermal Design Power
MEC	Multi-Access Edge Computing
BFA	Best-Fit Algorithm
GA	Greedy Algorithm
FFA	First-Fit Algorithm
FFTA	First-Fit with TOPSIS Algorithm
LRC	Low Resource Competition
MRC	Medium Resource Competition
HRC	High Resource Competition
PI	Performance Improvement
s	Seconds
kJ	Kilo Joules

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Eyhab Al-Masri, for his invaluable guidance, insights, and support throughout the research process. Without his help and encouragement, I would not have been able to complete this research.

I am also thankful to my research partner, Ashwin Meena Meiyappan, for their assistance when I was exploring different fields to find a suitable thesis topic.

I am also grateful to my committee members, Dr. Hossam Fattah and Dr. Wei Cheng, for their support and words of encouragement.

Lastly, I am indebted to my parents, my brother and sister-in-law, and my dearest nephew, for all their love and support. Without them, this research project would not have been possible.

Dedication

To my friends and family, for their encouragement and unwavering support.

Declaration

I declare that I am the sole author of this thesis. Parts of this thesis are based on existing publications and/or have been/will be submitted for consideration for possible conference and/or journal publication. Copyright may be transferred in the future without notice. A digital copy of this thesis may be available at the University of Washington ResearchWorks database.

The overall contributions of this thesis are based on the following publications:

1. A journal paper which, at the time of writing, is in progress.

2. A conference paper which, at the time of writing, is in progress.

Chapter 1: Introduction

The term IoT, or Internet of Things, refers to a network of connected devices that communicate with the cloud and amongst themselves [1]. IoT enables everyday “things” to connect to the internet and perform complex functionalities. This makes devices such as cameras, smoke alarms, toothbrushes, weighing machines, and other such machines smarter. With the recent shrinking of the size of computer chips which provide “smart” and high bandwidth telecommunication, there exist billions of devices that are connected to the internet. Such recent growth of IoT and related paradigms have become extremely important topics as they have major societal and economic impacts [2].

Generally, IoT has the following impacts on society.

- Better Health: Wearable technologies like smartwatches and fitness trackers offer users instant access to personalized services and information. In addition to this, the integration of IoT within the healthcare sector has yielded tangible benefits in terms of patient care and outcomes. Remote patient monitoring devices enable healthcare professionals to remotely track vital signs, medication adherence, and disease progression.
- Enhanced Security: Smart security devices such as cameras, motion sensors, and door locks, can now be remotely monitored and controlled via mobile devices or computers, empowering users to protect their premises from anywhere in the world. Moreover, IoT-driven biometric authentication mechanisms ensure that only authorized individuals gain entry to sensitive areas or digital assets.
- Improved Efficiency: IoT technology has brought about advancements in various sectors, particularly in manufacturing. Through the integration of sensors and connected devices, industries can now monitor and optimize their production processes in real-time. This integration has resulted in reduced downtime, minimized waste, and streamlined workflows.
- Enhanced Convenience: Smart home gadgets, such as thermostats, lighting systems, and household appliances, can now be remotely controlled via smartphones or voice commands. This capability empowers homeowners to adjust settings, monitor energy consumption, and automate household tasks resulting in unprecedented levels of convenience.

Generally, IoT has had the following impacts on the economy:

- Increased Productivity: The integration of IoT technology has spurred notable improvements in productivity across various sectors of the economy. By harnessing the power of interconnected devices and sensors, businesses can optimize their operations in real-time, resulting in enhanced output and efficiency.
- Job Creation: The advent of IoT has not only transformed existing job roles but also fostered the creation of new employment opportunities. As industries embrace IoT solutions to drive innovation and efficiency, there is a growing demand for skilled professionals with expertise in IoT development,

data analytics, cybersecurity, and system integration. Moreover, the deployment and maintenance of IoT infrastructure necessitates a workforce proficient in installation, troubleshooting, and ongoing support.

- Cost Savings: IoT offers significant cost-saving opportunities for businesses through improved operational efficiency and resource optimization. Furthermore, IoT-driven data analytics provide valuable insights into consumer behavior, market trends, and operational performance, enabling businesses to make informed decisions and allocate resources more effectively. As a result, organizations can achieve substantial cost savings, enhance profitability, and gain a competitive edge in the marketplace.
- New Revenue Streams: IoT presents businesses with opportunities to explore new revenue streams. Through IoT, companies can introduce value-added services such as predictive maintenance contracts, remote monitoring, and personalized customer experiences. Furthermore, IoT facilitates the creation of ecosystems and partnerships, allowing businesses to collaborate with other stakeholders and capitalize on complementary products and services.

The recent growth of IoT devices and related services has brought the IoT paradigm into the mainstream. According to IoT Analytics [3], IoT devices will grow at a rate of 16% every year, reaching a staggering figure of 29.7 billion connected IoT devices by 2027. Figure 1 shows the projected growth of IoT devices.

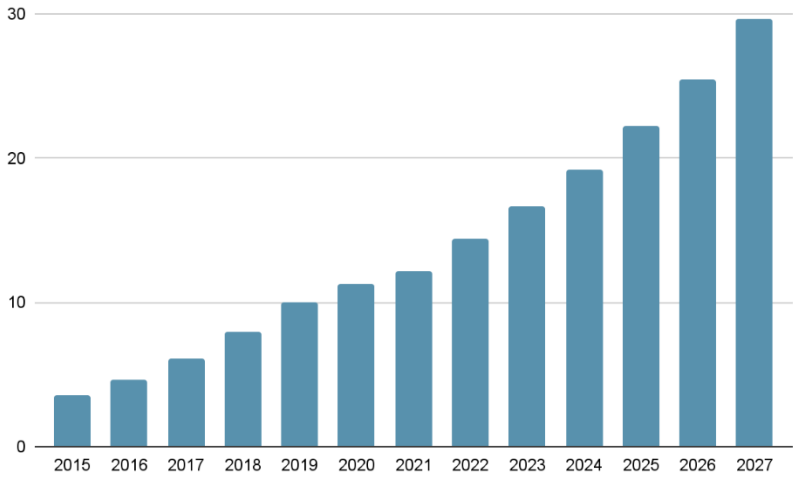

Figure 1. Global IoT Market Forecast (in billions of IoT-connected devices) [28]

To put this number into perspective, it is estimated that in the year 2027, the world population will be approximately 8.3 billion [4]. This translates into having about 3.57 IoT devices for every person on earth.

This growth in terms of IoT devices has resulted in the generation of an enormous amount of data that needs to be processed, analyzed, and acted upon in real-time. However, cloud computing, a traditional method of handling such data, has limitations in terms of latency, bandwidth, and cost. This has led to the emergence of distributed paradigms such as fog computing, where computation and data storage are moved closer to the

data source. The general fog computing architecture can be divided into three distinct layers, as shown in Figure 2.

As can be seen in Figure 2, the first is the IoT devices layer, it includes different devices such as smart thermostats, smart lighting systems, smartphones, industrial machinery, and self-driving cars. This layer generates data and offloads it to the fog computing layer through 3G, 4G, 5G, Wi-Fi, and Bluetooth technologies. The intermediate layer is the fog computing layer that includes various computing, networking, and storage devices. Finally, the upper layer is the cloud computing layer, which includes data centers with high computing power [5]. As we move away from the devices layer, the computation complexity increases, and the magnitude of resources decreases [6].

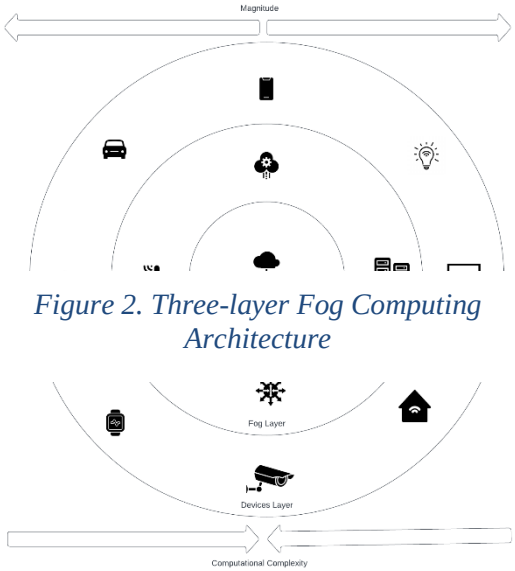

Figure 2. Three-layer Fog Computing Architecture

Fog computing has several advantages over cloud computing. First, it reduces the latency and bandwidth requirements as data is processed and analyzed closer to the data source, resulting in faster response times. Second, it reduces the cost of data transmission as only relevant, or summary data is transmitted to the cloud. Third, it enhances data privacy and security as data is processed and analyzed closer to the source, reducing the risk of data breaches. However, fog environments generally suffer from limited resources, translating into IoT devices competing to find relevant resources on fog nodes to complete or execute operations or tasks [7]. Therefore, resource scheduling is a crucial aspect of fog computing as it ensures that the available resources are efficiently utilized and allocated to different tasks. Resource scheduling involves assigning tasks to computing resources based on their capabilities and availability.

The billions of devices around the world are likely to generate several tasks, some of which would need to be offloaded to resources, as discussed above. There is no definitive mechanism to estimate how many tasks a typical IoT device would generate, as it would depend on several factors.

- Device Heterogeneity: Different devices have different needs. A simple temperature sensor might not have many complex offloadable tasks. However, a smart camera might have to send continuous video streams to analyze the footage.

- Device capabilities: Processing power and onboard storage affect how much data a device can process locally before offloading tasks to the cloud or fog. Powerful devices might perform local analysis, reducing tasks sent elsewhere.
- Configuration: How a device is configured significantly impacts how it interacts with the network and generates offloadable tasks. Some devices might be configured to optimize battery life. Such devices typically do not offload tasks frequently.

Despite having no definitive number, it is easy to see how these IoT devices generate billions of daily tasks. Considering there are only around 10,000 data centers on the earth [8], there are limited resources that can handle these tasks. This illustrates how important resource scheduling is in a fog environment to ensure there are no bottlenecks and the tasks are executed successfully.

The resource scheduling problem in fog environments is more challenging than in cloud computing due to the environment's resource constraints, heterogeneity, and dynamic nature [9]. IoT devices are typically resource-constrained and have limited processing power, memory, and storage [9]. Furthermore, resources in a fog environment are generally very heterogeneous, i.e., they differ in terms of their capabilities, such as processing power, memory, and storage [9]. The dynamic nature of the environment means that resources may become available or unavailable at any time, making resource scheduling even more challenging.

Furthermore, with the increasing energy consumption of computing devices, energy efficiency has become a critical factor in resource scheduling. Energy efficiency refers to the ability of a system to perform a task using the minimum amount of energy. Resource scheduling algorithms that keep energy consumption down while not compromising on the performance/execution time are the need of the hour. The energy demand from data centers accounts for 1.5% of global electricity consumption [8]. This figure is predicted to increase significantly in the coming years. The estimated global data center electricity consumption in 2022 was 240-340 TWh, which resulted in a staggering 330 metric tons of CO2 equivalent. The combined electricity use of just four major companies, Amazon, Microsoft, Google, and Meta was estimated to be around 72 TWh in 2021. Emerging services and technologies such as streaming, cloud gaming, blockchain, artificial intelligence, machine learning, and virtual reality will further boost this demand. All these figures highlight the pressing need for significant efficiency gains, and a resource scheduling algorithm that prioritizes energy efficiency can make a notable difference in reducing energy consumption.

TETRA will reduce the carbon footprint of fog nodes, servers, and data centers while still providing Quality-of-Life (QoL) guarantees to users. This will also lead to other advantages, such as higher revenues for service

providers due to lower operational costs. In addition, the costs to the users will be minimized due to the lower makespan to create a win-win situation for the users and the service providers.

While the existing research work in resource scheduling for fog environments has contributed significantly, it has limitations. Majority of these studies, for example, rely on generic algorithms such as First-Come-First-Serve (FCFS), Shortest-Job-First (SJF), or optimization algorithms like Ant Colony Optimization (ACO) and Particle-Swarm-Optimization (PSO). These algorithms have been shown to be effective in some scenarios. Nonetheless, these solutions have their drawbacks [5], such as limited scalability, lack of adaptability to heterogeneous environments, and inability to consider multiple criteria simultaneously. Moreover, the fog computing environment is highly dynamic, and resource availability can vary unpredictably, making the resource scheduling problem even more challenging. Therefore, a novel resource scheduling algorithm that addresses the limitations of the existing approaches while considering multiple criteria during scheduling decisions is needed. To this extent, we propose **TETRA** – A ***T***ime and ***E****nergy-aware* ***T****OPSIS-based* ***R***esource scheduling ***A***lgorithm for optimizing the decision-making of scheduling relevant resources.

Our contribution in this thesis project is as follows:

1. We conducted a comprehensive literary survey aimed at identifying the limitations present in current research. This process will assist us in strategically identifying the optimal field for our own research endeavors.
2. We want to develop a scalable algorithm that maintains a high level of performance in a resource constrained environment.
3. We implemented an energy aware algorithm that significantly reduces energy consumption while not compromising on a critical performance criterion: makespan.
4. We tested our algorithm extensively against several state-of-the-art baseline algorithms.
5. We developed an open-source library extends CloudSim Plus framework so that users can simulate resource scheduling across fog and cloud environments.

The rest of this thesis is organized as follows: Chapter 2 presents the related work. Chapter 3 presents the proposed model. Chapter 4 presents the experimental results. Chapter 5 presents our conclusions and plans for future work.

Chapter 2: Related Work

Optimally scheduling tasks on resources within cloud environments is an ongoing area of research that the research community has extensively investigated in recent years. However, little research has been conducted to optimize resource scheduling in resource-constrained environments such as fog environments. Additionally, multi-objective optimization approaches are often neglected in research. To this extent, we review the existing literature on resource scheduling in fog and related environments.

Resource scheduling comes under the umbrella of resource management in cloud environments. Effective resource management requires innovative techniques and strategies for resource allocation, scheduling, optimization, and allocation policies. Three important topics under the umbrella of resource management are computational offloading, resource allocation, and resource provisioning.

Computational offloading is the process of shifting compute-intensive tasks or workloads from edge devices to nearby edge servers or cloud data centers. This helps edge devices conserve their limited resources, like battery power and processing capacity. To achieve effective computational offloading, factors such as network latency, workload characteristics, and task priorities must be carefully considered.

Resource allocation involves assigning resources, like CPU, memory, and storage, to various tasks or applications running on edge devices. Resource allocation decisions are made based on factors such as workload demands, resource availability, and quality-of-service requirements. Effective resource allocation strategies can optimize resource utilization and improve application performance.

Resource provisioning refers to supplying resources, like virtual machines, containers, or serverless functions, to edge devices or servers as needed. Resource provisioning ensures that applications have access to the required resources at the right time and in the correct amount. Effective resource provisioning techniques require careful consideration of factors such as resource availability, deployment costs, and resource utilization efficiency.

While investigating related work for TETRA, we conducted a survey on the current state of resource management techniques in edge computing, including computational offloading, resource allocation, and resource provisioning. The survey aims to provide insights and guidance for researchers and practitioners in designing and implementing efficient and effective resource management solutions in edge computing.

2.1 Survey Methodology

This section provides the method we used to survey the resource management approaches in the field of edge computing. It is shown in Figure 3.

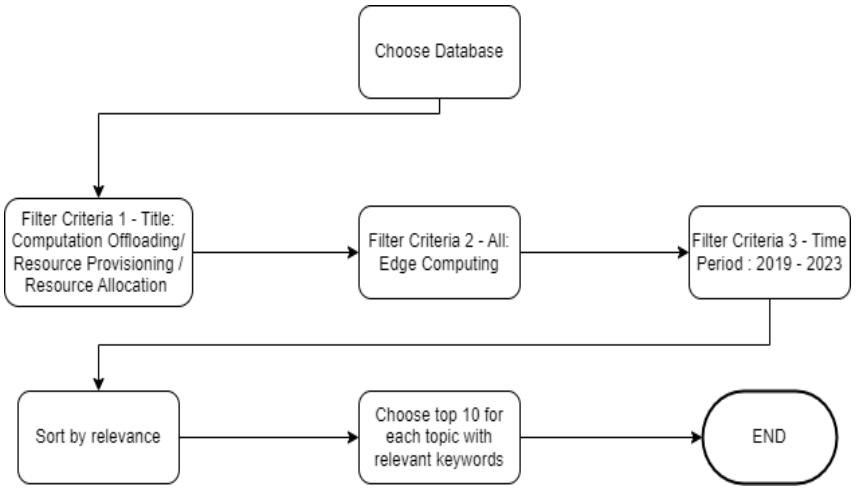

Figure 3. Paper Selection Method

We have considered three databases to avoid any bias:

1. Institute of Electrical and Electronics Engineers (IEEE) Xplore
2. Association of Computing Machinery (ACM)
3. Multidisciplinary Digital Publishing Institute (MDPI)

We looked at the three approaches of resource management: computation offloading, resource provisioning and resource allocation. To find the papers related to these topics, the following keywords were applied:

1. Computation offloading: “Computation Offloading”
2. Resource Provisioning: “Resource Provisioning”
3. Resource Allocation: “Resource Allocation”

These keywords were searched for in the Title of the papers in the above-mentioned databases.

To make sure the papers were relevant to the field of edge computing, we applied the keyword “Edge Computing”.

To ensure that the information regarding the field is up to date, we filtered for research papers that were published between 2019 and 2023.

Next, we filtered manually to ensure that the research papers returned were relevant to topics mentioned above. In this step, we chose ten papers from each database.

2.2 Survey Results

Based on the methodology above, we found a total of 6635 research papers across all three databases for computational offloading. Out of which, IEEE, MDPI and ACM account for

813,18, and 5794 research papers respectively, as seen in Figure 4.

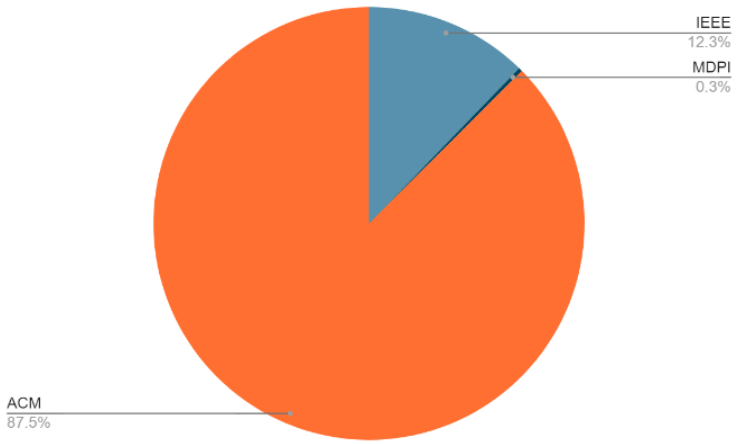

Figure 4. Distribution of Research Rapers - Computation Offloading

Similarly, we found a total of 148 research papers across all three databases for resource provisioning. Out of which, IEEE, MDPI and ACM account for 77,1, and 70 research papers respectively, as seen in Figure 5.

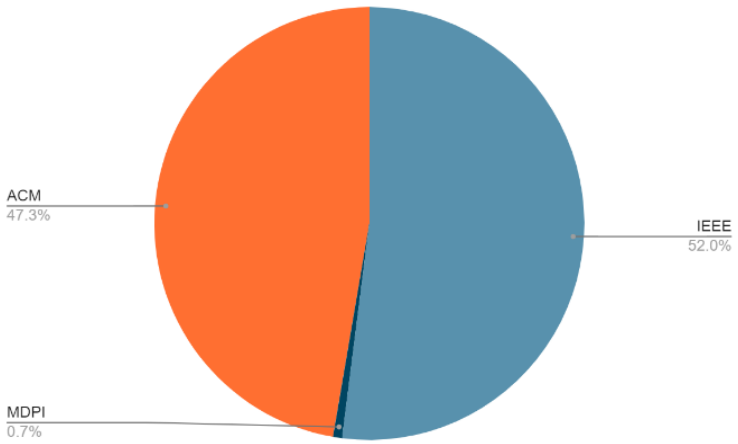

Figure 5. Distribution of Research Papers - Research Provisioning

Similarly, we found a total of 1663 research papers across all three databases for resource allocation. Out of which, IEEE, MDPI and ACM account for 1567,30, and 66 research papers respectively, as seen in Figure 6.

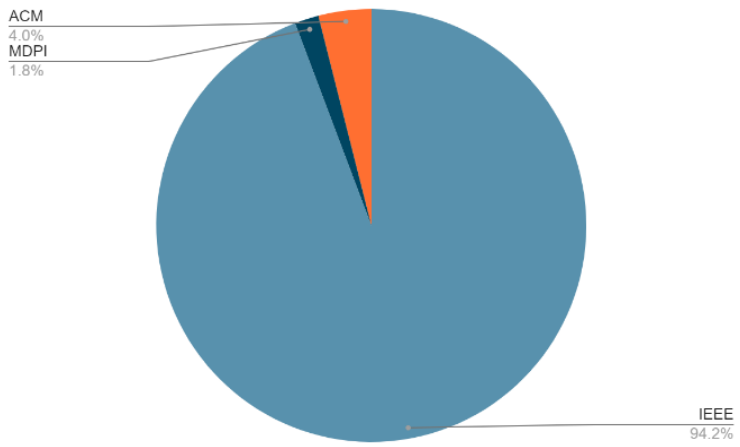

Figure 6. Distribution of Research Papers - Resource Allocation

2.2.1 Computational Offloading

Computation offloading in edge computing refers to the process of transferring computationally intensive tasks from a mobile device or endpoint to an edge device or server located closer to the user. Computation offloading can help reduce energy consumption and increase the battery life of endpoint devices, while also providing faster response times and improved user experience. Static and dynamic computation offloading are two different approaches to moving computation tasks from edge devices to servers in edge computing. Static computation offloading is more suitable for applications with predictable workload and resource requirements.

The following keywords were used to categorize the research papers on computational offloading.

- Partial: The process of distributing computation tasks between edge devices and the edge server/node to optimize the performance.
- Binary: This criterion specifies whether all the computation tasks are offloaded to edge nodes or none, that is 0 or 1.
- Distributed / Decentralized: Computation offloading technique where a computation task is divided into smaller sub-tasks and distributed across multiple edge devices for parallel processing.
- Centralized: Centralized technique to complete all computation tasks at an edge node.

- Simulation: If experiments were conducted to evaluate the algorithm for computation offloading.
- MEC: If the proposed computation offloading algorithm in the paper is applicable to multi access edge computing.
- Objective: This criterion specifies the goal of the research such as energy, latency, network congestion.
- Technique: The technique used to develop the algorithm.

Papers [10-38] were analyzed, and the results were summarized in Figures 7-13.

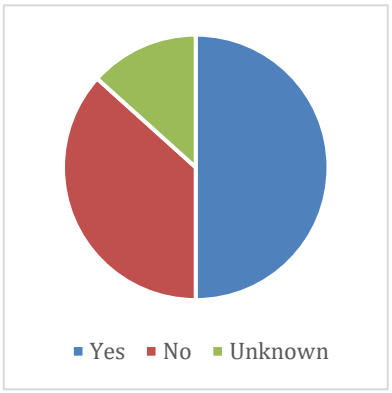

Figure 7. Category: Supports Partial Computation Offloading

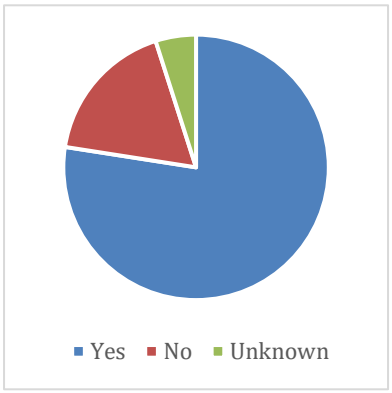

Figure 8. Category: Supports Binary Computation Offloading

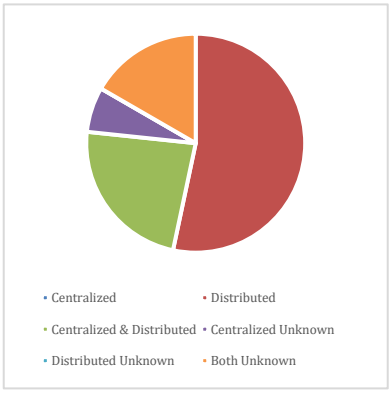

Figure 9. Category: Centralized / Distributed Computation Offloading

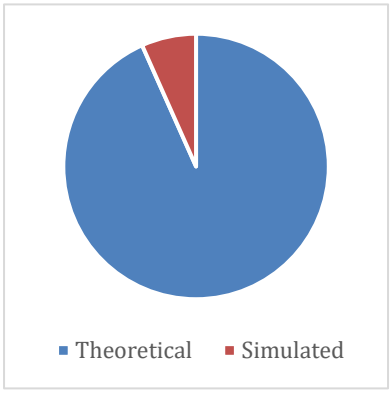

Figure 10. Category: Theoretical/ Simulated Computation Offloading

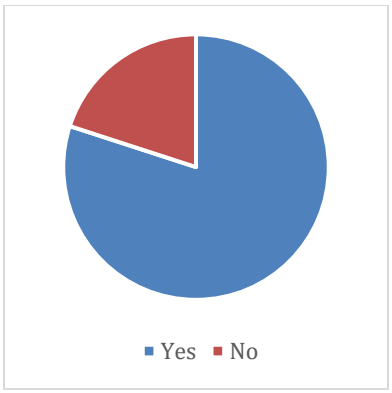

Figure 11. Category: Offers MEC support

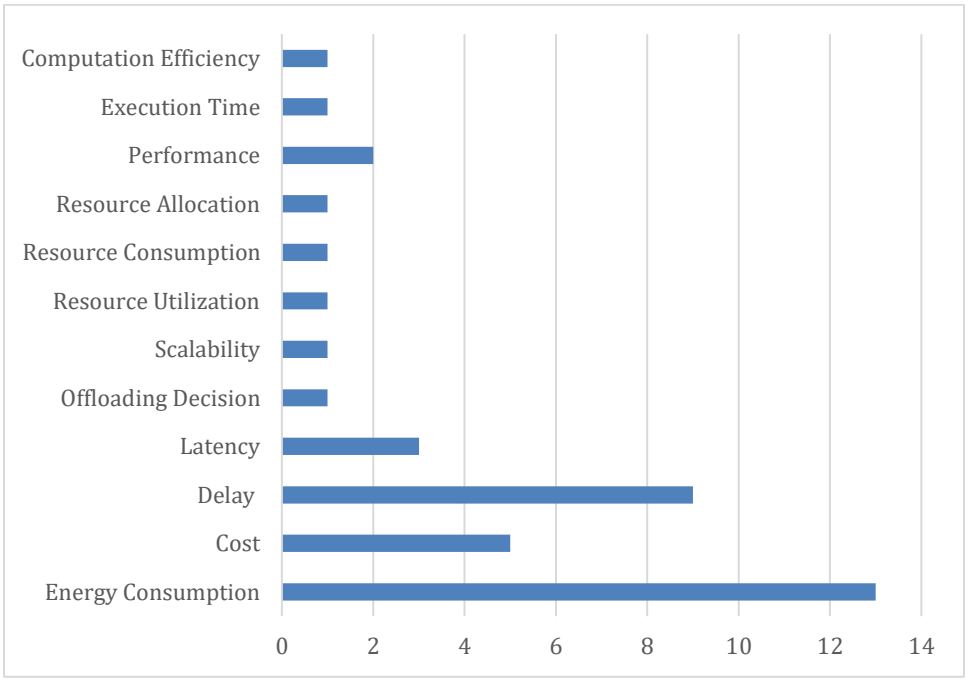

Figure 12. Category: Computational Offloading Objective

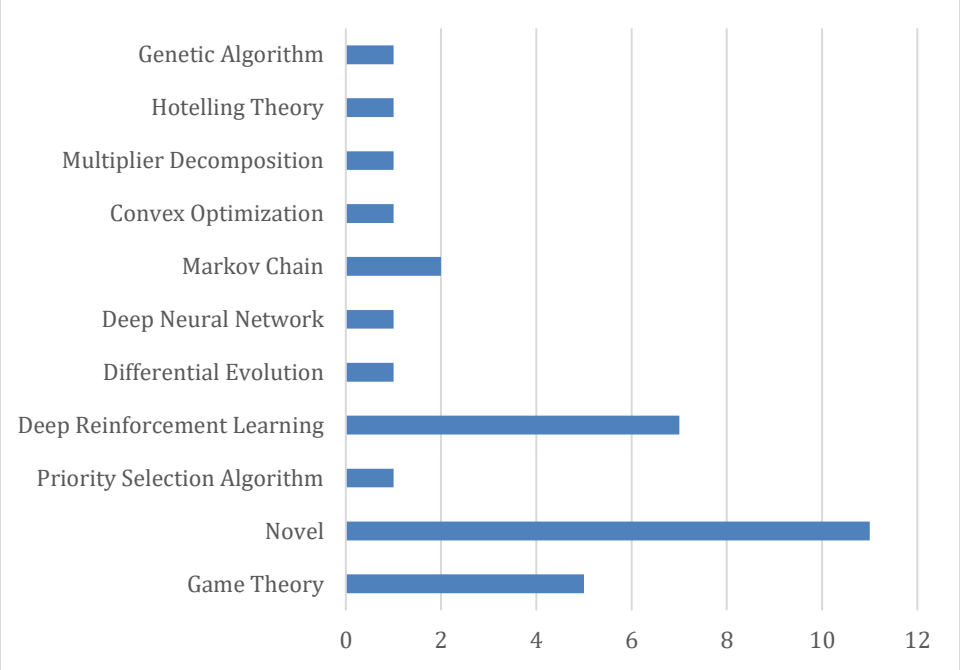

Figure 13. Category: Technique for Computational Offloading

2.2.2 Resource Provisioning

Resource provisioning refers to the process of preparing and making resources available for use by cloud/remote servers. It involves discovering available resources, preparing them for use, and making them available to applications and services. This can include tasks such as configuring servers, setting up network connections, and preparing storage resources. By provisioning resources at the edge, data can be processed locally, reducing the amount of data that needs to be transmitted over the network, and potentially lowering network congestion and bandwidth costs. Edge computing can help optimize bandwidth usage by reducing the need to send large amounts of data to the cloud for processing.

The following keywords were used to categorize the research papers on resource provisioning. The results can be found in Table 2.

- Static: This approach assumes that the resource requirements for edge computing workloads are known in advance and remain relatively constant over time.
- Dynamic: This approach involves allocating resources based on the current demands of the workload.
- MEC: If the proposed resource provisioning algorithm in the paper is applicable to multi access edge computing.
- Simulation: If experiments were conducted to evaluate the algorithm for resource provisioning.
- Objective: The goal of the research such as energy, latency, and network congestion.

- Technique: The technique used to develop the algorithm.

Papers [62-81] were analyzed, and the results were summarized in Figures 14-18.

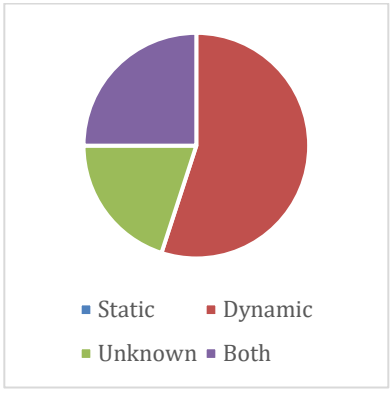

Figure 14. Category: Static/ Dynamic Resource Provisioning

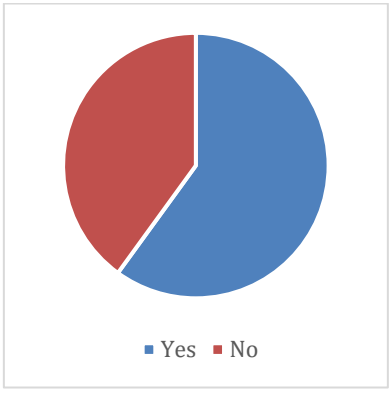

Figure 15. Category: Offers MEC Support

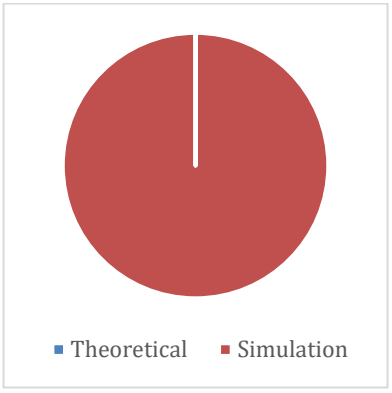

Figure 16. Category: Theoretical/ Simulated Resource Provisioning

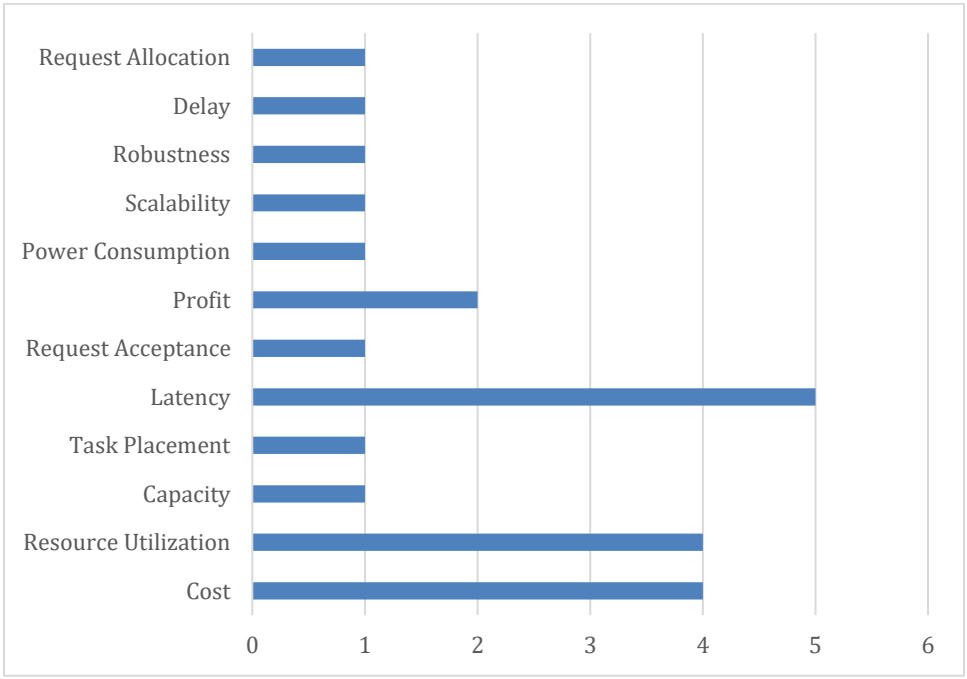

Figure 17. Resource Provisioning Objective

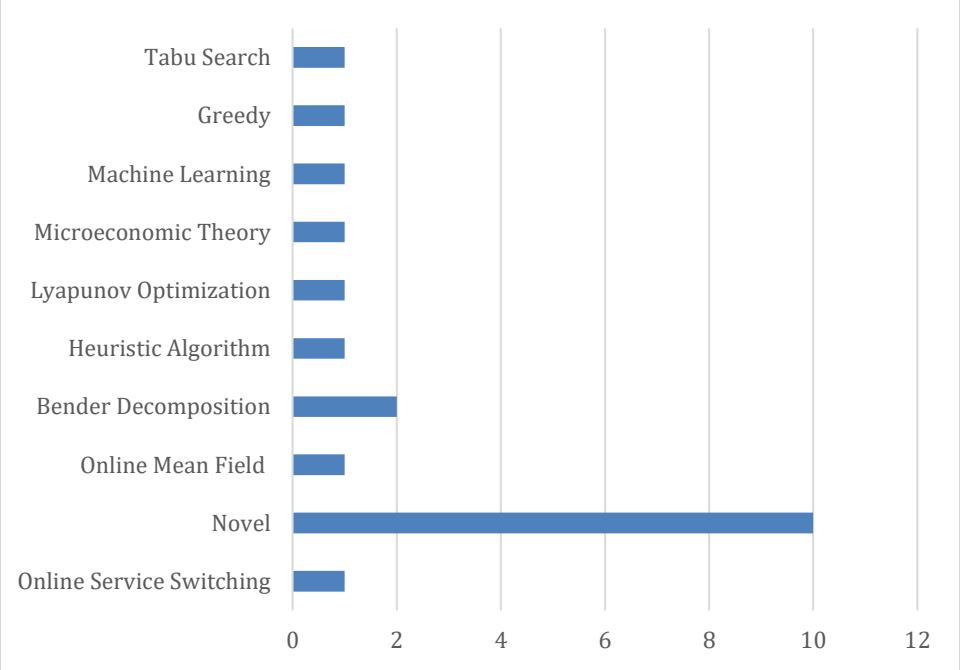

Figure 18. Category: Technique for Resource Provisioning

2.2.3 Resource Allocation

It refers to the process of assigning specific resources to individual applications or services. It involves managing the allocation of computing power, memory, storage, and network bandwidth to ensure that resources are used efficiently and effectively. This can include tasks such as assigning processing power to specific applications, allocating memory resources, and managing network bandwidth. Resource provisioning is a prerequisite for resource allocation, as resources must be prepared and made available before they can be allocated to applications and services. Resource allocation in edge computing refers to the process of distributing computing resources such as CPU, memory, storage, and bandwidth to various edge devices or nodes, based on their requirements and availability. Some of the common techniques used in resource allocation in edge computing include load balancing, dynamic resource allocation, and task scheduling algorithms.

The following keywords were used to categorize the research papers on resource allocation.

- Centralized technique: A central entity, such as a server/ edge node, controls and manages the allocation of resources to edge devices, such as sensors, mobile devices, and IoT devices.
- Distributed technique: Edge devices exchange information about their resource availability and usage patterns with each other and make decisions about resource allocation based on this information. This

can be done using various techniques, such as peer-to-peer networking, blockchain-based systems, or distributed algorithms.

- Storage: To filter resource allocation research papers if they account for storage requirements.
- Compute/ CPU: To filter resource allocation research papers if they account for CPU requirements.
- Networks/ Communication/ Bandwidth: To filter resource allocation research papers if the resources also involve the network requirements.
- MEC: If the proposed resource allocation algorithm in the paper is applicable to multi access edge computing.
- Objective: The goal of the research such as energy, and latency.
- Technique: The technique used to develop the algorithm.

Papers [59-87] were analyzed, and the results were summarized in Figures 19-23.

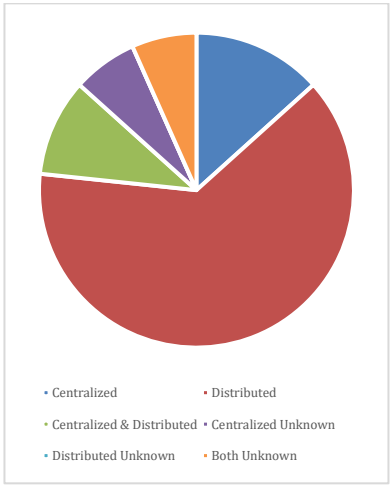

Figure 19. Category: Centralized/ Distributed Resource Allocation

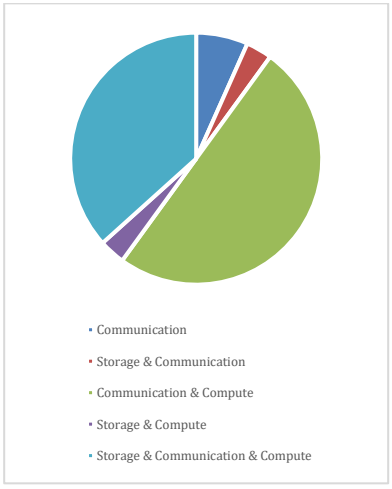

Figure 20. Category: Storage, Communication & Compute Focused Resource Allocation

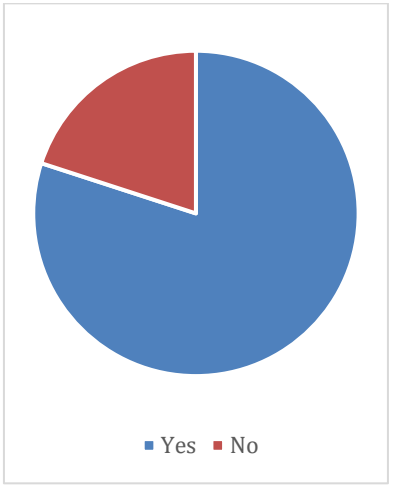

Figure 21. Category: Offers MEC Support

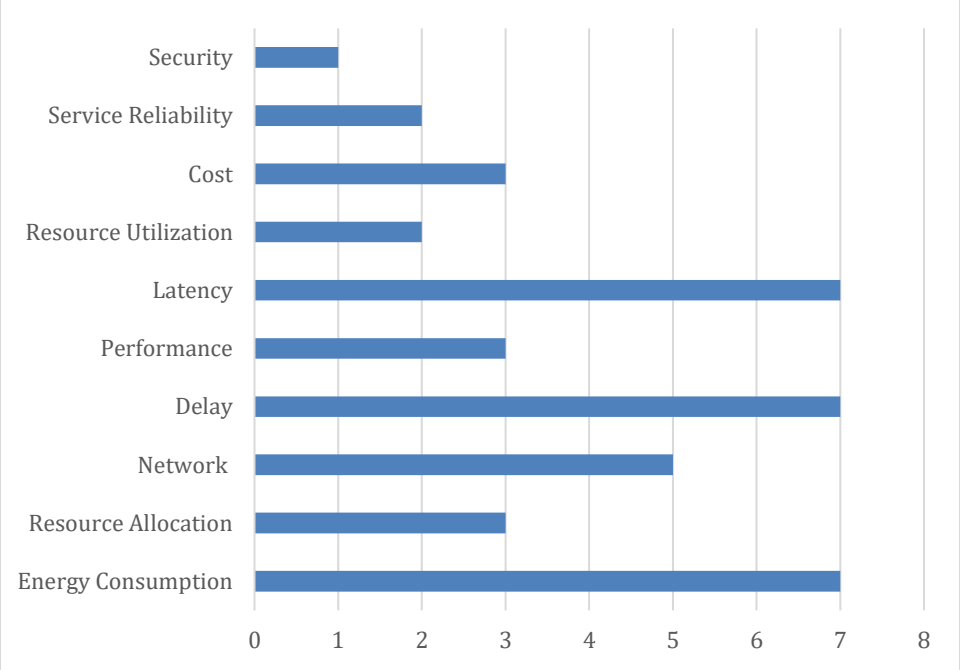

Figure 22. Category: Resource Allocation Objective

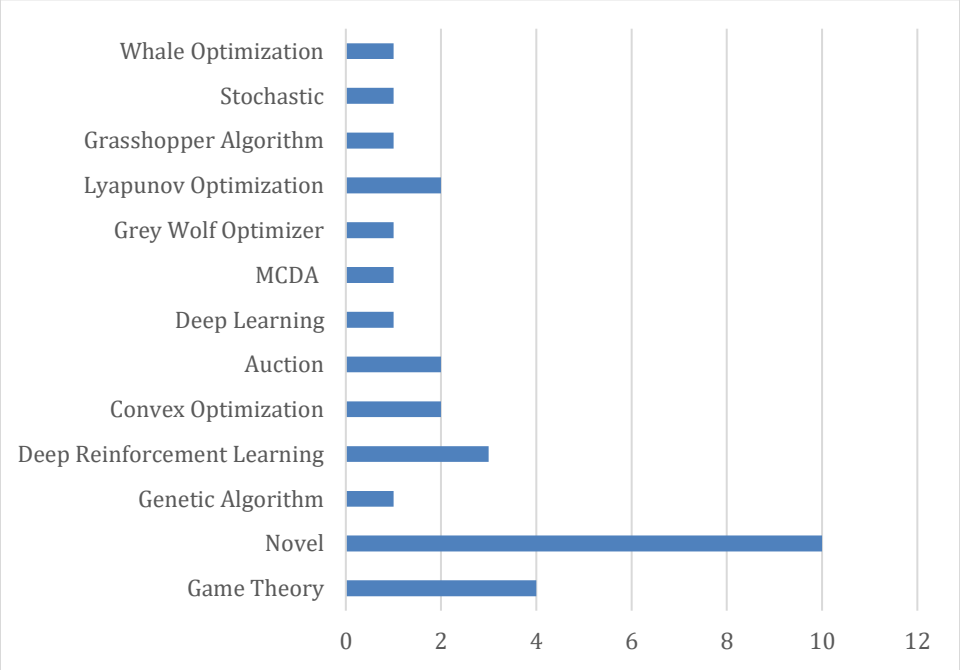

Figure 23. Category: Technique for Resource Allocation

2.2.4 Resource Scheduling

In our research, we are tackling a subset of Resource Allocation, which is resource scheduling. The following is research work which focuses on task scheduling algorithms.

- A scheduling approach based on virtualization technology is proposed by Mtshali et al. [88] to improve the average delay and energy consumption of real-time applications in a fog computing environment. iFogSim was used to run simulations. The findings show that compared to alternative algorithms, the FCFS scheduling policy resulted in a reduction in energy consumption, average task delay, network usage, and execution time.
- A scheduling technique is proposed by Mukherjee et al. [89] with the goal of guaranteeing network stability and maximizing the number of completed tasks executed within their respective deadlines. The method schedules tasks on fog nodes using the Lyapunov drift-plus-penalty function on the queue length. According to the simulation results, the suggested approach works better than baseline algorithms such as random scheduling and Lyapunov drift.
- A scheduling method based on the hybrid heuristic (HH) algorithm is put forth by Wang et al. [90] and is capable of processing jobs for terminal devices with high energy consumption and low computing power in an efficient and real-time manner. The outcomes of the experiment show that the suggested approach performs better than alternative approaches and is a good fit for maximizing the efficiency of smart manufacturing.
- Jamil et al. [91] have proposed a novel scheduling algorithm for fog computing that aims to optimize delay and network utilization for IoE (Internet of Everything) device service provisioning. They assessed their algorithm using iFogSim and compared it against current methods, with delay and energy consumption and network usage taken into account as performance measures. The results show that the suggested scheduling technique is effective in optimizing these performance measures, when compared to the First-Come-First-Serve (FCFS) method.
- To assign an optimal set of tasks to fog nodes while satisfying the quality of service (QoS) requirements of Cyber-Physical-Systems (CPS) applications in a way that minimizes the total execution time of tasks, Ghobaei-Arani et al. [92] propose a scheduling algorithm based on the moth-flame optimization algorithm. The proposed algorithm treats the reduction of transfer time and task execution as objective functions. With reduced overall execution time consumption than previous methods, the experimental findings from simulation based on the proposed algorithm demonstrate that the optimal solution for task scheduling and equal distribution of tasks to fog nodes has been achieved.

- A study by Azizi et al. [93] proposed two new resource scheduling methods, namely priority-aware semi-greedy (PSG) and PSG with multi-start procedure (PSG-M), that outperform existing techniques. These methods improve how tasks are assigned to computing resources, resulting in significant benefits. The proposed methods achieve a greater percentage of tasks meeting deadlines, lower overall energy consumption, a much larger reduction in the total deadline violation time, and faster completion times for all tasks. Overall, these new resource scheduling methods significantly improve the efficiency of the system in terms of deadline requirements and deadline violation time.
- Islam et al. [94] propose a Reinforcement Learning (RL) system to manage resources in a Spark cluster deployed on the cloud. They developed two schedulers based on Deep Reinforcement Learning (DRL) within the TF-Agents framework. These DRL-based schedulers enable agents to learn job characteristics, aiming to decrease both overall cluster VM usage costs and average job duration. The findings indicate that the DRL-based algorithms significantly reduce VM usage.
- Jayanetti et al. [95] present a deep reinforcement learning-based scheduling framework that uses a distinct hierarchical action space to distinguish between edge and cloud nodes. Evaluation metrics including energy usage, execution time, percentage of deadline adherence, and percentage of jobs completed are used to evaluate the framework against a variety of baseline algorithms. Their suggested framework performed significantly better than the baselines, by greatly optimizing energy consumption and execution time while keeping execution time and energy efficiency on par with the optimized baselines.

Unlike existing research efforts, we will build a resource scheduling algorithm that optimizes both time and energy, taking advantage of an MCDA-based approach. In addition, we will investigate this algorithm's performance in an environment with a relatively higher number of tasks and a lower magnitude of resources, which is typical of a fog environment. This will be done using the CloudSim Plus framework.

Table 1 summarizes the research papers considered for the related work and compares the related work with TETRA.

Table 1. Comparison of Research Rapers - Task Scheduling Algorithms

References	Multiple Objective Optimization	MCDA Algorithm	Fog Environment Support	CloudSim Plus Support
[88]	✓	X	✓	X

[89]	✓	X	✓	X
[90]	✓	X	✓	X
[91]	✓	X	✓	X
[92]	X	X	✓	X
[93]	✓	X	✓	X
[94]	✓	X	X	X
[95]	✓	X	X	X
[96]	X	X	✓	X
[97]	X	X	✓	X
[98]	X	X	✓	X
[99]	X	X	✓	X
[100]	✓	✓	X	X
[101]	X	X	✓	X
[102]	✓	X	✓	X
TETRA	✓	✓	✓	✓

Chapter 3: Proposed Model

This chapter delves into TETRA's resource scheduling optimization strategy, including its design and deployment. To optimize energy consumption and makespan, TETRA employs a multi-objective MCDA approach called TOPSIS. This approach considers available resources and their capabilities to determine the optimal mapping between cloudlets and resources. In addition, the model is compared with other baseline algorithms like Best-Fit, Greedy, and First-Fit algorithms.

The model consists of seven components: (a) pre-scheduling, (b) pre-processing, (c) post-filtration, (d) scheduling, (e) cloudlet execution, (f) energy consumption estimation, and (g) makespan estimation. Each component is explained in detail in this chapter. Before delving into the components, we will first discuss the deployment of this model. By introducing the TETRA model, we make the following hypothesis.

Hypothesis: MCDA methods such as TOPSIS can be utilized for effective scheduling in data center environments.

3.1 Model Deployment

TETRA can typically be deployed on IoT gateways on the fog layer. This placement enables it to be a hybrid system model; that is, it can allocate tasks/cloudlets to fog resources as well as cloud resources. TETRA uses a TOPSIS-based algorithm that considers the various specifications of resources that exist across both layers. By considering the specifications of all the available resources, finding the best matching resource that can be allocated for executing the task/ cloudlet is possible. At any given time, multiple resources might be available to execute a cloudlet. As part of our hybrid system model, we try to identify the 'ideal' match that would optimize energy consumption and makespan globally.

The following are key terminologies that will be used throughout the thesis.

- Device: A device is defined as the entity representing an IoT device residing in the outermost – devices layer.
- Task / Cloudlet: A task or a cloudlet is an executable piece of code that a device offloads to be executed on a more computationally complex resource than itself. An example of a task would be a smart camera at the entrance of a business requesting a Machine Learning (ML) service to identify authorized people within images.
- Gateway: An entity capable of receiving tasks from the devices layer and routing them to the appropriate resource.

- Resource: An entity capable of executing tasks. It represents a combination of physical hardware components such as RAM (random-access memory), storage, and processor. Resources vary in degree of computational complexity.

Figure 24 shows our proposed hybrid system model. The layers are explored further in this section.

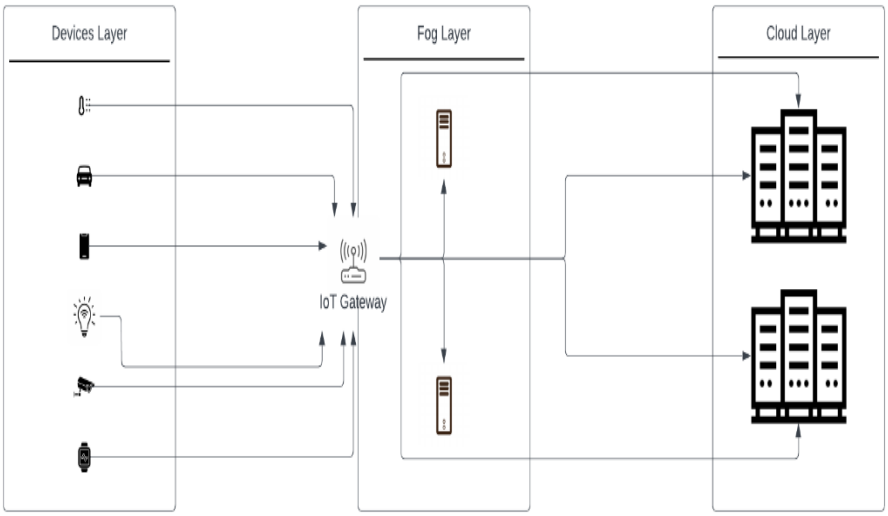

Figure 24. Model Deployment

3.1.1 Devices Layer

The devices layer serves as the outermost layer of the Internet of Things (IoT) ecosystem, encompassing a diverse array of interconnected devices equipped with specialized sensors designed to gather and transmit data. These devices represent a crucial link in the data collection chain, acting as the frontline data gatherers in various environments ranging from industrial settings to smart homes.

Through their sensors, these IoT devices gather a huge amount of data that must be processed to provide crucial functionality. However, due to inherent constraints such as limited processing power, storage capacity, and energy resources like battery life, these devices often find themselves challenged when executing computationally intensive tasks or performing complex data analysis.

To overcome these limitations, IoT devices leverage the collaborative power of the fog and cloud layers, strategically offloading tasks that exceed their capabilities to these higher-level computing platforms. The fog layer, comprising localized computing resources situated closer to the edge of the network, provides immediate processing capabilities for time-sensitive tasks while reducing latency and alleviating bandwidth congestion.

Meanwhile, the cloud layer offers vast computational resources and storage capacity, making it an ideal environment for executing complex analytics, machine learning algorithms, and long-term data storage. Examples of IoT devices span a wide spectrum of applications and industries, including but not limited to:

- Smart thermostats: Devices designed to regulate temperature in homes and commercial buildings by monitoring environmental conditions and adjusting heating or cooling systems accordingly.
- Industrial sensors: Embedded sensors within manufacturing equipment and machinery that collect real-time data on performance metrics, enabling predictive maintenance and process optimization.
- Wearable health monitors: Compact devices worn on the body to track vital signs such as heart rate, blood pressure, and activity levels, providing valuable insights for personal health management and medical research.
- Smart agriculture sensors: Deployed in agricultural fields to monitor soil moisture levels, temperature, and crop health, optimizing irrigation schedules, and maximizing yield.
- Connected vehicles: Equipped with sensors and telematics systems to gather data on driving behavior, vehicle performance, and environmental conditions, enhancing safety, efficiency, and convenience on the road.

In essence, the devices layer represents the foundational building blocks of the IoT ecosystem, connecting the physical world with the digital realm and paving the way for transformative innovations across industries and domains.

3.1.2 IoT Gateway

An IoT gateway serves as an intermediary node located at the edge of fog layer which contains information of all the available resources, spanning from localized fog resources to the expansive capabilities offered by the cloud layer. TETRA typically is designed to operate within a fog gateway. The gateway receives tasks from the devices layer and executes the optimization algorithm and allocates a resource to each task. Throughout this optimization process, the algorithm effectively attempts to keep energy consumption and makespan low.

3.1.3 Fog Layer

The fog layer represents the first layer which contains resources capable of executing tasks that have been offloaded by the devices layer. The resources at this level are not computationally complex but offer advantages such as low latency due to proximity to the device’s layer. Examples of fog resources are micro

data centers, fog gateways and other such resources that are powerful enough to execute tasks that do not require extremely high processing power, memory, or storage.

3.1.4 Cloud Layer

The cloud layer, while sharing similarities with the fog layer in its composition of multiple resources, stands apart in terms of its scale and sophistication. Unlike the fog layer, which boasts proximity advantages and rapid response times, the cloud layer is characterized by its expansive computational infrastructure, offering unparalleled processing power, vast memory capacities, and extensive storage capabilities.

Within the cloud layer, a myriad of high-performance servers and data centers converge to form a formidable computing ecosystem capable of handling a diverse array of workloads. This layer usually contains resources capable of executing tasks in fields such as Machine Learning (ML), Artificial Intelligence (AI), and big data.

3.2 Architecture

Our model operates on a layer above the CloudSim Plus framework. The architecture of our TETRA model is shown in Figure 25.

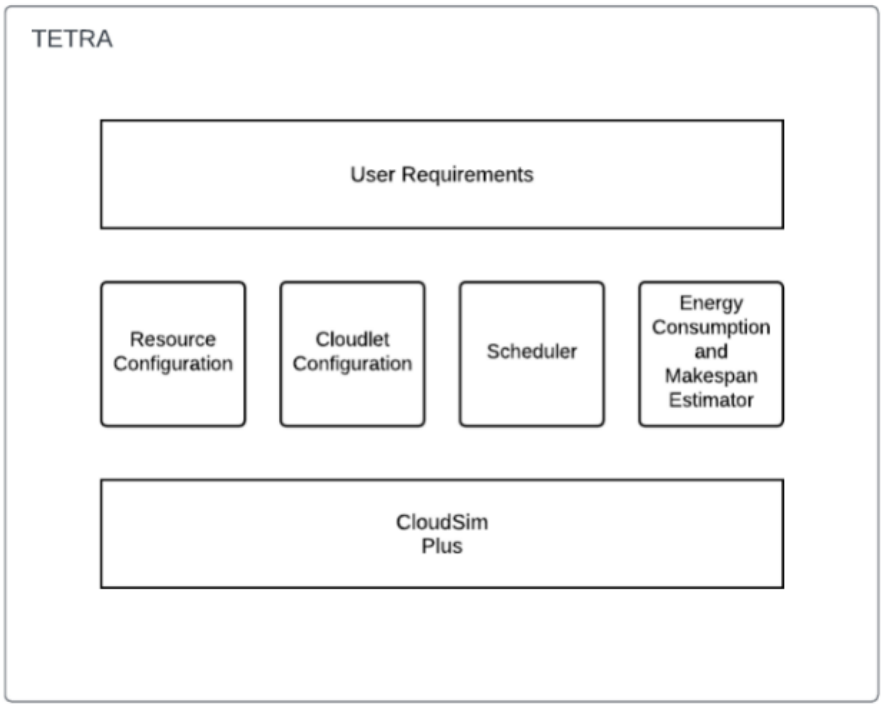

Figure 25. TETRA Architecture

TETRA builds on the CloudSim Plus framework, consisting of the following components.

- User Requirements: This component contains the main class where the user specifies various parameters such as number of resources, number of cloudlets and the scheduling algorithm to be used.
- Resource Configuration: This component contains multiple classes to deliver functionalities such as reading Host and VM specifications from excel sheets, creating data center, hosts and VMs using the specifications read from the sheets.
- Cloudlet Configuration: This component is used to create cloudlets. The user has control over parameters such as magnitude and size of cloudlet.
- Scheduler: The component contains the logic for the novel TOPSIS-based algorithm and the other baseline algorithms. Using these algorithms, each task is mapped to a resource. This mapping is forwarded to the broker in CloudSim Plus framework for execution. This broker oversees submission of cloudlets to VMs and the execution of these cloudlets on these VMs. It manages the creation and destruction of VMs.
- Energy Consumption and Makespan Estimator: This component contains the class "CalculateEnergyMakespanCost.java". It performs the following functionalities.

- Calculate the total energy consumption of all the resources in the datacenter.
- Calculate the makespan.

3.3 Process

This section describes the entire process flow of our proposed TETRA model. Figure 26 shows the flow-chart of the process that takes place within our model. It summarizes both the data and the logical flow of the model.

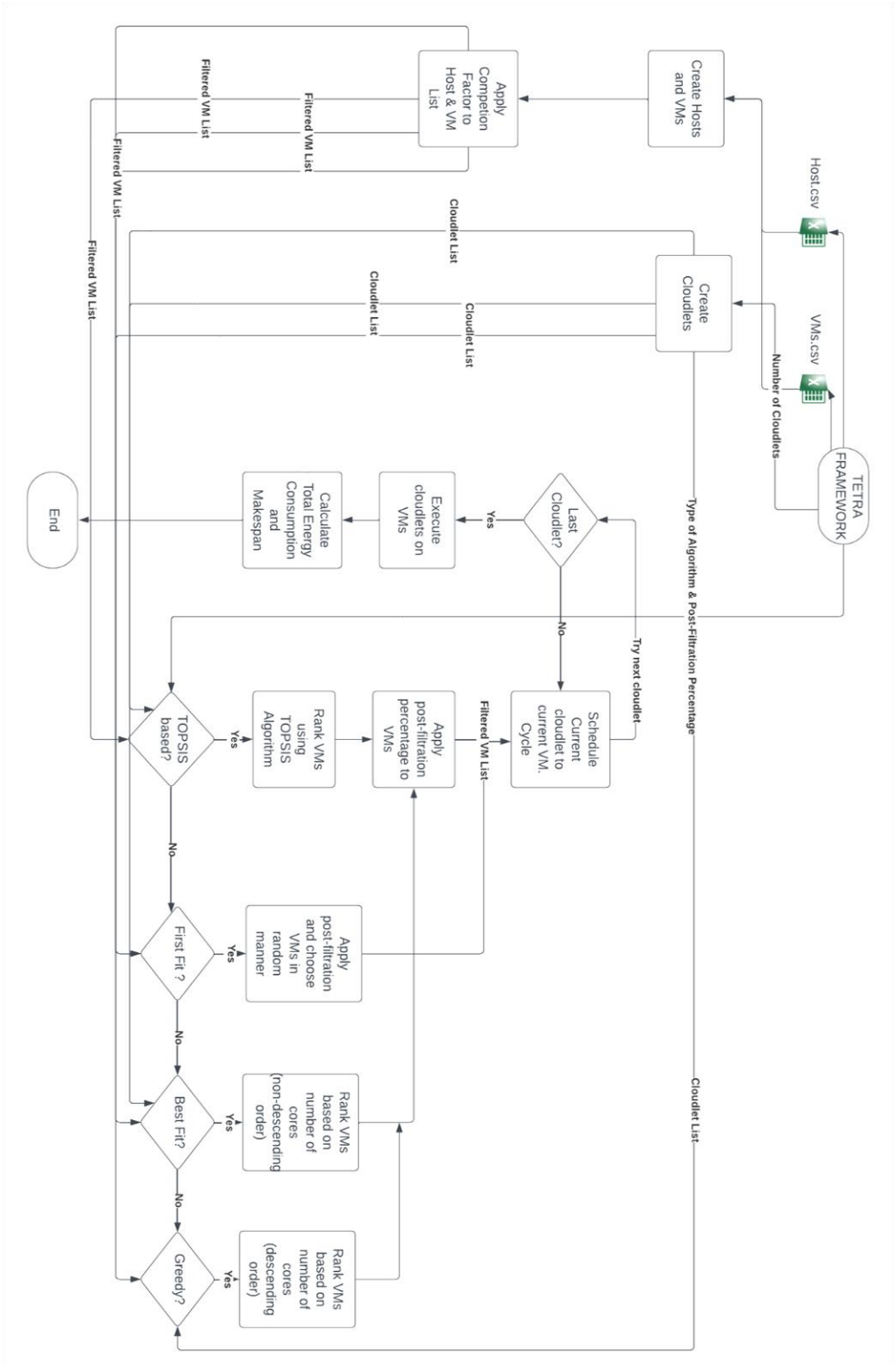

Figure 26. Model Process Flowchart

3.3.1 Inputs

Table 2 contains the inputs to the First-Fit Algorithm (FFTA) within the TETRA model.

Table 2. Model Inputs

Input	Description
Host.csv	Excel sheet that contains specifications of the Physical Machines on which VMs are hosted.
VM.csv	Excel sheet that contains specifications of VMs.
Number of cloudlets	The total number of cloudlets to be executed in each simulation.
Type of Algorithm	The type of algorithm to be used for each simulation. Options are TETRA, First-Fit, Best-Fit and Greedy.
Competition Factor	Used as a pre-filtration criterion. Used to decide the competition amongst resources (hosts + VMs)
Post filtration	Used as a post-filtration criterion only for TETRA algorithm.

3.3.2 Pre-Scheduling Steps

Before utilizing resource scheduling algorithms to allocate cloudlets to resources, there are certain pre-scheduling steps that are required in order to create a real-world IoT environment.

3.3.2.1 Create Datacenter

Using CloudSim Plus, we create a datacenter. This datacenter contains physical machines (Hosts). These hosts are of different configurations based on the specifications provided in the Host.csv file. The competition factor determines how many hosts are created. A lower number indicates higher competition whereas a higher number indicates a lower competition amongst the resources as there are more resources available.

3.3.2.2 Create Virtual Machines

As a next step, we create VMs (resources) which typically are hosted on the Hosts created in the previous step. These VMs are capable of executing cloudlets. The competition factor determines how many VMs can be created. A lower number indicates higher competition whereas a higher number indicates a lower competition amongst the resources as there are more resources available.

3.3.2.3 Create Cloudlets

We create a specific number of cloudlets as specified by the user input. Table 3 contains the specifications of each cloudlet.

Table 3. Cloudlet Specifications

Attribute	Description
fileSize	Size (in bytes) before execution
outputSize	Size (in bytes) after execution
numberOfCPUCores	Number of VM CPU cores required by the cloudlet for execution

3.3.3 Pre-Processing

TETRA compares our novel TOPSIS-based algorithm against three baseline resource scheduling algorithms. The following are the pre-processing steps for the various algorithms we employed. This step involves arranging the resources in a specific order based on the algorithm.

3.3.3.1 First-Fit Algorithm

No pre-processing is involved for this algorithm.

3.3.3.2 Best-Fit Algorithm

In this step, we arrange the VMs based on their computational capabilities. In particular, we sort the VMs based on the total number of CPU cores they possess in a non-decreasing order. This ensures that cloudlets can be mapped to resources with a lower number of cores first. Such a prioritization will typically results in lower energy consumption and higher resource utilization as there is a lower wastage of cores.

3.3.3.3 Greedy Algorithm

This algorithm sorts the VMs based on their CPU cores in a decreasing order. This ensures that the cloudlets are mapped to resources with a higher number of cores first. Such a prioritization typically results in reduced resource utilization. Nevertheless, under specific circumstances, it may yield advantages in terms of diminished makespan.

3.3.3.4 TOPSIS Algorithm

A critical component of TETRA is the multi-criteria-decision-analysis (MCDA) method that is used to rank the resources so that the appropriate allocation decision can be taken. MCDA methods are widely used in various fields such as Operations Research in order to evaluate alternatives based on multiple criteria. These methods often help decision-makers to select the best alternative from a set of options, where the options have different trade-offs between different criteria. Some common MCDA methods are TOPSIS, AHP, ELECTRE, and PROMETHEE [103-104]. Among these methods, we have identified the TOPSIS method for usage within our proposed TETRA model due to its simplicity, flexibility, and ability to handle both quantitative and qualitative criteria. One of the advantages of using TOPSIS over the other MCDA methods is that it provides a clear ranking of alternatives, making it easy for decision-makers to choose the best option.

3.3.3.4.1 MCDA Method – TOPSIS

This method is based on the concept of identifying the alternative (resource) that is closest to the ideal solution and furthest from the worst solution. TOPSIS uses a set of weighted criteria to evaluate the options and ranks them based on their similarity to the ideal solution.

Table 4 contains the attributes we selected to identify and rank the available resources.

Table 4. Resource Attributes

Attribute	Description
MIPS	MIPS, or Million Instructions Per Second is an approximate measure of computer's raw processing power, representing the number of instructions it can execute within a single second [105].
CPU Cores	The number of cores the resource possesses.
RAM	Random-access memory is the main memory of the resource.
TDP	Thermal Design Power refers to the power consumption under the maximum theoretical load [106].

For each cloudlet, a table is created for all the available resources. Table 5 presents a sample decision matrix for the TOPSIS algorithm.

Table 5. Mock Decision Matrix

VM #	MIPS	CPU Cores	RAM (GB)	TDP (W)
VM1	400	4	8	150
VM2	500	8	16	300

VM3	110	4	16	115
VM4	300	8	16	215
VM5	700	16	16	300

Based on these values, a normalized matrix is created by populating each cell of the matrix using Equation 1.

$$\bar{x}_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{n} x_{ij}^2}}$$

Equation 1[107]

where $\bar{x}_{ij}$ is the normalized value of the cell with coordinates (i, j). x_{ij} refers to the value in the cell with coordinates (i, j) in the original matrix. As a next step, the weighted normalized decision matrix is created by multiplying each value in a column with the corresponding user-specified weight of the attribute associated with that column. Then, the ideal best and the ideal worst are calculated. This depends on whether the attribute is beneficial or not. In case an attribute is beneficial, the ideal best is the maximum value in that particular column and the ideal worst is the minimum value in that particular column. The case is reversed if the attribute is not beneficial.

Now, we calculate the Euclidean distance from the ideal best for elements in all rows, using Equation 2.

$$S_i^+ = \sqrt{\sum_{j=1}^{m} \left(v_{ij} - v_j^+\right)^2}$$

Equation 2 [107]

where v_{ij} is the value in the cell with coordinates (i, j), v_j^+ is the ideal best of column j and m is the number of alternatives. Similarly, the Euclidean distance from the ideal worst is calculated using Equation 3.

$$S_i^- = \sqrt{\sum_{j=1}^{m} \left(v_{ij} - v_j^-\right)^2}$$

Equation 3 [107]

Then the performance score for each alternative (resource) will be calculated using Equation 3.

$$P_i = \frac{S_i^-}{S_i^+ + S_i^-}$$ *Equation 4[107]*

where P_i is the performance score of row i.

Finally, the alternatives (resources) will be ranked in decreasing order of their performance score. So, the resource with the highest performance score will be the best alternative and the resource with the lowest performance score will be the worst alternative.

3.3.3.4.2 Weight Estimation Method

TETRA uses a manual weight estimation method. The user gets to choose the weights for each attribute. Among other factors, changing the weights of attributes changes the ranking of the alternatives as well. The user should distribute the weights in such a way that globally energy consumption is reduced while keeping the makespan low.

Table 6 contains a sample weight distribution for the attributes.

Table 6. Sample attribute weight distribution

MIPS (%)	CPU Cores (%)	RAM (%)	TDP (%)
40	30	5	25

In Chapter 6, we will conduct several experiments where the weight distribution among the attributes is changed while other experiment parameters are kept constant to arrive at a configuration/s that satisfy the optimization goals.

3.3.4 Post Filtration

Once the resources are ranked using the TOPSIS algorithm. Executing tasks on higher ranked resources will lead to lower energy consumption and lower makespan. To ensure this, the user can specify post-filtration criterion p. The top p percentage of resources are filtered and selected for task execution. To make sure there is no bias and other algorithms have access to similar number of resources, the same post filtration criterion is applied in other scheduling algorithms as well.

The First-Fit algorithm scheduler applies this criterion by choosing a p percentage of resources randomly.

The Greedy and Best-Fit algorithm scheduler applies this criterion by choosing the top p percentage of resources.

3.3.5 Resource Scheduling

In this step we iterate through a list of cloudlets, mapping each one to the first available resource capable of executing it, and cycle through the resource list in a circular manner.

3.3.6 Cloudlet Execution

The mapping the previous step is passed to the broker in CloudSim Plus framework for execution. The broker returns a list of finished cloudlets using the "getCloudletFinishedList()" method.

3.3.7 Energy Consumption Estimation

The total energy consumption of the datacenter is estimated by calculating the total energy consumed by executing all the cloudlets. It is represented by the following pseudocode:

```
# Initialize total energy variable to zero Joules.
Total_energy = 0

# Iterate over each finished cloudlet
for cloudlet in finished_cloudlets:

    # Get the VM mapping of the current cloudlet
    currVM = cloudlet.getVM()
    # Get the number of cores of currVM
    currCoresVM = currVM.getCores()
    # Get the number of cores of the Host of currVM
    currCoresHost = currVM.getHost().getCores()
    # Get the TDP of currVM
    tdp = currVM.getTDP()
    # Get the execution time of the current cloudlet
    execTime = cloudlet.finishedTime() - cloudlet.startTime()

    # Calculate the Energy in kWh
    energy = (((currCoresVM/currCoresHost)*tdp)/1000)*(execTime/3600)
    # Convert to Joules
    energy *= 3600000
    # Append energy to total_energy
    total_energy += energy

# Return the total energy consumed to execute all cloudlets
return total_energy
```

Figure 27. Energy consumption estimation pseudocode

With the energy consumption estimated above, the total data center cost associated with executing all the cloudlets can also be estimated. For instance, if the data center is in a state where the electricity rate is 13 ¢/kWh. Then the total cost can be estimated using Equation 5.

$$Total\_Cost = energy * 13 \qquad \textit{Equation 5}$$

3.3.8 Makespan Estimation

The total makespan is defined as the maximum execution time of a cloudlet across all VMs. It is calculated by the following the process:

```
# Initialize makespan
makespan = -math.inf

# Initialize the total execution time array
vmExecTimes = []
for I in range(0,vmList.size()):
    vmExecTimes.append(0)

# Iterate over each finished cloudlet
for cloudlet in finished_cloudlets:

    # Get the VM mapping of the current cloudlet
    currVM = cloudlet.getVM()
    # Get the execution time of the current cloudlet
    execTime = cloudlet.finishedTime() - cloudlet.startTime()
    # Get the VM id
    vmIdx = cloudlet.getVM().getId()
    # Get the current execution time for the VM
    curreExTime = vmExecTimes[vmIdx]
    # Append current cloudlet execution time
    vmExecTimes[vmIdx] += execTime

# Iterate over the total execution time array
for totalExecTime in vmExecTimes:
    if totalExecTime > makespan:
        makespan = totalExecTime

# Return the makespan
return makespan
```

Figure 28. Makespan estimation pseudocode

Chapter 4: Evaluation

The primary goal of our experiments is to identify the ideal weight configurations at which our FFTA optimizes both energy consumption and makespan algorithm under a wide range of conditions. To this extent, we test our algorithm under various scenarios, varying in the degree of available VMs and cloudlets. The number of cloudlets varies from 100 to 1000. The initial number of hosts and VMs vary from 73 and 153, respectively to 292 and 612, respectively.

Table 7 shows the attributes we consider for ranking the resources.

Table 7. Resource Attributes for TOPSIS Algorithm

Attribute	Beneficial
MIPS	✓
CPU Cores	X
TDP	X

We regard MIPS as a beneficial attribute because the higher the MIPS of a virtual machine (VM), the greater the number of instructions it can execute per second, thereby leading to faster execution times. This contributes to minimizing the makespan, ensuring quicker completion of tasks.

We consider the number of cores of VM (CPU Cores) to be a non-beneficial attribute. Typically, resources with higher core counts are not as energy efficient. Thus, our aim is to identify resources that closely match the CPU core requirements of the cloudlets. This strategy results in reduced energy consumption.

Next, we regard TDP as a non-beneficial attribute. This is because the lower the TDP rating of a resource, the less energy it consumes.

In this section we group the results into three groups. The first group represents low competition amongst the resources. The second and the third groups represent medium and high competition. As we move from the first group to the third group, the number of resources drastically reduces, creating a progressively more challenging environment for resource scheduling optimization.

We repeat a series of experiments for each group. Each experiment differs based on the TOPSIS weight distribution for each attribute. The weight for each attribute is tested across the range (0,100). Table 8 presents the weight distribution for each of the experiments we conducted.

Table 8. Attribute Weight Distribution

Experiment #	Host Count	VM Count	Post-Filtration (%)	Final VM Count	Competition	MIPS (%)	CPU Cores (%)	TDP (%)
1	292	612	60	367	Low	0	0	100
2	292	612	60	367	Low	20	0	80
3	292	612	60	367	Low	50	0	50
4	292	612	60	367	Low	80	0	20
5	292	612	60	367	Low	0	100	0
6	292	612	60	367	Low	0	80	20
7	292	612	60	367	Low	0	50	50
8	292	612	60	367	Low	0	20	80
9	292	612	60	367	Low	100	0	0
10	292	612	60	367	Low	80	20	0
11	292	612	60	367	Low	50	50	0
12	292	612	60	367	Low	20	80	0
13	292	612	60	367	Low	50	25	25
14	292	612	60	367	Low	25	50	25
15	292	612	60	367	Low	25	25	50
16	146	306	60	183	Medium	0	0	100
17	146	306	60	183	Medium	20	0	80
18	146	306	60	183	Medium	50	0	50
19	146	306	60	183	Medium	80	0	20
20	146	306	60	183	Medium	0	100	0
21	146	306	60	183	Medium	0	80	20

22	146	306	60	183	Medium	0	50	50
23	146	306	60	183	Medium	0	20	80
24	146	306	60	183	Medium	100	0	0
25	146	306	60	183	Medium	80	20	0
26	146	306	60	183	Medium	50	50	0
27	146	306	60	183	Medium	20	80	0
28	146	306	60	183	Medium	50	25	25
29	146	306	60	183	Medium	25	50	25
30	146	306	60	183	Medium	25	25	50
31	73	153	60	91	High	0	0	100
32	73	153	60	91	High	20	0	80
33	73	153	60	91	High	50	0	50
34	73	153	60	91	High	80	0	20
35	73	153	60	91	High	0	100	0
36	73	153	60	91	High	0	80	20
37	73	153	60	91	High	0	50	50
38	73	153	60	91	High	0	20	80
39	73	153	60	91	High	100	0	0
40	73	153	60	91	High	80	20	0
41	73	153	60	91	High	50	50	0
42	73	153	60	91	High	20	80	0
43	73	153	60	91	High	50	25	25
44	73	153	60	91	High	25	50	25
45	73	153	60	91	High	25	25	50

5.1 Low Resource Competition (LRC)

This group of experiments represents cases where resources are abundant. There are 612 VMs hosted on 292 hosts. The post-filtration VM count is 367. The number of cloudlets ranges from 100 to 1000 in increments of 100. Thus, the ratio of cloudlets to VMs is ~3:1.

Experiment 1-15 represent this group.

5.1.2 Low Resource Competition Results

Results in Figures 29-43 represent the experiments 1 through 15 which vary in terms of the weight configuration ranging between 0 to 100 for each of the considered attributes including MIPS, CPU cores, and TDP. Each figure presents energy consumption and makespan results across varying number of cloudlets (tasks) along with a comparison among four specific types of algorithms including First Fit Algorithm (FFA), First Fit with TOPSIS Algorithm (FFTA), Best Fit Algorithm (BFA), and Greedy Algorithm (GA).

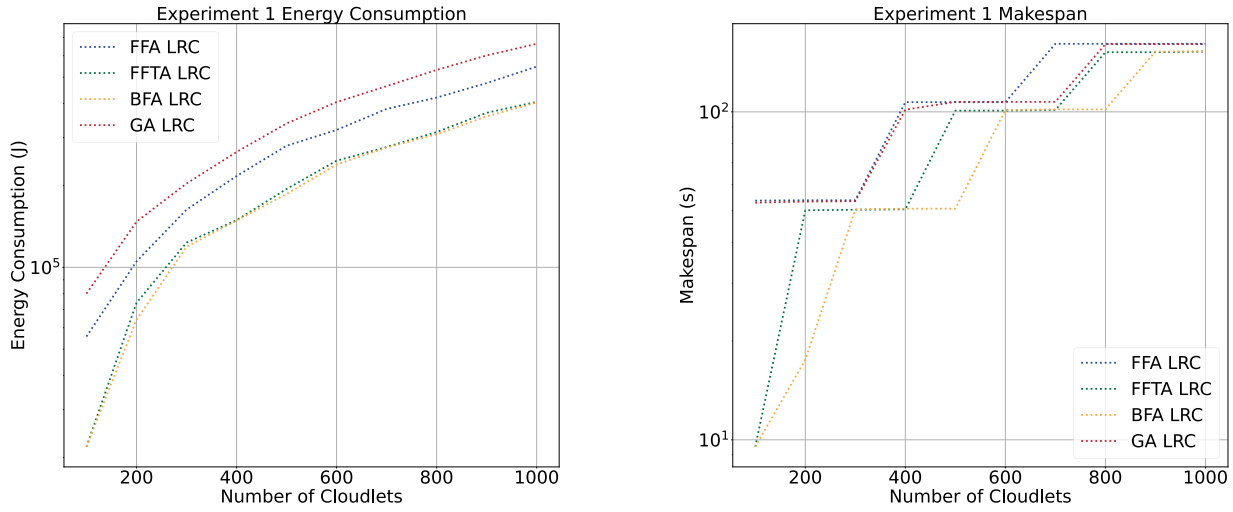

Figure 29. Experiment 1 FFTA weight configuration: MIPS=0%, CPU Cores=0%, TDP=100% (a) Energy Consumption (b) Makespan

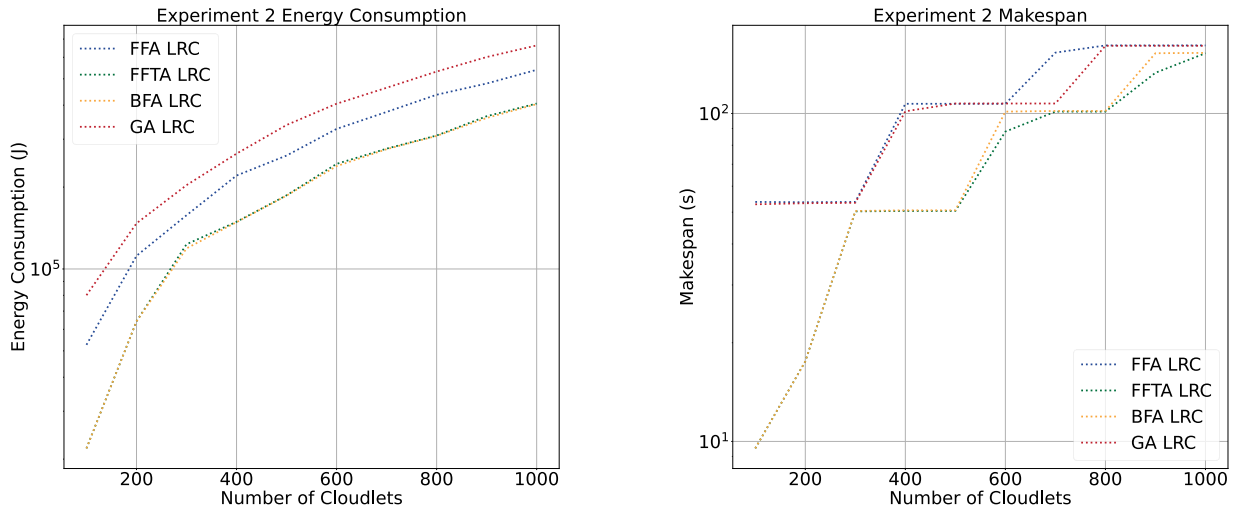

Figure 30. Experiment 2 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

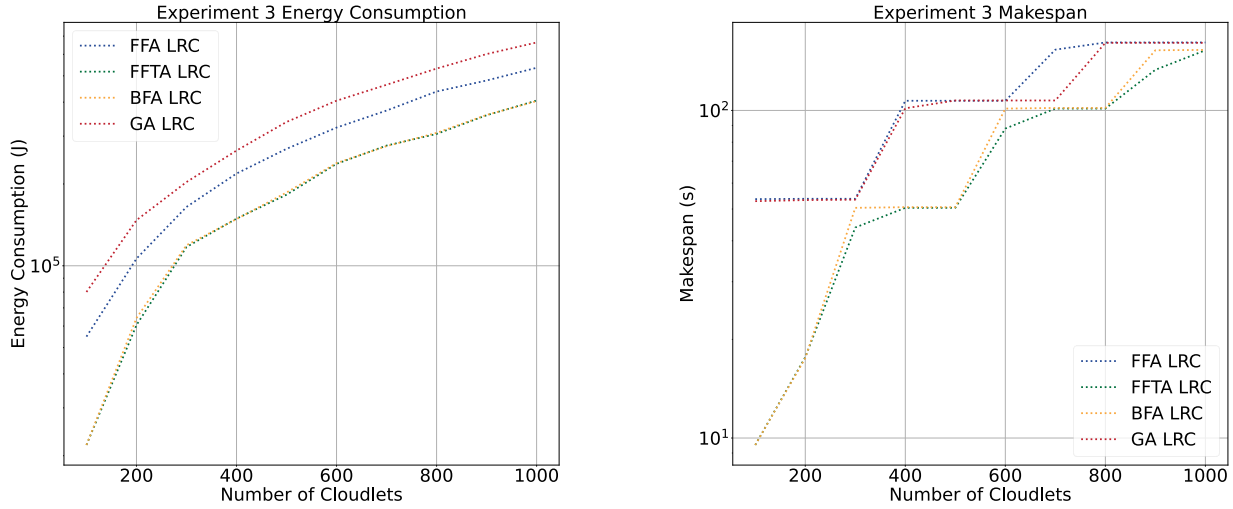

Figure 31. Experiment 3 FFTA weight configuration: MIPS=50%, CPU Cores=0%, TDP=50% (a) Energy Consumption (b) Makespan

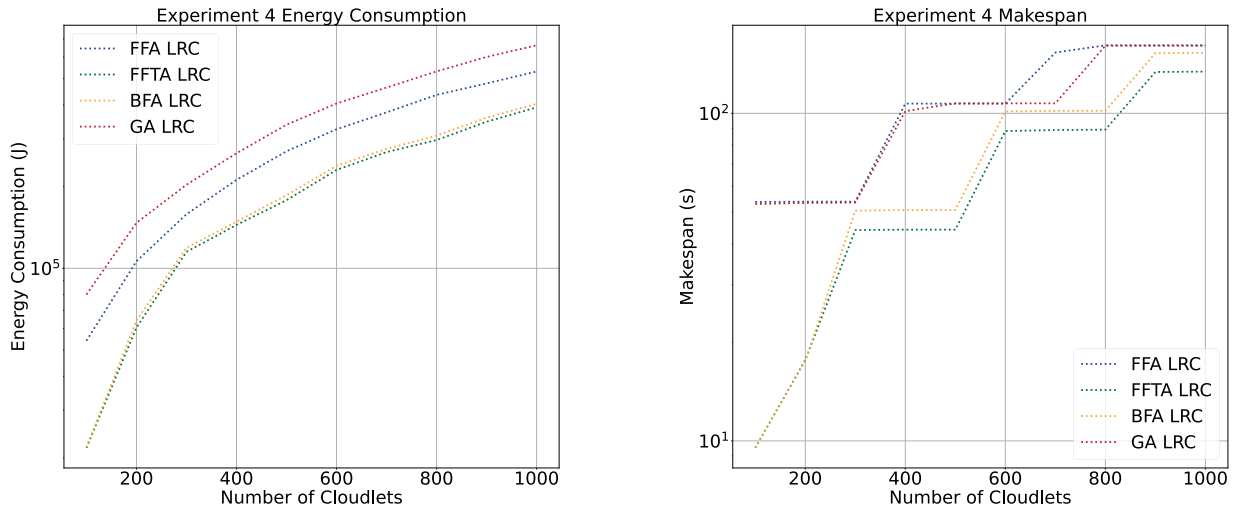

Figure 32. Experiment 4 FFTA weight configuration: MIPS=80%, CPU Cores=0%, TDP=20% (a) Energy Consumption (b) Makespan

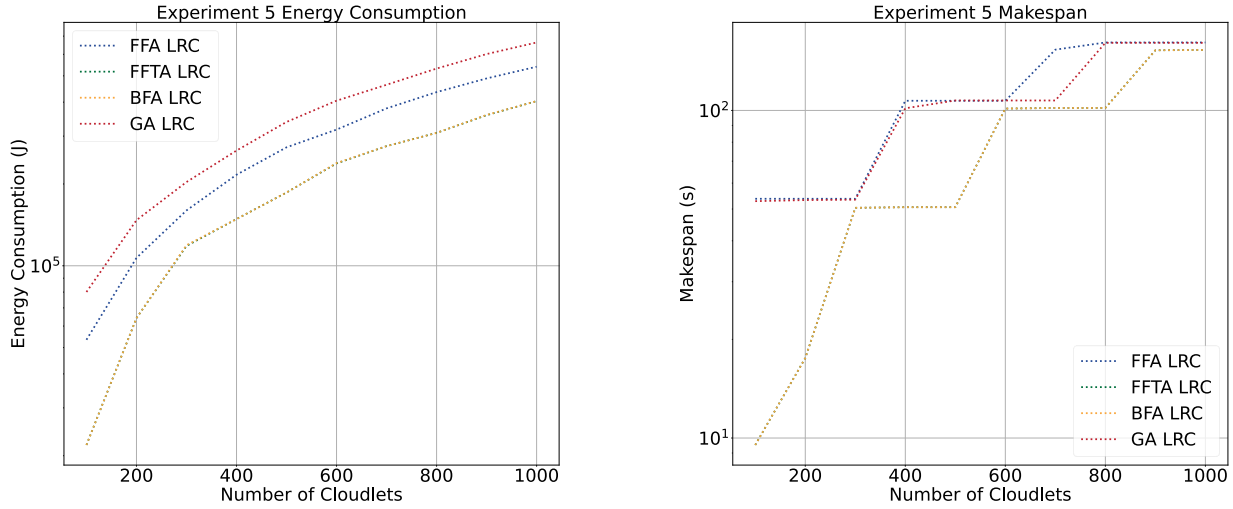

Figure 33. Experiment 5 FFTA weight configuration: MIPS=0%, CPU Cores=100%, TDP=0% (a) Energy Consumption (b) Makespan

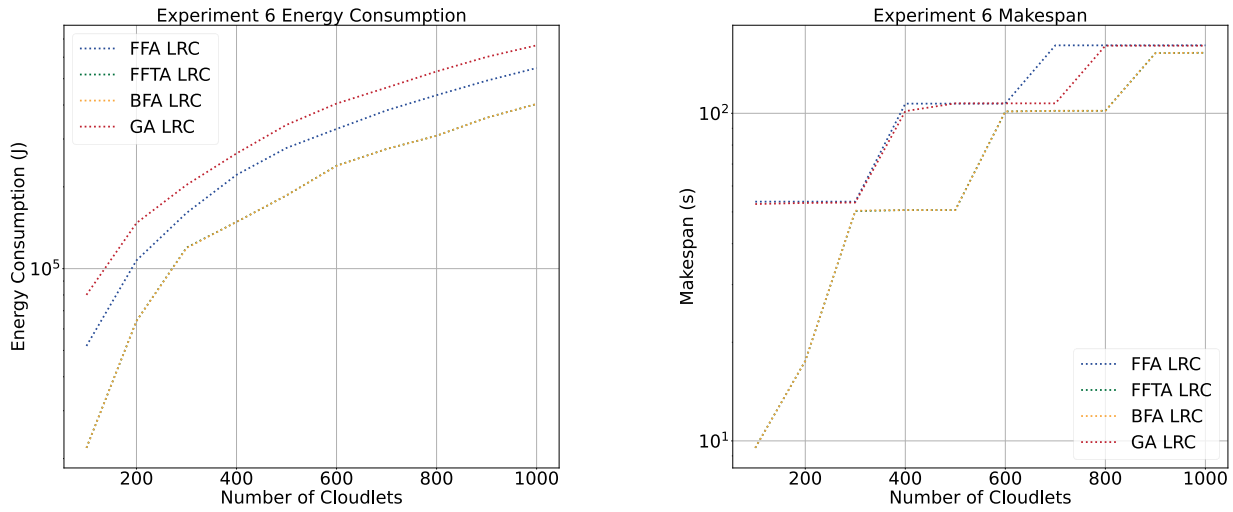

Figure 34. Experiment 6 FFTA weight configuration: MIPS=0%, CPU Cores=80%, TDP=20% (a) Energy Consumption (b) Makespan

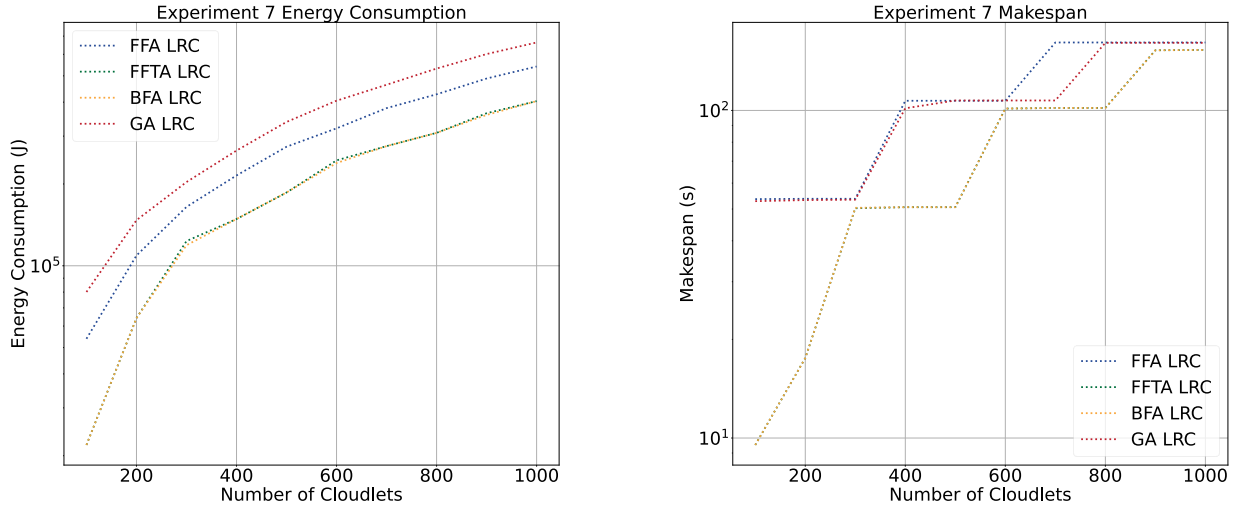

Figure 35. Experiment 7 FFTA weight configuration: MIPS=0%, CPU Cores=50%, TDP=50% (a) Energy Consumption (b) Makespan

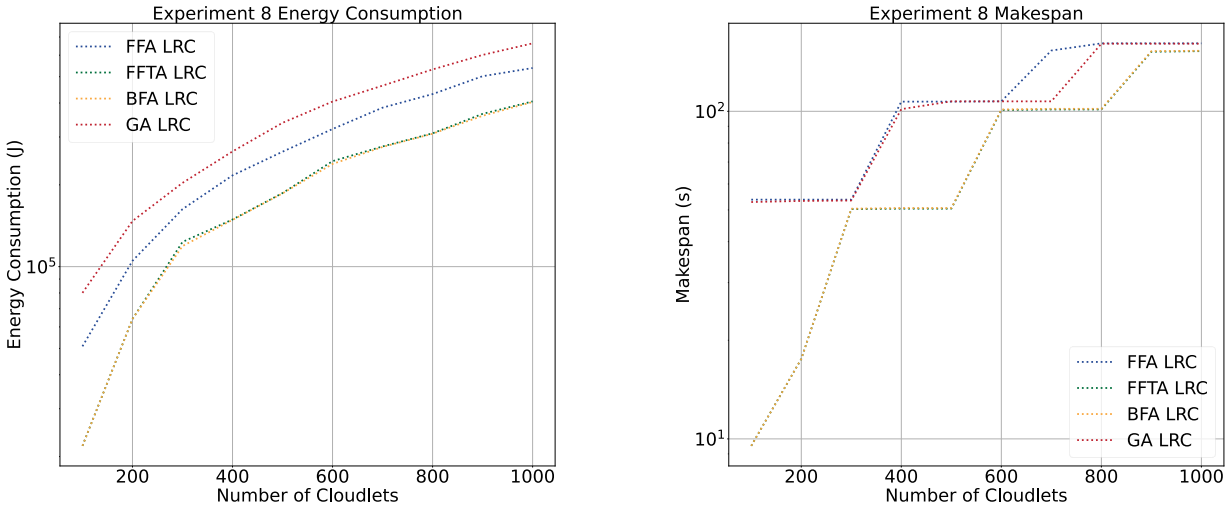

Figure 36. Experiment 8 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

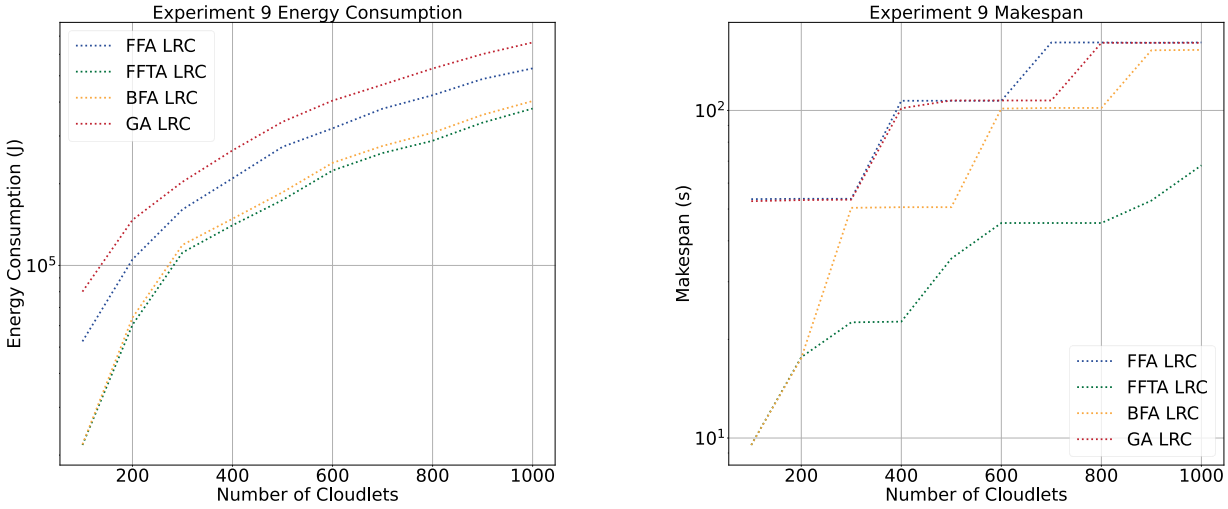

Figure 37. Experiment 9 FFTA weight configuration: MIPS=100%, CPU Cores=0%, TDP=0% (a) Energy Consumption (b) Makespan

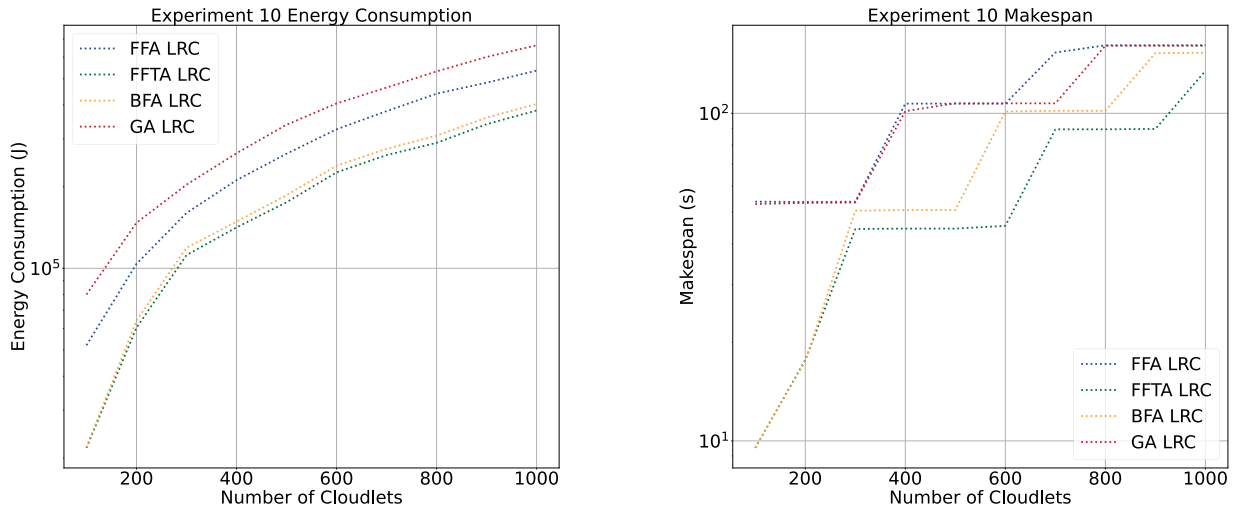

Figure 38. Experiment 10 FFTA weight configuration: MIPS=80%, CPU Cores=20%, TDP=0% (a) Energy Consumption (b) Makespan

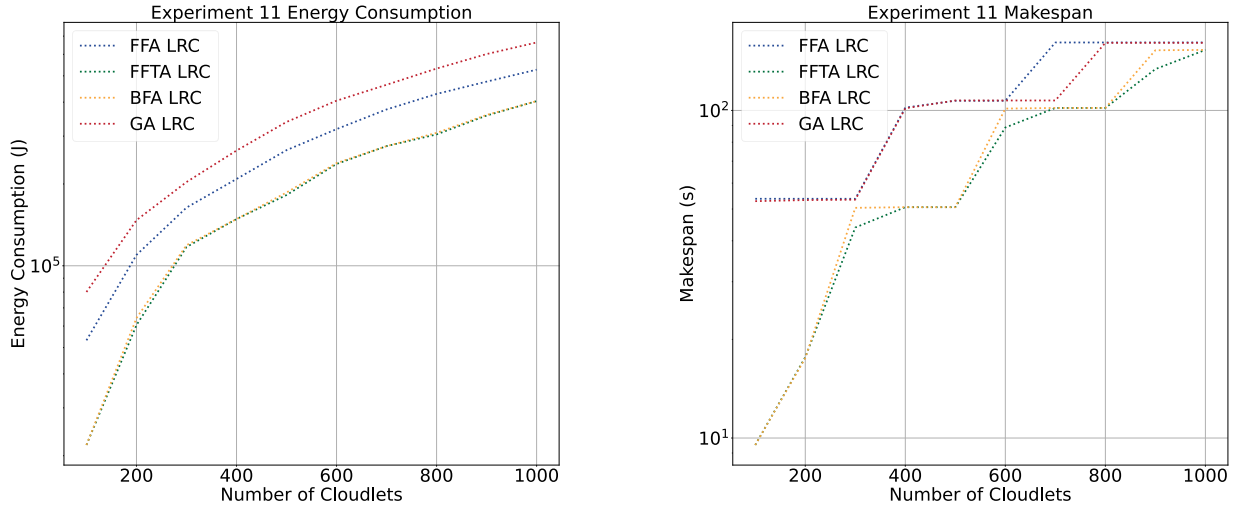

Figure 39. Experiment 11 FFTA weight configuration: MIPS=50%, CPU Cores=50%, TDP=0% (a) Energy Consumption (b) Makespan

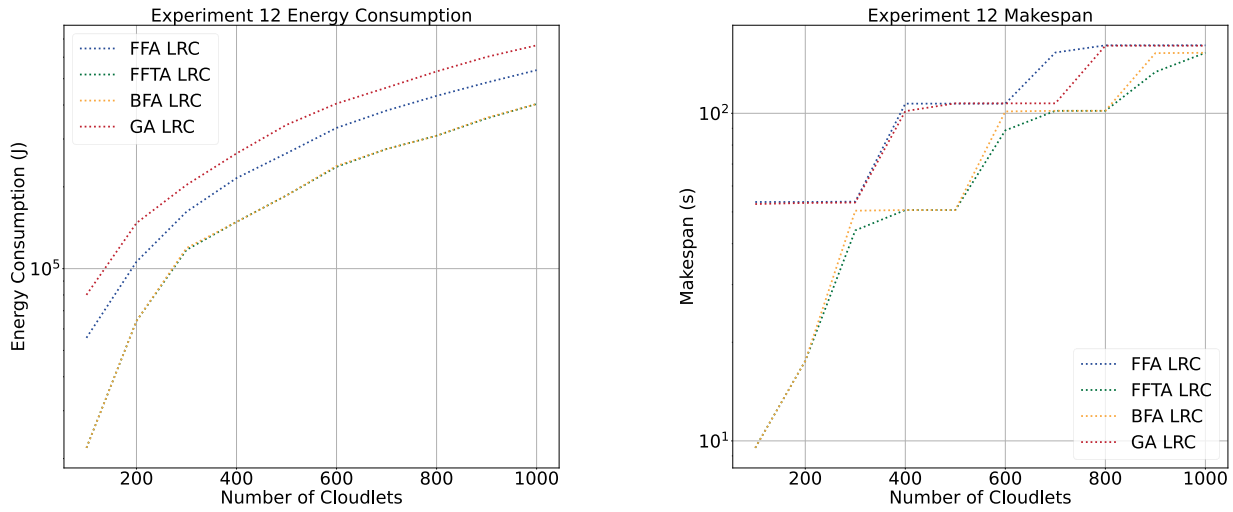

Figure 40. Experiment 12 FFTA weight configuration: MIPS=20%, CPU Cores=80%, TDP=0% (a) Energy Consumption (b) Makespan

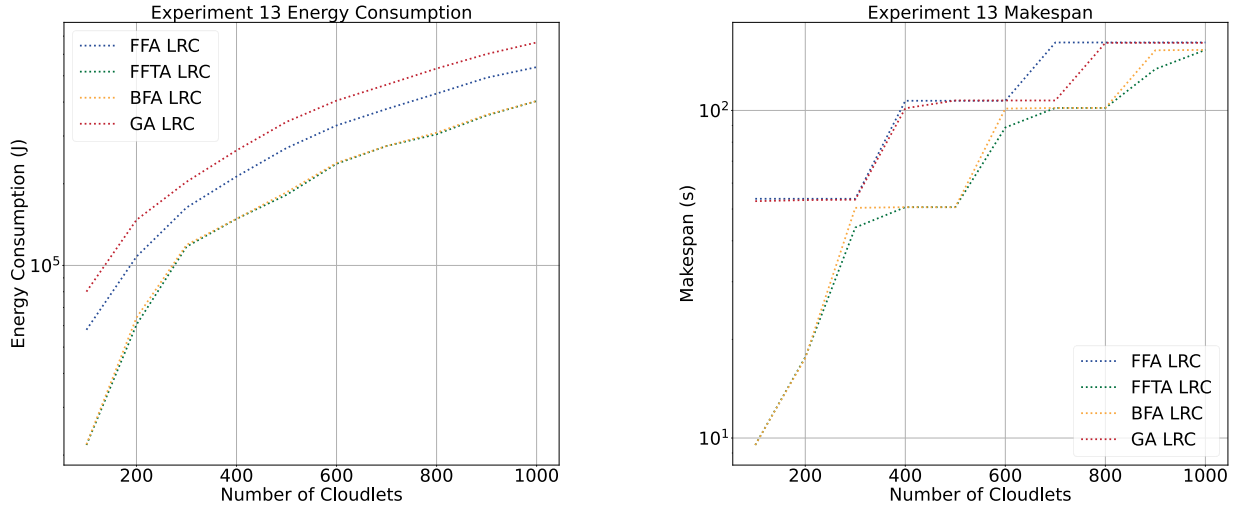

Figure 41. Experiment 13 FFTA weight configuration: MIPS=50%, CPU Cores=25%, TDP=25% (a) Energy Consumption (b) Makespan

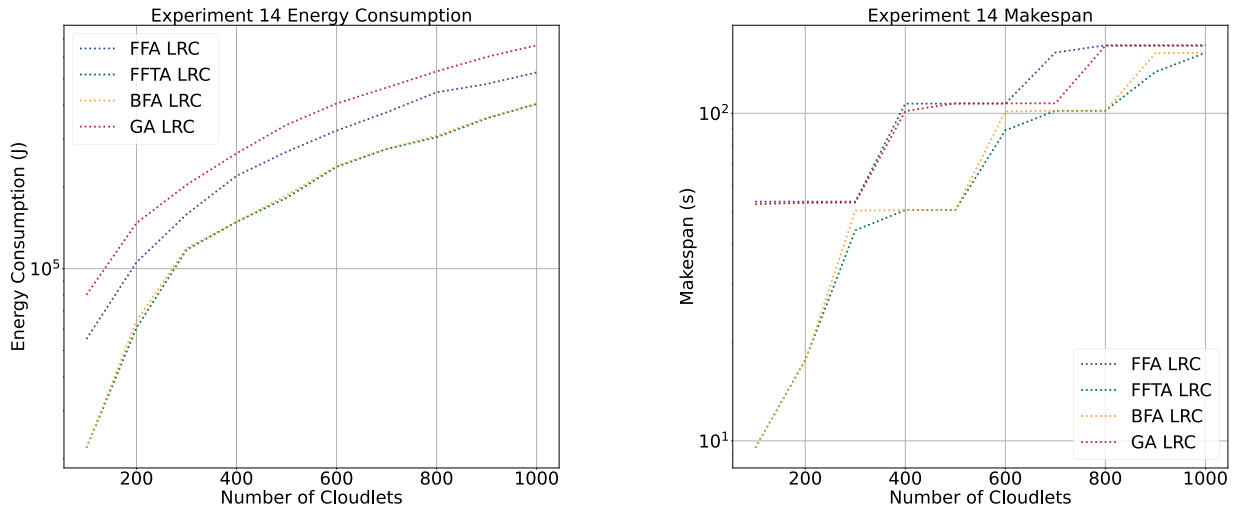

Figure 42. Experiment 14 FFTA weight configuration: MIPS=25%, CPU Cores=50%, TDP=25% (a) Energy Consumption (b) Makespan

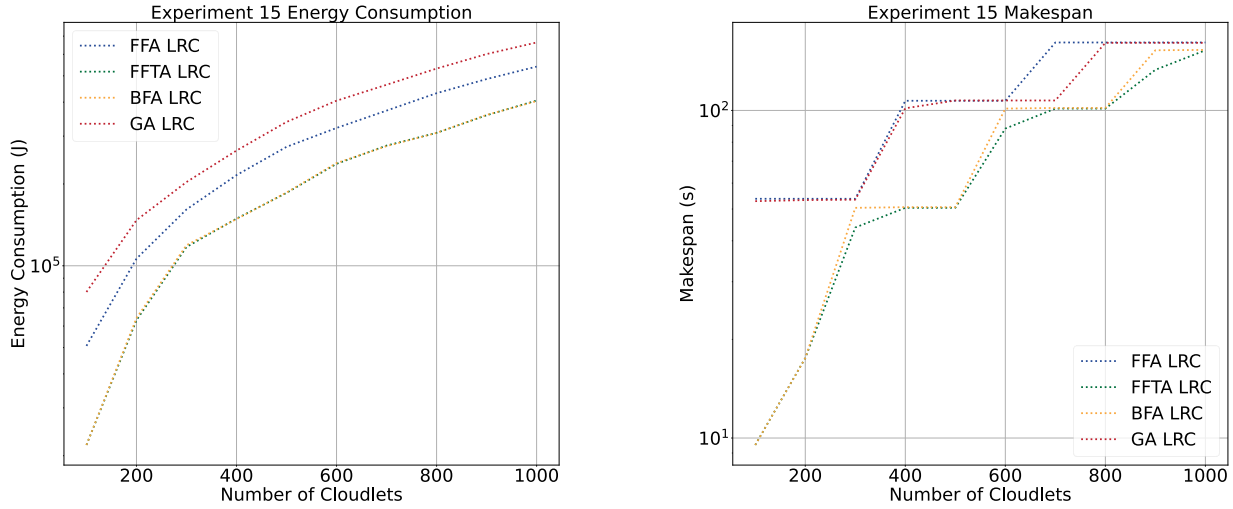

Figure 43. Experiment 15 FFTA weight configuration: MIPS=25%, CPU Cores=25%, TDP=50% (a) Energy Consumption (b) Makespan

Table 9 presents detailed results summarizing experiments 1 through 15 for LRC.

Table 9. LRC Results for Experiments 1-15

Experiment #	TEC FFTA (kJ)	TM FFTA (s)	TEC FFA (kJ)	TM FFA (s)	TEC BFA (kJ)	TM BFA (s)	TEC GA (kJ)	TM GA (s)
1	2175.80	920.04	2965.58	1127.14	2124.87	789.33	3698.73	1065.37
2	2144.60	754.64	2965.48	1119.21	2124.87	789.33	3698.73	1065.37
3	2114.45	748.38	2962.14	1119.21	2124.87	789.33	3698.73	1065.37
4	2054.28	693.63	2948.15	1119.21	2124.87	789.33	3698.73	1065.37
5	2122.63	789.33	2970.37	1119.34	2124.87	789.33	3698.73	1065.37
6	2125.88	789.18	3001.31	1127.26	2124.87	789.33	3698.73	1065.37
7	2139.79	789.18	2974.82	1127.14	2124.87	789.33	3698.73	1065.37
8	2146.32	786.37	2977.32	1119.57	2124.87	789.33	3698.73	1065.37
9	1994.45	364.71	2942.60	1126.81	2124.87	789.33	3698.73	1065.37
10	2009.06	608.81	2954.62	1119.21	2124.87	789.33	3698.73	1065.37
11	2107.73	751.39	2927.26	1122.24	2124.87	789.33	3698.73	1065.37
12	2119.34	751.28	2969.70	1119.09	2124.87	789.33	3698.73	1065.37
13	2107.27	751.39	2977.28	1127.26	2124.87	789.33	3698.73	1065.37
14	2107.73	751.39	2955.94	1119.34	2124.87	789.33	3698.73	1065.37
15	2122.59	748.27	2960.07	1127.26	2124.87	789.33	3698.73	1065.37

5.1.2.1 Energy Consumption Results

Table 10 presents a comparison of the energy consumption performance of FFTA with FFA, BFA, and GA in LRC environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates an optimization in terms of the total energy consumption. As the goal of TETRA is to optimize energy consumption compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 10. LRC Energy Consumption Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
1	-26.63	+2.39	-41.17
2	-27.6	+0.92	-42.01
3	-28.6	-0.49	-42.83
4	-30.31	-3.32	-44.45
5	-28.53	-0.10	-42.61
6	-29.16	+0.00	-42.52
7	-28.00	+0.70	-42.14
8	-27.9	+1.00	-41.97
9	-32.22	-6.13	-46.07
10	-32.00	-5.450	-45.68
11	-28.66	-0.80	-43.01
12	-28.63	-0.26	-42.70
13	-28.71	-0.82	-43.02
14	-28.69	-0.80	-43.01
15	-28.29	-0.10	-42.61

5.1.2.2 Makespan Results

Table 11 presents a comparison of the makespan performance of FFTA with FFA, BFA, and GA in a LRC data center environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates an optimization in terms of the total makespan. As the goal of TETRA is to optimize makespan compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 11. LRC Makespan Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
1	-18.37	+16.55	-13.64
2	-32.57	-4.39	-29.16
3	-33.13	-5.18	-29.75
4	-38.025	-12.12	-34.89
5	-29.48	0.00%	-25.90
6	-29.99	-0.019	-25.92
7	-29.98	-0.019	-25.92
8	-29.76	-0.37	-26.18
9	-67.63	-53.79	-65.76
10	-45.60	-22.86	-42.85
11	-33.04	-4.80	-29.47
12	-32.86	-4.82	-29.48
13	-33.34	-4.80	-29.47
14	-32.87	-4.80	-29.47
15	-33.62	-5.20	-29.76

5.2 Medium Resource Competition (MRC)

In this group of experiments, we increase the competition rate amongst resources. Unlike LRC, MRC environment therefore has higher level of competition between tasks finding available resources to execute them. In this MRC setting, we employed 306 VMs hosted on 146 hosts. The post-filtration VM count is 183. The number of cloudlets ranges from 100 to 1000 in increments of 100. Thus, the ratio of cloudlets to VMs is ~5:1. Experiments 16-30 represent the ones that involve MRC setting.

5.2.1 Medium Resource Competition Results

Results in Figures 44-58 represent experiments 16 through 30 which vary in terms of the weight configuration ranging between 0 to 100 for each of the considered attributes including MIPS, CPU cores, and TDP. Each figure presents energy consumption and makespan results across varying number of cloudlets (tasks) along with a comparison among four specific types of algorithms including First Fit Algorithm (FFA), First Fit with TOPSIS Algorithm (FFTA), Best Fit Algorithm (BFA), and Greedy Algorithm (GA).

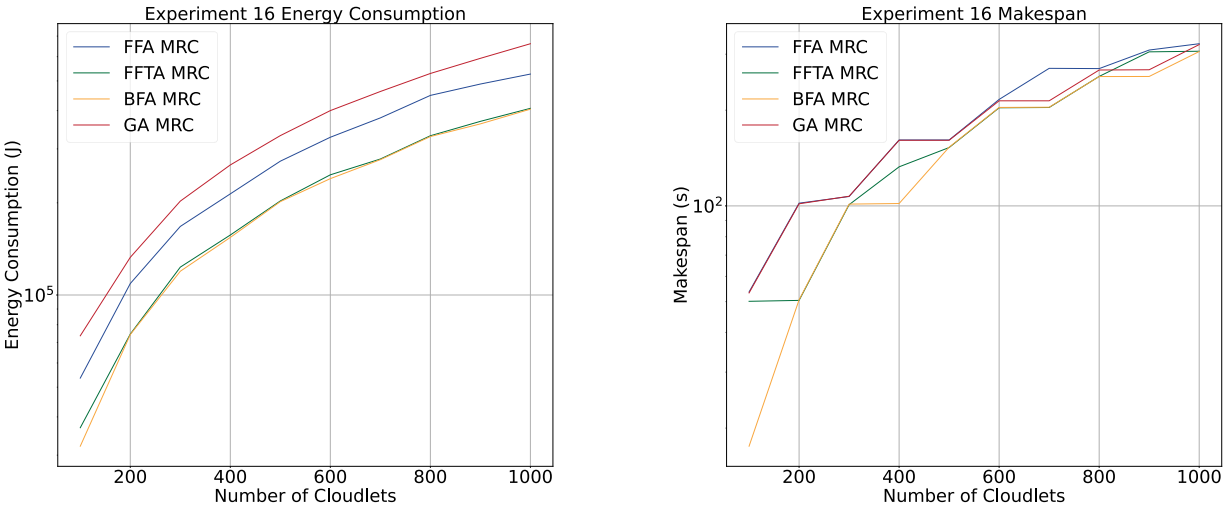

Figure 44. Experiment 16 FFTA weight configuration: MIPS=0%, CPU Cores=0%, TDP=100% (a) Energy Consumption (b) Makespan

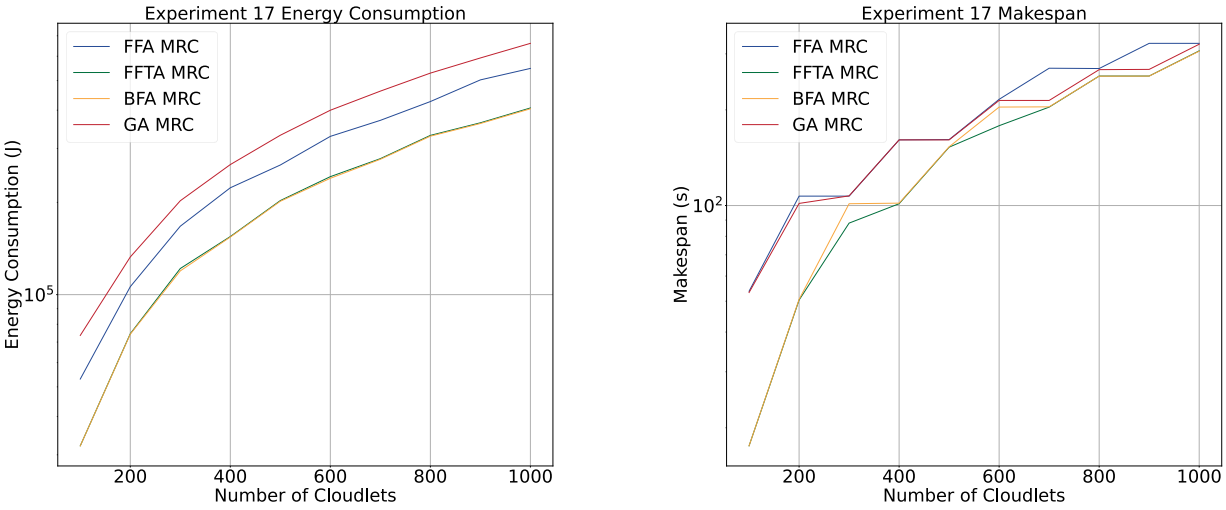

Figure 45. Experiment 17 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

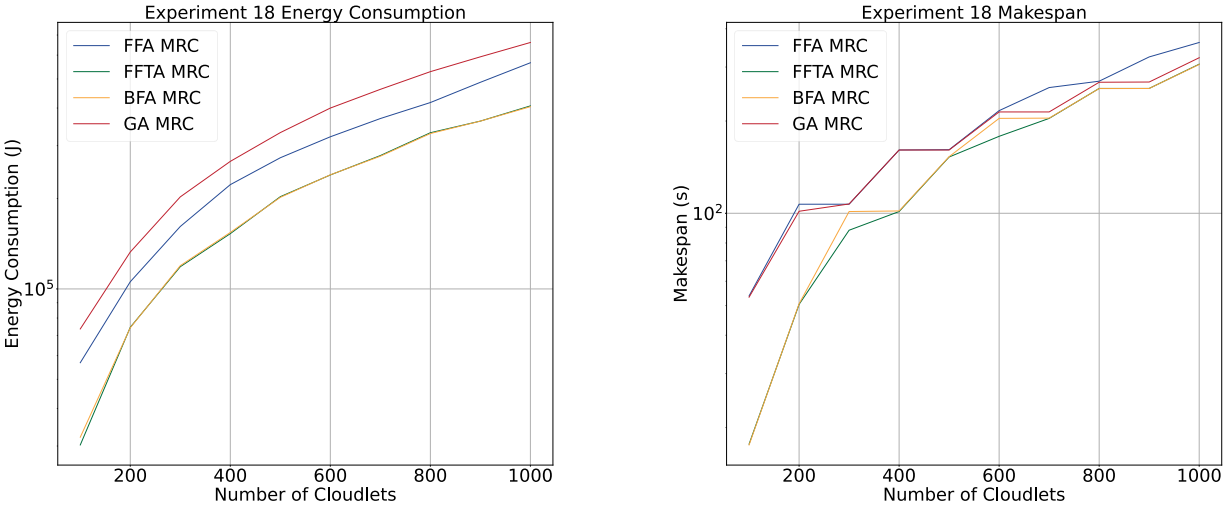

Figure 46. Experiment 18 FFTA weight configuration: MIPS=50%, CPU Cores=0%, TDP=50% (a) Energy Consumption (b) Makespan

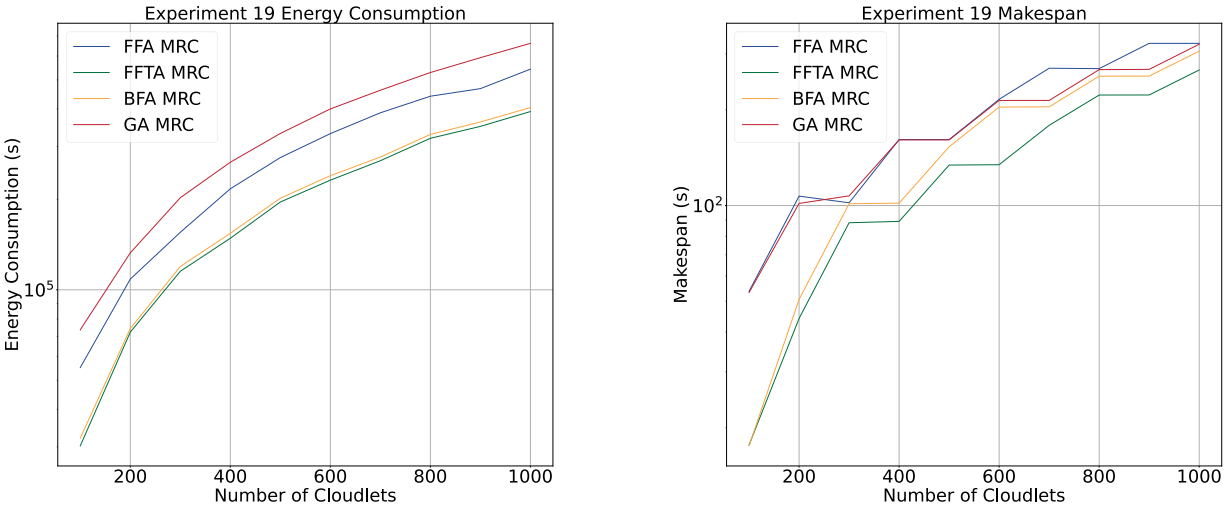

Figure 47. Experiment 19 FFTA weight configuration: MIPS=80%, CPU Cores=0%, TDP=20% (a) Energy Consumption (b) Makespan

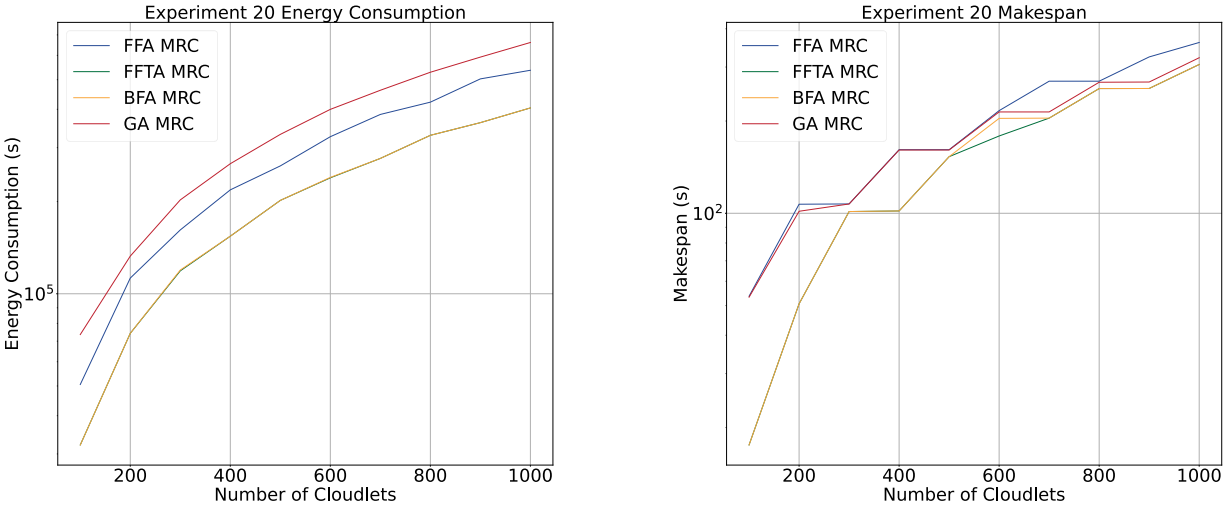

Figure 48. Experiment 20 FFTA weight configuration: MIPS=0%, CPU Cores=100%, TDP=0% (a) Energy Consumption (b) Makespan

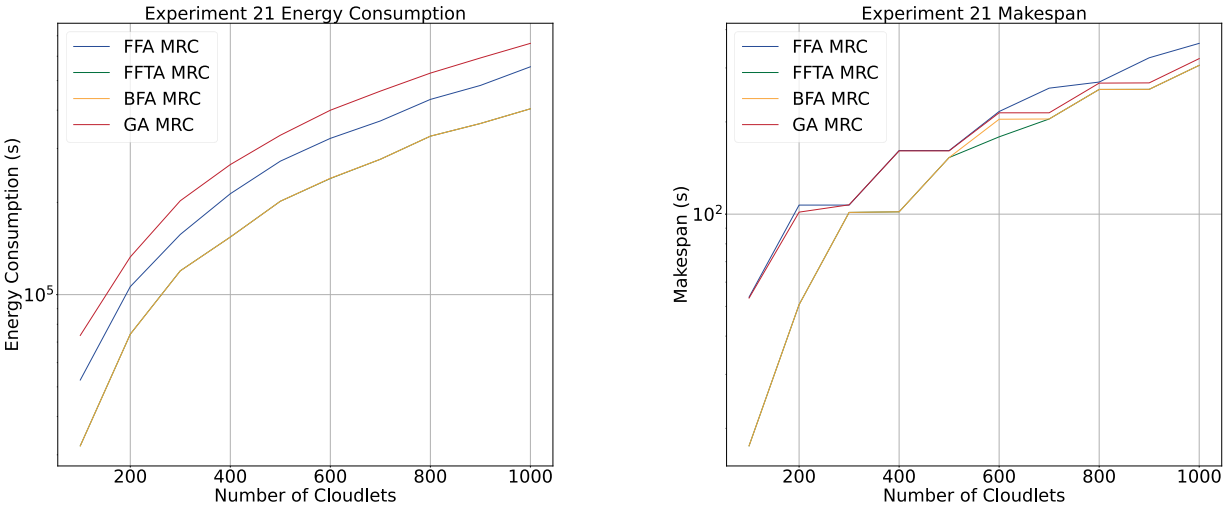

Figure 49. Experiment 21 FFTA weight configuration: MIPS=0%, CPU Cores=80%, TDP=20% (a) Energy Consumption (b) Makespan

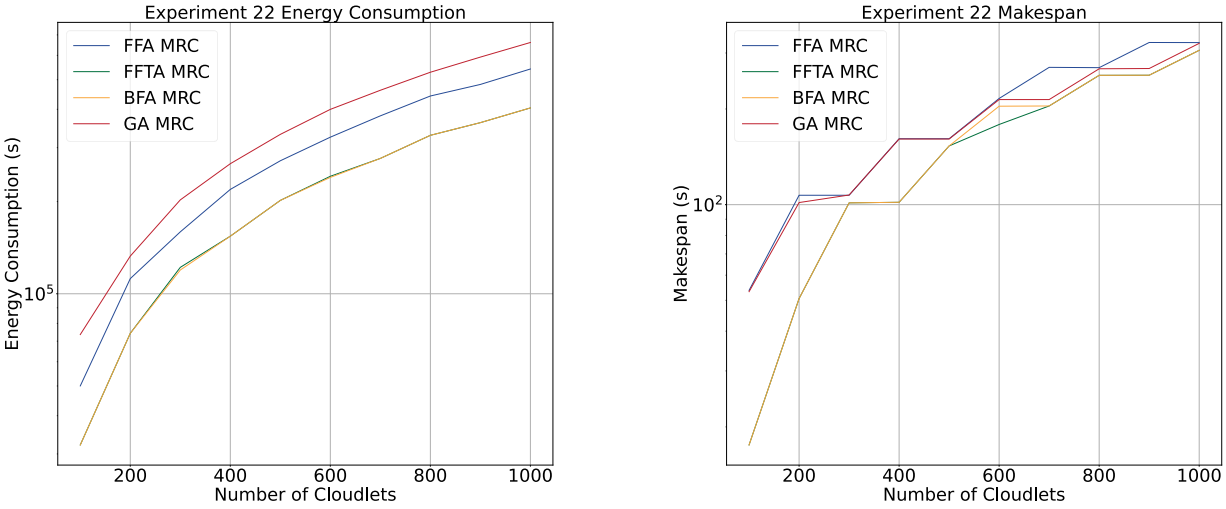

Figure 50. Experiment 22 FFTA weight configuration: MIPS=0%, CPU Cores=50%, TDP=50% (a) Energy Consumption (b) Makespan

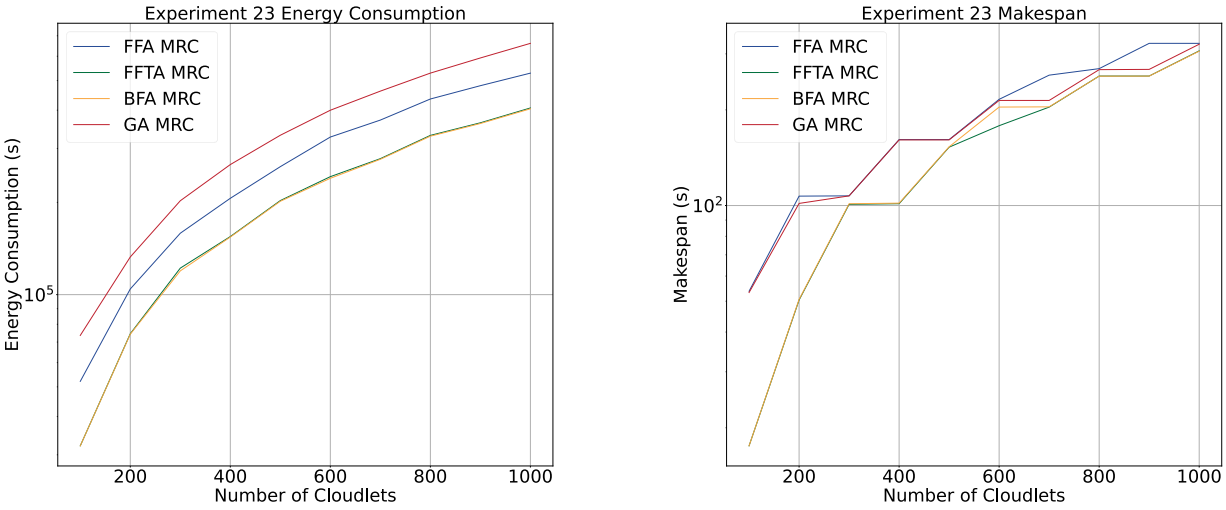

Figure 51. Experiment 23 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

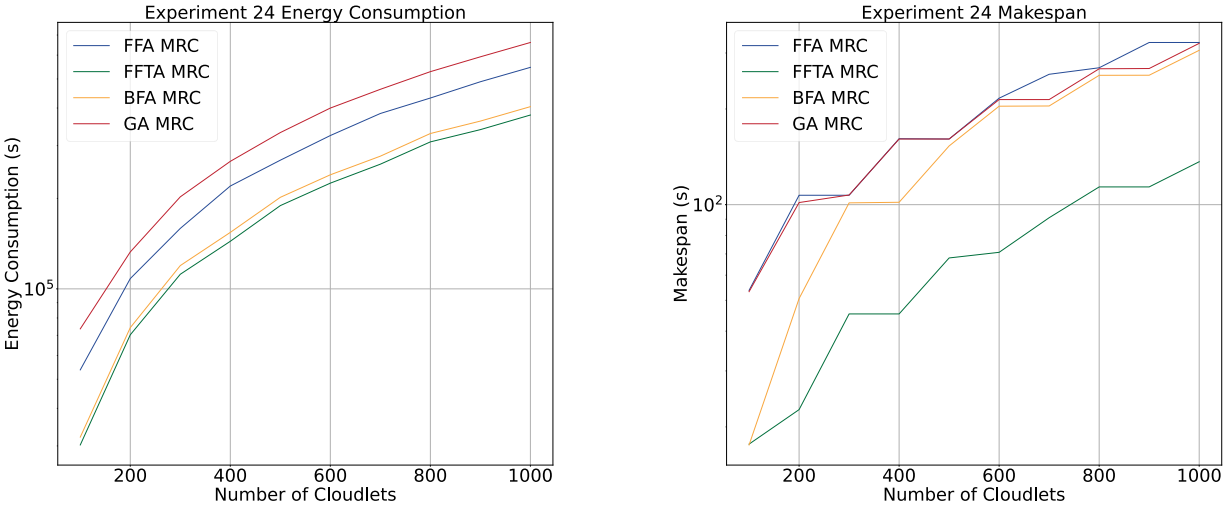

Figure 52. Experiment 24 FFTA weight configuration: MIPS=100%, CPU Cores=0%, TDP=0% (a) Energy Consumption (b) Makespan

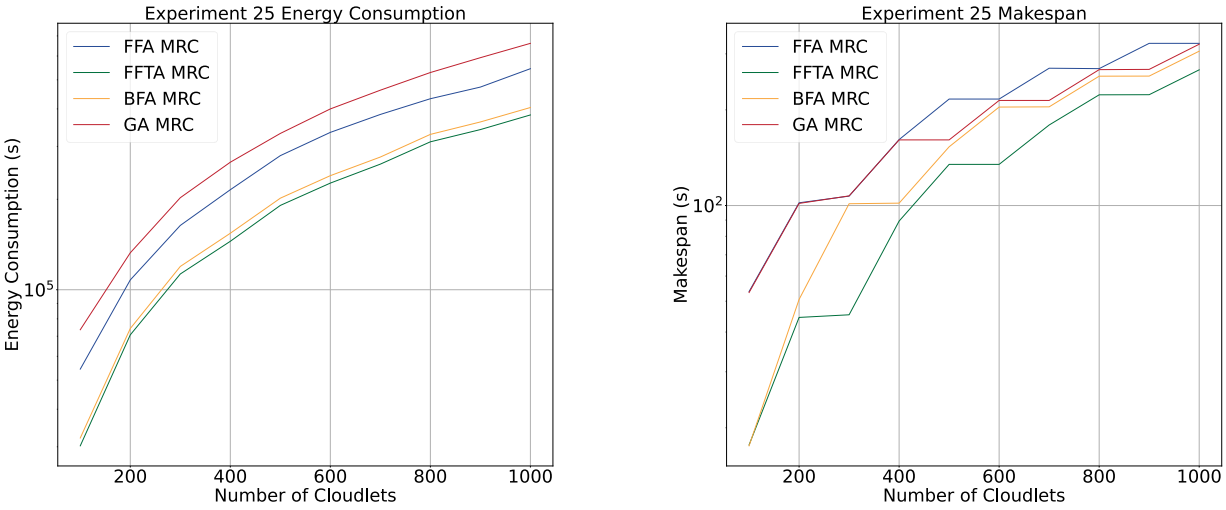

Figure 53. Experiment 25 FFTA weight configuration: MIPS=80%, CPU Cores=20%, TDP=0% (a) Energy Consumption (b) Makespan

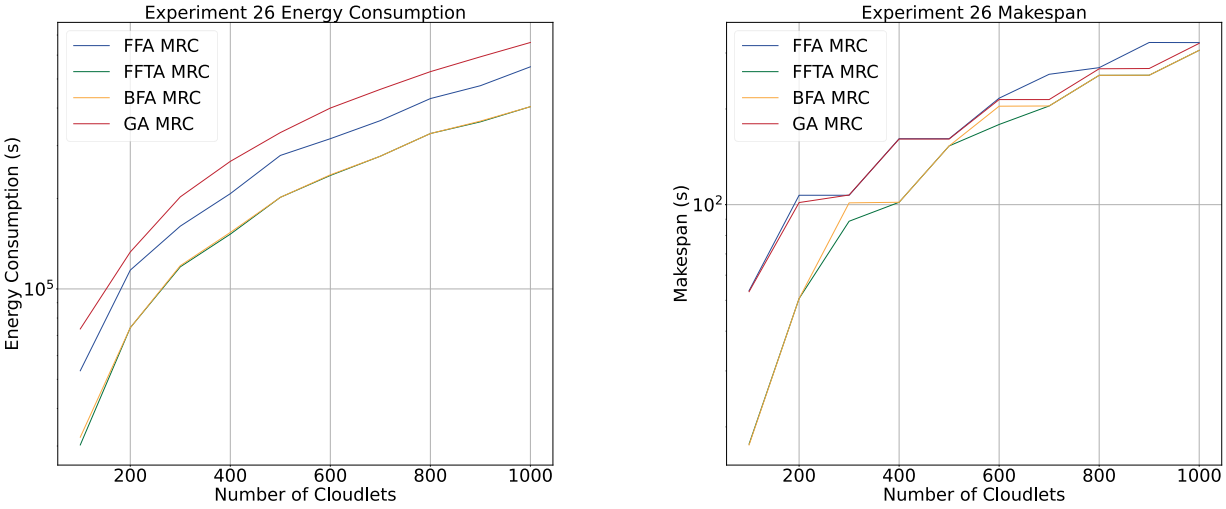

Figure 54. Experiment 26 FFTA weight configuration: MIPS=50%, CPU Cores=50%, TDP=0% (a) Energy Consumption (b) Makespan

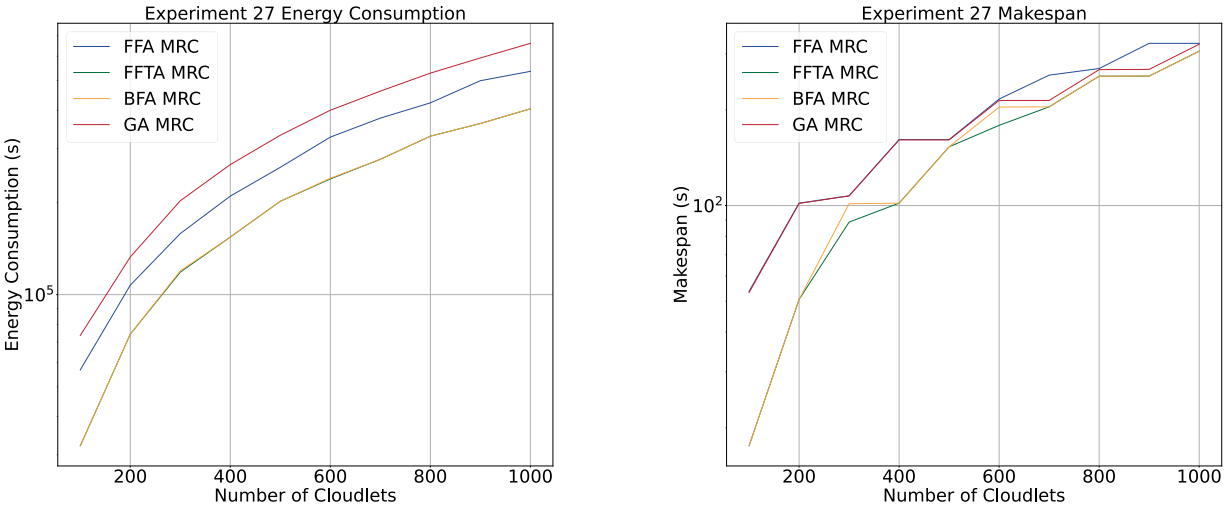

Figure 55. Experiment 27 FFTA weight configuration: MIPS=20%, CPU Cores=80%, TDP=0% (a) Energy Consumption (b) Makespan

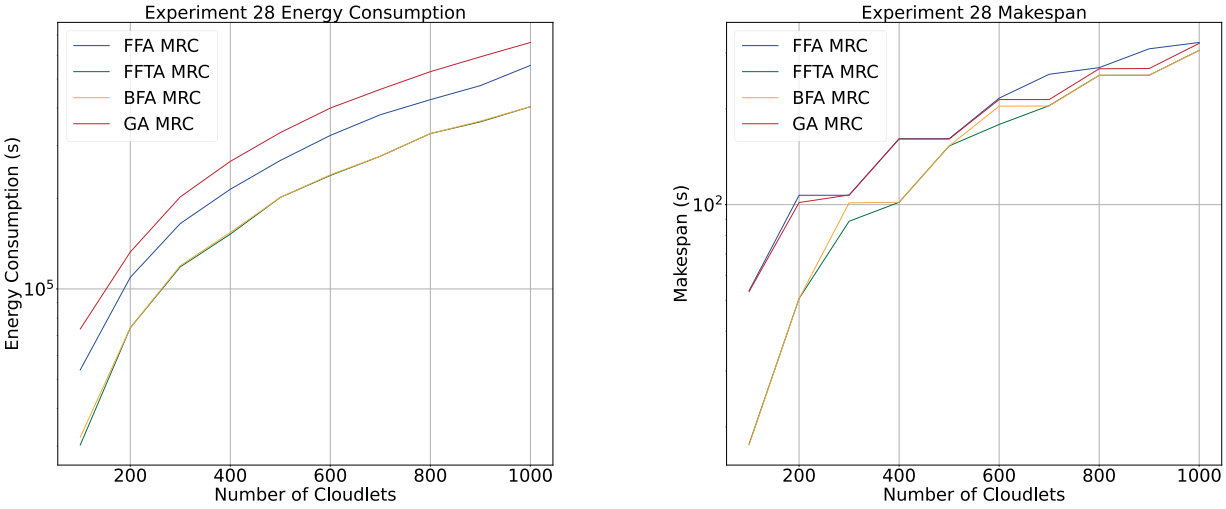

Figure 56. Experiment 28 FFTA weight configuration: MIPS=50%, CPU Cores=25%, TDP=25% (a) Energy Consumption (b) Makespan

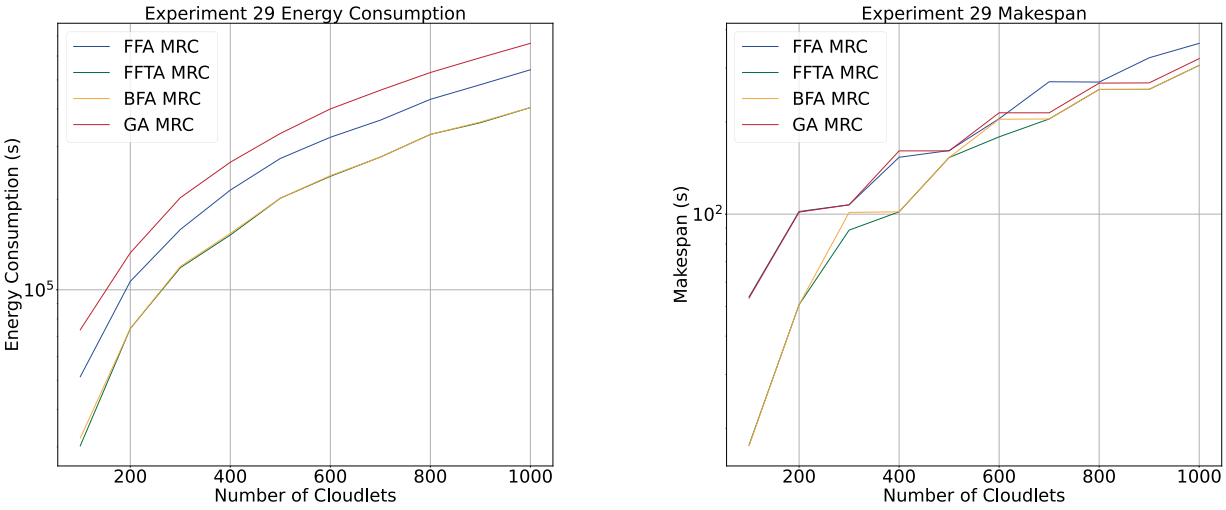

Figure 57. Experiment 29 FFTA weight configuration: MIPS=25%, CPU Cores=50%, TDP=25% (a) Energy Consumption (b) Makespan

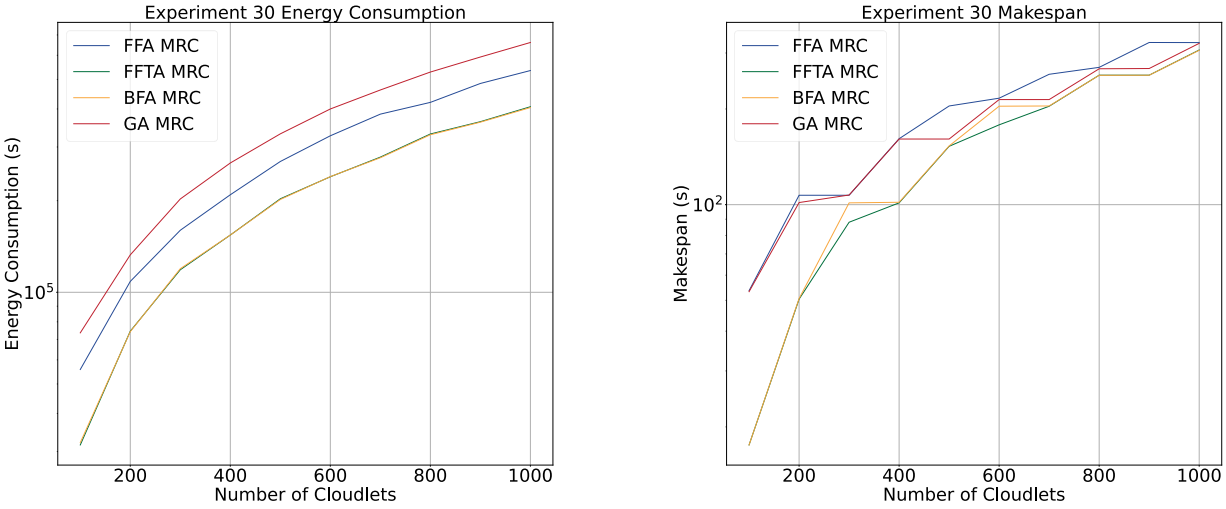

Figure 58. Experiment 30 FFTA weight configuration: MIPS=25%, CPU Cores=25%, TDP=50% (a) Energy Consumption (b) Makespan

Table 12 presents detailed results summarizing the experiments 16 through 30 for MRC. The results are summarized in Table 12.

Table 12. MRC Results for Experiments 16-30

Experiment #	TEC FFTA (kJ)	TM FFTA (s)	TEC FFA (kJ)	TM FFA (s)	TEC BFA (kJ)	TM BFA (s)	TEC GA (kJ)	TM GA (s)
16	2225.94	1761.62	2986.39	1975.54	2193.45	1649.23	3649.98	1869.44
17	2208.28	1609.87	2990.72	1993.29	2193.45	1649.23	3649.98	1869.44
18	2196.20	1609.98	2980.27	2017.59	2193.45	1649.23	3649.98	1869.44
19	2123.99	1398.80	2981.30	1988.38	2193.45	1649.23	3649.98	1869.44
20	2191.96	1623.94	2976.82	2030.63	2193.45	1649.23	3649.98	1869.44
21	2193.18	1623.94	2964.88	2018.35	2193.45	1649.23	3649.98	1869.44
22	2197.88	1623.94	2983.89	1993.36	2193.45	1649.23	3649.98	1869.44
23	2208.63	1622.69	2926.16	1980.43	2193.45	1649.23	3649.98	1869.44
24	2057.41	724.69	2984.38	1980.31	2193.45	1649.23	3649.98	1869.44
25	2072.09	1358.84	2988.01	2043.72	2193.45	1649.23	3649.98	1869.44
26	2185.38	1611.43	2949.72	1980.47	2193.45	1649.23	3649.98	1869.44
27	2185.13	1611.87	2971.23	1966.26	2193.45	1649.23	3649.98	1869.44
28	2185.13	1611.87	2971.23	1966.26	2193.45	1649.23	3649.98	1869.44
29	2185.38	1611.43	2946.10	2007.00	2193.45	1649.23	3649.98	1869.44
30	2200.29	1609.87	2953.50	2024.24	2193.45	1649.23	3649.98	1869.44

5.2.1.1 Energy Consumption Results

Table 13 presents a comparison of the energy consumption performance of FFTA with FFA, BFA, and GA in an MRC environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates an optimization in terms of energy consumption. As the goal of TETRA is to optimize the

energy consumption compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 13. MRC Energy Consumption Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
16	-25.46	+1.48	-39.01
17	-26.16	+0.67	-39.49
18	-26.30	+0.12	-39.82
19	-28.75	-3.16	-41.8
20	-26.06	-0.06	-39.94
21	-26.02	-0.01	-39.91
22	-26.34	-0.20	-39.78
23	-24.52	+0.69	-39.48
24	-31.06	-6.20	-43.63
25	-30.65	-5.53	-43.23
26	-25.91	-0.36	-40.12
27	-26.45	-0.37	-40.13
28	-26.40	-0.30	-40.13
29	-25.82	- 0.30	-40.12
30	-25.50	+ 0.31	-39.71

5.2.1.2 Makespan Results

Table 14 presents a comparison of the makespan performance of FFTA with FFA, BFA, and GA in an MRC environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates an optimization in terms of the total makespan. As the goal of TETRA is to optimize makespan compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 14. MRC Makespan Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
16	-10.82	+ 6.81	- 5.76
17	-19.23	-2.38	-13.88
18	-20.20	-2.38	-13.88
19	-29.65	-15.18	-25.17
20	-20.02	-1.53	-13.13
21	-19.54	-1.53	-13.13
22	-18.53	-1.53	-13.13
23	-18.06	-1.60	-13.19
24	-63.40	-56.05	- 61.23
25	-33.51	-17.60	-27.31
26	-18.63	-2.29	-13.80
27	-18.02	-2.26	-13.77
28	-18.02	-2.26	-13.77
29	-19.70	-2.29	-13.80
30	-19.78	-2.38	-13.88

5.3 High Resource Competition (HRC)

In this section, we present the results from experiments in which the resource competition is its highest level in terms of the number of available VM (resources) and the total number of tasks require execution. The high resource competition (HRC) setting is twice the competition rate of that of MRC and quadruple that of the LRC. The HRC can be considered as a scenario in which we test TETRA under high stress as the competition amongst resources is the highest. For HRC, we set the number of VMs to 153 hosted on 73 hosts. The post-filtration VM count is 91. The number of cloudlets ranges from 100 to 1000 in increments of 100. Thus, the ratio of cloudlets to VMs is ~11:1. Experiment 31-45 represent ones that utilize a higher competition rate compared to those of LRC and MRC.

5.3.2 High Resource Competition Results

Results in Figures 59-58 represent the experiments 31 through 45 which vary in terms of the weight configuration ranging between 0 to 100 for each of the considered attributes including MIPS, CPU cores, and TDP. Each figure presents energy consumption and makespan results across varying number of cloudlets (tasks) along with a comparison among four specific types of algorithms including First Fit Algorithm (FFA), First Fit with TOPSIS Algorithm (FFTA), Best Fit Algorithm (BFA), and Greedy Algorithm (GA).

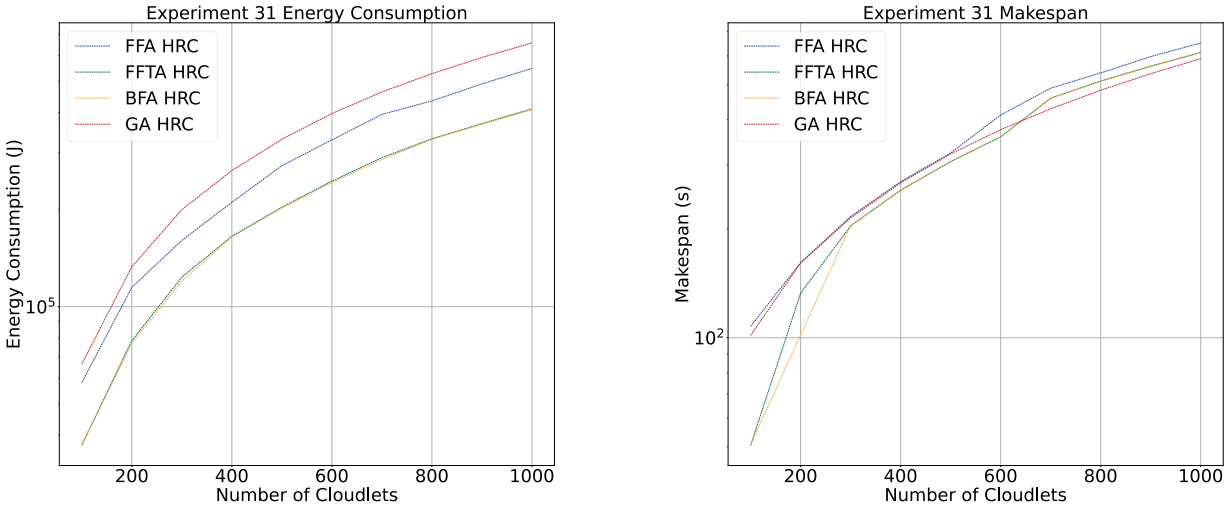

Figure 59. Experiment 31 FFTA weight configuration: MIPS=0%, CPU Cores=0%, TDP=100% (a) Energy Consumption (b) Makespan

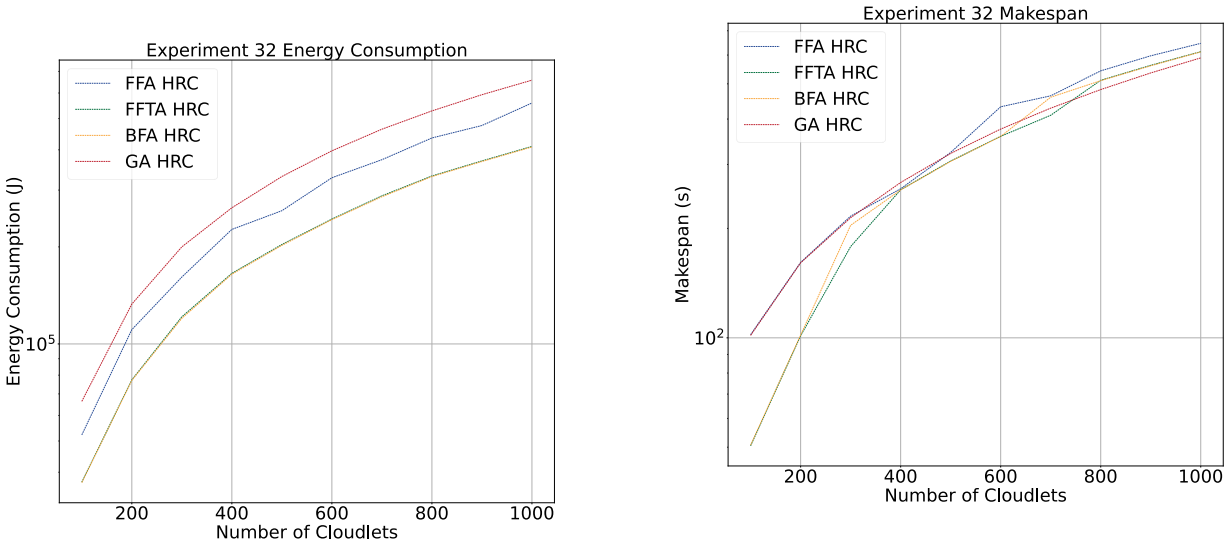

Figure 60. Experiment 32 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

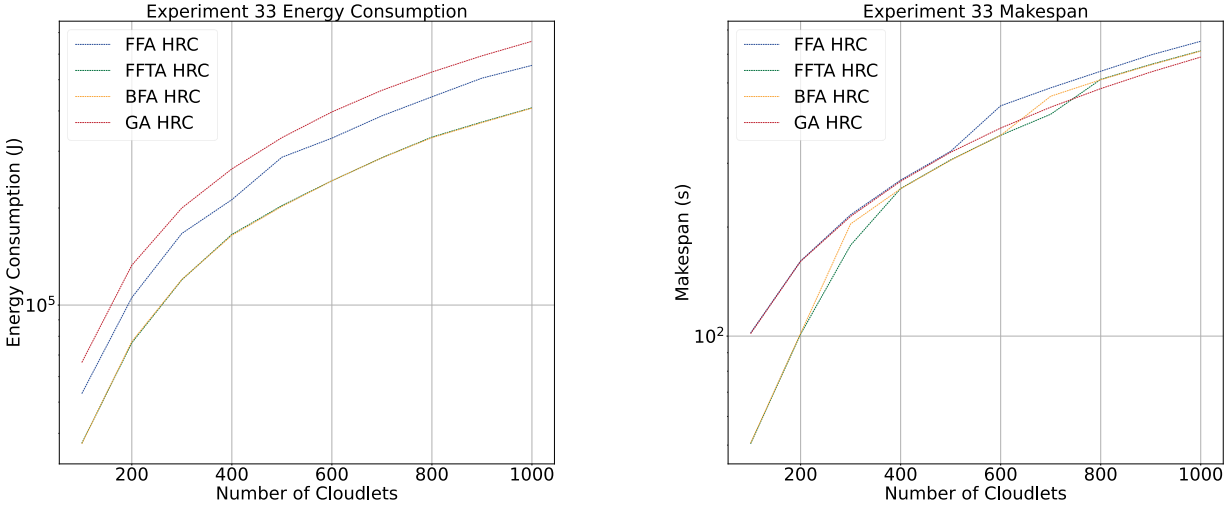

Figure 61. Experiment 33 FFTA weight configuration: MIPS=50%, CPU Cores=0%, TDP=50% (a) Energy Consumption (b) Makespan

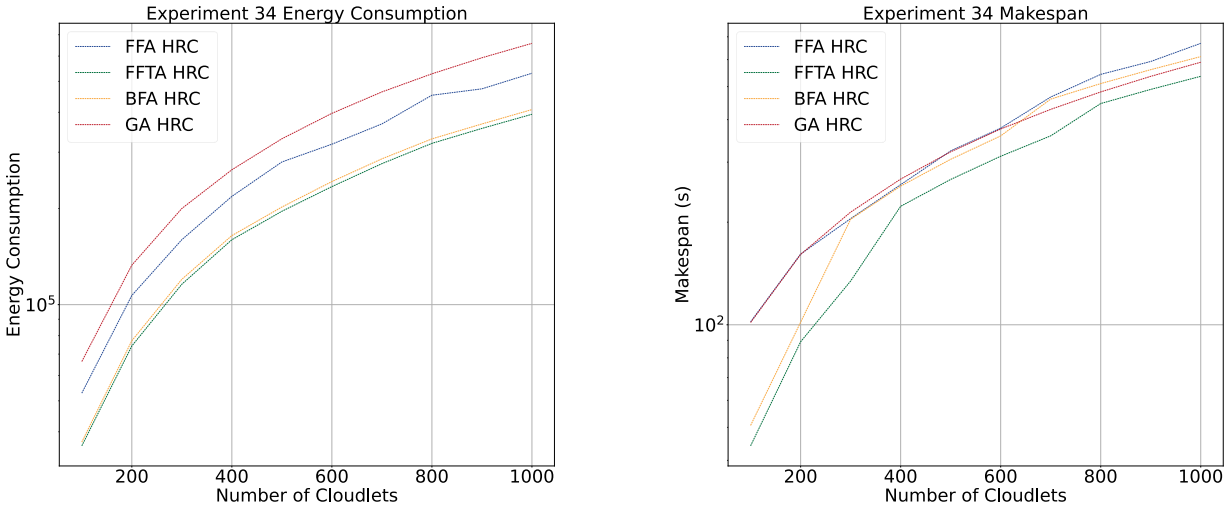

Figure 62. Experiment 34 FFTA weight configuration: MIPS=80%, CPU Cores=0%, TDP=20% (a) Energy Consumption (b) Makespan

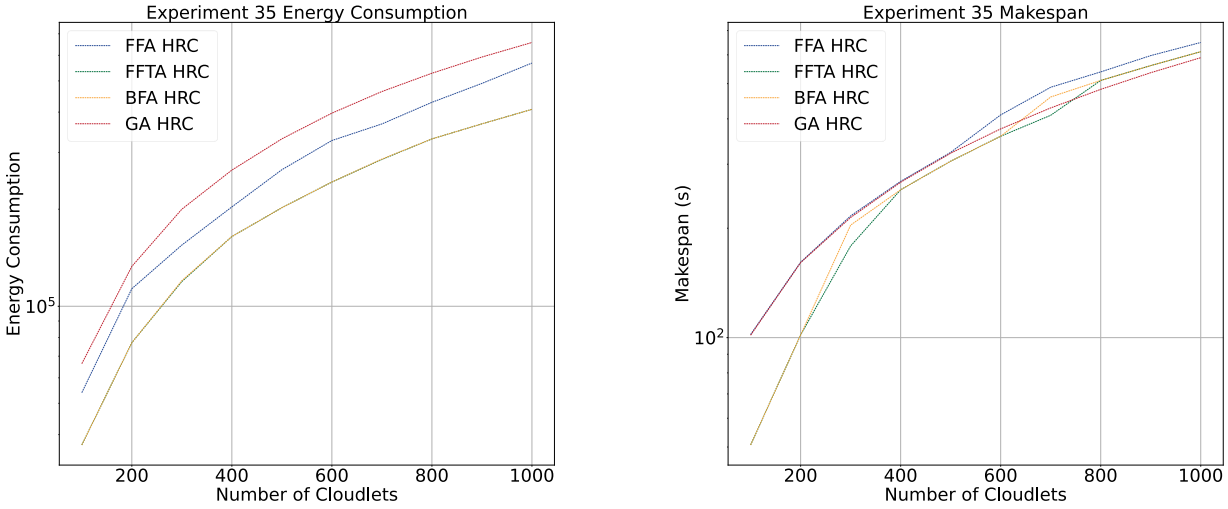

Figure 63. Experiment 35 FFTA weight configuration: MIPS=0%, CPU Cores=100%, TDP=0% (a) Energy Consumption (b) Makespan

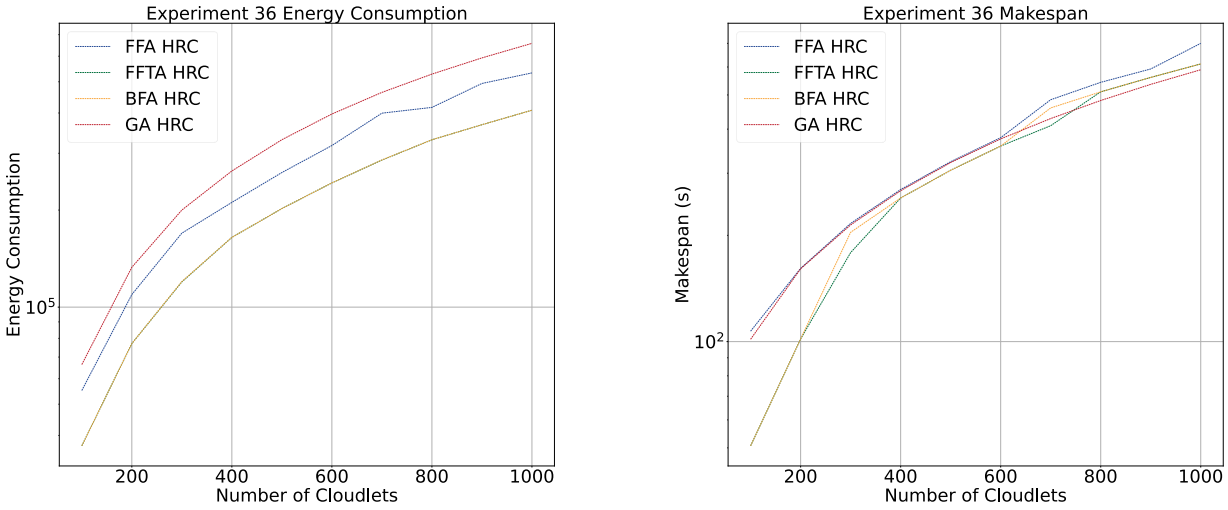

Figure 64. Experiment 36 FFTA weight configuration: MIPS=0%, CPU Cores=80%, TDP=20% (a) Energy Consumption (b) Makespan

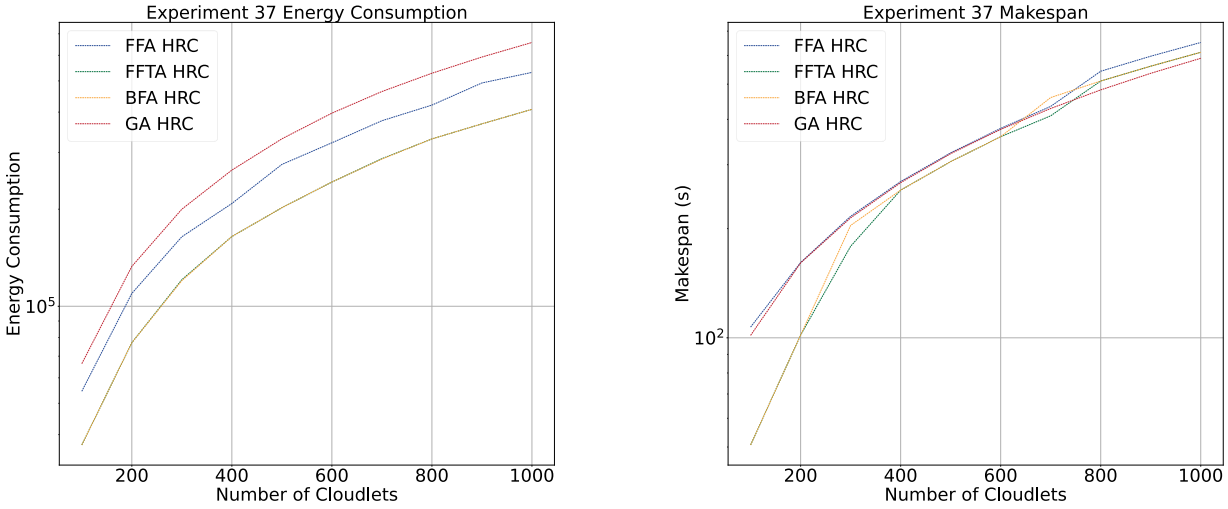

Figure 65. Experiment 37 FFTA weight configuration: MIPS=0%, CPU Cores=50%, TDP=50% (a) Energy Consumption (b) Makespan

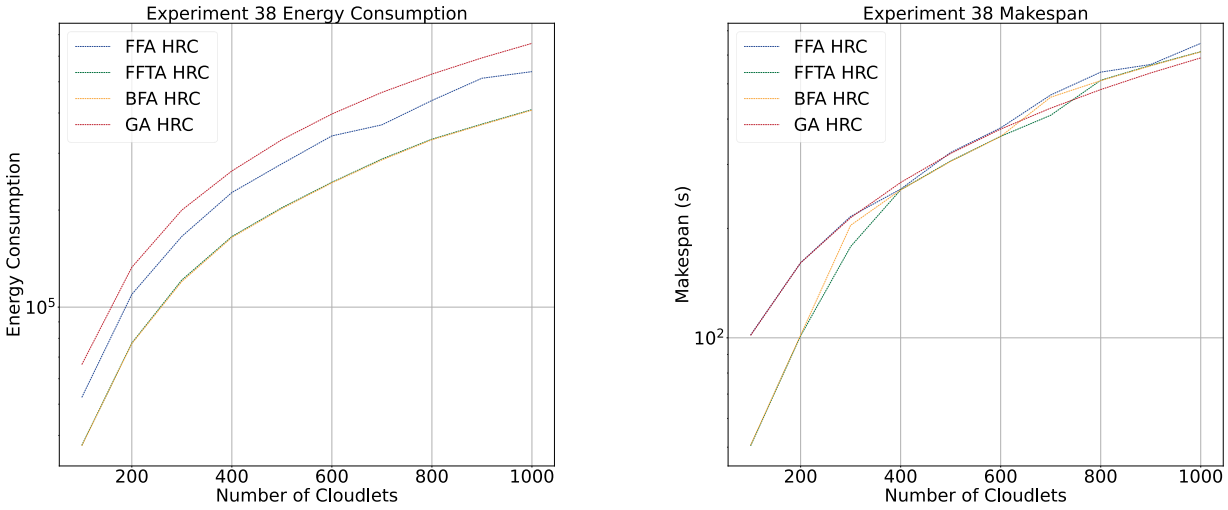

Figure 66. Experiment 38 FFTA weight configuration: MIPS=0%, CPU Cores=20%, TDP=80% (a) Energy Consumption (b) Makespan

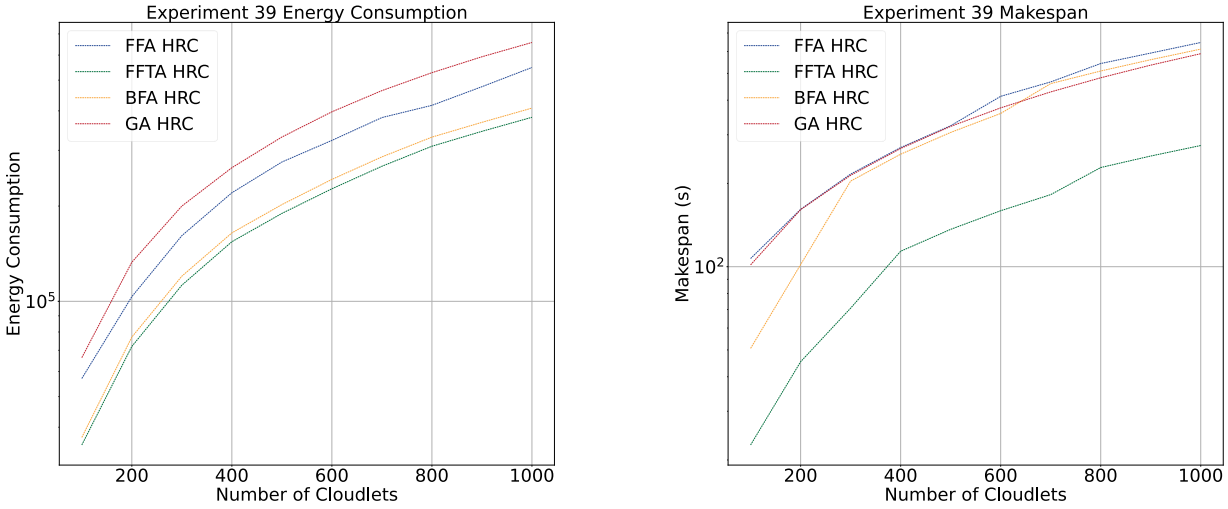

Figure 67. Experiment 39 FFTA weight configuration: MIPS=100%, CPU Cores=0%, TDP=0% (a) Energy Consumption (b) Makespan

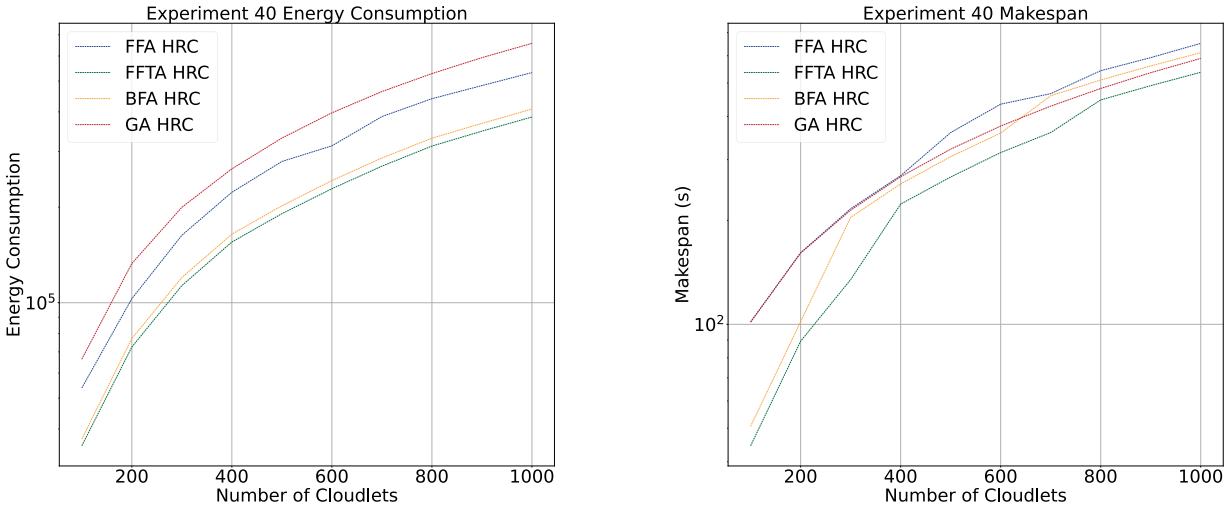

Figure 68. Experiment 40 FFTA weight configuration: MIPS=80%, CPU Cores=20%, TDP=0% (a) Energy Consumption (b) Makespan

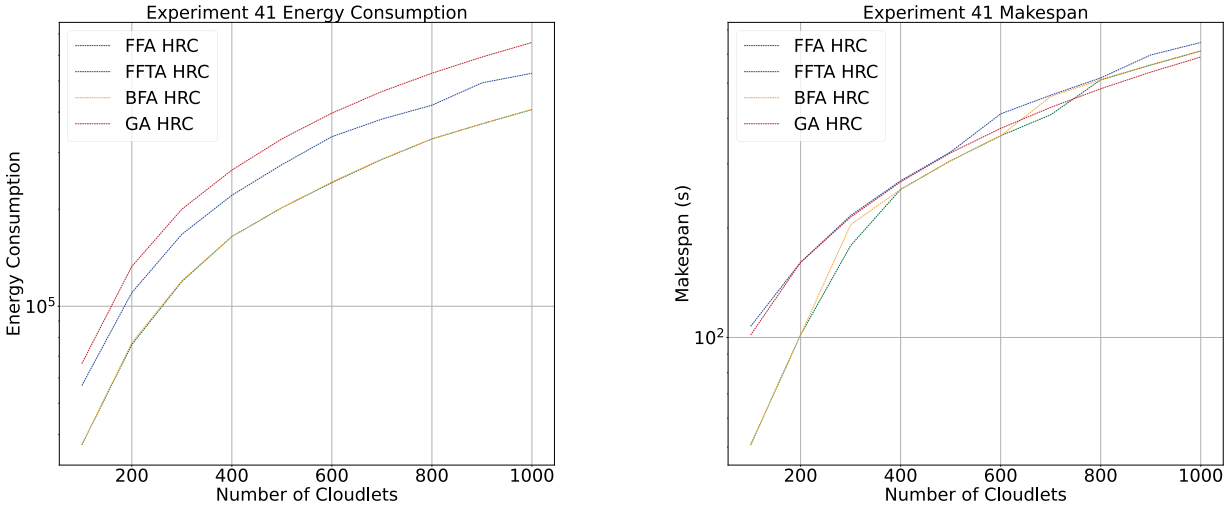

Figure 69. Experiment 41 FFTA weight configuration: MIPS=50%, CPU Cores=50%, TDP=0% (a) Energy Consumption (b) Makespan

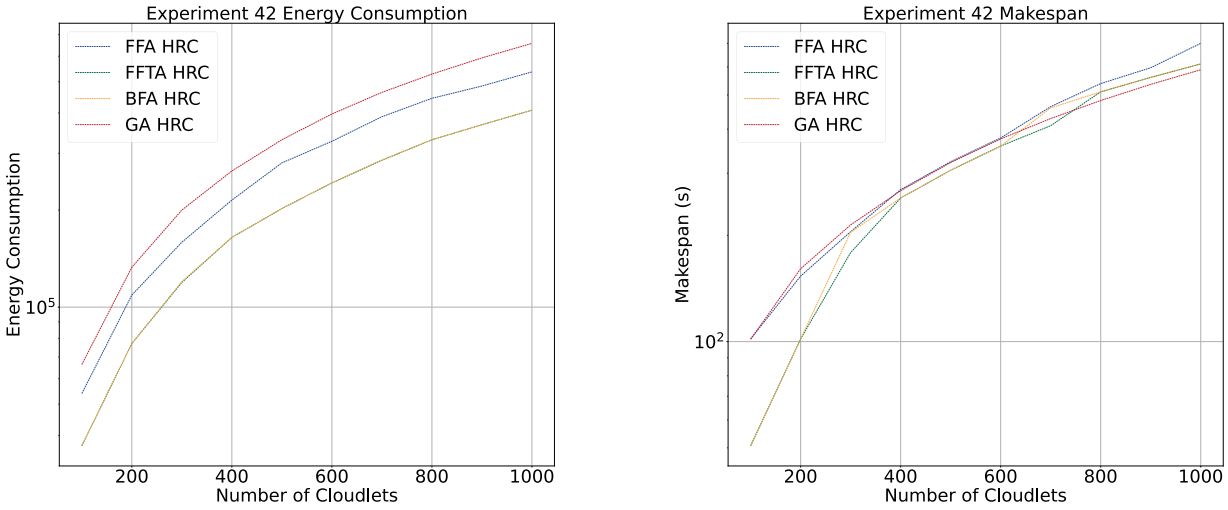

Figure 70. Experiment 42 FFTA weight configuration: MIPS=20%, CPU Cores=80%, TDP=0% (a) Energy Consumption (b) Makespan

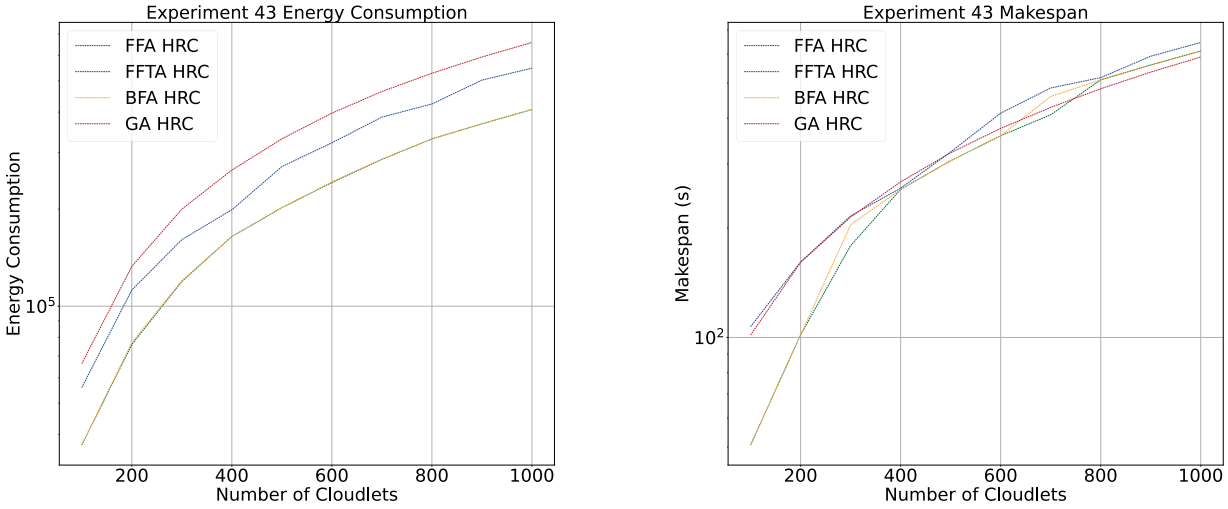

Figure 71. Experiment 43 FFTA weight configuration: MIPS=50%, CPU Cores=25%, TDP=25% (a) Energy Consumption (b) Makespan

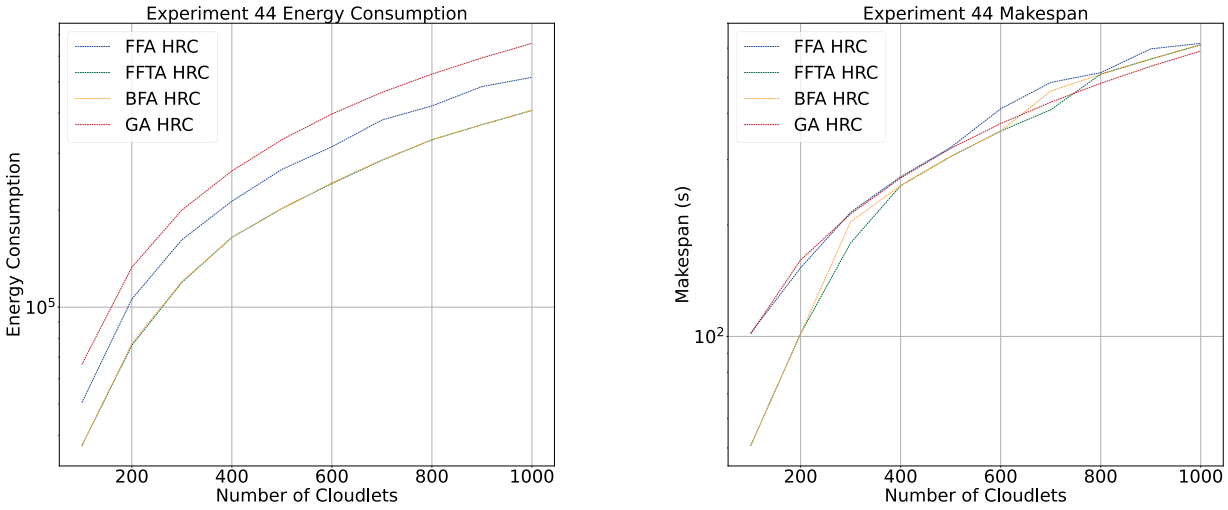

Figure 72. Experiment 44 FFTA weight configuration: MIPS=25%, CPU Cores=50%, TDP=25% (a) Energy Consumption (b) Makespan

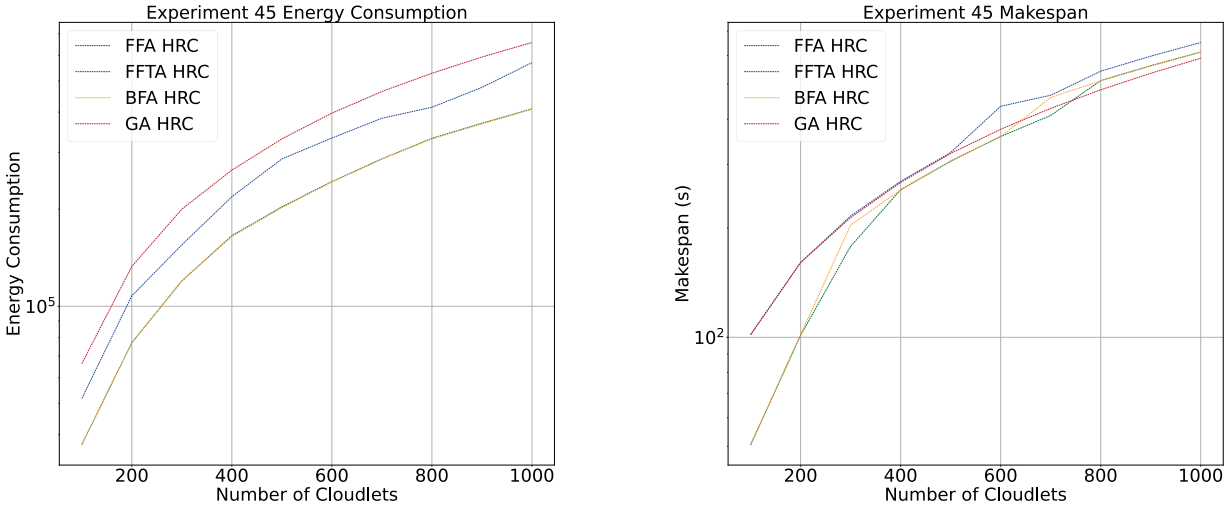

Figure 73. Experiment 45 FFTA weight configuration: MIPS=25%, CPU Cores=25%, TDP=50% (a) Energy Consumption (b) Makespan

Table 15 presents detailed results summarizing the experiments 31 through 45 for HRC. The results are summarized in Table 15.

Table 15. HRC Results for Experiments 31-45

Experiment #	TEC FFTA (kJ)	TM FFTA (s)	TEC FFA (kJ)	TM FFA (s)	TEC BFA (kJ)	TM BFA (s)	TEC GA (kJ)	TM GA (s)
31	2255.31	3452.80	3015.75	3766.33	2236.11	3418.03	3633.30	3477.22
32	2247.87	3347.10	2979.66	3742.52	2236.11	3418.03	3633.30	3477.22
33	2242.14	3347.54	3042.01	3776.96	2236.11	3418.03	3633.30	3477.22
34	2162.80	2900.14	2959.86	3697.84	2236.11	3418.03	3633.30	3477.22
35	2234.62	3342.45	2972.64	3755.43	2236.11	3418.03	3633.30	3477.22
36	2235.42	3342.45	2966.15	3776.03	2236.11	3418.03	3633.30	3477.22
37	2237.74	3342.45	2953.96	3681.19	2236.11	3418.03	3633.30	3477.22
38	2247.87	3347.10	3027.33	3652.93	2236.11	3418.03	3633.30	3477.22
39	2092.72	1484.62	2962.41	3738.95	2236.11	3418.03	3633.30	3477.22
40	2110.72	2907.96	2976.95	3794.75	2236.11	3418.03	3633.30	3477.22
41	2231.97	3342.89	2986.83	3711.74	2236.11	3418.03	3633.30	3477.22
42	2234.62	3342.45	2998.67	3729.73	2236.11	3418.03	3633.30	3477.22
43	2231.88	3342.89	2980.33	3718.72	2236.11	3418.03	3633.30	3477.22
44	2231.97	3342.89	2912.98	3690.96	2236.11	3418.03	3633.30	3477.22
45	2243.94	3347.10	2995.67	3763.94	2236.11	3418.03	3633.30	3477.22

5.3.2.1 Energy Consumption Results

Table 16 presents a comparison of the energy consumption performance of FFTA with FFA, BFA, and GA in a HRC environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates a saving in energy consumption. As we aim to reduce energy consumption compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 16. HRC Energy Consumption Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
31	-25.21	+0.85	-37.92
32	-24.55	+0.52	-38.13
33	-26.29	+0.52	-38.28
34	-26.92	-3.27	-40.47
35	-24.82	-0.06	-38.49
36	-24.63	-0.03	-38.47
37	-24.55	+0.07	-38.41
38	-25.74	+0.52	-38.13
39	-29.35	-6.41	-42.40
40	-29.09	-5.60	-41.90
41	-25.27	-0.18	-38.56
42	-25.47	-0.06	-38.49
43	-25.11	-0.18	-38.57
44	-23.37	-0.18	-38.56
45	-25.093	+0.35	-38.23

5.3.2.2 Makespan Results

Table 17 presents a comparison of the makespan performance of FFTA with FFA, BFA, and GA in a HRC environment. The comparison is in terms of the percentage of performance improvement. A negative value indicates a saving in makespan. As we aim to reduce makespan compared to the baseline algorithms, the negative values are favorable outcomes and are color-coded green in the table. In cases where FFTA performs worse, the values are color-coded red.

Table 17. HRC Makespan Results Comparison

Experiment #	PI FFA (%)	PI BFA (%)	PI GA (%)
31	-25.21	-0.52	-0.70
32	-10.56	-2.07	-3.74
33	-11.36	-2.06	-3.72
34	-21.57	-15.15	-16.59
35	-10.99	-2.211	-3.87
36	-11.48	-2.21	-3.87
37	-9.20	-2.21	-3.87
38	-8.37	-2.07	-3.74
39	-60.29	-56.56	-57.30
40	-23.36	-14.92	-16.37
41	-10.37	-2.19	-3.86
42	-10.38	-2.21	-3.87
43	-10.10	-2.19	-3.86
44	-9.43	-2.19	-3.86
45	-11.07	-2.07	-3.74

5.4 Results Summary

We compare the optimization performance of FFTA in terms of the competition rate. To this extent, we group the results and analyze them based on two main categories, energy consumption and makespan. Tables 18 and 19 present the overall aggregation of the individual results presented in earlier sections relative to FFTA.

Table 18. Energy Consumption Results Summary

	FFA		BFA		GA	
	PI (%)	Savings (kJ)	PI (%)	Savings (kJ)	PI (%)	Savings (kJ)
LRC	-28.94	12,860	-0.88	281	-43.05	23,889
MRC	-26.76	11,937	-0.89	284	-40.42	22,132
HRC	-25.70	11,489	-0.87	300	-39.00	21,257
	Avg.	Total	Avg.	Total	Avg.	Total
	-27.1	**36,286**	**-0.88**	**865**	**-40.82**	**67,278**

As can be seen in Table 18, FFTA outperforms in terms of optimizing the energy consumption across all of the three experiments' settings including LRC, MRC and HRC resulting in a total optimization of 36286 kJ, 865 kJ, and 67278 kJ, respectively.

Table 19. Makespan Results Summary

	FFA		BFA		GA	
	PI (%)	Savings (s)	PI (%)	Savings (s)	PI (%)	Savings (s)
LRC	-34.68	5841	-7.11	842	-31.17	4982
MRC	-23.14	6951	-6.96	1724	-17.92	5027
HRC	-16.25	8466	-7.15	3738	-8.8681	4626
	Avg.	Total	Avg.	Total	Avg.	Total
	-24.69	**21258**	**-7.15**	**6304**	**-19.32**	**14635**

As can be seen in Table 19, FFTA outperforms in terms of optimizing the total makespan across all of the three experiments' settings including LRC, MRC and HRC resulting in a total optimization of 21258 seconds, 6304 seconds, and 14635 seconds, respectively.

As shown in Tables 18 and 19, our proposed algorithm effectively optimizes both energy consumption and makespan in various competition environments under various competition levels. In addition, our proposed algorithm achieves an effective optimization rate while under various weight configurations as presented

earlier in Table 13, 14, 16 and 17. Further, results from our experiments demonstrate that our algorithm achieves best performance when the TOPSIS algorithm optimizes for resources with high MIPS, while not completely ignoring CPU cores, as can be evidenced by Experiments 9, 24, 39, 10, 25 and 40. Therefore, MCDA methods are effective in scheduling tasks for execution within fog environments.

In addition, one interesting insight is that the results from the experiments also show that the energy consumption performance suffers relatively when the TOPSIS algorithm optimizes for resources with low TDP and relatively higher MIPS, as can be seen in Experiments 1, 16, 31 and 2, 17 and 32. While this might seem counter-intuitive at first glance, a possible explanation could be that resources with low TDP are typically resources that are less powerful, this results in a longer time to execute tasks. This could potentially lead to higher energy consumption.

Chapter 5: Conclusion and Future Work

5.1 Contributions and Findings of TETRA

The results in the previous chapter prove our hypothesis Multi-Criteria Decision Analysis methods such as TOPSIS are highly effective at scheduling tasks within data center environments as our proposed algorithm successfully optimizes both energy consumption and makespan in a simulated data center environment. These results can be translated to a real-world environment as our proposed algorithm has proven to be scalable as is evidenced by the consistent performance when the competition for resources increases. Also, to ensure real-world performance, we created a data center with hosts and VMs that resemble the specifications of real-world hosts and VMs. This enables data centers to perform well while reducing their carbon emission levels. This would help large companies with their sustainability goals. In addition, through our research we have identified the key parameters that must be considered by the TOPSIS algorithm to effectively rank the resources. Based on the above findings, TETRA provides a very promising foundation for further exploration of hybrid resource scheduling in IoT systems.

5.2 Future Work

There is room for improvement in TETRA. Currently, TETRA accounts for fog and cloud levels implicitly by creating a huge diversity of resources that resemble resources that exist in these levels respectively. We could improve this model by explicitly specifying the level of each resource and using that as a parameter in the TOPSIS based ranking of resources. This would have huge implications for IoT systems. It would make resource scheduling even smarter. For instance, a time sensitive task would give more preference to a fog resource whereas a computationally complex task would give more preference to a cloud resource. This adds a layer of sophistication to TETRA. We can also offer customization of tasks such allowing the user to specify the type of application associated with it, in addition to having tasks with varying sizes. We would also like to explore the applicability of other alternate MCDA algorithms such as PROMETHEE, VIKOR and ELECTRE [107]. Currently, our model only allows manual configurations of weights. We would like to automate this process by using algorithms such as Entropy and CRITIC [108-109].

References

[1] “What is IoT? - Internet of Things Explained - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/iot/

[2] “Societal benefits of the IoT.” https://www.arduino.cc/education/societal-benefits-of-the-iot

[3] S. Sinha, “State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally,” IoT Analytics, Jan. 26, 2024. https://iot-analytics.com/number-connected-iot-devices/

[4] “World population projections - Worldometer.” https://www.worldometers.info/world-population/world-population-projections/

[5] K. Matrouk and K. Alatoun, “Scheduling Algorithms in Fog Computing: A Survey,” *International Journal of Networked and Distributed Computing*, vol. 9, no. 1, p. 59, 2021, doi: 10.2991/ijndc.k.210111.001.

[6] H. Mohamed, E. Al-Masri, O. Kotevska, and A. Souri, “A Multi-Objective Approach for Optimizing Edge-Based Resource Allocation Using TOPSIS,” *Electronics*, vol. 11, no. 18, p. 2888, Sep. 2022, doi: https://doi.org/10.3390/electronics11182888.

[7] D. Bermbach, D. Pallas, D. Pérez, P. Plebani, M. Anderson, R. Kat, and S. Tai, "A research perspective on fog computing," In Service-Oriented Computing–ICSOC 2017 Workshops: ASOCA, ISyCC, Revised Selected Papers (pp. 198-210). Springer, 2018.

[8] J. Minnix, “105 Data center stats you should know in 2024,” Brightlio - Technology Iluminated, Jan. 19, 2024. https://brightlio.com/data-center-stats/ H. Atlam, R. Walters, and G. Wills, “Fog Computing and the Internet of Things: A Review,” Big Data and Cognitive Computing, vol. 2, no. 2, p. 10, Apr. 2018, doi: https://doi.org/10.3390/bdcc2020010.

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

[10] Y. Zhang, X. Dong and Y. Zhao, "Decentralized Computation Offloading over Wireless-Powered Mobile-Edge Computing Networks," 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, 2020, pp. 137-140, doi: 10.1109/ICAIIS49377.2020.9194840.

[11] J. Zhang and X. Zhao, "An Overview of User-Oriented Computation Offloading in Mobile Edge Computing," 2020 IEEE World Congress on Services (SERVICES), Beijing, China, 2020, pp. 75-76, doi: 10.1109/SERVICES48979.2020.00029.

[12] Z. Wu and D. Yan, "Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network," in China Communications, vol. 18, no. 11, pp. 26-41, Nov. 2021, doi: 10.23919/JCC.2021.11.003.

[13] G. Guo and L. Yang, "Hotelling Model based Computation Offloading in Multi-access Edge Computing Assisted Mobile Network," 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 2020, pp. 697-700, doi: 10.1109/CVIDL51233.2020.00049.

[14] D. Gupta, A. Moudgil, S. Wadhwa and V. Solanki, "Efficient Data Caching and Computation Offloading Strategy for Edge Network," 2022 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2022, pp. 1-5, doi: 10.1109/ESCI53509.2022.9758379.

[15] J. Xia, P. Wang, B. Li and Z. Fei, "Intelligent task offloading and collaborative computation in multi-UAV-enabled mobile edge computing," in China Communications, vol. 19, no. 4, pp. 244-256, April 2022, doi: 10.23919/JCC.2022.04.018.

[16] M. Pan and Z. Li, "Multi-user Computation Offloading Algorithm for Mobile Edge Computing," 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT), Sanya, China, 2021, pp. 771-776, doi: 10.1109/CECIT53797.2021.00140.

[17] Y. Wang, H. Ge, A. Feng, W. Li, L. Liu and H. Jiang, "Computation Offloading Strategy Based on Deep Reinforcement Learning in Cloud-Assisted Mobile Edge Computing," 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 2020, pp. 108-113, doi: 10.1109/ICCCBDA49378.2020.9095689.

[18] M. Gao et al., "Computation Offloading With Instantaneous Load Billing for Mobile Edge Computing," in IEEE Transactions on Services Computing, vol. 15, no. 3, pp. 1473-1485, 1 May-June 2022, doi: 10.1109/TSC.2020.2996764.

[19] Q. Li, J. Zhao, Y. Gong and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for Internet of Everything," in China Communications, vol. 16, no. 3, pp. 32-41, March 2019, doi: 10.12676/j.cc.2019.03.004.

[20] Qinglin Yang, Xiaofei Luo, Peng Li, Toshiaki Miyazaki, and Xiaoyan Wang. 2019. Computation offloading for fast CNN inference in edge computing. In Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS '19). Association for Computing Machinery, New York, NY, USA, 101–106. https://doi-org.offcampus.lib.washington.edu/10.1145/3338840.3355669

[21] Lei Huang, Yifeng Zheng, Jingmin Yang, Wenjie Zhang, Liwei Yang, and Chai Kiat Yeo. 2021. Distributed Algorithm for Computation Offloading in Mobile Edge Computing. In Proceedings of the 2021 9th International Conference on Communications and Broadband Networking (ICCBN '21). Association for Computing Machinery, New York, NY, USA, 129–133. https://doi-org.offcampus.lib.washington.edu/10.1145/3456415.3456435

[22] Shuang Yuan, Yanfang Fan, and Ying Cai. 2020. A Survey on Computation Offloading for Vehicular Edge Computing. In Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City (ICIT '19). Association for Computing Machinery, New York, NY, USA, 107–112. https://doi-org.offcampus.lib.washington.edu/10.1145/3377170.3377228

[23] Yuben Qu and Jiajia Liu. 2019. Computation offloading for mobile edge computing with accuracy guarantee. In Proceedings of the ACM Turing Celebration Conference - China (ACM TURC '19). Association for Computing Machinery, New York, NY, USA, Article 21, 1–5. https://doi-org.offcampus.lib.washington.edu/10.1145/3321408.3321582

[24] Slađana Jošilo and György Dán. 2020. Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing. IEEE/ACM Trans. Netw. 28, 2 (April 2020), 667–680. https://doi-org.offcampus.lib.washington.edu/10.1109/TNET.2020.2968209

[25] Shihao Shen, Yiwen Han, Xiaofei Wang, and Yan Wang. 2019. Computation Offloading with Multiple Agents in Edge-Computing–Supported IoT. ACM Trans. Sen. Netw. 16, 1, Article 8 (February 2020), 27 pages. https://doi.org/10.1145/3372025

[26] Young-Min Lee and Joon-Sung Yang. 2019. Computation offloading of acoustic model for client-edge-based speech-recognition: work-in-progress. In Proceedings of the International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion (CASES '19). Association for Computing Machinery, New York, NY, USA, Article 1, 1–2. https://doi-org.offcampus.lib.washington.edu/10.1145/3349569.3351534

[27] Hanine Tout, Azzam Mourad, Nadjia Kara, and Chamseddine Talhi. 2021. Multi-Persona Mobility: Joint Cost-Effective and Resource-Aware Mobile-Edge Computation Offloading. IEEE/ACM Trans. Netw. 29, 3 (June 2021), 1408–1421. https://doi-org.offcampus.lib.washington.edu/10.1109/TNET.2021.3066558

[28] Yiwen Han, Ding Li, Haotian Qi, Jianji Ren, and Xiaofei Wang. 2019. Federated learning-based computation offloading optimization in edge computing-supported internet of things. In Proceedings of the ACM Turing Celebration Conference - China (ACM TURC '19). Association for Computing Machinery, New York, NY, USA, Article 25, 1–5. https://doi-org.offcampus.lib.washington.edu/10.1145/3321408.3321586

[29] Bai, Y.; Li, X.; Wu, X.; Zhou, Z. Dynamic Computation Offloading with Deep Reinforcement Learning in Edge Network. Appl. Sci. 2023, 13, 2010. https://doi.org/10.3390/app13032010\

[30] Huang, Y.-Y.; Wang, P.-C. Computation Offloading and User-Clustering Game in Multi-Channel Cellular Networks for Mobile Edge Computing. Sensors 2023, 23, 1155. https://doi.org/10.3390/s23031155\

[31] Wei, D.; Wang, R.; Xia, C.; Xia, T.; Jin, X.; Xu, C. Edge Computing Offloading Method Based on Deep Reinforcement Learning for Gas Pipeline Leak Detection. Mathematics 2022, 10, 4812. https://doi.org/10.3390/math10244812

[32] Qi, H.; Zhou, Z. Computation Offloading and Trajectory Control for UAV-Assisted Edge Computing Using Deep Reinforcement Learning. Appl. Sci. 2022, 12, 12870. https://doi.org/10.3390/app122412870\

[33] Wu, J.; Jia, M.; Zhang, L.; Guo, Q. DNNs Based Computation Offloading for LEO Satellite Edge Computing. *Electronics* **2022**, *11*, 4108. https://doi.org/10.3390/electronics11244108

[34] Wang, Y.; Wang, J.; Ke, H.; Sun, Z. COPP-DDPG: Computation Offloading with Privacy Preservation in a Vehicular Edge Network. Appl. Sci. 2022, 12, 12522. https://doi.org/10.3390/app122412522

[35] Liu, L.; Zhao, Y.; Qi, F.; Zhou, F.; Xie, W.; He, H.; Zheng, H. Federated Deep Reinforcement Learning for Joint AeBSs Deployment and Computation Offloading in Aerial Edge Computing Network. Electronics 2022, 11, 3641. https://doi.org/10.3390/electronics11213641

[36] Wang, S.; Peng, H.; Guo, D. Resource- and Time-Efficient Computation Offloading in Vehicular Edge Computing: A Max-Min Fairness Oriented Approach. Mathematics 2022, 10, 3735. https://doi.org/10.3390/math10203735

[37] Wei, Z.; Yu, X.; Zou, L. Multi-Resource Computing Offload Strategy for Energy Consumption Optimization in Mobile Edge Computing. Processes 2022, 10, 1762. https://doi.org/10.3390/pr10091762

[38] Chen, J.; Chang, Z.; Guo, W.; Guo, X. Resource Allocation and Computation Offloading for Wireless Powered Mobile Edge Computing. Sensors 2022, 22, 6002. https://doi.org/10.3390/s22166002

[39] H. Badri, T. Bahreini, D. Grosu and K. Yang, "Risk-Based Optimization of Resource Provisioning in Mobile Edge Computing," 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 2018, pp. 328-330, doi: 10.1109/SEC.2018.00033.

[40] S. Dai, L. Hai, Y. Li and Z. Zhang, "An Incentive Auction-based Cooperative Resource Provisioning Scheme for Edge Computing over Passive Optical Networks," 2019 18th International Conference on Optical Communications and Networks (ICOCN), Huangshan, China, 2019, pp. 1-3, doi: 10.1109/ICOCN.2019.8933864.

[41] X. Duan, H. Lu and S. Li, "Elastic Service Provisioning for Mobile Edge Computing," 2021 IEEE 7th International Conference on Big Data Intelligence and Computing (DataCom), Huizhou, China, 2021, pp. 57-60, doi: 10.1109/DataCom53700.2021.00016.

[42] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin and X. Shen, "Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," in IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 968-980, 1 July-Sept. 2021, doi: 10.1109/TCC.2019.2903240.

[43] Y. Ma, W. Liang, M. Huang, W. Xu and S. Guo, "Virtual Network Function Service Provisioning in MEC Via Trading Off the Usages Between Computing and Communication Resources," in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2949-2963, 1 Oct.-Dec. 2022, doi: 10.1109/TCC.2020.3043313.

[44] J. Liu, S. Guo, K. Liu and L. Feng, "Resource Provision and Allocation Based on Microeconomic Theory in Mobile Edge Computing," in IEEE Transactions on Services Computing, vol. 15, no. 3, pp. 1512-1525, 1 May-June 2022, doi: 10.1109/TSC.2020.3000050.

[45] S. Hu, W. Shi and G. Li, "CEC: A Containerized Edge Computing Framework for Dynamic Resource Provisioning," in IEEE Transactions on Mobile Computing, vol. 22, no. 7, pp. 3840-3854, 1 July 2023, doi: 10.1109/TMC.2022.3147800.

[46] A. Abouaomar, S. Cherkaoui, Z. Mlika and A. Kobbane, "Resource Provisioning in Edge Computing for Latency-Sensitive Applications," in IEEE Internet of Things Journal, vol. 8, no. 14, pp. 11088-11099, 15 July15, 2021, doi: 10.1109/JIOT.2021.3052082.

[47] L. Li et al., "Data-Driven Optimization for Cooperative Edge Service Provisioning With Demand Uncertainty," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4317-4328, 15 March15, 2021, doi: 10.1109/JIOT.2020.3028242.

[48] Y. Nakamura, T. Mizumoto, H. Suwa, Y. Arakawa, H. Yamaguchi and K. Yasumoto, "In-Situ Resource Provisioning with Adaptive Scale-out for Regional IoT Services," 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 2018, pp. 203-213, doi: 10.1109/SEC.2018.00022.

[49] Ruozhou Yu, Guoliang Xue, Yinxin Wan, Jian Tang, Dejun Yang, and Yusheng Ji. 2020. Robust resource provisioning in time-varying edge networks. In Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20). Association for Computing Machinery, New York, NY, USA, 21–30. https://doi-org.offcampus.lib.washington.edu/10.1145/3397166.3409146

[50] Klervie Toczé, Ali J. Fahs, Guillaume Pierre, and Simin Nadjm-Tehrani. 2023. VioLinn: Proximity-aware Edge Placementwith Dynamic and Elastic Resource Provisioning. ACM Trans. Internet Things 4, 1, Article 7 (February 2023), 31 pages. https://doi-org.offcampus.lib.washington.edu/10.1145/3573125

[51] Xiaofeng Cao, Guoming Tang, Deke Guo, Yan Li, and Weiming Zhang. 2020. Edge Federation: Towards an Integrated Service Provisioning Model. IEEE/ACM Trans. Netw. 28, 3 (June 2020), 1116–1129. https://doi-org.offcampus.lib.washington.edu/10.1109/TNET.2020.2979361

[52] Xiaofeng Cao, Guoming Tang, Deke Guo, Yan Li, and Weiming Zhang. 2020. Edge Federation: Towards an Integrated Service Provisioning Model. IEEE/ACM Trans. Netw. 28, 3 (June 2020), 1116–1129. https://doi-org.offcampus.lib.washington.edu/10.1109/TNET.2020.2979361

[53] Xiaofeng Cao, Guoming Tang, Deke Guo, Yan Li, and Weiming Zhang. 2020. Edge Federation: Towards an Integrated Service Provisioning Model. IEEE/ACM Trans. Netw. 28, 3 (June 2020), 1116–1129. https://doi-org.offcampus.lib.washington.edu/10.1109/TNET.2020.2979361

[54] Zhiyuan Wang, Jiancheng Ye, and John C.S. Lui. 2021. An Online Mean Field Approach for Hybrid Edge Server Provision. In Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21). Association for Computing Machinery, New York, NY, USA, 131–140. https://doi.org/10.1145/3466772.3467042

[55] Jinxi Li, Deke Guo, Junjie Xie, and Sheng Chen. 2023. Availability-aware Provision of Service Function Chains in Mobile Edge Computing. ACM Trans. Sen. Netw. 19, 3, Article 57 (August 2023), 28 pages. https://doi-org.offcampus.lib.washington.edu/10.1145/3565483

[56] Zhi Zhou, Xu Chen, Weigang Wu, Di Wu, and Junshan Zhang. 2019. Predictive Online Server Provisioning for Cost-Efficient IoT Data Streaming Across Collaborative Edges. In Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19). Association for Computing Machinery, New York, NY, USA, 321–330. https://doi-org.offcampus.lib.washington.edu/10.1145/3323679.3326530

[57] Dimitrios Spatharakis, Ioannis Dimolitsas, Eleftherios Vlahakis, Dimitrios Dechouniotis, Nikolaos Athanasopoulos, and Symeon Papavassiliou. 2023. Distributed Resource Autoscaling in Kubernetes Edge Clusters. In Proceedings of the 18th International Conference on Network and Service Management (CNSM '22). International Federation for Information Processing, Laxenburg, AUT, Article 1, 1–7.

[58] Fang, J.; Chen, Y.; Lu, S. Energy-Efficient Resource Provisioning Strategy for Reduced Power Consumption in Edge Computing. Appl. Sci. 2020, 10, 6057. https://doi.org/10.3390/app10176057

[59] J. X. Liao and X. W. Wu, "Resource Allocation and Task Scheduling Scheme in Priority-Based Hierarchical Edge Computing System," 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Xuzhou, China, 2020, pp. 46-49, doi: 10.1109/DCABES50732.2020.00021.

[60] S. Zhang, G. Cui, Y. Long and W. Wang, "Joint computing and communication resource allocation for satellite communication networks with edge computing," in China Communications, vol. 18, no. 7, pp. 236-252, July 2021, doi: 10.23919/JCC.2021.07.019.

[61] Y. Tang, "Minimizing Energy for Caching Resource Allocation in Information-Centric Networking with Mobile Edge Computing," 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Fukuoka, Japan, 2019, pp. 301-304, doi: 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00062.

[62] S. Ma, S. Guo, K. Wang, W. Jia and M. Guo, "A Cyclic Game for Service-Oriented Resource Allocation in Edge Computing," in IEEE Transactions on Services Computing, vol. 13, no. 4, pp. 723-734, 1 July-Aug. 2020, doi: 10.1109/TSC.2020.2966196.

[63] F. D. Tilahun, A. T. Abebe and C. G. Kang, "Delay-aware Joint Resource Allocation in Cell-Free Mobile Edge Computing," 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Korea, Republic of, 2022, pp. 81-82, doi: 10.1109/APCC55198.2022.9943562.

[64] X. Liu, J. Yu, Z. Feng and Y. Gao, "Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing," in China Communications, vol. 17, no. 9, pp. 220-236, Sept. 2020, doi: 10.23919/JCC.2020.09.017.

[65] J. Liu, S. Guo, K. Liu and L. Feng, "Resource Provision and Allocation Based on Microeconomic Theory in Mobile Edge Computing," in IEEE Transactions on Services Computing, vol. 15, no. 3, pp. 1512-1525, 1 May-June 2022, doi: 10.1109/TSC.2020.3000050.

[66] X. Sun, J. Jia, Z. Liu, Y. Li, B. Sun and D. Liu, "Resource Allocation and Load Balancing Based on Edge Computing in Industrial Networks," 2022 IEEE International Conference on Smart Internet of Things (SmartIoT), Suzhou, China, 2022, pp. 250-251, doi: 10.1109/SmartIoT55134.2022.00048.

[67] P. Wang, C. Yao, Z. Zheng, G. Sun and L. Song, "Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2872-2884, April 2019, doi: 10.1109/JIOT.2018.2876198.

[68] Senyu Yu, Yan Guo, Ning Li, Duan Xue, and Cuntao Liu. 2023. Dynamic resource allocation on Vehicular edge computing and communication. In Proceedings of the 8th International Conference on Communication and Information Processing (ICCIP '22). Association for Computing Machinery, New York, NY, USA, 205–211. https://doi-org.offcampus.lib.washington.edu/10.1145/3571662.3571696

[69] Abdukodir Khakimov, Aleksandr Loborchuk, Ibodulaev Ibodullokhodzha, Dmitry Poluektov, Ibrahim A. Elgendy, and Ammar Muthanna. 2021. Edge Computing Resource Allocation Orchestration System for Autonomous Vehicles. In Proceedings of the 4th International Conference on Future Networks and Distributed Systems (ICFNDS '20). Association for Computing Machinery, New York, NY, USA, Article 3, 1–7. https://doi.org/10.1145/3440749.3442594

[70] Jiaqi Wang, Zac Lu, and Eyhab Al-Masri. 2020. Edgify: Resource Allocation Optimization for Edge Clouds Using Stable Matching. In Companion Proceedings of the Web Conference 2020 (WWW '20). Association for Computing Machinery, New York, NY, USA, 128–130. https://doi.org/10.1145/3366424.3382732

[71] Xutao Yang, Xuejie Zhang, Weidong Li, and Jixian Zhang. 2020. A Truthful Auction Mechanism for Cumulative Resource Allocation in Mobile Edge Computing. In Proceedings of the 2020 4th International Conference on High Performance Compilation, Computing and Communications (HP3C 2020). Association for Computing Machinery, New York, NY, USA, 63–69. https://doi.org/10.1145/3407947.3407976

[72] Tiao Tan, Ming Zhao, and Zhiwen Zeng. 2022. Joint Offloading and Resource Allocation Based on UAV-Assisted Mobile Edge Computing. ACM Trans. Sen. Netw. 18, 3, Article 36 (August 2022), 21 pages. https://doi.org/10.1145/3476512

[73] Erfan Meskar and Ben Liang. 2020. Fair multi-resource allocation in mobile edge computing with multiple access points. In Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20). Association for Computing Machinery, New York, NY, USA, 11–20. https://doi.org/10.1145/3397166.3409144

[74] Yuhu Sun, Qiang He, Lianyong Qi, Wajid Rafique, and Wanchun Dou. 2020. DPODA: Differential Privacy-based Online Double Auction for Pervasive Edge Computing Resource Allocation. In Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI '20). Association for Computing Machinery, New York, NY, USA, 130–141. https://doi.org/10.1145/3384943.3409429

[75] Zhaowu Huang, Fang Dong, Dian Shen, Huitian Wang, Xiaolin Guo, and Shucun Fu. 2023. Enabling Latency-Sensitive DNN Inference via Joint Optimization of Model Surgery and Resource Allocation in Heterogeneous Edge. In Proceedings of the 51st International Conference on Parallel Processing (ICPP '22). Association for Computing Machinery, New York, NY, USA, Article 71, 1–11. https://doi.org/10.1145/3545008.3545071

[76] Guifu Ma, Haoran Li, Xiaowei Wang, Xiaolong Chen, Yougang Bian, Manjiang Hu, Xuepeng Wang, and Jin Zhang. 2022. Mobility-aware Task Splitting and Computation Resource Allocation for Distributed Multi-access Edge Computing Enabled Vehicular Network. In 2021 International Conference on Mechanical, Aerospace and Automotive Engineering (CMAAE 2021). Association for Computing Machinery, New York, NY, USA, 164–170. https://doi.org/10.1145/3518781.3519206

[77] Ashish Pandey, Prasad Calyam, Saptarshi Debroy, Songjie Wang, and Mauro Lemus Alarcon. 2021. VECTrust: trusted resource allocation in volunteer edge-cloud computing workflows. In Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing (UCC '21). Association for Computing Machinery, New York, NY, USA, Article 19, 1–10. https://doi.org/10.1145/3468737.3494099

[78] Cui, T.; Yang, R.; Fang, C.; Yu, S. Deep Reinforcement Learning-Based Resource Allocation for Content Distribution in IoT-Edge-Cloud Computing Environments. Symmetry 2023, 15, 217. https://doi.org/10.3390/sym15010217

[79] Dong, L.; He, W.; Yao, H. Task Offloading and Resource Allocation for Tasks with Varied Requirements in Mobile Edge Computing Networks. Electronics 2023, 12, 366. https://doi.org/10.3390/electronics12020366

[80] Tong, M.; Li, S.; Wang, X.; Wei, P. Inter-Satellite Cooperative Offloading Decision and Resource Allocation in Mobile Edge Computing-Enabled Satellite–Terrestrial Networks. Sensors 2023, 23, 668. https://doi.org/10.3390/s23020668

[81] Qiu, S.; Zhao, J.; Lv, Y.; Dai, J.; Chen, F.; Wang, Y.; Li, A. Digital-Twin-Assisted Edge-Computing Resource Allocation Based on the Whale Optimization Algorithm. Sensors 2022, 22, 9546. https://doi.org/10.3390/s22239546

[82] Cao, C.; Su, M.; Duan, S.; Dai, M.; Li, J.; Li, Y. QoS-Aware Joint Task Scheduling and Resource Allocation in Vehicular Edge Computing. Sensors 2022, 22, 9340. https://doi.org/10.3390/s22239340

[83] Ali, Z.; Qureshi, K.N.; Mustafa, K.; Bukhsh, R.; Aslam, S.; Mujlid, H.; Ghafoor, K.Z. Edge Based Priority-Aware Dynamic Resource Allocation for Internet of Things Networks. Entropy 2022, 24, 1607. https://doi.org/10.3390/e24111607

[84] Fang, C.; Zhang, T.; Huang, J.; Xu, H.; Hu, Z.; Yang, Y.; Wang, Z.; Zhou, Z.; Luo, X. A DRL-Driven Intelligent Optimization Strategy for Resource Allocation in Cloud-Edge-End Cooperation Environments. Symmetry 2022, 14, 2120. https://doi.org/10.3390/sym14102120

[85] Mahmood, O.A.; Abdellah, A.R.; Muthanna, A.; Koucheryavy, A. Distributed Edge Computing for Resource Allocation in Smart Cities Based on the IoT. Information 2022, 13, 328. https://doi.org/10.3390/info13070328

[86] Chen, J.; Chang, Z.; Guo, W.; Guo, X. Resource Allocation and Computation Offloading for Wireless Powered Mobile Edge Computing. Sensors 2022, 22, 6002. https://doi.org/10.3390/s22166002

[87] Ke, H.; Wang, H.; Sun, H. Multi-Agent Deep Reinforcement Learning-Based Partial Task Offloading and Resource Allocation in Edge Computing Environment. Electronics 2022, 11, 2394. https://doi.org/10.3390/electronics11152394

[88] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun and P. Mudali, "Multi-Objective Optimization Approach for Task Scheduling in Fog Computing," *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, Winterton, South Africa, 2019, pp. 1-6, doi: 10.1109/ICABCD.2019.8851038.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[89] 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," in IEEE Communications Letters, vol. 24, no. 2, pp. 307-311, Feb. 2020, doi: 10.1109/LCOMM.2019.2957741.

[90] J. Wang and D. Li, "Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing," *MDPI*, Feb. 28, 2019. https://www.mdpi.com/1424-8220/19/5/1023.

[91] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, Nov. 2019, doi: 10.1002/cpe.5581.

[92] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, Oct. 2019, doi: 10.1002/ett.3770

[93] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," Journal of Network and Computer Applications, vol. 201, p. 103333, May 2022, doi: https://doi.org/10.1016/j.jnca.2022.103333.

[94] M. T. Islam, S. Karunasekera, and R. Buyya, "Performance and Cost-Efficient Spark Job Scheduling Based on Deep Reinforcement Learning in Cloud Computing Environments," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 7, pp. 1695–1710, Jul. 2022, doi: https://doi.org/10.1109/tpds.2021.3124670.

[95] A. Jayanetti, S. Halgamuge, and R. Buyya, "Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge-cloud computing environments," Future Generation Computer Systems, Jun. 2022, doi: https://doi.org/10.1016/j.future.2022.06.012.

[96] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing," *Future Generation Computer Systems*, vol. 111, pp. 539–551, Oct. 2020, doi: https://doi.org/10.1016/j.future.2019.09.039.

[97] S. Sharma and H. Saini, "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment," *Sustainable Computing: Informatics and Systems*, vol. 24, p. 100355, Dec. 2019, doi: https://doi.org/10.1016/j.suscom.2019.100355.

[98] T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, "Cost-Aware Task Scheduling in Fog-Cloud Environment," *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, Jun. 2020, doi: https://doi.org/10.1109/rtest49666.2020.9140118.

[99] H. Tan, W. Chen, L. Qin, J. Zhu, and H. Huang, "Energy-aware and Deadline-constrained Task Scheduling in Fog Computing Systems," *2020 15th International Conference on Computer Science & Education (ICCSE)*, Aug. 2020, doi: https://doi.org/10.1109/iccse49874.2020.9201710.

[100] R. Mo, F. Dai, Q. Liu, W. Dou, and X. Xu, "Multi-objective Cross-layer Resource Scheduling for Internet of Things in Edge-Cloud Computing," *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, Oct. 2020, doi: https://doi.org/10.1109/cloud49709.2020.00055.

[101] K. P. N. Jayasena and B. S. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, Dec. 2019, doi: https://doi.org/10.1109/smartcloud.2019.0001

[102] S. Ghanavati, J. H. Abawajy, and D. Izadi, "An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment," *IEEE Transactions on Services Computing*, pp. 1–1, 2020, doi: https://doi.org/10.1109/tsc.2020.3028575.

[103] Wikipedia Contributors, "Multiple-criteria decision analysis," Wikipedia, Feb. 15, 2019. https://en.wikipedia.org/wiki/Multiplecriteria_decision_analysis

[104] Y. Zhu, D. Tian, and Y. Feng, "Effectiveness of entropy weight method in Decision-Making," Mathematical Problems in Engineering, vol. 2020, pp. 1–5, Mar. 2020, doi: 10.1155/2020/3564835.

[105] GfG, "What is MIPS(Million of Instructions Per Second)?," GeeksforGeeks, Jul. 07, 2022. https://www.geeksforgeeks.org/what-is-mipsmillion-of-instructions-per-second/

[106] "About TDP(Thermal Design Power) - GIGABYTE Global." https://www.gigabyte.com/Glossary/tdp

[107] W. Sałabun, J. Wątróbski, and A. Shekhovtsov, "Are MCDA Methods Benchmarkable? A Comparative Study of TOPSIS, VIKOR, COPRAS, and PROMETHEE II Methods," *Symmetry*, vol. 12, no. 9, p. 1549, Sep. 2020, doi: https://doi.org/10.3390/sym12091549.

[108] Y. Zhu, D. Tian, and Y. Feng, "Effectiveness of entropy weight method in Decision-Making," Mathematical Problems in Engineering, vol. 2020, pp. 1–5, Mar. 2020, doi: 10.1155/2020/3564835.

[109] Krishnan AR, Kasim MM, Hamid R, Ghazali MF. A Modified CRITIC Method to Estimate the Objective Weights of Decision Criteria. Symmetry. 2021; 13(6):973. https://doi.org/10.3390/sym13060973