DEVELOPMENT OF A MODEL FOR OPTIMIZED RESOURCE MANAGEMENT IN MULTI CLOUD ENVIRONMENT

Thesis Submitted for the Award of the Degree of

DOCTOR OF PHILOSOPHY

in

Computer Applications

By
Ramanpreet Kaur

Registration Number: 41800765

Supervised By
Dr. Divya (24844)
Computer Science and Engineering
(Assistant Professor)
Lovely Professional University

Co-Supervised by
Dr. Upinder Kaur
Computer Science and Engineering
(Assistant Professor)
Akal University

Co-Supervised by
Prof.(Dr.) Sahil Verma
Chandigarh University
Mohali, India

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2024

Declaration

I, hereby declared that the presented work in the thesis entitled "DEVELOPMENT OF A MODEL FOR OPTIMIZED RESOURCE MANAGEMENT IN MULTI CLOUD ENVIRONMENT" in fulfilment of degree of **Doctor of Philosophy (Ph. D.)** is outcome of research work carried out by me under the supervision Dr. Divya, working as Assistant Professor, in School of Computer Science and Engineering of Lovely Professional University, Punjab, India. In keeping with general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of other investigator. This work has not been submitted in part or full to any other University or Institute for the award of any degree.

Ramanpreet Kaur

(Signature of Scholar)

Name of the scholar: Ramanpreet Kaur

Registration No.: 41800765

Department/school: Computer Applications

Lovely Professional University,

Punjab, India

Certificate

This is to certify that the work reported in the Ph. D. thesis entitled "DEVELOPMENT OF A MODEL FOR OPTIMIZED RESOURCE MANAGEMENT IN MULTI CLOUD ENVIRONMENT" submitted in fulfillment of the requirement for the reward of degree of **Doctor of Philosophy (Ph.D.)** in the computer application, is a research work carried out by Ramanpreet Kaur, 41800765, is bonafide record of his/her original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma or equivalent course.

(Signature of Supervisor)
Name of supervisor: Dr. Divya
Designation: Assistant Professor
Department/school: Computer Science and Engineering.
University: Lovely Professional University

(Signature of Co-Supervisor)
Name of supervisor: Dr. Upinder Kaur
Designation: Assistant Professor
Department/school: Computer Science and Engineering
University: Akal University

(Signature of Co-Supervisor)
Name of Co-Supervisor
Prof. (Dr.) Sahil Verma
University: Chandigarh University
Mohali, India

Abstract

Cloud computing has transformed the IT industry by giving organisations instant access to a pool of computer resources, allowing them to efficiently extend their applications. However, as cloud data centres expand, optimising several objectives to reconcile competing performance indicators becomes crucial. This research provides a model that overcomes these difficulties by creating an intelligent scheduling system that is flexible to dynamic and diverse workloads and is specifically tailored for practical applications in cloud computing. Effective task scheduling becomes even more important in multi-datacenter scenarios where cloud services cover multiple geographical zones. The suggested solution recognises the difficulties caused by datacenter heterogeneity as well as the need to optimise resource allocation across many data centres. In the world of cloud computing, where massive volumes of data are processed and analysed, efficient task scheduling is critical to optimising resource utilisation and overall system efficiency. The Cyber Shake Seismogram (CSS) process stands out as a significant and demanding application among the different workflows that benefit from effective task scheduling. The Modified Best Fit Decreasing (MBFD) method is included into an upgraded MET framework, resulting in a location and resource-aware scheduling system.

Traditional work scheduling algorithms frequently concentrate on a particular goal, such as minimising job completion time, resulting in inefficient solutions. To address this shortcoming, the research proposes steps to improve the Flower Pollination Algorithm, transforming it into a multi-objective task scheduling tool. This method seeks to optimise numerous competing objectives at the same time, achieving a balance between diverse performance measures.

The multi-data centre architecture is being investigated as a significant development in meeting the increasing demands of modern applications and services. The study provides a toolset for creating and managing virtual machines (VMs) and physical hosts (PMs) in a virtualized cloud environment, as well as for simulating various

scenarios based on real-world cloud usage trends. The extensive results and evaluations show a diverse spectrum of real-world circumstances and issues. The results show that our integrated strategy greatly improves resource utilisation, effectively decreases, and overall improves the performance of the scheduling network. Our analysis goes deeply into the impact of important parameters on the performance of the scheduling system. Each characteristic, such as MIPS, makespan, used energy, CO2 emission, user distance, throughput, VM load distribution ratio, and projected cost, has a significant impact on the quality of the scheduling results. We obtain significant insights into the strengths and adaptability of our strategy by meticulously studying these elements, allowing us to fine-tune the scheduling system for optimal results.

The combination of Q-learning with flower pollination raises the bar in resource allocation and job scheduling. The combination of these advanced methodologies enables our solution to handle complicated and dynamic scheduling settings quickly, making it suited for a wide range of practical applications. The algorithm finds the most promising option by using Q-values to drive the pollination process, enhancing efficiency and efficacy in discovering optimal solutions. The performance analysis includes adjusting the number of users and increasing the workload while comparing the proposed method to five previous studies (MET, Hu et al., Cai et al., Jena et al. Saurabh et al.). At a workload of 100,000 MIPS, the proposed algorithm's performance in comparison to previous studies demonstrate a significant improvements across a variety of parameters. The proposed approach consistently delivers lower energy usage, with values ranging from 0.39987KJ to 0.716416KJ, representing a savings of up to 47% when compared to MET, Hu et al., Cai et al., Jena et al., and Saurabh et al. In terms of cost-effectiveness, the proposed algorithm has an average total cost range of Rs24.1422 to Rs238.069, showing possible savings of up to 20% over the previous methods. Furthermore, it has fewer CO2 emissions, ranging from 16.27168893Mt to 141.8717559Mt, representing up to a 35% improvement over existing techniques. The suggested approach maintains competitive complexity metrics, has fast training and inference times, and achieves high throughput rates ranging from 8276.669 to 8659.64 Mbps, showing better data transport and

processing capabilities. Task scheduling and completion are managed efficiently, with a competitive makespan range of 13.81982 to 14.17695 seconds, resulting in up to 12% improvements over competing methods. In summary, the suggested algorithm significantly improves energy efficiency, cost-effectiveness, and environmental impact while remaining competitive in complexity, throughput, and job completion time, making it a promising alternative for future implementations. The task scheduling solution for multi-cloud systems is proposed here, which incorporates flower pollination and Q-learning, stands out as a uniquely efficient strategy that outperforms current state-of-the-art algorithms. Extensive testing using simulations on various datasets simulating real-world scenarios consistently demonstrates the suggested method's higher performance. The algorithm's adaptability, scalability, and effectiveness in managing dynamic workloads and different user populations highlight its practical application in both cloud computing and multi-datacenter systems. The suggested solution's new features distinguish it as a possible improvement in solving the issues connected with task scheduling in complex and developing computing landscapes.

Acknowledgement

I would like to express my deep appreciation to my mentors, Dr. Divya, Dr. Upinder Kaur and Prof. (Dr.) Sahil Verma for their unwavering guidance, inspiration, and encouragement throughout my thesis journey. Without their invaluable support and constructive feedback, this research endeavour would not have been possible.

Furthermore, I extend my heartfelt gratitude to my academic family at BFGI (Baba Farid Group of Institutions, Bathinda) for their motivation and encouragement in pursuing my Ph.D. I owe a special debt of thanks to Mr. Jagmeet Singh, my husband, for his unwavering cooperation, moral support, and emotional encouragement at every step.

Finally, I wish to express my profound gratitude to my parents, in-laws, siblings, and all my family members who provided me with strength and unwavering spirit through the enduring memories. I am especially thankful to my sister, Gagandeep Kaur, who always believed in my capabilities, saying that my sister can achieve anything. Last, but not least, I am filled with gratitude for my adorable baby, Lehareen Kaur, who was born and grew up during my PhD journey.

Table of Contents

Declaration ..... **ii**

Certificate ..... **iii**

Abstract ..... **iv**

Acknowledgement ..... **vii**

Table of Contents ..... **viii**

List of Tables ..... **xiii**

List of Figures ..... **xiv**

CHAPTER 1: INTRODUCTION **1**

1.1 Introduction 1

1.2 Evolution of Cloud Computing 2

1.2.1 Parallel Computing 4

1.2.2 Grid Computing 5

1.2.3 Utility Computing 5

1.2.4 Virtualization 5

1.2.5 Centralized computing 6

1.3 Characteristics of Cloud Computing 6

1.3.1 On-demand Self-service 6

1.3.2 Broad Network Service 6

1.3.3 Resource Pools 7

1.3.4 Fast Flexibility 7

1.3.5 Measured Service 7

1.3.6 Significance of Resource Allocation 7

1.4 Task Scheduling.....8
1.4.1 Advantages of Task scheduling algorithms.....9
1.4.2 Task Scheduling in Cloud Computing.....9
1.5 Types of Cloud Computing.....10
1.5.1 Public Cloud.....11
1.5.2 Private Cloud.....13
1.5.3 Hybrid Cloud.....15
1.5.4 Community Cloud.....16
1.5.5 Multi Cloud.....17
1.6 Job Scheduling for Optimal Usage of Resources.....20
1.6.1 First-Come, First-Served (FCFS) Scheduling.....20
1.6.2 Shortest Job Next (SJN) Scheduling.....20
1.6.3 Round Robin (RR) Scheduling.....21
1.6.4 Priority-Based Scheduling.....21
1.6.5 Genetic Algorithm (GA) Scheduling.....21
1.6.6 Minimum Execution Time (MET).....21
1.7 Various parameters of Cloud computing.....22
1.7.1 Makespan.....22
1.7.2 Energy Consumption.....23
1.7.3 CO_2 Emission.....24
1.7.4 Execution Cost.....26
1.8 Machine Learning and its Evolution in Job Scheduling Under Cloud Computing.....26
1.8.1 Supervised.....29
1.8.2 Un-supervised.....29

1.8.3 Semi-supervised 31

1.9 Metaheuristic Algorithms 33

1.9.1 Characteristics of Metaheuristic 35

1.10 Organisation of the Thesis 35

CHAPTER 2: REVIEW OF LITERATURE 36

2.1 Literature Review 36

2.2 Review Summary 69

2.3 Research Gaps 72

2.4 Research Objectives 73

2.5 Summary of the Chapter 73

CHAPTER 3: ENHANCEMENT IN MET FOR CYBER SHAKE SEISMOGRAM 74

3.1 Cyber Shake Seismogram Work Flow 74

3.2 Minimum Execution Time and its Correlation with CSS Workflow 75

3.3 Proposed Work Algorithm for the Enhancement 79

3.4 Summary of the Chapter 83

CHAPTER 4: APPLICATION AND IMPROVISATION IN FLOWER-POLLINATION ALGORITHM FOR IMPROVED EFFICIENCY IN TASK SCHEDULING 84

4.1 Background 84

4.2 Scenario of Development and the Optimization Issue 85

4.2.1 The FPA Algorithm 86

4.2.2 Significance in Job Scheduling in Cloud Computing 88

4.2.3 Switching Probability 92

4.3 Proposed Work towards improving FPA using Q-learning 93

4.3.1 Multi-datacenter deployment, benefit and illustration 93
4.3.2 Integration of Q-learning 97
4.3.2.1 Q-learning Application in Proposed Work Case Scenario 102
4.3.3 Neural Networks and its Applicability 105
4.3.3.1 Integration of the Neural Network 106
4.4 Application over the real time multi-cloud 113
4.4.1 Application Initialization at AWS cloud 114
4.4.2 Execution of cloud server 116
4.5 Summary of the Chapter 123
CHAPTER 5: RESULTS AND DISCUSSION 125
5.1 Result Evaluation Based on Increasing Number of Users 129
5.2 Results Evaluation based on Increasing Load 136
5.2.1 Importance of Load Variation Analysis 136
5.2.2 Energy Consumption Analysis 138
5.2.2.1 Minimum and Maximum Value Scenarios 140
5.2.2.2 Percentage Improvement 140
5.2.2.3 Discussion of Improvement 142
5.2.3 Cost Analysis 143
5.2.3.1 Minimum and Maximum Scenario for Cost 144
5.2.3.2 Percentage Improvement in Cost 145
5.2.3.3 Discussion on Improvement in Cost 147
5.2.4 CO_2 Emission Analysis 147
5.2.4.1 Minimum and Maximum Value Scenarios 149
5.2.4.2 Percentage Improvement 149
5.2.4.3 Discussion of Improvement 151

5.2.5 Throughput Analysis.....153

5.2.5.1 Minimum maximum value scenarios for Throughput 154

5.2.5.2 Percentage Improvement 154

5.2.5.3 Discussion of Improvement 155

5.2.6 Makespan Analysis 156

5.2.6.1 Minimum maximum value scenarios for Makespan.....157

5.2.6.2 Percentage Improvement 158

5.2.6.3 Discussion of Improvement 159

5.2.7 Comparative Analysis.....160

5.3 Summary of the Chapter 166

CHAPTER 6: CONCLUSION.....168

6.1 Conclusion for Simulation Architecture 1 - Increasing Number of Users.....169

6.2 Conclusion for Simulation Architecture 2 - Increasing Load Amount.....169

6.3 Comparative Analysis.....170

6.4 Future Scope 174

List of Publications 175

Bibliography 176

List of Tables

Table 1.1 Difference between Supervised and Unsupervised ML Techniques 30
Table 1.2 Comparison of Machine Learning Algorithms 31
Table 2.1 Comparative Analysis of Existing Scheduling Approaches 59
Table 2.2 Comparative Analysis of Evaluation Parameters 70
Table 4.1 Attributes and Description 97
Table 4.2 System Configuration 109
Table 4.3 Performance Analysis of Machine Learning Techniques 112
Table 5.1 Evaluation Based on Number of Users 130
Table 5.2 Comparative Analysis Based on Energy Consumption 139
Table 5.3 Comparative Analysis based on Cost 143
Table 5.4 Comparative Analysis Based on CO_2 Emission.....147
Table 5.5 Comparative Analysis Based on Throughput 153
Table 5.6 Comparative Analysis Based on Makespan.....156
Table 5.7 Comparative Analysis 160

List of Figures

Figure 1.1 Cloud Computing Environment [3] 2
Figure 1.2 Developments in Cloud Computing [8] 3
Figure 1.3 Characteristics of Cloud Computing [18] 6
Figure 1.4 Task Scheduling Structure [105] 8
Figure 1.5 Task Scheduling System Model in Cloud Environments [106] 10
Figure 1.6 Cloud Computing Classification [107] 11
Figure 1.7 Public Cloud [12] 12
Figure 1.8 Private Cloud [12] 14
Figure 1.9 Comparison of Hybrid and Multi Cloud [12] 19
Figure 1.10 Machine Learning Approach [36] 29
Figure 1.11 Supervised and Unsupervised Approach [34] 30
Figure 4.1 The Flower Pollination Process [125] 86
Figure 4.2 Job Allocation Process in Proposed Work 88
Figure 4.3 Datacentre modelling in Cloudsim 96
Figure 4.4 Proposed Work Using Q-Learning [135] 100
Figure 4.5 Overall work architecture 101
Figure 4.6 The applied Q-learning algorithm 104
Figure 4.7 Implementation of Q-learning algorithm in the cloud-sim environment . 104
Figure 4.8 Running Instances on AWS Multicloud 116
Figure 4.9 AWS Instance Creation 117
Figure 4.10 AWS Dashboard with application and space 118
Figure 4.11 Execution of cloud instances for current algorithm 119
Figure 4.12 Result of execution on 100% completion 119
Figure 4.13 Data Migration between multiple clouds 121
Figure 4.14 Execution Outcome at server 1 based on loads 121
Figure 4.15 Read and Write Operations on server 2 for data storage 122
Figure 5.1 Energy vs Number of Users 131
Figure 5.2 Cost vs Number of Users 132

Figure 5.3 CO_2 vs Number of Users 132
Figure 5.4 Comparative Analysis Based on Energy Consumption 139
Figure 5.5 Energy Consumption Improvement 141
Figure 5.6 Comparative Analysis based on Cost 144
Figure 5.7 Cost Improvement 146
Figure 5.8 Comparative Analysis based on CO_2 Emission 148
Figure 5.9 CO_2 Emission Improvement 150
Figure 5.10 Realtime AWS based Multicloud Comparison with Local Servers 152
Figure 5.11 Comparative Analysis based on Throughput 154
Figure 5.12 Throughput Improvement 155
Figure 5.13 Comparative Analysis based on Makespan 157
Figure 5.14 Makespan Improvement 158
Figure 5.15 Comparative Analysis of Energy 160
Figure 5.16 Comparative Analysis of Cost 161
Figure 5.17 Comparative Analysis of CO_2 Emission 162
Figure 5.18 Complexity Analysis in Terms of Time 163
Figure 5.19 Comparative Analysis of Throughput 164
Figure 5.20 Comparative Analysis of Makespan 165

CHAPTER 1: INTRODUCTION

The chapter goes into a lot of detail about cloud computing, including its history, traits, and security issues brought on by how widely used it is. To build the groundwork for the current research project, the multilayered cloud computing structure and the idea of virtualization are also explained here.

1.1 Introduction

Within the field of computing, mobile computing in the cloud is acknowledged as a developing commercial concept. Cloud computing" refers to a system that offers quick, easy, and ubiquitous network access to a shared pool of flexible computing resources, including servers, network connections, storage, services, and applications. These resources can be quickly allocated and released with minimal administrative effort or service provider involvement [1, 2]. The entities responsible for furnishing these services are termed cloud providers, typically charging users based on their usage of cloud computing resources. CC encompasses a technological framework that employs principles of connectivity, virtualization, resource sharing, and data exchange among various devices over the internet. It delivers on-demand services to users by incorporating crucial elements like security, scalability, distribution, and isolation. [3].

When referring to online-based services and apps, such as distributed processing, machine virtualization, and secure web services, we use the term "cloud computing.". The challenges encompass a spectrum from handling diverse resources to efficiently allocating resources based on user requests, effectively scheduling planned requests to designated resources, and addressing potential contingencies associated with both the workload and the system [4]. Server virtualization technology plays a crucial role in streamlining cloud resource management, enabling optimized resource utilization through the sharing of resources via virtual machines

Figure 1.1 shows the architectural representation of CC. It entails offering computing services including storage, database management systems, networking, analytics, programmes, and more over the Internet or through cloud platforms.

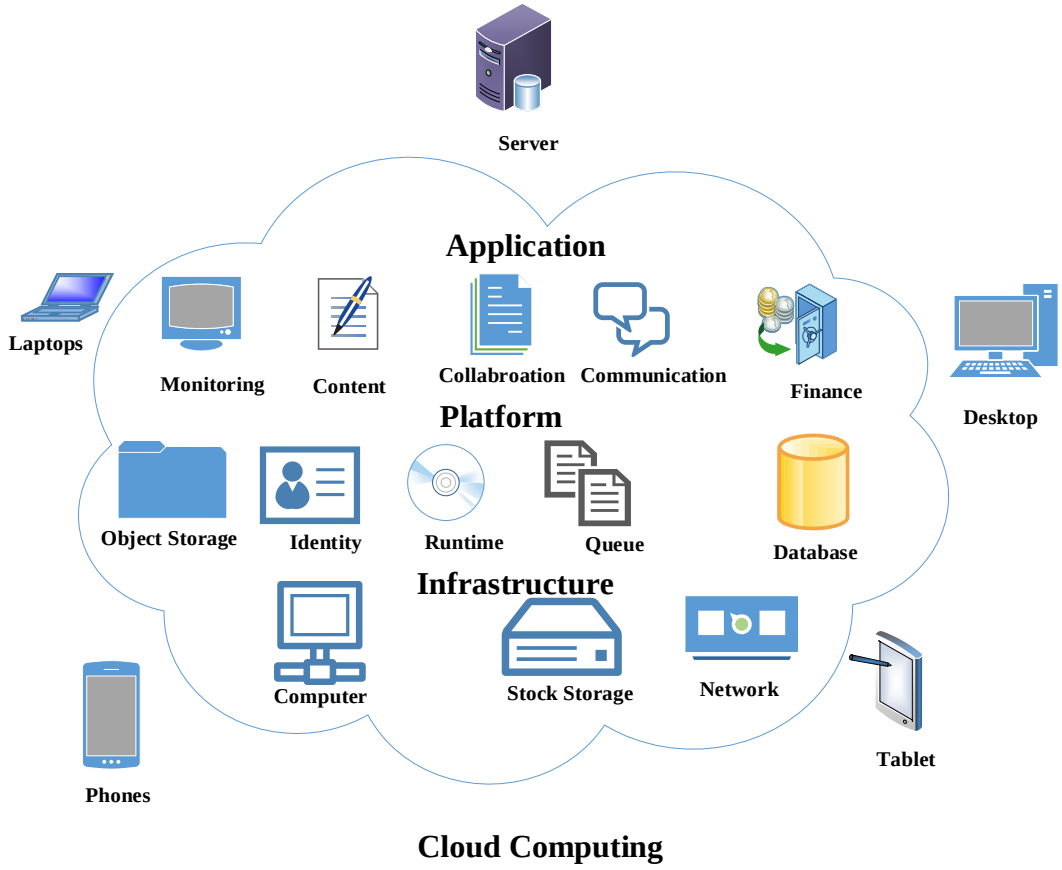

Figure 1.1 Cloud Computing Environment [3]

The fundamental structure of the CC environment is illustrated in Figure 1.1. [4]. NIST defines "cloud computing" as a methodology that permits smooth, practical, and immediate network connectivity to share computing resource pools. Quick allocation and release of assets is possible with minimum involvement from administration contacts [5].

1.2 Evolution of Cloud Computing

The concept of CC has its roots dating back to the 1960s. During this era, John McCarthy envisioned the future use of 'national utilities' for computer computing. Although Cloud Computing appears as a relatively recent phenomenon, its origins

trace back to the early 1950s when mainframes enabled multiple users to access a central computer. In the 1960s, several developments emerged that foreshadowed the essence of present-day cloud computing, such as J. K. R. Liklider's concept of the "intergalactic computer network." The philosophy of cloud computing gained momentum in 2007, driven by the expansion of communication channels and the exponentially increasing demand from both business and private users to horizontally scale their information systems [1, 6].

Virtualization brought the mainframe to a new level during the 1970s, while telecommunication companies started offering connectivity for virtual private networks (VPNs) in the 1990s. In 1999, Salesforce.com pioneered the delivery of enterprise applications through network connections. These applications could be accessed by multiple users simultaneously through a web browser, and they were available at an affordable cost[7].

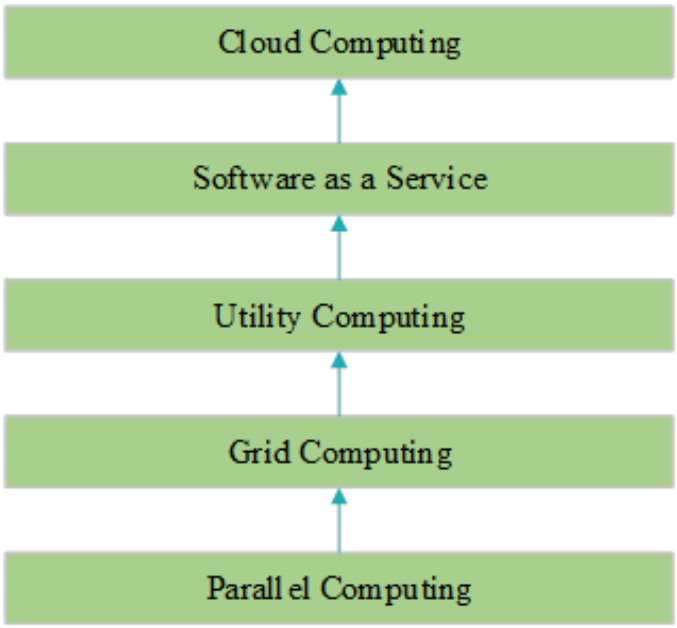

Figure 1.2 Developments in Cloud Computing [8]

Figure 1.2 shows the various developments in CC. The notion of the cloud computing gained practical traction through the initiatives of various enterprises, including Google. An illustrative case in point is Google's implementation of the concept with its Google Docs service. This service empowers users to collaboratively work on official documents directly through a web browser, exemplifying the essence of cloud

computing. [9]. The development of Amazon Web Services (AWS) by Amazon.com, a well-known online retailer, marked the beginning of the cloud computing revolution. In 2006, this marked the inception of contemporary 'clouds' [10]. AWS provides an extensive array of services, including robust computing capabilities and data warehousing. These services represent a highly reliable and cutting-edge cloud web service infrastructure. Following Amazon.com's lead, other major players like Microsoft, Google, Apple, and IBM also entered the cloud computing arena. Consequently, the CC market is fast growing. [11].

The most current technology for providing users with access to shared resource pools in response to their demands, according to the National Institute of Standards and Technology (NIST), is cloud computing. In a projection dating back to 1969, Leonard Kleinrock, a prominent scientist involved in the original Research Projects Network (ARPANET), accurately predicted the ongoing evolution and expanding utilization of computer networks. Furthermore, he anticipated the expansion of computer applications during the 1990s.[8].

Numerous technological advancements have played a pivotal role in shaping the landscape of cloud computing paradigms. These encompass:

- Network Computing
- Grid Computing
- Parallel Computing
- Utility Computing
- Virtualization
- Local Computing

The subsequent section delves into the most prominent technologies in this domain;

1.2.1 Parallel Computing

Parallel computation involves dividing the entire computational task into smaller segments that can be processed concurrently. This methodology encompasses various aspects of concurrent programming and the creation of efficient adaptations for existing hardware. A parallel computer constitutes a highly cohesive ensemble of

components that collaboratively and communicatively address extensive computational challenges, resulting in accelerated problem-solving capabilities [12].

1.2.2 Grid Computing

The idea of grid computing has received a lot of attention, and it's noteworthy that important market participants like Hewlett-Packard, International Business Machines and Sun Microsystems have shown a strong interest in this technology. The primary challenge in grid computing lies in allocating geographically dispersed resources to diverse user groups, often referred to as 'virtual organizations,' collaborating on tasks. This challenge underscores the absence of both a central focal point and centralized control, emphasizing the significance of trust and cooperation in collaborative endeavors.

In the context of a grid system, the establishment, management, and utilization of relationships among potential project participants' resources are of paramount importance. Achieving interoperability within a network environment necessitates adherence to universally accepted protocols. These protocols dictate the behavior and format of exchanged information, governing the interaction between elements within a distributed system. Consequently, the grid architecture essentially functions as a protocol framework that defines the foundational mechanisms of interaction [13].

1.2.3 Utility Computing

Users get services as required, and fees are charged in accordance with that requirement. Service providers strive to respond to individual tastes while customizing their services to fit clients' budgetary restrictions. The service provider possesses the ability to allocate resources based on user requests while implementing a pricing structure that optimizes utility utilization and minimizes resource costs. [14].

1.2.4 Virtualization

Virtualization serves as a method to generate a simulated representation of resources, dissociated from hardware (in this context, software-related). This concept extends to virtualizing servers, storage, network resources, applications, and desktops. Essentially, virtualization involves segregating physical components to furnish users

with virtualized resources. [15]. A standard server has the capacity to accommodate various virtual machine instances, which can then be accessed by users as needed.

1.2.5 Centralized computing

Centralized Computer Systems, often referred to as Centralized Computing, encompass a unified location where computer system resources, including processors, memory, and storage, are collectively shared and interconnected. This approach can be observed in parallel or distributed configurations within multiple data centres or may be adopted as a framework within architectural or cloud computing contexts. [16].

1.3 Characteristics of Cloud Computing

This section describes the fundamental characteristics related to CC.

1.3.1 On-demand Self-service

In response to consumer needs, cloud service providers provide users with immediate services like network storage that may be accessed anytime, anywhere, all without requiring human intervention. [17, 18].

1.3.2 Broad Network Service

Figure 1.3 refers to characteristics of CC in which various applications are hosted in a manner that enables access through various network-connected devices like laptops, desktops, mobile phones, tablets, and workstations.

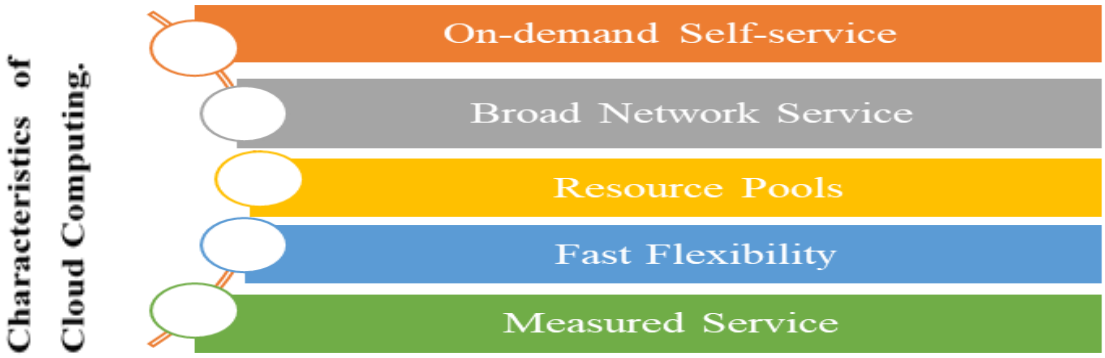

Figure 1.3 Characteristics of Cloud Computing [18]

Typically, these applications are accessed using a built-in web browser on the device, which serves as a common client, allowing widespread network access.

1.3.3 Resource Pools

It represents a concept where diverse organizations collaborate to utilize a common physical cloud infrastructure. The redistribution of virtual, not actual, resources is determined by consumer requirements. Typically, consumers are unaware of the specific location of a given resource; nevertheless, they may identify a broader level of location abstraction. Examples of these resources encompass processing power, storage capacity, memory, and network bandwidth. [14, 19, 20].

1.3.4 Fast Flexibility

Fast flexibility entails the capability to offer scalable services that enable users to effortlessly request varying capacities or types of services within the cloud environment. This attribute reflects a system's capacity to align resources with the demands of workloads, by provisioning resources to closely match present requirements. From the consumer's perspective, the availability of resources often appears unlimited and accessible on demand. This elasticity empowers customers of cloud providers to attain cost efficiencies, typically being a fundamental driver for adopting cloud services.

1.3.5 Measured Service

The cloud system autonomously manages user resource utilization by gauging the system's measuring capabilities. Measurement services serve as benchmarks for cloud providers to gauge, monitor, and manage services, encompassing aspects like billing, resource optimization, and comprehensive forecasting strategies [21–24].

1.3.6 Significance of Resource Allocation

Improper distribution of resources can result in service shortages. To tackle this issue, Resource Supply comes into play as a mechanism facilitating the provisioning of services by managing the resources of individual components. The pivotal element here is the Resource Allocation Strategy (RAS), a framework devised to enhance the

efficiency and equitable distribution of limited resources within the boundaries of the cloud environment. Consequently, this approach caters to the specific demands of diverse cloud applications. [21]. The effectiveness of the strategy depends on clearly defining the volume and categories of resources necessary for each application to effectively accomplish user assignments. The Resource Allocation Strategy (RAS) takes into account inputs such as temporal data and the succession of resource demands. Structured in its design, RAS effectively addresses the ensuing challenges. [22].

1.4 Task Scheduling

Algorithms for task scheduling are used to assign jobs to resources in a computing system in order to maximise resource utilisation, reduce execution time, increase energy efficiency, and accomplish specified goals. These algorithms establish the guidelines and techniques for deciding which tasks should be executed, when they should be executed, and which resources they should be assigned to. Their purpose is to efficiently manage the allocation of tasks in a computing system to maximize performance and meet desired goals, and is shown is Figure 1.4.

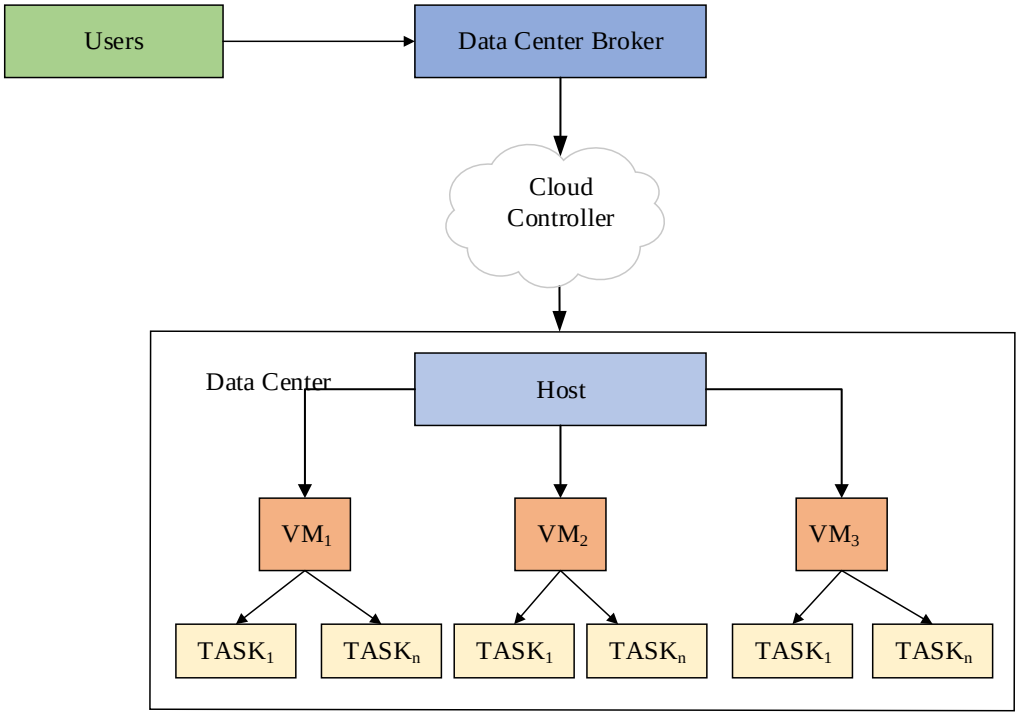

Figure 1.4 Task Scheduling Structure [105]

1.4.1 Advantages of Task scheduling algorithms

Task scheduling brings several advantages to computing systems. It ensures optimal resource utilization by assigning tasks in a balanced and optimized manner, maximizing the use of available resources and improving system performance. By minimizing execution time through intelligent task assignments based on resource availability and task requirements, overall system productivity and responsiveness are enhanced. Task scheduling algorithms also contribute to improved system performance by optimizing task allocation, considering load balancing, resource availability, and task dependencies. Energy-aware task scheduling algorithms further optimize energy utilization, reducing operational costs in energy-constrained systems. By taking task deadlines, priority levels, and resource capabilities into account, task scheduling also plays a significant role in upholding Service Level Agreements (SLAs) and achieving Quality of Service (QoS) criteria. Task scheduling algorithms are scalable and flexible, capable of handling large-scale computing systems and dynamic workloads. They ensure load balancing by distributing the workload evenly across resources, preventing resource overutilization and underutilization. This leads to a balanced system and avoids performance degradation. Additionally, effective task scheduling contributes to system stability by preventing resource starvation, ensuring fair resource allocation, and avoiding excessive queuing delays, resulting in a more stable and reliable computing environment.

1.4.2 Task Scheduling in Cloud Computing.

In CC, tasks are efficiently assigned to available resources using a task scheduling system. It maximises resource utilisation, cuts down on the duration of execution, and makes sure the cloud infrastructure runs without any issues. Key considerations of a task scheduling system include task management, resource monitoring, scheduling policies, load balancing, task prioritization, energy efficiency, fault tolerance, scalability, and performance optimization. The system receives tasks from users or applications, monitors the availability and performance of resources, and employs scheduling policies and algorithms to assign tasks to appropriate resources. Load balancing ensures an even distribution of the workload, while task prioritization

considers deadlines and priorities. Energy-aware algorithms minimize energy consumption, fault tolerance mechanisms handle failures, and the system scales dynamically to handle varying workloads. Performance monitoring and optimization improve system efficiency, and the task scheduling system model in cloud environments is shown in Figure 1.5.

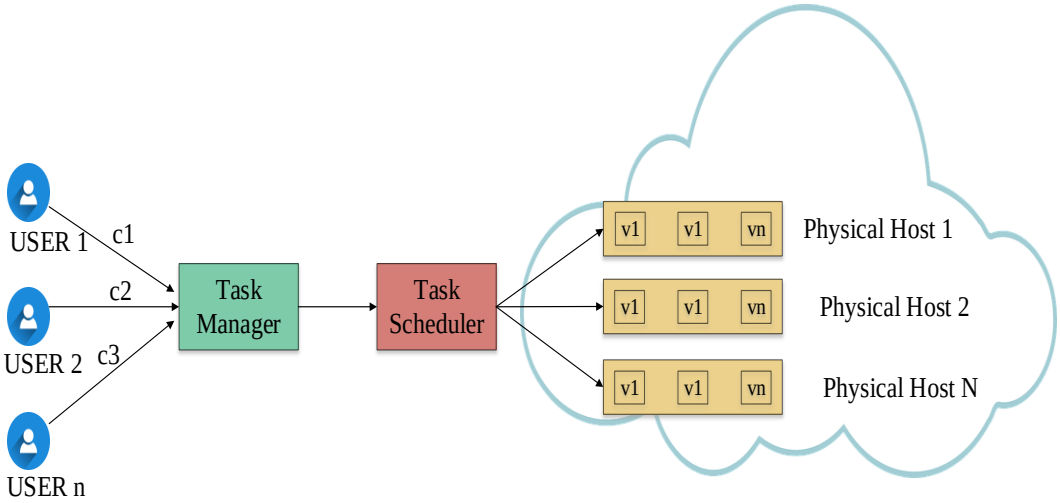

Figure 1.5 Task Scheduling System Model in Cloud Environments [106]

1.5 Types of Cloud Computing

In CC, there are different types of clouds that depend on their deployment and service models. These types of clouds can be combined with service models like IaaS, PaaS, and SaaS to provide different levels of control and management to users based on their specific requirements. Although the terms homogeneous and heterogeneous are sometimes used in cloud computing discussions, they typically refer to the composition of resources within a specific cloud infrastructure or data centre. Homogeneous refers to an infrastructure with identical components, while heterogeneous means the infrastructure includes diverse components. The main types are shown in Figure 1.6.

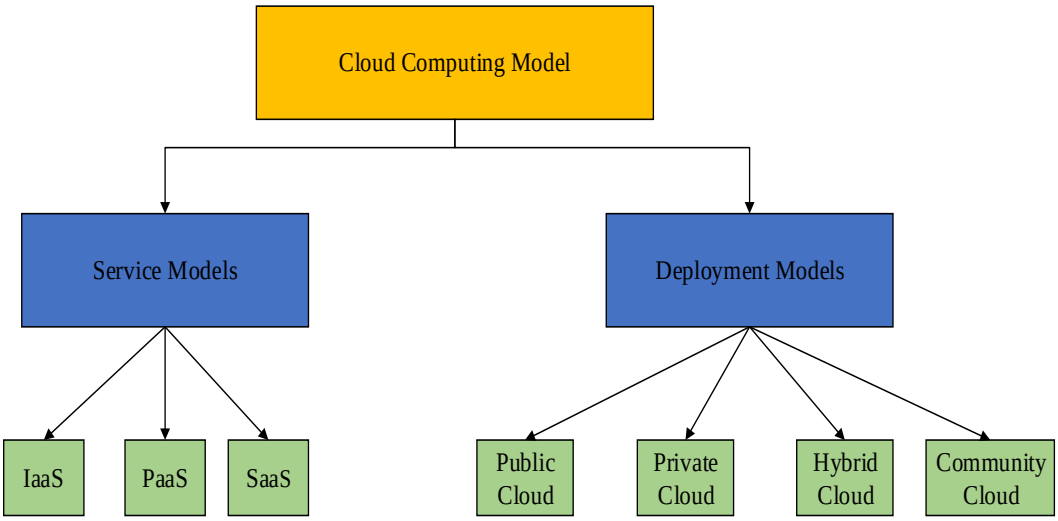

Figure 1.6 Cloud Computing Classification [107]

1.5.1 Public Cloud

A third-party supplier manages and controls this kind of cloud, which provides resources for computers over the internet to numerous customers or businesses. It allows for cost efficiency and scalability as the infrastructure is shared among users. The general block diagram of the public cloud is shown in Figure 1.7.

a) Advantages of Public Cloud:

- **Scalability:** According to demand, public clouds allow for scaling up or down. You can easily allocate more resources when needed and release them when demand decreases.
- **Cost-Efficiency:** With public clouds, you only pay for the resources you utilise because they operate on a pay-as-you-go model. This can lead to cost savings, as you avoid the upfront investment in hardware and infrastructure.
- **Accessibility:** With publicly accessible cloud services, data and apps may be accessed remotely from anywhere over the internet, which is very useful for scattered teams and remote work settings.
- **Quick Deployment:** Setting up and provisioning resources in a public cloud environment is faster compared to traditional infrastructure setups, allowing you to get applications up and running quickly.

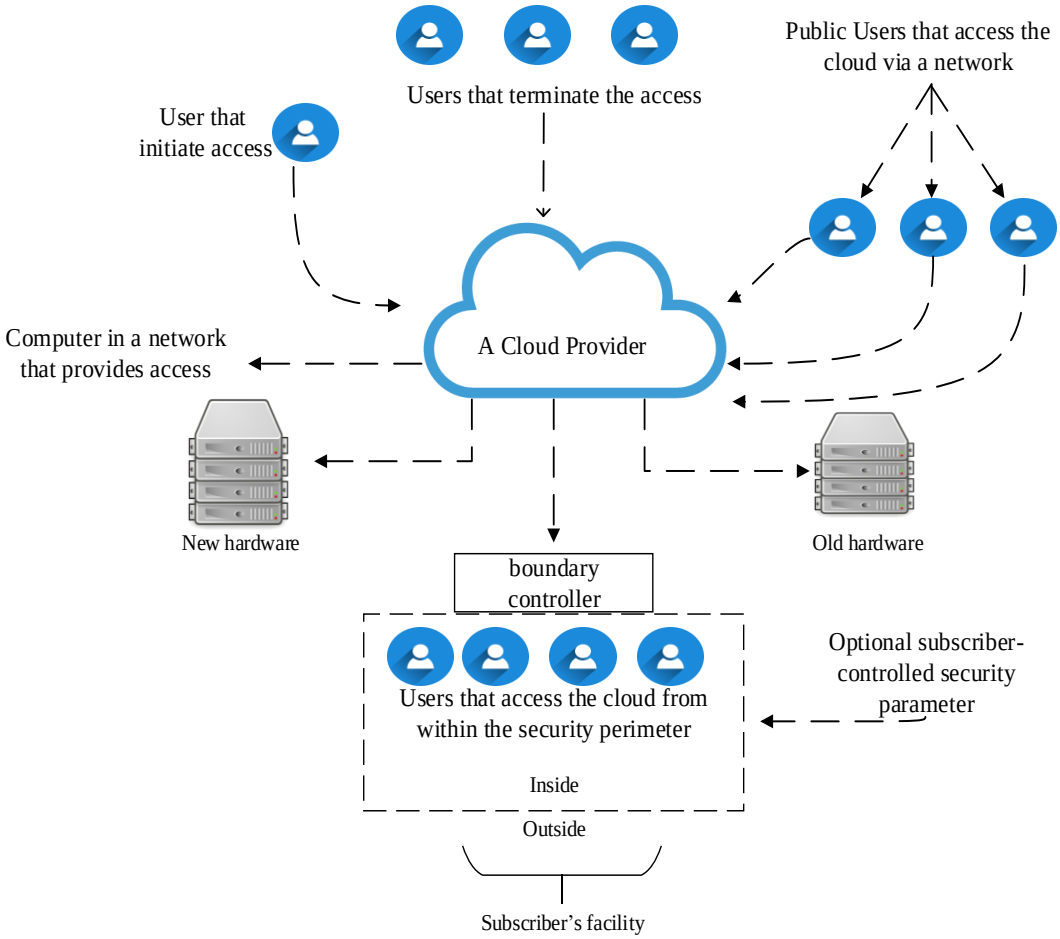

Figure 1.7 Public Cloud [12]

b) Limitations of Public Cloud:

- **Security and Privacy Concerns:** Applications and sensitive data stored on a public cloud might pose security and privacy problems. To secure their data, organisations must make sure that the necessary security measures are in place.
- **Dependency on Internet Connectivity:** Public cloud services require a reliable internet connection. You might not be able to access and use cloud resources if your internet connection is erratic or sluggish.
- **Downtime and Outages:** Although public cloud providers strive for high availability, they can still experience outages. Your applications and services may be affected during these incidents.

- **Cost Management Complexity**: While the pay-as-you-go model can be cost-efficient, it can also lead to unpredictable costs if resources are not managed properly. Monitoring and managing expenses require ongoing attention.

1.5.2 Private Cloud

A computing ecosystem that is solely dedicated to one company is known as a private cloud. In terms of scalability and virtualization, it provides many of the same advantages as public clouds but works on the company's internal infrastructure. Private clouds are intended to give more control, security, and customisation compared to public cloud options. They can be maintained by the company's information technology department or by a third-party supplier. The private cloud is shown graphically in Figure 1.8.

a) Advantages of Private Cloud

- **Enhanced Security:** Private clouds provide greater control over data security and compliance. Organizations can implement customized security measures and access controls to meet specific regulatory requirements.
- **Customization:** Private clouds allow organizations to tailor the infrastructure and environment to their specific needs, enabling optimal performance for applications and services.
- **Performance Control:** Since resources are dedicated to a single organization, there is more predictable performance without the performance fluctuations that can occur in shared public cloud environments.
- **Compliance:** Private clouds are well-suited for industries with strict regulatory requirements, such as healthcare and finance, where data governance and compliance are crucial.
- **Reduced Downtime:** Private clouds can offer higher availability and reduced downtime compared to public clouds, as organizations have more control over maintenance schedules and resource allocation.

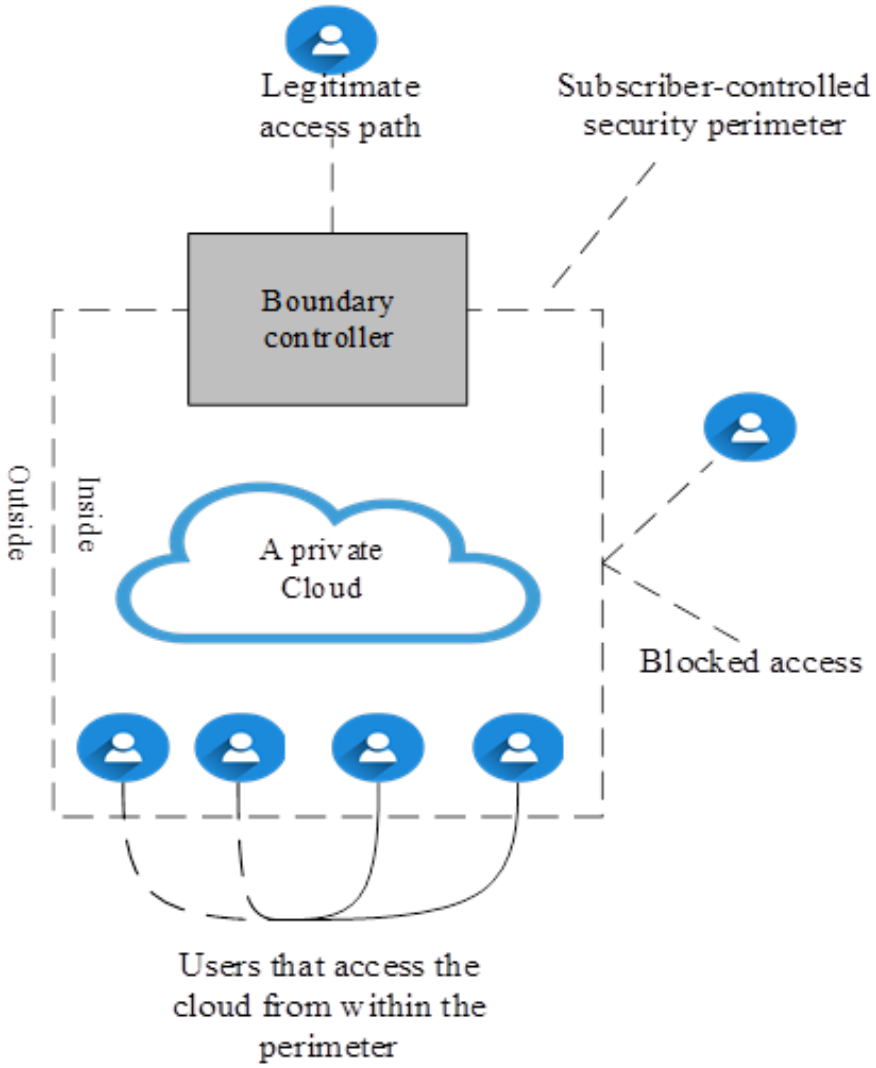

Figure 1.8 Private Cloud [12]

b) Limitations of Private Cloud:

- **Higher Costs:** Private cloud solutions are less affordable than public cloud options since establishing and managing a private cloud infrastructure needs a sizable upfront investment in hardware, software, and maintenance.
- **Complexity:** It may be difficult to manage a private cloud, therefore you need knowledgeable IT staff that are familiar with virtualization and cloud computing.

- **Scalability Challenges:** While private clouds can be scaled, achieving the same level of scalability as public clouds may be more challenging due to infrastructure limitations.
- **Limited Innovation:** Private clouds may not have the same rapid access to cutting-edge technologies and features that public cloud providers regularly introduce.

1.5.3 Hybrid Cloud

A hybrid cloud allows for the exchange of apps and data between private and public cloud environments. It provides additional mobility by letting businesses use the advantages of the public as well as private clouds while taking care of particular workloads and data needs.

a) **Pros of Hybrid Cloud:**
   - **Flexibility:** Organizations can choose where to host different workloads based on factors like security, performance, and compliance, achieving an optimal balance between the two cloud models.
   - **Scalability:** Hybrid clouds allow bursting into the public cloud during high-demand periods, ensuring resources are available when needed without overprovisioning private infrastructure.
   - **Cost Efficiency:** By using public clouds for temporary or variable workloads, organizations can avoid the upfront costs of overprovisioning private resources.
   - **Data Control:** To address security and compliance issues, sensitive data can be stored in a cloud that is private whereas non-sensitive information data is processed on a public cloud.

b) **Cons of Hybrid Cloud:**
   - **Complexity:** Integrating and managing two distinct cloud environments can be complex, requiring specialized skills and coordination between the private and public components.

- **Security Challenges:** Ensuring consistent security measures across both environments can be challenging, leading to potential vulnerabilities if not managed properly.
- **Data Transfer Costs:** Moving data between private and public clouds may incur costs, especially if large volumes of data need to be transferred frequently.
- **Vendor Lock-In:** Organizations might face vendor lock-in issues if they heavily rely on a specific public cloud provider's tools and services.
- **Management Overhead:** Administering and monitoring a hybrid cloud requires additional effort to ensure smooth operation, resource optimization, and cost control.

1.5.4 Community Cloud

Several organisations share community clouds, which are used to address issues like compliance and security. They offer a platform for collaboration for organisations with comparable needs and can be administered internally or by a third-party service.

a) Merits of Community Cloud:

- **Shared Resources:** Organizations within a specific community can pool their resources to create a collaborative and cost-effective cloud solution.
- **Data Control:** Community clouds allow organizations to maintain control over their data and applications while benefiting from shared infrastructure.
- **Compliance:** Community clouds can be tailored to meet specific regulatory and compliance requirements common to the industry or community members.
- **Cost Savings:** By sharing infrastructure costs among community members, each organization can achieve cost savings compared to setting up their private clouds.
- **Customization:** Community clouds offer flexibility without compromising data security since they may be tailored to the specific demands of the community members.

b) **Demerits of Community Cloud:**

- **Complexity:** Coordinating among multiple organizations within the community can be complex, leading to potential management challenges.
- **Security Concerns:** While community clouds can offer improved security compared to public clouds, they still need careful security measures to ensure data privacy and protection.
- **Limited Scalability:** Depending on the size and growth of the community, scalability might be limited compared to larger public cloud environments.
- **Shared Risks:** Since resources are shared, any downtime or issues affecting one organization might impact others as well.

1.5.5 Multi Cloud

The use of services from various cloud service providers (CSPs) to meet an organization's computing needs is known as multi-cloud. In a multi-cloud environment, an organization distributes its workloads, applications, and services across different cloud platforms rather than relying solely on a single provider. This approach offers a range of benefits but also presents certain challenges that organizations need to manage effectively. Figure 1.9 represents the comparison of Multi and Hybrid clouds. There are several variations and strategies within the multi-cloud approach. Here are some different variations:

i. **Multi-Provider Multi-Cloud:** The most popular form of the multi-cloud approach involves an organisation using services from a variety of different cloud providers in order to prevent vendor lock-in and to take use of each provider's advantages.

ii. **Hybrid Multi-Cloud:** This approach combines a mix of private, public, and multiple public cloud providers to create a hybrid infrastructure. It allows organizations to keep sensitive data in private clouds while utilizing public clouds for specific workloads.

iii. **Geographical Multi-Cloud:** In this variation, an organization uses different cloud providers in different geographic regions to optimize for performance

and data residency requirements. This can help reduce latency and comply with data regulations.

iv. **Specialized Multi-Cloud:** This strategy involves using different cloud providers that offer specialized services optimized for specific tasks. For instance, using one provider for AI/ML capabilities and another for high-performance computing.

v. **Cost Optimization Multi-Cloud:** This strategy involves using different cloud providers based on cost. Organizations choose providers with the most cost-effective offerings for specific workloads.

vi. **Data-Driven Multi-Cloud:** Depending on the kind of data being processed, this strategy makes use of several cloud service providers. Use one supplier for big data analytics and another for processing transactional data, for instance.

vii. **Application-Centric Multi-Cloud:** Here, different cloud providers are chosen based on the specific needs of different applications. This could involve optimizing costs for non-critical apps while ensuring high availability for mission-critical apps.

a) Benefits of Multicloud Strategy:

- **Vendor Diversity:** By utilizing multiple cloud providers, organizations can avoid vendor lock-in. This means they are not tied to a single provider's ecosystem and can choose the best solutions from different providers.
- **Best-of-Breed Services:** Different cloud providers specialize in various services. A multi-cloud approach enables organizations to select the most suitable provider for each specific service, capitalizing on each provider's strengths.
- **Cost Optimization:** Different providers offer varied pricing models and discounts. By strategically distributing workloads, organizations can optimize costs based on the specific requirements of each workload.

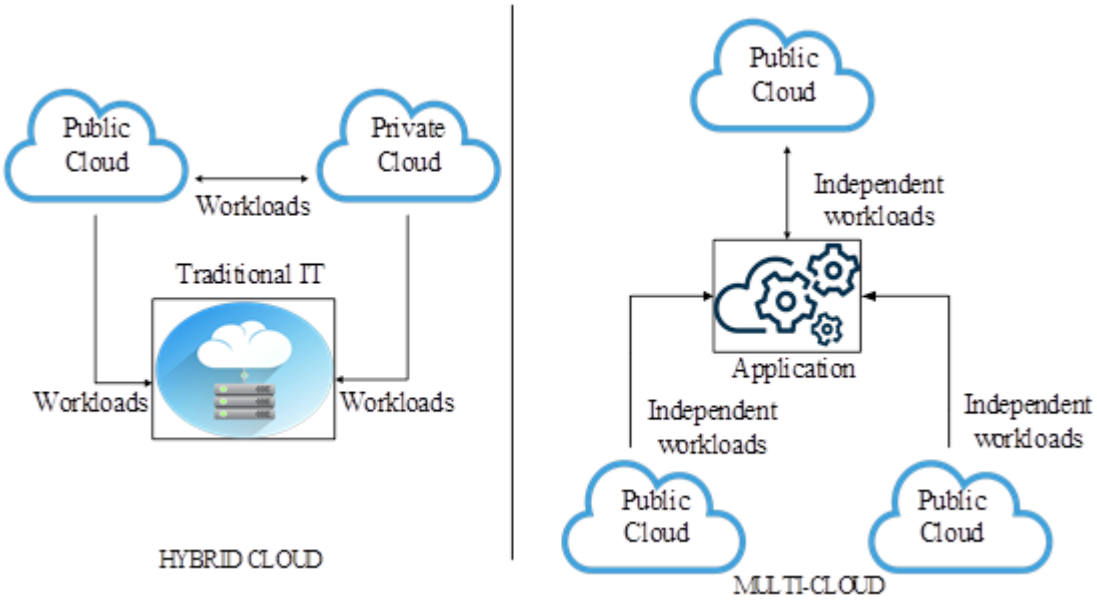

Figure 1.9 Comparison of Hybrid and Multi Cloud [12]

b) Challenges of Multicloud Strategy:

- **Complexity:** Managing multiple cloud providers introduces complexity in terms of billing, provisioning, security, and governance. This complexity can increase the workload for IT teams.
- **Data Transfer Costs:** Transferring data between different cloud platforms might incur costs, particularly if not planned and managed carefully.
- **Security and Compliance:** Ensuring consistent security measures and compliance across multiple providers can be intricate, requiring thorough planning and monitoring.
- **Security and Governance:** Develop consistent security measures, access controls, and compliance policies across all providers.
- **Monitoring and Management:** Implement robust monitoring and management tools that provide visibility into the performance and health of all deployed services.
- **Data Management:** Plan data storage, backup, and migration strategies to ensure seamless data movement between providers.

1.6 Job Scheduling for Optimal Usage of Resources

In cloud computing environments, scheduling algorithms hold significant importance in enhancing resource utilization and overall performance. Their primary duties involve effectively allocating computational resources and controlling how activities and assignments are carried out on the cloud infrastructure. We will look at a number of well-known scheduling algorithms used in the context of CC in this study, exploring their workings and benefits [25]. We will also go through each algorithm's minimum execution time, which shows how long it takes to finish the scheduling process.

1.6.1 First-Come, First-Served (FCFS) Scheduling

A straightforward scheduling system called FCFS bases its operations on the time that tasks arrive. Without taking into account their priority or execution time, the tasks are completed in the order they are received. This algorithm ensures fairness, as it follows a first-come, first-served approach. However, FCFS may result in subpar performance when long-running activities are completed early, delaying subsequent shorter jobs for an extended period of time. The number of tasks and their execution times affect FCFS's minimum execution time [23].

1.6.2 Shortest Job Next (SJN) Scheduling

Shortest Job Next (SJN) scheduling strives to minimize the average waiting period by prioritizing the task with the briefest execution duration for the subsequent execution. To implement this algorithm effectively, advance knowledge of the execution duration for each task is essential. By prioritizing short tasks, SJN reduces waiting time and improves system efficiency. However, SJN can be problematic in scenarios where the execution time is unknown or cannot be accurately predicted. The distribution of task execution times and the number of tasks determine the minimal SJN execution time. [24].

1.6.3 Round Robin (RR) Scheduling

RR is a popular scheduling method that cyclically allots a similar time slice to each activity. It provides equity and prevents any task from having a resource monopoly for an extended period of time. Each task is allowed to execute for a predefined time quantum before being pre-empted and moved to the back of the scheduling queue. RR is suitable for scenarios with diverse task execution times, as it provides equal opportunities for all tasks. The time quantum and the number of tasks determine the RR's minimal execution time [26].

1.6.4 Priority-Based Scheduling

Tasks are given priority levels according to their relevance or criticality in priority-based scheduling. The scheduler executes tasks with higher priority levels first, ensuring that critical tasks receive immediate attention. This algorithm allows for better control over task execution and resource allocation. However, improper prioritization can lead to starvation of low-priority tasks. The number of jobs and the number of priority levels determine the priority-based scheduling's minimum execution time [27].

1.6.5 Genetic Algorithm (GA) Scheduling

The metaheuristic method known as GA was influenced by the idea of natural selection. It evolves a population of potential schedules and finds the best solution using genetic operators like mutation, crossover, and selection [28]. GA-based scheduling considers several parameters, such as task execution time, resource availability, and task dependencies, to provide efficient schedules. The minimal execution time of GA-based scheduling depends on the size of the population, the number of generations, and the complexity of the problem [29].

1.6.6 Minimum Execution Time (MET)

MET is similar to the shortest job first but it also considers the allocation cost of the job. If a work has to be executed on more than one cloud host, MET will broadcast

the demand and will allocate the job to the server with the shortest execution time [30].

As time has passed by, the evolution of ML methods has been observed to be applied in cloud scheduling approaches. The aim is to learn from previous allocations and to assign a rank so that overall power consumption can be reduced [31]. Hence the preliminary aim of ML based algorithm can be defined as follows.

$$\boldsymbol{Obj_{ML} = argmin(PC)} \quad \textbf{(1.1)}$$

Where PC is the power consumption in assigning a job to a host system.

The overall quantity of power that is used over a certain amount of time is referred to as energy consumption.

$$\boldsymbol{E = \int_{i=1}^{t} P\, dt} \quad \textbf{(1.2)}$$

1.7 Various parameters of Cloud computing

Cloud computing involves various parameters that define its characteristics and capabilities. Here are some of the key parameters in CC that have been used popularly in the research community for determining the effectiveness of cloud scheduling.

1.7.1 Makespan

Makespan is the entire amount of time needed to complete a series of tasks or procedures in a cloud environment. It displays the total time required for each task, beginning with the first one. In cloud computing systems, the makespan is a crucial performance parameter for assessing the effectiveness and efficiency of resource allocation and job scheduling algorithms.

The makespan can be calculated using the following formulas:

1. **Start Time (ST):** This indicates the moment a task starts to be completed.
2. **Finish Time (FT):** This shows the moment a task completes execution.

3. **Makespan (MS):** It is computed by deducting the time the initial operation began from when the prior job was completed. In other words, it shows the total time required to complete all tasks.

The makespan may be calculated mathematically using the following formula:

$$\boldsymbol{MS = FT_{last} - ST_{first}} \quad \textbf{(1.3)}$$

Here, FT_{last} refers to the finish time of the last task, and ST_{first} refers to the start time of the first task.

The makespan in cloud computing is optimised using a variety of scheduling and allocation of resources approaches. These techniques aim to minimize the makespan by efficiently utilizing cloud resources and distributing tasks effectively among the available resources. By reducing the makespan, cloud systems can achieve improved performance, enhanced resource utilization, and faster task completion.

1.7.2 Energy Consumption

The phrase "energy consumption" describes how much energy the different components of the cloud architecture—such as servers, networking hardware, storage, and cooling systems—consume. Effective management of energy consumption is essential to reduce operational costs and minimize the environmental impact of cloud computing.

Factors that influence energy consumption in cloud computing include:

1. **Server Power Consumption:** The energy consumed by servers depends on factors such as their processing capacity, utilization levels, and the energy efficiency of hardware components.
2. **Cooling and HVAC Systems**: Data centres require cooling systems to maintain the optimal temperature for servers. For the purpose of removing the heat produced by the servers, these cooling systems use electricity.
3. **Networking Equipment:** Networking devices such as switches and routers also consume energy, influenced by factors like data traffic volume, network topology, and utilization.

4. **Storage Systems:** Energy consumption in storage devices, such as hard drives and SSDs, varies based on their capacity, utilization, and energy efficiency.

Several metrics are used to measure energy consumption in CC:

a. **Power Usage Effectiveness (PUE):** PUE compares the total energy consumed by a data centre (including IT equipment and cooling) to the energy consumed by IT equipment alone. Lower PUE values indicate higher energy efficiency.

$$\boldsymbol{PUE = \frac{Total\ data\ center\ power}{IT\ Devices\ Power}} \quad \textbf{(1.4)}$$

b. **Data Center Infrastructure Efficiency (DCIE):** DCIE is the reciprocal of PUE and represents the percentage of energy consumed by IT equipment relative to the total energy consumed by the data centre infrastructure.

$$\boldsymbol{DCIE = \frac{1}{PUE} = \frac{IT\ Devices\ Power}{Total\ Data\ center\ Power}} \quad \textbf{(1.5)}$$

c. **Energy Proportionality:** Energy proportionality refers to a system's ability to dynamically adjust its energy consumption based on workload demands. Systems with higher energy proportionality consume less energy during periods of low workload.

$$\boldsymbol{EP = \frac{Energy\ Resued}{Total\ power}} \quad \textbf{(1.6)}$$

Employing energy-efficient hardware, streamlining workload allocation, and putting power management strategies like server consolidation and dynamic voltage and frequency scaling (DVFS) into practise are all part of optimising energy use in cloud computing.

1.7.3 CO_2 Emission

CO_2 emissions in computing result from the energy consumption and operation of cloud infrastructure, releasing carbon dioxide and other greenhouse gases. Despite the

potential environmental benefits of cloud computing, it still contributes to carbon emissions due to energy usage Carbon Usage Effectiveness is a metric used to determine CO2 emissions. As defined by CUE:

$$\boldsymbol{CUE} = \frac{\boldsymbol{Eco_2}}{\boldsymbol{EIT}} \quad (1.7)$$

Where

Eco_2 = Total energy used by the data centre, which results in a total CO2 output.

EIT = Total amount of energy used by IT devices

Several factors contribute to CO_2 emissions in CC:

1. **Energy Source:** The type of energy used to power the cloud infrastructure impacts CO_2 emissions. Carbon-intensive energy sources, like fossil fuels, lead to higher emissions compared to renewable energy sources.
2. **Data Center Operations**: Data centres, which house the cloud infrastructure, consume significant amounts of electricity. Powering servers, cooling systems, and other equipment contribute to CO_2 emissions, particularly if the energy comes from non-renewable sources.
3. **Cooling and HVAC Systems:** Data centres require cooling systems to maintain optimal temperatures. The energy consumed by these systems, including air conditioning and ventilation, adds to CO_2 emissions.
4. **Server Utilization**: Efficient server utilization is crucial for reducing CO_2 emissions. Underutilized servers consume excess energy, resulting in higher emissions. Techniques such as server virtualization and load balancing improve utilization and reduce energy waste.

Efforts to reduce CO_2 emissions in cloud computing include:

a. **Renewable Energy Adoption:** Reducing CO2 emissions by switching to renewable energy sources, such solar or wind power, to power data centres and cloud infrastructure.
b. **Energy Efficiency Measures:** Implementing energy-efficient hardware, optimizing cooling systems, and adopting energy-saving practices like dynamic power management help reduce overall energy consumption and lower emissions.

c. **Green Data Centers:** Designing and operating data centres with eco-friendly approaches, such as using energy-efficient equipment and cooling methods, contributes to reduced CO_2 emissions.
d. **Carbon Offsetting**: Some cloud providers offer carbon offset programs, investing in projects that offset the carbon emissions generated during cloud operations.

Monitoring and reporting CO_2 emissions in cloud computing ensures transparency and accountability. It enables cloud providers and users to understand the environmental impact and make informed decisions regarding energy consumption and sustainability initiatives.

1.7.4 Execution Cost

It is the sum of the job scheduling costs calculated across various physical equipment or data centres. When users are to be assigned to the PMs in the early phase of allocation, the scheduling design wants to achieve a minimum cost. Most of the time, the cost may be calculated using the following relationship between energy use and makespan

$$\boldsymbol{Overall_{Cost} = energy_{consumption} \times \alpha + \beta} \quad \textbf{(1.8)}$$

Where the unit cost is α and β is the quantity of energy that will be used due to malicious processing.

1.8 Machine Learning and its Evolution in Job Scheduling Under Cloud Computing

The area of machine learning (ML), which falls under artificial intelligence (AI), aims to create methods and algorithms that let computers take knowledge from data and use it to forecast and make decisions. ML has revolutionized various industries by providing solutions to complex problems and enabling intelligent automation [32, 33]. In recent years, the merging of machine learning (ML) with cloud computing has propelled its adoption and capabilities to even greater heights. The inclusion of a section on ML in the opening chapter of the study work is significant because it

underlines the significance of thoroughly knowing various machine learning algorithms before adopting or integrating them into the research. This understanding serves as the foundation for planning and creating efficient resource management solutions in multi-cloud systems. Researchers can find the best algorithms for tackling specific difficulties like workload prediction, resource allocation, and performance improvement by experimenting with various machine learning methodologies. This ensures that the suggested model has the ability to adapt to changing cloud settings and produce optimal results. Thus, a detailed evaluation of machine learning algorithms is required to inform the design and execution of the resource management model, thereby increasing its effectiveness and practical application in real-world multi-cloud settings.

- **Early Adoption:**

The initial stages of ML deployment were characterized by on-premises infrastructure, where organizations had to manage and maintain their own hardware and software resources. However, as ML algorithms became more complex and required substantial computational power and storage, cloud computing emerged as an ideal platform for ML applications. Cloud providers offered scalable and flexible infrastructure, allowing users to deploy ML models and leverage powerful hardware resources on-demand.

- **Scalability and Elasticity:**

Cloud computing provides the scalability and elasticity required for ML workloads. ML models often demand significant computing resources, and cloud platforms enable users to scale up or down their infrastructure based on the workload's requirements. This ensures optimal resource utilization and cost efficiency. Cloud providers offer services like virtual machines, containers, and serverless computing, making it easier to deploy and maintain ML applications at scale.

- **Data Storage and Processing:**

Platforms for cloud computing offer strong data processing and storage capacities, which are essential for machine learning processes. Large datasets are essential to the training and validation of machine learning algorithms. Large datasets may be securely and economically stored with the help of cloud storage systems like Google Cloud Storage and Amazon S3. Additionally, cloud-based data processing frameworks like Apache Spark or Google Cloud Dataflow provide distributed processing capabilities, allowing ML practitioners to efficiently pre-process and analyse vast amounts of data.

- **Managed ML Services:**

Cloud providers have introduced managed ML services that simplify the development, training, and deployment of ML models. They also offer automated model training, hyperparameter tuning, and model serving capabilities, reducing the complexity and time required to build and deploy ML applications.

- **Integration of AI Services:**

Cloud computing has enabled the integration of ML with other AI services. Cloud providers offer APIs and pre-trained models that allow developers to incorporate advanced AI functionalities into their applications without the need for extensive ML expertise. This integration of ML with cloud-based AI services empowers businesses to deliver more intelligent and personalized experiences to their users.

- **Federated Learning and Edge Computing:**

The evolution of ML in CC has also shown the way for emerging paradigms such as federated learning and edge computing. Federated learning enables ML models to be trained on distributed devices while preserving data privacy. Cloud infrastructure acts as an orchestrator, coordinating the training process across multiple devices, which accelerates model development and reduces the need for data transfer. Edge computing brings ML capabilities closer to the data source, allowing real-time inference and reducing latency. This is particularly valuable in applications like autonomous vehicles, IoT, and healthcare.

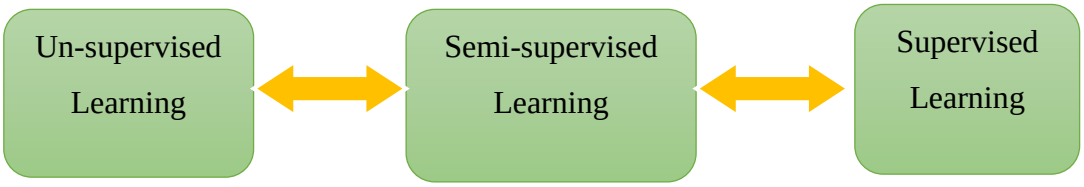

Figure 1.10 Machine Learning Approach [36]

Two of the most common approaches to machine learning are supervised and unsupervised learning [34]. Both options enable the user to supply the computer with a vast amount of data records in order for it to understand and establish relationships. This collected data is commonly referred to as a “feature vector” and the various machine-learning approaches are shown in Figure 1.10.

1.8.1 Supervised

Supervised learning comes in two flavours: regression and classification. The difference between the two is the core of their manufacturing vector. If the result is different, in the form of a real value, it is referred to as regression [35]. For instance, attributes like size, weight, height, and economic value belong to the realm of continuous variables. Conversely, classification comes into play when the outcome variable assumes a class or category format. Distinguishing between options like "pink," "white," "high," or "short" aligns with classification. When data input is categorized into precisely two classes, it is deemed binary classification. However, when data is categorized into more than two classes, it qualifies as a multi-class classification.

1.8.2 Un-supervised

In this ML technique, the data is not labelled. The machine must find the correct target without any prior knowledge, and therefore detect unknown patterns in the data. Algorithms must thus be created in a way that allows them to identify the right patterns and structures for them inside the data. Unsupervised learning can take two forms: association and clustering [35, 36]. When developers want to find trends or order in a large group of uncategorized results, they use clustering. Where the aim is

to identify relationships between various data items in broad datasets, the association is used. These approaches are graphically represented in Figure 1.11.

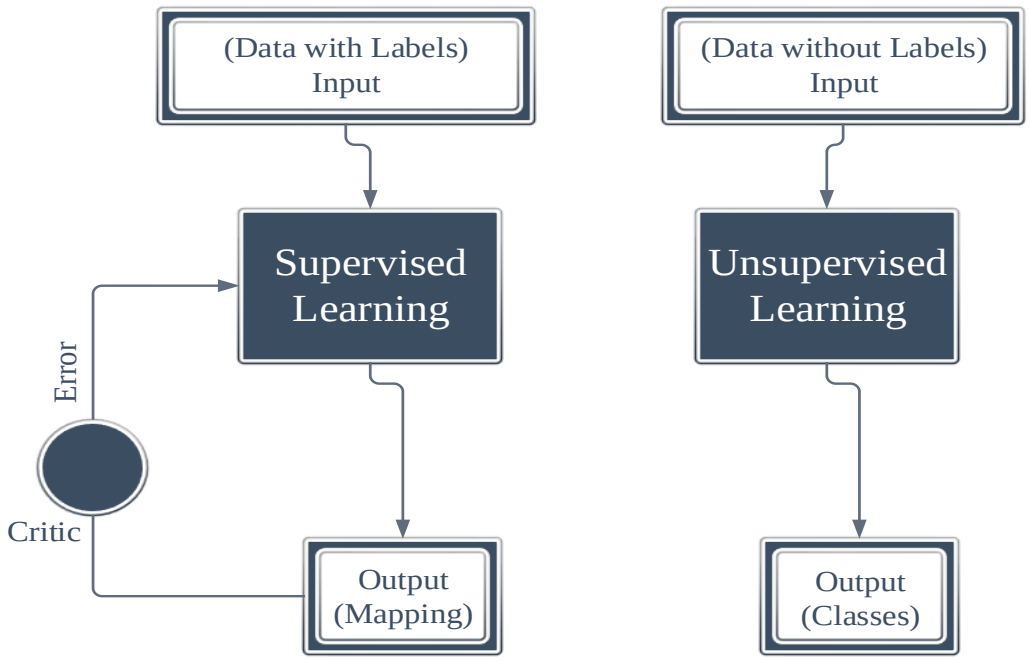

Figure 1.11 Supervised and Unsupervised Approach [34]

The key differences between the supervised and unsupervised learning approaches are listed in Table 1.1.

Table 1.1 Difference between Supervised and Unsupervised ML Techniques

SUPERVISED	**UNSUPERVISED**
Labelled data is provided as input	Unlabelled data is provided as input
Used trained dataset	Use just input dataset
Used for data prediction	Used for data analysis
Includes classification and regression	Used clustering approach, estimate density, and reduce dimension
e.g. ANN, CNN, SVM	e.g. K-means

1.8.3 Semi-supervised

Most deep learning classifiers require large numbers of labelled samples for good generalization, but obtaining such data is expensive and complex. To deal with this limitation, supervised learning in addition to unsupervised learning is presented, which is known as semi-supervised learning. It is a class of methods that use a set of labelled data along with a large amount of unlabelled data. Untagged data, when combined with a limited amount of labelled data, has been found to greatly increase training performance by many machine learning researchers. In the last decade, researchers have presented that semi-controlled learning can be used in combination with the performance of a small number of labelled data classifiers for IDSs that require less time and expense. Each machine learning algorithm has its own advantages and limitations that suit it best for some particular situations. The worthiness of most popular algorithms can be accessed from the following Table 1.2.

Table 1.2 Comparison of Machine Learning Algorithms

Machine Learning Techniques	**Advantages**	**Limitations**	**Remarks**
Naïve Bayes (NB)	• NB is an effective technique to extract the subjective sentence. • Suitable for small training set • Simple and straight forward to use • The interpretation is very easy. • The training data required for	• Bias outcomes when the number of training sets increases • Assumes all features to be independent which is not the case in the real world.	This is a robust and effective technique but still, there is a need for primary knowledge as it is sensitive to how the input data is prepared.

	processing is low to initiate the work.		
Decision Tree (DT)	• Can handle categorical data • The interpretation is easy to understand.	• Classes need to be mutually exclusive • Highly sensitive to noisy data and outliers • Increased risk of overly complex trees	The performance can be good on large datasets
K-nearest Neighbour (KNN)	• Nonparametric • No cost associated with the learning process • Implementation is easy • Robust for outlier prediction	• Result interpretation is very hard • Lacks explicit model training • Sensitive enough in case of measuring the distance. • Expensive computation in the case of a large dataset	The classification speed is slow when the training set is large.
Random Forest (RF)	• Easier to tune • Hard for overfitting	• An increased number of trees makes the algorithm slower • Not suitable for categorical analysis with a number of levels	Slows down in real-time prediction due to the generation of a larger number of trees
Support Vector Machine (SVM)	• Training is relatively easy • For practice and theoretical aspects, there is a good generalization.	• An appropriate kernel function is required to choose. • There is a problem of interpretation.	• The performance is very good. • For better performance knowledge of the kernel is

	• The feature space has less dependency on dimensionality.		required.
Neural Network (NN)	• Delivers good performance even in case of noisy data • The execution time is less.	• The implementation and interpretation are difficult. • Requires a large sample size for better performance. • There is a high memory usage. • Many parameters to be finely tuned.	Performance gets challenged for smaller sample sizes.
Q-learning	• Learns optimal policy directly • Computation cost is less • Comparatively fast • The perfect option for offline learning	• Uses biased samples • Per sample variance is high • Not efficient for online learning	A good option when offline leaning has to be performed

1.9 Metaheuristic Algorithms

Researchers across the globe have observed a prevailing trend in merging machine learning algorithms with metaheuristic algorithms to amplify the efficiency of existing scheduling procedures. A metaheuristic acts as a recurrent governing mechanism that directs and improves the operations of subordinate heuristics, producing the production of high-quality solutions. It is capable of manipulating either a single solution or a group of solutions in each iteration. These subordinate heuristics encompass methods varying in complexity levels, ranging from high to low, which could involve simple local searches or construction techniques.

Metaheuristics can be conceptualized as a set of foundational principles that facilitate the formulation of heuristic approaches adaptable to a diverse spectrum of problems. They embody a general algorithmic framework that offers versatility, requiring minimal adaptations to suit specific issues across a range of optimization problems. Metaheuristic techniques entail a repetitive sequence of exploration actions, rendering them potent tools for effectively tackling intricate optimization challenges. Predominantly, metaheuristics draw inspiration from natural processes, yet a subset also takes cues from biological systems, human activities, nature, and physical systems.[37, 38], etc.

- **Genetic Algorithm:** It is one of the simplest algorithms that is based on the prospective solutions observed as a population. In this type of metaheuristics, the new solutions are generated by a number of genetic operations such as mutation and crossover [39]. These new solutions replace the existing solutions. The GA is based on Darwin's idea of survival of the fittest.
- **Simulated Annealing:** This metaheuristic is inspired by the metallurgy and describes the cooling process carried on in the furnace. In simulated annealing the probability of reaching a low-valued solution considerably decreases with an increase in algorithm runs [40, 41]. Hence, the probability of reaching the best global solution significantly increases and this proved to efficiently resolve various complex optimization problems.
- **Swarm Intelligence:** It covers algorithms that are inspired by the group or swarm-based intelligent behaviour of the organisms. The group behaviour here means the collective and collaborative work of organisms such as colonies of ants. The major reasons behind exhibiting such group behaviour are mainly foraging for food, evading prey or relocation of the colony. In most of the cases, the communication was performed using pheromones (ants), proximity (fish) or dance (bees). Some of the widely used nature-inspired algorithms that fall under this category are PSO [42], ABC [43], ACO [44], Firefly [45] and Grasshopper Optimization Algorithm [46].

1.9.1 Characteristics of Metaheuristic

There are different characteristics of metaheuristic [47] that can be defined as follows:

- By effectively searching the search space, a global technique that uses the metaheuristic approach to solve problems ensures that the best or nearly the best answer will be found.
- Heuristics that are governed by the higher-level strategy are used by metaheuristics to access domain-specific knowledge.
- Metaheuristic methods are problem-independent and more flexible as compared to exact methods.
- Traditional methods are not able to handle voluminous data efficiently. On the other hand, metaheuristics handles high-dimensional data in a viable manner.

1.10 Organisation of the Thesis

The chapters of the thesis are organized in the following manner.

- Chapter 1 introduces the scheduling architecture that is involved in the Cloud Computing paradigm. The introduction section includes a description of the usage and evolution of the ML algorithms in scheduling architecture as well as the hybridization of the scheduling architecture via ML and meta-heuristic approaches.
- Chapter 2 illustrates the related work section that incorporates the articles that list hybrid scheduling architectures or the architectures that support meta-heuristic algorithms to support the approach of the proposed algorithm.
- Chapter 3 introduces the MET algorithm and cyber shake work flow and discusses the proposed work algorithm for the enhancement.
- Chapter 4 is dedicated to the application and improvisation of the flower pollination algorithm while involving Q-learning.
- Chapter 5 reviews the findings and provides a thorough analysis of the work that is planned.
- Chapter 6 concludes with the evaluated results and future work.

CHAPTER 2: REVIEW OF LITERATURE

The cloud computing environment has attracted masses due to its potential services offered to service users as well as the service providers. It has shown tremendous grown in a very small-time span in the service industry. However, there are some adjoining challenges and issues that need to be constantly addressed to ensure its advantages to overcome its associated limitations. The literature survey discussed in this chapter focuses on discussing the various research publications and articles that have been presented to address the scheduling in cloud in energy efficient manner. The following studies also show that the research community is constantly looking forward to minimizing CO_2 emissions in order to ensure green computing by employing various optimization and machine learning algorithms.

2.1 Literature Review

Lucas-Simarro et al. introduced an extensible broker framework that allowed for the best possible deployment of virtual services across several clouds. The design enabled a variety of scheduling strategies, taking into account user constraints, environmental considerations, and numerous optimisation criteria. The study looked at how different clustered services, such as an HPC cluster and a cluster of web servers, were deployed in a multi-cloud context. Different conditions, constraints, and optimisation criteria were covered by the analysis. Overall, the study demonstrated the efficiency and advantages of the modular broker architecture for deploying virtual services optimally across various clouds [48].

Dhanalakshmi and Basu combined a modified Max-min algorithm with a VM placement algorithm designed to use less energy. This method was presented to accomplish the twin goals of minimising energy use and shortening the duration of tasks. Cloud Sim was utilised to simulate the experimental outcomes. The study's findings led to the conclusion that the proposed method successfully reduced both energy consumption and response time [49].

Brar and Rao aimed to improve resource utilization and minimize execution time in cloud-based workflows. By analysing previous works in the field, the researchers identified the limitations of existing scheduling techniques. The findings revealed that the Max-Min algorithm effectively optimized workflow scheduling, resulting in improved resource allocation and reduced execution time. The widely used scientific workflow such as Cyber Shake, Montage, and Sipht were employed for processing the incoming requests. The study shed important information on the use of scheduling algorithms in CC, emphasising the Max-Min algorithm's importance in improving workflow management in cloud environments. It was concluded that the Max-Min achieved better results with Cyber Shake in comparison to Data ware [50].

Panda and Jana introduced three task scheduling techniques, MCC, MEMAX, and CMMN, were presented and are intended for heterogeneous multi-cloud systems. The algorithms sought to reduce makespan and maximise average cloud usage. MCC was a single-phase scheduling technique, in contrast to MEMAX and CMMN, which used a two-phase approach. The algorithms were thoroughly validated with an emphasis on makespan and average cloud utilisation metrics using a variety of benchmark and fake datasets**.** The authors conducted tests on two benchmark datasets and one synthetic dataset to see how well the recommended methods worked. The outcomes were contrasted with the multi-cloud task scheduling algorithms RR, CLS, CMMS, and CMAXMS based on their applicability. The comparison showed that, in terms of makespan and average cloud utilisation, the suggested algorithms performed better than the ones already in use [51].

Hosseinimotlagh et al. introduced SEATS, a VM scheduling algorithm, to optimize utilization and energy consumption in a Cloud environment. The algorithm allocated more computing power to VMs on a host, aiming for optimal utilization. A VM scheduling algorithm based on maximising utilisation was additionally suggested to reduce energy usage while maintaining QoS. The study used simulations to show that the suggested algorithms significantly reduced real-time task turnaround times by 94% and overall energy consumption by 60%. Also increasing by 96% was the

acceptance rate of incoming tasks. Overall, the study showed how effective SEATS are at maximising energy consumption and improving task performance [52].

Patel et al. conducted research on a number of work scheduling strategies, including Minimum Execution Time. For static meta-task scheduling, they also recommended modifications to the load-balanced Min-Min (ELBMM) algorithm. The effects of the load-balanced Min-Min algorithm on static Meta Task Scheduling in grid computing were thoroughly examined prior to designing the enhanced method. In order to successfully exploit underutilised resources, the Enhanced Load Balanced Min-Min algorithm (ELBMM), which was based on the Min-Min technique, required task rescheduling. It assigned the assignment to the relevant resources after determining which one would take the longest to finish. The EELBM outperforms other algorithms in terms of resource utilisation and makespan, according to the results [26].

Ismail and Fardoun created an energy-aware tasks scheduling (EATS) approach to divide and schedule massive data in the cloud. EATS's primary objectives were to improve application efficiency and cut back on the energy consumption of the underlying resources**.** Various workloads were used to measure a computing server's power usage. The results of the experiments showed that there was a 1.3 energy consumption ratio between the peak performance and idle state, highlighting the need for efficient resource utilisation without affecting performance. As a result, data centres use less energy a result of cloud providers adopting such strategies [53].

Hemamalini and Srinath evaluated the effectiveness of various task-scheduling algorithms for resource discovery and management in the cloud. The performance analysis was aimed at achieving a balanced minimum execution time for the task scheduling. Different task scheduling algorithms, including Minimum Execution Time, Min-Min, Load Balanced, and Min-Min, were studied in the process with regard to makespan, completion time, execution time, and load balancing. According to the research, the balanced minimum execution time outperformed the other conventional scheduling algorithms covered in the paper [54].

Maheshwari et al. provided a method for multi-site process scheduling that used dynamic probes to estimate feasible network throughput across sites and performance models to forecast resource execution times. They also established the concept of workflow skeletons and the SKOPE framework to examine and analyse the computational and data transportation features of processes. Real-world applications were run in two different computing environments—distributed across various clusters and parallelized across numerous clouds—using the Swift parallel and distributed execution framework. The outcomes showed that their method successfully reduced the overall workflow execution time by up to 60%. When compared to the default scheduling method, the approach also showed improved resource utilisation and shorter execution times [55].

In a cloud computing setting, Jasraj et al. developed a unique meta-heuristic cost-effective genetic algorithm to reduce process execution costs while assuring deadline adherence. They carried out a thorough performance test of the proposed approach utilising well-known scientific workflows of various sizes, including Montage, LIGO, Cyber Shake, and Epigenetics. The novel scheme includes various operations of genetic algorithm involving initialization of the population, encoding, performing mutations and crossovers. By minimizing execution costs while meeting deadlines, the proposed approach strikes a balance between efficient resource utilization and meeting user requirements [56].

Panda and Jana introduced two SLA-based task scheduling algorithms, namely SLA-MCT and SLA-Min-Min, designed for heterogeneous multi-cloud environments. The algorithms were tailored to support three different customer-determined SLA levels. SLA-Min-Min employed a two-phase scheduling strategy, whereas SLA-MCT utilized a single-phase strategy. To evaluate the algorithms' performance, the study conducted simulations using both benchmark and synthetic datasets. Execution-Min-Min and Profit-Min-Min were two two-phase scheduling algorithms that were evaluated alongside SLA-Min-Min. On the other side, SLA-MCT was contrasted with CLS, Execution-MCT, and Profit-MCT, three single-phase task scheduling methods. The examination used four important performance metrics: makespan, average cloud

utilisation, gain, and penalty cost of services. The study's findings showed that in terms of striking a balance between makespan and the gain cost of services, the proposed algorithms consistently beat the alternatives. This indicated that SLA-MCT and SLA-Min-Min have the potential to effectively manage task scheduling while satisfying SLA requirements within heterogeneous multi-cloud environments [57].

Maharana et al. identified the limitations of existing energy-aware scheduling approaches for real-time tasks in cloud environments. These methods relied on determinism and pre-calculated schedule decisions, which weren't appropriate for environments with dynamic execution. The paper took into account a number of energy-efficiency factors, including energy cost, CPU power efficiency, carbon emission rate, and workload, to address these issues. The authors suggested real-time, aperiodic, independent task scheduling strategies that are nearly optimal in terms of energy efficiency for cloud data centres. The proposed EDVS algorithm was designed to provide Quality of Service (QoS) while lowering operational costs. Results from experiments showed that the EDVS algorithm performed better in cloud environments when compared to related algorithms [58].

Madni et al. conducted a study comparing the performance of heuristic algorithms for task scheduling in an IaaS CC environment. Three well-known heuristic algorithms—Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), and Ant Colony Optimisation (ACO)—were examined in the study. The authors also highlighted the trade-off between execution time and resource utilization. The evaluation was performed against various heuristic algorithms including First Come First Serve (FCFS), Minimum Execution Time (MET), minimum Completion Time (MCT) and Max-min, and Min-min algorithms for job scheduling in the cloud. Among all, the MET algorithm demonstrated better performance for optimal job scheduling [59].

Douik et al. addressed the complex task of maximizing network-wide utility by associating users with clouds and scheduling them to specific Processing Zones (PZs) while accommodating practical constraints. These constraints encompassed allowing users to be assigned to one cloud at most, multiple Base Stations (BSs) within a cloud, and multiple distinct PZs within the frame of the BSs. To tackle this challenge, the

study employed graph theory methods and constructed a conflict graph. This graph facilitated the transformation of the scheduling problem into a maximum-weight independent set problem. The authors proposed both heuristic and optimal distributed network algorithms to solve the issue. These distributed algorithms efficiently addressed the maximum-weight independent set problem by leveraging the structure of the conflict graph. Simulation outcomes demonstrated that the hybrid scheduling strategies, combining optimal and heuristic approaches, offered substantial enhancements in comparison to scheduling-level coordinated networks. These improvements were achieved with only a minimal impact on signal-level coordination, showcasing the potential of the proposed strategies to optimize network utility while considering practical constraints [60].

Praveen et al. plan to balance the load among diverse resources of numerous cloud providers . Task scheduling and resource allocation were the two phases of the plan. To allocate resources effectively, the researchers used the social group optimisation algorithm. To schedule tasks, they used the shortest-job-first scheduling method to reduce makespan time and increase throughput. The researchers experimented with synthetic data in a diverse cloud environment to gauge how well their suggested strategy performed. They contrasted the experimental findings with those of first-in, first-out scheduling and a shortest-job-first technique based on genetic algorithms. The outcomes demonstrated the suggested method's validity and usefulness in improving the system's performance, with considerable increases in makespan time and throughput [61].

Duan et al. proposed an incremental Genetic Algorithm (GA) approach for minimizing makespan in task scheduling problems. They incorporated adaptive probabilities of crossover and mutation, adjusting them based on generations and varying them between individuals. The researchers conducted experiments using randomly generated tasks to simulate Cloud-based scheduling scenarios. Their adaptive incremental GA was tested against Min-Min, Max-Min, Simulated Annealing, and Artificial Bee Colony Algorithm. The outcomes demonstrated that

their suggested approach produced workable solutions with a reasonable makespan while needing less computing time [62].

Panda and Jana developed four task-scheduling algorithms for heterogeneous multi-cloud environments. The first two used traditional normalization techniques, while the third introduced distribution scaling and nearest radix scaling. The algorithms were tested using synthetic and benchmark datasets, showing superior performance in makespan and average cloud utilization. However, they failed to consider execution and transfer costs across diverse clouds [63].

Hu et al. present a multi-objective scheduling (MOS) method designed for scientific processes in a multi-cloud context. The algorithm was carefully designed to reduce both the cost and the length of the operation while upholding dependability requirements. Leveraging PSO technology, the algorithm was designed with a coding strategy that accounted for task execution locations and the sequence of data transmission. To assess the algorithm's performance, the researchers conducted extensive simulation experiments using real-world scientific workflow models. The results of the evaluation revealed the superiority of the MOS algorithm. It consistently outperformed both the CMOHEFT and RANDOM algorithms across all multi-objective metrics, underscoring its potential as a highly efficient and effective approach for optimizing task scheduling and resource allocation within multi-cloud environments. [64].

Lin et al. introduced a power efficiency model for cloud servers and proposed the ECOTS algorithm for energy-efficient task scheduling. The algorithm considered task resource requirements, server power efficiency, and performance degradation to minimize energy consumption while maintaining performance levels. Simulation experiments in a heterogeneous cluster environment showed that ECOTS achieved the highest energy efficiency while satisfying task resource requirements. According to various workloads, the suggested power efficiency model provided an appropriate assessment of server power efficiency. Min-Min and Performance First scheduling were exceeded by ECOTS, which led to energy savings of 21.4% and 21.9%, respectively [65].

Jena and Mohanty introduced a Genetic Algorithm-driven strategy for Customer-Conscious Resource Allocation and Task Scheduling in the context of multi-cloud computing. This approach was tailored for a multi-cloud federation scenario. The method was split into two phases: a task scheduling technique that gave priority to the smallest tasks first, and resource allocation controlled by genetic algorithms. The main purpose was to efficiently distribute work among virtual machines (VMs) in the multi-cloud federation in order to reduce makespan time and increase customer satisfaction. To validate their approach, the researchers meticulously conducted experiments using synthetic data and juxtaposed their simulation results with those of pre-existing scheduling algorithms. The outcomes of the simulations substantiated that the proposed algorithm surpassed the performance of existing algorithms, especially in terms of pertinent metrics [66].

Mishra et al. undertook a study concerning energy consumption within the cloud environment. Their research centred around diverse services, with the overarching goal of promoting environmentally conscious practices in cloud computing, commonly referred to as green cloud computing. Their primary focus was on curtailing overall energy consumption in the system, an objective tackled through the resolution of the task allocation predicament prevalent in cloud computing. To realize this objective, they introduced an adaptive task allocation algorithm specially designed to cater to the complexities of heterogeneous cloud environments. The algorithm successfully demonstrated its ability to reduce both makespan and energy usage through simulation in the Cloud SIM environment. The simulation results demonstrated that the suggested algorithm outperformed previous methods and became an energy-efficient solution in the cloud environment [67].

Hazra et al. embarked on an extensive exploration of diverse scheduling algorithms, driven by the overarching goal of curbing energy consumption during the allocation of varying tasks within a cloud setting. Their investigation encompassed a comprehensive assessment of the merits and demerits intrinsic to these pre-existing algorithms. During that period, energy consumption by distinct computing resources held notable significance, with a substantial portion of consumed energy being

directed toward task execution. Task schedulers bore the responsibility of orchestrating the mapping of tasks to their corresponding resources. The pivotal objective was to execute this mapping in a manner that optimized energy efficiency, consequently yielding a significant reduction in the overall energy consumption within cloud systems. The paper furnished a succinct analysis of pre-existing energy-conscious task-scheduling algorithms that were prevalent at the time [68].

Gawali and Shinde provided a heuristic method for allocating resources and scheduling tasks in cloud computing. Their strategy used a number of approaches, such as the modified analytic hierarchy process (MAHP), the longest expected processing time pre-emption (LEPT), divide-and-conquer tactics, and bandwidth-aware divided scheduling (BATS) + BAR optimisation. This combination seeks to improve the effectiveness of resource allocation and work scheduling procedures within the cloud environment. Through experimental comparisons with existing frameworks, their approach showed improved performance in terms of turnaround time and response time. It efficiently allocated resources, including CPU, memory, and bandwidth, leading to enhanced resource utilization. By considering bandwidth as a resource and evaluating system performance based on it, their approach demonstrated its effectiveness in optimizing resource allocation and improving overall system efficiency [69].

According to their article, TangXiaoyong et al. undertook a thorough project to reduce energy usage across cloud data centre servers, network components, and cooling systems. Their approach commenced with the establishment of an energy-efficient cloud data centre ecosystem, encompassing the design of its architecture and the formulation of models pertaining to job assignments and power consumption. To anticipate short-term workload trends within the cloud data centre, they integrated linear regression and wavelet neural network methodologies, giving rise to a predictive technique termed MLWNN. Additionally, they put forth a heuristic solution for energy-efficient job scheduling, leveraging workload predictions. This included a dual approach that included resource management and an online algorithm for energy-efficient task scheduling. The outcomes unveiled the solution's

commendable performance, establishing its efficacy, particularly in the context of cloud data centres with lower workloads [70].

To overcome the drawbacks of work consolidation and scheduling algorithms, Panda and Jana presented the Energy-Efficient Work Scheduling Algorithm (ETSA). The algorithm was developed with a focus on optimizing energy consumption and minimizing makespan in a heterogeneous environment. It incorporated factors such as task completion time and overall resource utilization, utilizing a normalization technique to guide scheduling decisions. To assess its efficacy, the researchers conducted a comprehensive evaluation of the ETSA algorithm. They evaluated its performance to a number of current methods, including MaxUtil, round-robin, random, dynamic cloud list scheduling, and task consolidation with consideration for energy use. The outcomes showed that the suggested ETSA algorithm was more capable than the alternatives currently available in terms of balancing energy efficiency and makespan [71].

Panda et al. introduced allocation-aware task scheduling algorithms tailored for multi-cloud settings. These algorithms extended traditional Min-Min and Max-Min approaches, adapting them to suit multi-cloud requirements. Divided into matching, allocating, and scheduling phases, these algorithms seamlessly integrated into the multi-cloud environment. Simulations using benchmark and synthetic datasets demonstrated their efficacy, with improved metrics such as makespan, cloud utilization, and throughput. Comparative analysis against existing methods affirmed the proposed algorithms' effectiveness in enhancing scheduling efficiency within multi-cloud environments, contributing to optimized resource allocation and improved system performance [72].

Kaur et al. proposed a deep-Q learning-driven heterogeneous earliest finish time scheduling algorithm for cloud-based scientific workflows. They aimed to enhance workflow execution efficiency. The algorithm optimally allocated resources based on the earliest finish time, leveraging Deep-Q learning to make dynamic decisions. The study focused on heterogeneous environments and aimed to improve resource utilization. Through simulation, they demonstrated the algorithm's superiority in terms

of task completion times and resource utilization compared to baseline methods. The research contributed to efficient cloud-based workflow scheduling [73]

Gupta et al. focused on workflow scheduling in cloud computing using the Jaya algorithm. The study aimed to optimize task assignment to minimize makespan and execution costs. In the process, comparisons were made with other nature-inspired algorithms like PSO, GA, ACO, honey bee, and CSO using benchmark functions such as Montage, CyberShake, Inspiral, and Sipht during evaluation. The simulation analysis demonstrated that the Jaya algorithm showed superior performance while converging quickly and producing similar results in less time. Thus, the study contributed to the effectiveness of workflow scheduling, emphasizing its potential in optimizing cloud computing resources [74].

Rehman et al. proposed a Multi-Objective Genetic Algorithm (MOGA) for workflow scheduling in the cloud. The algorithm aimed to optimize makespan while considering budget and deadline constraints, as well as achieve energy efficiency through dynamic voltage frequency scaling. Comparative evaluations were conducted against genetic algorithms focusing on individual objectives and Multi-objective Particle Swarm Optimization (MOPSO) with the same objectives. The results demonstrated that MOGA outperformed the other algorithms in terms of budget, deadline, energy efficiency, and resource utilization. The proposed algorithm showcased significant improvements across diverse objectives, highlighting its effectiveness compared to existing solutions [75].

An extensive survey was carried out by Arunarani et al. with a focus on work scheduling techniques and related metrics that apply in cloud computing settings. The study looked at several scheduling techniques, addressed their drawbacks, and identified important factors to take into account or leave out of a certain system. The poll was divided into three sections based on three perspectives: work scheduling techniques, applications, and parameter-based measurements. By exploring these perspectives, the research aimed to provide insights into the diverse range of scheduling approaches and metrics used in the literature [76].

Zhang et al. proposed a replica-aware task scheduling algorithm and a load-balanced cache placement algorithm to address response delay and optimize data access in multi-cloud environments. The task scheduling algorithm considered node locality and transferred both computation and data, replicating only non-local unassigned and failed map tasks' input data. The cache placement algorithm predicted the next executing task using Bayesian networks and selected cache prefetching files based on caching profit and recycling cost. Extensive experiments demonstrated that the proposed algorithms outperformed benchmark approaches in terms of node locality ratio, job response time, prefetching hit ratio, and execution time-saving ratio [77].

Energy-oriented Flower Pollination Algorithm (E-FPA), developed by Usman et al. is a novel strategy that has been tailored for Virtual Machine (VM) allocation in cloud data centre environments. Their methodology encompassed the creation of a systematic framework that prioritized energy-efficient allocation of diverse VMs onto Physical Machines (PMs). This allocation strategy, known as Dynamic Switching Probability (DSP), was designed to balance global exploration and local exploitation in the search process, ultimately facilitating efficient VM allocation while considering PM's processor, storage, and memory constraints. Throughout the study, real-world workload data were employed in simulations conducted on Multi-Rec Cloud-Sim. The results highlighted the superior performance of E-FPA compared to other techniques. Notably, E-FPA outperformed Genetic Algorithm for Power-Aware (GAPA) by 21.8%, Order of Exchange Migration (OEM) ant colony systems by 21.5%, and First Fit Decreasing (FFD) by 24.9%. This robust performance solidified E-FPA's position as a significant contributor to elevating data centre performance while advancing environmental sustainability through energy-efficient VM allocation [78].

Masdari and Zangakani undertook an extensive survey that furnished a comprehensive panorama of inter-cloud scheduling strategies. The main goal was to efficiently distribute user-generated activities and workflows among appropriate virtual machines scattered over various cloud infrastructures, all while taking into account different goals and factors. The paper systematically categorized scheduling

schemes devised for various inter-cloud environments, delving into their architectural intricacies, prominent attributes, and inherent benefits. Furthermore, the study encompassed a comparative analysis of distinct inter-cloud scheduling approaches, spotlighting the disparities in their features. The culmination of the research entailed concluding observations derived from the findings, as well as the identification of lingering research challenges within the realm of multi-cloud scheduling [79].

Natesan and Chokkalingam presented the Whale Genetic Optimization Algorithm, a hybrid approach combining the WOA and GA to minimize both makespan and cost in task scheduling. Through simulations using the Cloudsim toolkit, the proposed algorithm demonstrated significant improvements in execution time compared to classical WOA and standard GA methods. This innovative technique showcased the potential to provide higher-quality solutions for task scheduling challenges, contributing to the advancement of optimization strategies in this domain [80].

Xu and Buyya addressed the issue of carbon emissions in data centres by proposing a strategy that involves shifting workloads among multi-cloud environments located in different time zones. They developed models to quantify the energy usage, carbon emissions, and solar power availability at different locations. By leveraging this information, they aimed to minimize the reliance on brown energy and maximize the utilization of renewable energy sources. Specifically, they focused on managing the carbon footprint and renewable energy usage in data centres located in Europe, that are situated in different time zones. The results of their study demonstrated that by implementing workload shifting, they were able to reduce carbon emissions by approximately 40% compared to a baseline scenario, while still maintaining an acceptable average response time for user requests [81].

Hosseinzadeh et al. conducted a study in the realm of metaheuristic multi-objective optimization, with a specific focus on multi-objective scheduling strategies within diverse cloud computing settings. Their research encompassed a thorough survey and analysis of these strategies, categorizing them based on the multi-objective optimization algorithms utilized. The study provided insights into how these algorithms have been effectively applied to tackle scheduling challenges.

Furthermore, the research included a comparative analysis of the various multi-objective scheduling schemes. This evaluation not only illuminated the strengths and weaknesses of different approaches but also highlighted directions for future research in this field. Overall, the study contributed to a deeper understanding of the landscape of multi-objective scheduling in cloud computing and provided valuable insights for researchers exploring further advancements in this domain [82].

Sharma and Garg proposed HIGA, a hybrid metaheuristic algorithm, for energy-efficient task scheduling in cloud data centres. HIGA combines genetic algorithm and harmony search to explore optimal regions efficiently. HIGA aimed to improve energy efficiency and performance while minimizing active racks, indirectly reducing cooling energy. Simulations with task applications and real-world scientific tasks showed significant energy savings of up to 33%, a 47% improvement in application performance, and a reduced execution overhead of 39%. HIGA demonstrated effectiveness in enhancing energy efficiency, application performance, and resource utilization in cloud data centres [83].

A 3-layer distributed multi-access edge computing (MEC) network made up of clouds, MEC servers, and edge devices (EDs) was the subject of research by Zhang et al. The network's goal was to effectively provide application-driven computing activities while lowering system latency by using EDs and MEC servers. The authors suggested a distributed system based on multi-agent reinforcement learning to solve resource rivalry amongst cloud centres (CCs). This system allowed CCs to jointly decide on task offloading and resource allocation while taking into account the choices made by other CCs. The suggested technique produced decreased system latency compared to existing approaches, according to simulation findings. The effect of the number of CCs, MEC servers, and EDs on latency performance was also covered in the study [27].

Aziza and Krichen investigated scientific workflow scheduling in cloud computing, aiming to optimize task execution time, reduce computational costs, and adhere to deadlines and budgets. They proposed a hybrid approach that combines a genetic algorithm with the heterogeneous earliest finish time (HEFT) heuristic model. HEFT

is employed to create the initial population for the genetic algorithm. Through simulations using real-world datasets, their approach demonstrated superior performance compared to existing methods, highlighting its efficiency. The proposed hybrid method was applied to various workflow families and was integrated into the WorkflowSim framework using CloudSim, facilitating its practical implementation for cloud workflow scheduling [84].

Bezdan et al. uses an improved flower pollination technique to solve the scheduling problem in cloud computing. The goal of the study was to raise work scheduling in cloud systems' effectiveness and performance. The classic flower pollination algorithm was modified by the authors to improve its exploration and exploitation capabilities. In terms of job execution time, resource utilisation, and overall system performance, the suggested approach showed better results. Through the utilisation of its dynamic selection mechanism and adaptive search behaviour, the algorithm successfully tackled the difficulties associated with cloud computing job scheduling. The study facilitates better resource management and effective work scheduling in cloud environments by optimising cloud resource allocation and task allocation [85].

Sharma and Garg presented a novel method for developing an autonomous task scheduler that uses little energy by using a supervised neural network. The main objective was to reduce the number of active racks in cloud settings, as well as makespan, energy consumption, and execution overhead. Through thorough comparison testing with other current algorithms, simulation experiments were carried out on cloud configurations with and without heavy loads. This research presents a novel strategy that aims to enhance the energy efficiency of task scheduling in cloud environments, with a focus on reducing multiple performance parameters simultaneously. The results demonstrated the superiority of the proposed scheduler. In heavily loaded environments, it achieved a 59% improvement in makespan, a 45% reduction in energy consumption, an 88% decrease in execution overhead, and a 70% reduction in the number of active racks. Similarly, in lightly loaded environments, it showed a 64% improvement in makespan, a 71% reduction in energy consumption, a

43% decrease in execution overhead, and a 70% reduction in the number of active racks [86].

Wen et al. developed an energy-efficient scheduling algorithm for data-driven industrial workflow applications in private SDWAN-connected data centres. The algorithm aimed to minimize the cloud provider's revenue and non-renewable energy usage while considering real-world electricity prices and green energy availability. By optimizing application execution and data transfers, the algorithm effectively prioritized the utilization of green energy. The experimental results demonstrated that the algorithm achieved similar expenditure to the base algorithm for large workflows and significantly increased green energy utilization by nearly 200% for smaller workflows with a marginal increase in electricity cost. Overall, the findings emphasized the algorithm's effectiveness in promoting energy efficiency and revenue management in data-driven industrial workflow scheduling [87].

Mohanraj and Santhosh presented a Multi-Swarm Optimization model designed to elevate Quality of Service (QoS) within a multi-cloud setting. Their method focused on multi-cloud scheduling and outperformed well-known methods like single-objective Particle Swarm optimisation scheduling and adaptive energy-efficient scheduling. Experimental results solidified the effectiveness of the proposed method, substantiating its ability to outperform existing approaches and thereby enhance QoS levels. This research contributes to the advancement of efficient multi-cloud scheduling strategies, emphasizing the improved quality of service achieved through their Multi-Swarm Optimization model [88].

Velliangiri et al. developed the Hybrid Electro Search with a Genetic Algorithm (HESGA) to improve task scheduling in multi-cloud environments. The algorithm combined genetic and electro-search strengths, enhancing critical parameters like makespan, load balancing, resource utilization, and cost efficiency. HESGA consistently outperformed existing scheduling methods, demonstrating the potential of combining optimization techniques [89].

Pirozmand et al. proposed the GAECS based on the Genetic Algorithm for task scheduling, considering energy and time constraints. The job prioritisation and task assignment phases make up the GAECS algorithm. The Energy-Conscious Scheduling Heuristic model is used for job assignment, prioritisation, and generation of main chromosomes. The algorithm selects optimal chromosomes based on time and energy criteria and assigns them to available resources. Eight additional algorithms were compared to the GAECS algorithm's performance. The findings showed that, in terms of makespan and energy usage, the GAECS algorithm beat the comparator algorithms [90].

Ahmad and Alam introduced a List Scheduling with Task Duplication (LSTD) algorithm to optimize the makespan of workflow applications while avoiding a rise in overall time complexity. Comprising three key phases—task rank calculation, selective task duplication for efficiency improvement, and processor assignment through an insertion-based policy—the LSTD algorithm aimed to enhance scheduling outcomes. The research contrasted LSTD against existing algorithms, including HEFT, CPOP, and PEFT, to validate its effectiveness. The paper proved LSTD's superiority over alternative scheduling algorithms through an experimental analysis incorporating Big Data processes like CyberShake, Montage, and LIGO. When comparing different criteria, such as schedule length ratio, percentage of best results, and average running time, LSTD consistently beat its competitors. This research highlighted LSTD's potential to significantly improve workflow scheduling outcomes while maintaining a manageable time complexity, thus contributing to more efficient task allocation and overall enhanced scheduling efficiency [91].

In a research, Renugadevi and Geetha developed a multi-cloud system made up of data centres spread out geographically. Each data center's solar power generation differed according to variables including location, electricity cost, carbon emissions, and carbon tax. The workload allocation algorithm's energy management was heavily influenced by the particular features of the apps that were being evaluated. They proposed an adaptive workload allocation algorithm that took into account the nature of the tasks and was aware of renewable energy availability. Additionally, a migration

policy was included to assess its impact on carbon emission, total energy cost, as well as brown and renewable power consumption [24].

Walia et al. proposed an HS Algorithm HS that ingeniously fuses the Genetic Algorithm (GA) and FPA Algorithm to tackle challenges within cloud environments. The study focused on evaluating the algorithm's performance through a comprehensive set of metrics, including completion time, resource utilization, cost of computation, and energy consumption. To gauge its effectiveness, the HS algorithm was rigorously compared against established scheduling approaches, namely GA and FPA. The results of extensive simulations unequivocally showcased the remarkable superiority of the HS algorithm across these crucial metrics, underscoring its potential to transform cloud scheduling dynamics. Particularly noteworthy was its significant enhancement in resource utilization, along with its impressive reductions in completion time and energy consumption. These findings were consistent in both homogeneous and heterogeneous environments, affirming the versatility and effectiveness of the HS algorithm in optimizing cloud scheduling practices [92].

Pradhan and Satapathy undertook a study to address energy consumption challenges in cloud data centres. They proposed an innovative solution called the EACTS algorithm. This algorithm incorporated conventional heuristics like min-min, max-min, and suffrage while integrating an energy model. It was specifically designed for deployment in a heterogeneous cloud environment. To assess the algorithm's effectiveness, the researchers conducted experiments using a benchmark dataset. The outcomes from the EACTS algorithm demonstrated a well-balanced trade-off between energy efficiency and makespan, showcasing its ability to optimize both aspects. The algorithm achieved this by estimating energy consumption while taking into account makespan and cloud utilization [93].

In the study conducted by Pradhan and Satapathy**,** a scheduling approach grounded in the PSO algorithm was employed to generate a collection of schedules or solutions. These solutions were subjected to evaluation based on QoS parameters, including criteria such as makespan, cloud utilization, and energy consumption. The objective was to identify the most optimal solution that met the desired QoS standards for task

allocation within a heterogeneous multi-cloud environment. The algorithm's efficiency was verified through simulations and testing, employing benchmark datasets. The findings showed that the proposed method outperformed current cloud scheduling techniques such as genetic algorithms, min-min, max-min, CMMS, CMAXMS scheduling algorithms, and others [94].

Jena and Mohanty introduced a genetic algorithm-centered strategy to address task mapping and priority scheduling within a multi-cloud context. This algorithm was structured around two pivotal phases: mapping and scheduling. The researchers subjected the algorithm to comprehensive simulations, employing synthetic data to assess its efficiency within a diverse multi-cloud environment. A comparative analysis was conducted against conventional FIFO mapping and scheduling methods. The outcomes derived from the simulations illuminated the superior performance of the proposed mapping and scheduling algorithm. Notably, it exhibited enhanced system performance concerning metrics such as makespan time and throughput in comparison to the existing FIFO approach. Through these findings, the study effectively affirmed the prowess of the suggested algorithm for optimizing task mapping and scheduling dynamics within a multi-cloud framework [95].

The goal of the task-scheduling algorithm Mangalampalli et al. suggested was to reduce data centre power and energy costs. The method effectively mapped tasks onto appropriate virtual machines (VMs) by modelling the task and VM priorities using the Whale Optimisation algorithm. A multi-objective fitness function that took power cost and energy use into account was assessed. The CloudSim Simulator was used to run simulations and compare the suggested algorithm's performance to that of the PSO and CS algorithms that are already in use. The simulation results demonstrated that the proposed algorithm outperformed the existing algorithms in terms of minimizing energy consumption and power cost [96].

Sangeetha et al. introduced a resource allocation framework designed to achieve optimal resource utilization within a multi-cloud setting. The architecture harnessed GWO in combination with a deep neural network to enhance both service provisioning and scalability. The deep neural network managed routing decisions

based on factors like input data rate and available storage, with the goal of minimizing processing and storage delays. This framework was structured around two primary modules: one for data processing and routing operations, and another control plane leveraging the deep neural network for effective resource allocation. Through simulations on the Java (NetBeans) platform and evaluation using the CloudSim toolkit, the study examined the framework's performance. Notably, it exhibited improvements in data delivery, resource allocation, and storage efficiency within the multi-cloud environment. The experimental results encompassed various performance metrics, including time delays and the cost associated with resource allocation in the context of multi-cloud setups. In summary, the research by Sangeetha et al. offered a comprehensive framework that combined GWO and deep neural networks to optimize resource allocation, resulting in improved efficiency across data delivery, resource management, and storage within the multi-cloud ecosystem [29].

Miglani et al. proposed an innovative approach for task scheduling in cloud environments, addressing reliability and resource allocation challenges. Their scheduler, based on a modified FPA, aimed to improve task scheduling efficiency. Through experiments with various scientific workflows, the study showcased the superiority of their approach over genetic and GWO in terms of reliability and resource utilization. This research highlighted the importance of combining reliability strategies and optimization algorithms for effective cloud workflow scheduling [97].

Chen et al. put forth an innovative approach for conserving energy in job scheduling, with specific consideration for task dependencies within cloud computing. This method meticulously accounted for the diverse characteristics of data centres, embracing their heterogeneous nature. The model also entailed the formulation of energy consumption patterns, grounded in factors such as virtual machine CPU frequency and kernel count. The study further introduced inventive strategies for monitoring energy consumption within cloud computing data centres. The core objective of this method revolved around the segmentation of each job into multiple tasks and their subsequent allocation to virtual machines. Through simulations, the research contrasted the total execution time under varying conditions, specifically

comparing scenarios with and without job segmentation. The outcomes were evaluated using virtual machines tasked with handling either 1000 or 2000 jobs. The findings demonstrated that job segmentation, or "job cutting," consistently yielded improvements in both total execution time and total energy consumption, irrespective of task dependencies. Remarkably, this approach not only effectively reduced energy consumption but also managed to curtail the job discard rate, showcasing its potency in achieving energy efficiency and task optimization within cloud computing environments**.** [98].

Marri and Rajalakshmi introduced a comprehensive approach aimed at augmenting the efficiency of task scheduling in cloud computing through a multi-objective framework. Their model ingeniously combined the GA with an energy-aware component to optimize both energy consumption and makespan values. This innovative hybrid algorithm, referred to as the multi-objective energy-aware genetic algorithm, incorporated considerations for CPU energy consumption within virtual machines and leveraged an energy model based on voltage and frequency distribution. Task interdependencies were captured through a directed acyclic graph representation. Through simulations, the research outcomes underscored the remarkable superiority of the proposed multi-objective model when contrasted with alternative algorithms. This model achieved a notable 5% reduction in makespan compared to the MODPSO algorithm, as well as a 0.7% reduction compared to the HEFT algorithm. Notably, the proposed approach yielded an energy consumption of 125 joules for 50 active virtual machines. These results collectively validate the efficacy of the model in enhancing both task scheduling efficiency and energy conservation within cloud computing environments. [99].

Kruekaew and Kimpan presented an innovative solution to the problem of load balancing in cloud computing setting using a hybrid artificial bee colony algorithm with reinforcement learning for multi-objective job scheduling optimization. This novel approach seeks to optimize resource allocation and distribution among cloud servers. The researchers want to improve system performance and efficiency by combining these two optimization strategies. The study emphasized the importance of

balancing computing workloads to enhance resource utilization while minimizing latency. The findings contribute to ongoing attempts to improve cloud computing technology for better service delivery. This paper emphasizes the need of using intelligent algorithms to solve complicated optimization problems in cloud environments, opening up new paths for future research and development [100].

Saif et al. introduced the Multi-Objectives Grey Wolf Optimizer (MGWO) algorithm as a solution to minimize both delay and energy consumption within fog brokerage for task distribution. The researchers conducted simulations to conduct a performance comparison between MGWO and other algorithms, focusing on their effectiveness in reducing delay and energy consumption. The results demonstrated that MGWO surpassed the alternative algorithms by achieving notably lower delay and energy consumption levels. Notably, the algorithm exhibited stability and linear scalability as workloads increased, underscoring its capacity to manage substantial request loads from IoT devices. The study provided robust evidence of MGWO's effectiveness in curtailing delay and energy consumption, positioning it as a superior alternative to existing algorithms**.** [101].

Malathi and Priyadarsini proposed a load balancer algorithm for cloud computing using heuristic techniques. They made two key contributions: a hybrid technique that improved load balancing applicability and a genetic algorithm modified with Lion Optimizer's global search criteria. The hybrid methodology exhibited remarkable effectiveness in aspects like maximum turnaround time and the efficient utilization of virtual machine resources. The integration of the Lion Optimizer contributed to optimal parameter choices, and the incorporation of fitness criteria that took into account task and virtual machine attributes further enriched the selection mechanism. Empirical outcomes substantiated the proficiency of the hybrid approach combining the Lion Optimizer with a genetic algorithm, affirming the viability and success of their proposed strategy [102].

Mahilraj et al. presented a novel application of machine learning, specifically the LSTM technique, to address the escalating concerns of carbon emissions and energy usage through efficient power task scheduling. The proposed scheduling approach

took into account various factors including task completion time, exclusive resource utilization, and standardization processes. To enhance the efficacy of LSTM, the researchers incorporated the Novel Black Window (NBW) approach to reduce the model's weight. The effectiveness of the LSTM-NBW model was evaluated using simulated analysis across a range of parameters, including makespan, power consumption, job completion time, and resource utilisation. According to the study's findings, compared to the original LSTM model, the recommended model only required an extra 400KWh for an 80kB user workload [103].

Jaiprakash et al. proposed an innovative method aimed at enhancing the energy efficiency of workflow scheduling within cloud computing. This was achieved through the utilization of the MaxUtil model in conjunction with the FPA. The primary goals of this approach encompassed the reduction of both energy consumption and workflow processing duration. The core methodology revolved around the allocation of tasks to VMs and the subsequent scheduling of these tasks, all of which were guided by optimal criteria. The study was the first to focus on optimizing energy consumption and makespan using FPA. The proposed algorithm demonstrated advantages in convergence speed and feasible solutions. Extensive studies on scientific workflows from different fields showed that it outperformed traditional scheduling algorithms based on PSO, GSA, and GA. Based on the ANOVA analysis, the study concluded that the suggested algorithm exhibited superior performance compared to the existing methods Simulation results consistently supported the superiority of the proposed algorithm, and statistical analysis confirmed its effectiveness compared to existing methods [104].

Boopalan and Goswami proposed a cluster selection approach and a virtual machine (FFD) algorithm, called dynamic PUE genetic algorithm (CRA-DP-GA), to deploy virtual machines (VMs) in data centres with reduced energy consumption while maintaining performance. Their approach included a host selection algorithm for load balancing and dynamic adjustment of cooling load. By supporting VM clustering based on workload and bandwidth requirements, their solution improved efficiency and availability. The proposed approach achieved reduced task execution and

assignment times compared to previous techniques, contributing to enhanced energy efficiency and performance in data centres [105].

Cai et al. introduced an encompassing multi-cloud distributed scheduling model encompassing six distinct objective functions, tailored to cater to both user and cloud provider interests. Their approach entailed the proposition of an intelligent algorithm rooted in the sine function, adeptly equipped to address intricate scheduling quandaries. The algorithm's efficacy was scrutinized through evaluation against the DTLZ test suite, in conjunction with several comparison algorithms concurrently employed within the scheduling model. Through an assessment across six objectives, the algorithm and model underwent meticulous analysis concerning the distribution of individuals within the resultant population. The empirical findings underscored exceptional scheduling efficiency, translating into heightened security measures. This research presented an innovative perspective on tackling the intricate data processing intricacies intrinsic to the realm of IoT, introducing a fresh approach that promises to address pertinent challenges [106]. The comparative analysis of existing scheduling Approaches highlighting their findings and shortcomings are discussed in Table 2.1.

Table 2.1 Comparative Analysis of Existing Scheduling Approaches

Authors	Proposed/ Objective	Technologies used	Evaluated Parameters	Findings	Limitations and Future Scope	Publisher
Dhanalakshmi and Basu (2014) [49]	Energy-aware task scheduling	Modified Max-min algorithm and VM placement algorithm	Energy consumption, makespan and minimum execution time	Implementation of the proposed work lowered the energy consumption by 8%-10% and makespan	In some problem sets makespan is high that will form the basis of future research.	International Journal of Engineering Research and Technology

				by 1%- 2%.		
Brar and Rao (2015) [50]	Optimized Workflow scheduling	Max-Min algorithm	Workflows (Cyber shake, Montage, Sipht), minimum execution time	The Max-Min achieved better results with Cyber Shake in comparison to Data ware improved resource utilization and minimised execution time.	Analysis for cost and energy was missing that will be a part of future research	International Journal of Computer Applications
Panda & Jana (2015) [51]	Efficient task scheduling	MCC, MEMAX, and CMMN algorithm	Increase average cloud utilisation and minimise makespan	Results show that the makespan is minimised and average cloud utilization is increased	Only makespan and cloud utilization were considered, and will be evaluated over more parameters in future research.	The Journal of Supercomputing
Patel et al.	static Meta	Minimum	Resource	EELBM	The average	Procedia

(2015) [26]	task scheduling	Execution Time and Modified ELBMM Algorithm	utilization and makespan	achieves makespan of 84.3s.	response time for smaller tasks was increased and thus a comprehensive research in required in future.	Computer Science
Ismail & Fardoun (2016) [53]	Energy-aware task scheduling (EATS)	Non-linear programming model	Energy consumption, CPU Utilization	Experiments showed that there was a 1.3 energy consumption ratio between the peak performance and idle state.	EATS missed utilization in cloud computing environment and should be utilized in future to evaluate its performance.	Procedia Computer Science
Hemamalini and Srinath (2016) [54]	Efficient Grid task scheduling	Min-Min, load-balanced, and minimum execution time algorithms	Makespan, completion time, execution time and load balancing.	The balanced minimum execution time outcasted the other traditional scheduling	Work lacks in hybrid algorithm that should have been developed to combine all four algorithms.	International Journal of Communication and Networking System

				algorithms	This will also be a part of future research.	
Jasraj et al. (2016) [56]	Meta-heuristic optimization Approach workflow scheduling	CEGA	Workflows (Montage, LIGO, Cyber Shake, and Epigenetics).	For workflows with hard and soft deadlines, the suggested CEGA algorithm was registered at 83.5% and 62%, respectively	Implementing termination delays in future is a good idea because they have a negative impact on the workflow's overall execution costs.	IEEE
Panda and Jana (2017) [57]	Heterogeneous multi-cloud task scheduling	Algorithms SLA-MCT and SLA-Min-Min	Makespan, typical cloud usage, gain, and service cost penalties.	The algorithms outperformed the other algorithms in terms of makespan, penalty and gain of services	Further work aimed to create more efficient algorithms suitable for the heterogeneous multi-cloud environment	The Journal of Supercomputing

Madni et al. (2017) [59]	Heuristic methods for task scheduling	Genetic Algorithm (GA), Ant Colony Optimisation (ACO), and Particle Swarm Optimisation (PSO).	FCFS (First Come, First Served), MET (Minimum Execution Time), MCT (Minimum Completion Time), and Mmax.	MET algorithm demonstrated better performance for the optimal job scheduling	The trade-off between execution speed and resource use is more complex that will be a part of future work.	PLOS ONE
Lin et al. (2018) [65]	Power efficiency model for energy-efficient task scheduling.	ECOTS algorithm	Energy efficiency, energy consumption	ECOTS algorithm resulted in energy savings ranging from 21.4% to 21.9%.	A comprehensive model will be implemented that considers various server components to improve and evaluate power efficiency.	Sustainable computing: informatics and systems

Gawali et al. (2018) [69]	Heuristic method for scheduling tasks	Divide-and-Conquer strategies, the Modified Analytic Hierarchy Process (MAHP), BATS + BAR optimisation, and LEPT.	CPU Efficiency, Processing Time, and Response Time	The heuristic approach achieved a significantly lower TAT value of 2033.72 ms while exhibiting a lower RT value of 3.7 ms.	Future research will concentrate on developing more efficient scheduling algorithms.	Journal of Cloud Computing
Gupta et al. (2019) [74]	Workflow scheduling using the Jaya algorithm	Ant colony optimisation (ACO), genetic algorithms (GA), honey bees, cat swarms, particle swarm optimisation (PSO), and ACO optimization (CSO)	Workflows (Montage, CyberShake, Inspiral, and Sipht)	Jaya algorithm showed superior performance while converging quickly and producing similar results in less time	Some objectives like meeting deadlines, optimizing energy consumption, and achieving load balancing can also be considered.	Concurrency and computation: practice and experience

Usman et al. (2019) [78]	Energy-oriented Flower Pollination Algorithm (E-FPA)	Dynamic Switching Probability (DSP).	Memory Constraints, Energy-oriented Allocation	The proposed work outperformed GAPA, OEM, and FFD energy by 21.8%, 21.5%, and 24.9% respectively	To improve the present work resources of the data centre will be consolidated using a multi-objective E-FPA strategy.	Journal of Bionic Engineering
Natesan and Chokkalingam (2020) [80]	Hybridised multi-objective task scheduling algorithm	Genetic algorithms (GA) and whale optimisation algorithms (WOA)	Makespan, cost and enactment amelioration rate (EAR)	Makespan for WGOA = 1434.50s against 100 VMs and 500 tasks with some reduction in cost and EAR.	Further work needs to be evaluated for energy consumption, security, and reliability.	Wireless Personal Communications
Aziza and Krichen (2020) [84]	Scientific workflow scheduling	Genetic algorithm and the heterogeneous earliest finish time (HEFT).	Optimizing task execution time, reducing computational costs	The outcomes show how effective the suggested method is, making it potentially	Work needs to be evaluated on the issue of power consumption in data centres while	Neural Computing and Applications

				appropriate for Cyber Shake, LIGO, Epigenomics, Montage, and SIPHT.	planning workflows in cloud environments and will be a part of future scope.	
Bezdan et al. (2021) [85]	Efficient task scheduling	Enhanced Flower pollination algorithm	Execution time, resource utilization, and overall system performance	The algorithm demonstrated superior results in terms of resource management and efficient task scheduling in cloud environments	Other parameters will be included in future work as present work mainly focused on makespan	Proceedings of ICTIS, Springer Singapore
Ahmad and Alam (2021) [91]	List Scheduling with Task Duplication (LSTD) algorithm	HEFT, CPOP, and PEFT algorithm.	workflows (CyberShake, Montage, and LIGO)	The investigation of the Cyber Shake, Montage, and LIGO Big Data workflows	Following parameters need to be considered: acquisition delays, VM's heterogeneity, performance	Concurrency and Computation: Practice and Experience

				showed that LSTD performed better than alternative scheduling strategies.	variation, and deadline for improvement in future work.	
Miglani et al. (2022) [97]	Multi-objective reliability-based workflow scheduler.	Modified Flower Pollination Algorithm	Workflows (Genome, LIGO, Montage and Cyber shake).	Compared to other algorithms, the improved Flower Pollination Algorithm increases the dependability of task scheduling. and resource utilization.	Reliability parameters must be included. Future study will involve not only considering the reliability value but also minimizing the associated cost.	Concurrency and Computation: Practice and Experience
Kruekaew and Kimpan (2022) [100]	Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud	Artificial Bee Colony Algorithm, Reinforcement Learning	Task scheduling optimization, Load balancing	Improved resource allocation and distribution, increased system performanc	Limited evaluation scope, possible scalability issues, Future research into	IEEE Access

	Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning			e and efficiency	real-world deployment scenarios	
Jaiprakash et al. (2023) [104]	Energy-efficient workflow scheduling and MaxUtil model	Flower Pollination Algorithm (FPA)	Energy consumption and makespan	The analysis confirmed its effectiveness in terms of makespan and energy consumption.	Future research will work to develop a strategy for container migration that aims to reduce energy consumption while maintaining service quality at an optimal level.	Authorea

Boopalan and Goswami (2023) [105]	Energy-efficient Virtual Machine Allocation	Dynamic PUE genetic algorithm (CRA-DP-GA)	Energy consumption, efficiency and availability	Task assignment time and execution time were reduced while consuming 60% of power.	Service processing latency needs to be further modified in future research.	Mathematical Statistician and Engineering Applications

2.2 Review Summary

The literature surveys have explored various task scheduling algorithms to tackle energy efficiency and workflow optimization challenges. Energy-aware task scheduling approaches, such as the Modified Max-min and VM placement algorithms, focus on reducing energy consumption while minimizing makespan and execution time. The Max-Min algorithm is commonly used for optimized workflow scheduling, aiming to minimize execution time for workflows like CyberShake, Montage, and Sight. Efficient task scheduling algorithms like MCC, MEMAX, and CMMN aim to increase average cloud utilization and minimize makespan, improving overall system performance. Static meta task scheduling techniques, including Minimum Execution Time and Modified ELBMM algorithms, optimize resource utilization and minimize makespan. Energy-aware task scheduling (EATS) approaches, such as non-linear programming models, consider minimizing energy consumption and maximizing CPU utilization. Effective task scheduling algorithms aim to reduce makespan, completion time, execution time, carbon emission, and power usage and achieve load balancing. Overall, it has been observed that the existing studies have evaluated their studies for a number of parameters as illustrated using Table 2.2. Most of the studies have utilized a combination of 2 to 3 parameters to demonstrate the effectiveness of their work. However, none of the studies have addressed energy, cost and carbon

emission together in a single study. This fact has inspired the researcher to present an efficient scheduling framework that would evaluate these three parameters together.

Table 2.2 Comparative Analysis of Evaluation Parameters

Authors and Citation	Energy Consumption	Power Usage	Resource Utilization/ Scheduling	Load Balancing	Makespan	Carbon Emission	Memory and Storage	Service Quality	CPU Utilization	Execution/ Completion Time	Cost	Response Time
Patel et al., 2015 [26]	-	-	✓	-	✓	-	-	-	-	-	-	-
Sangeetha et al., 2022 [29]	-	-	✓	-	-	-	-	-	-	✓	✓	-
Dhanalakshmi & Basu, 2014 [49]	✓	-	-	-	-	-	-	-	-	-	-	✓
Singh Brar & Rao, 2015 [50]	-	-	✓	-	-	-	-	-	-	-	-	✓
Panda & Jana, 2015 [51]	-	-	✓	-	✓	-	-	-	-	-	-	-
Hosseinimotlagh et al., 2015 [52]	✓	-	-	-	-	-	-	-	-	✓	-	-
Ismail & Fardoun, 2016 [53]	-	✓	✓	-	-	-	-	-	-	-	-	-
Hemamalini & Srinath, 2016 [54]	-	-	-	✓	✓	-	-	-	-	✓	-	-
Maheshwari et al., 2016 [55]	-	-	✓	-	-	-	-	-	-	✓	-	-
Jasraj et al., 2016 [56]	-	-	✓	-	-	-	-	-	-	-	✓	-
Panda & Jana, 2017 [57]	-	-	✓	-	✓	-	-	-	-	-	✓	-
Madni et al., 2017 [59]	-	-	✓	-	-	-	-	-	-	✓	-	-
Praveen et al., 2018 [61]	-	-	✓	-	✓	-	-	✓	-	✓	-	-
Duan et al., 2018 [62]	-	-	-	-	✓	-	-	-	-	✓	-	-

Panda & Jana, 2018 [63]	-	-	✓	-	✓	-	-	-	-	-	-	-
Lin et al., 2018 [65]	✓	-	✓	-	-	-	-	-	-	-	-	-
Jena & Mohanty, 2018 [66]	-	-	✓	-	✓	-	-	-	-	-	-	-
Mishra et al., 2018 [67]	✓	-	-	-	-	-	-	-	-	-	-	-
Gawali & Shinde, 2018 [69]	-	-	-	-	✓	-	-	-	✓	✓	-	✓
Gupta et al., 2019 [74]	-	-	-	-	✓	-	-	-	-	-	✓	-
Usman et al., 2019 [78]	✓	-	-	-	-	-	✓	-	-	-	-	-
Natesan & Chokkalingam, 2020 [80]	-	-	-	-	✓	-	-	-	-	-	✓	-
Xu & Buyya, 2020 [81]	✓	-	-	✓	-	✓	-	-	-	-	-	✓
Aziza & Krichen, 2020 [84]	-	-	-	-	-	-	-	-	-	✓	✓	-
Bezdan et al., 2021 [85]	-	-	✓	-	-	-	-	-	-	✓	-	-
Velliangiri et al., 2021 [89]	-	-	✓	✓	✓	-	-	-	-	-	✓	-
Ahmad & Alam, 2021 [91]	-	-	✓	-	-	-	-	-	-	-	-	-
Walia et al., 2021 [92]	✓	-	✓	-	-	-	-	-	-	✓	✓	-
Mangalampalli et al., 2022 [96]	✓	✓	-	-	-	-	-	-	-	-	-	-
Mahilraj et al., 2023 [103]	✓	✓	✓	-	✓	✓	-	-	-	✓	-	-
Jaiprakash et al., 2023 [104]	✓	-	-	-	✓	-	-	-	-	-	-	-
Boopalan & Goswami, 2023 [105]	✓	✓	-	-	-	-	-	-	-	-	-	-

Different goals have been considered in the research through the use of heterogeneous multi-cloud task scheduling algorithms such as SLA-MCT and SLA-Min-Min. These consist of gains, penalty costs, makespan, and average cloud utilisation. Heuristic algorithms like GA, PSO, and ACO, along with scheduling algorithms like FCFS, MET, MCT, and Max-min, are commonly used for task scheduling. Power efficiency models like ECOTS aim to improve energy efficiency and reduce consumption. Workflow scheduling algorithms such as Jaya, WOA, GA, and others optimize makespan, cost, and EAR for workflows like Montage, CyberShake, Inspiral, and

Sight. Scientific workflow scheduling techniques like GA and HEFT optimize task execution time and computational costs. Enhanced Flower Pollination Algorithm and LSTD improve execution time, resource utilization, and system performance. Multi-objective reliability-based workflow schedulers using the Modified Flower Pollination Algorithm consider reliability and performance for workflows like Genome, LIGO, Montage, and CyberShake. Energy-efficient workflow scheduling and the MaxUtil model leverage FPA to reduce energy consumption and makespan. Dynamic PUE Genetic Algorithm ensures energy-efficient VM allocation.

Overall, these diverse task scheduling algorithms, including the Flower Pollination Algorithm for CyberShake workflow and other related approaches, provide valuable solutions for energy efficiency, makespan reduction, minimum execution time optimization, resource utilization, and workflow optimization in various computing environments.

2.3 Research Gaps

There are a number of interlinked research gaps observed during the literature survey.

1. Innovative Collaborative Approaches:

   Limited research on the integration of natural computing algorithms like Flower Pollination with reinforcement learning techniques e.g. Q-learning in multi-cloud environments.

   A gap in evaluating the synergy between different algorithms and learning techniques to optimize multi-cloud operations.
2. Multi-Objective Optimization:

   Lack of comprehensive strategies that concurrently optimize various objectives like cost, energy, and CO2 emissions in heterogeneous multi-cloud environments.

   Need for a robust framework that can balance and optimize multiple conflicting objectives in real-world applications.
3. Adaptability and Scalability:

   Limited studies focus on the adaptability and scalability of the integrated approaches in diverse and dynamic multi-cloud environments.

Exploration is required in how such collaborative approaches can adapt to evolving workloads, technologies, and cloud infrastructures.

4. Energy and Environmental Sustainability:
   Insufficient research on how integrated approaches can contribute to long-term energy efficiency and environmental sustainability in multi-cloud computing. Exploration is needed to uncover innovative solutions that can minimize ecological impacts while maximizing operational efficiency.

Addressing these research gaps can lead to the development of more refined, scalable, and adaptable solutions leveraging integrated approaches like Flower Pollination with Q-learning, to optimize the operations in heterogeneous multi-cloud environments, ensuring ecological sustainability, and operational efficiency.

2.4 Research Objectives

The objectives of the study are as follows.

1. To enhance the efficiency of heterogeneous task scheduling in cloud computing using minimum execution time in Cyber Shake Seismogram workflow.
2. To propose an optimized resource management model using the Enhanced Flower Pollination algorithm in a heterogeneous environment.
3. To implement the proposed optimized model for multi-cloud computing in a heterogeneous environment.
4. To validate the proposed model.

2.5 Summary of the Chapter

The chapter presents a detailed literature survey for the scheduling-based research work involving several parameters. The review mainly comprises the studies that have taken advantage of the minimum execution time algorithm, cyber shake and flower pollination algorithm in addition to related techniques. The observed research gaps are discussed in the later part of the chapter followed by the research objectives of the study.

CHAPTER 3: ENHANCEMENT IN MET FOR CYBER SHAKE SEISMOGRAM

The chapter is designed to address the first objective of the study and presents the integration of the MET algorithm in the Cyber shake seismogram workflow. Initially, the Cyber shake seismogram and MET algorithm are discussed followed by the proposed work algorithm aimed at the enhancement of MET.

3.1 Cyber Shake Seismogram Work Flow

In cloud computing, where massive volumes of data are processed and examined, effective task scheduling is essential to maximise resource use and enhance system efficiency. Effective job scheduling is beneficial to many processes, but the Cyber Shake Seismogram (CSS) workflow is particularly important and demanding. This sophisticated workflow is employed in seismology to simulate seismic events, generating valuable insights for earthquake prediction and hazard assessment. The intricate nature of the CSS workflow necessitates careful attention to scheduling heterogeneous tasks in order to minimize execution time and achieve optimal results. The CSS workflow encompasses a series of complex computational tasks that simulate the behaviour of seismic waves based on a given seismic event scenario. These simulations involve intricate algorithms and sophisticated modelling techniques that require significant computational resources and data processing capabilities. The workflow typically involves multiple tasks with varying degrees of complexity and interdependencies, making efficient scheduling a challenging endeavour [107].

The significance of effective task scheduling in the CSS workflow cannot be overstated. By optimizing the allocation of computational resources and minimizing execution time, researchers and seismologists can expedite the generation of seismograms, leading to faster and more accurate earthquake predictions and hazard assessments. This has a direct impact on public safety, allowing authorities to make

timely decisions and implement preventive measures to mitigate the potential impact of seismic events.

Moreover, effective task scheduling in the CSS process aids in resource optimisation in cloud computing settings, where computational resources are shared by several users and applications. By intelligently allocating resources based on task requirements and priorities, the overall system performance and throughput can be enhanced, enabling the simultaneous execution of multiple workflows while meeting quality-of-service objectives [108].

Efficient scheduling in the CSS workflow also holds economic significance. By reducing the execution time and resource consumption, cloud service providers can offer cost-effective solutions to their clients. This makes cloud computing more accessible and affordable for organizations and researchers involved in seismology, enabling them to leverage the power of high-performance computing resources without incurring exorbitant costs. To address the challenges and exploit the opportunities presented by the CSS, researchers and practitioners have developed and employed various scheduling algorithms and optimization techniques. These algorithms aim to intelligently allocate computational resources, minimize communication overhead, and prioritize tasks based on their dependencies and computational requirements. Additionally, advances in workflow management systems have facilitated the design and execution of the Cyber Shake Seismogram workflow, enabling seamless integration of tasks, efficient data transfer, and fault tolerance mechanisms [109].

3.2 Minimum Execution Time and its Correlation with CSS Workflow

Effective task-scheduling algorithms are critical to optimising resource utilisation and attaining high-performance results in the cloud computing area. The Minimum Execution Time (MET) algorithm has become a well-liked method for efficient job scheduling among the many algorithms. This algorithm focuses on minimizing the execution time of tasks by intelligently allocating computational resources in cloud environments [59]. The MET algorithm holds great significance in the context of the CSS workflow, where precise and efficient task scheduling is crucial for accurate

earthquake prediction and hazard assessment. The CSS workflow is a sophisticated application employed in seismology to simulate and analyse seismic events. It involves complex computational tasks that generate seismograms based on given seismic event scenarios. These simulations utilize intricate algorithms and advanced modelling techniques, demanding significant computational resources and efficient task scheduling to achieve accurate and timely results.

The correlation between the MET algorithm and the CSS workflow lies in the shared goal of optimizing task execution time. The MET algorithm accomplishes this by assigning tasks to suitable computational resources based on their execution time requirements. By identifying resources with the shortest expected execution time, the algorithm aims to expedite the completion of tasks, thereby reducing the overall workflow execution time [110]. In the context of the CSS workflow, the MET algorithm can have a profound impact on the efficiency and accuracy of seismic simulations. By minimizing execution time, the algorithm enables faster generation of seismograms, leading to more timely and precise earthquake predictions. This, in turn, facilitates prompt decision-making by authorities and enhances public safety by enabling proactive measures to mitigate potential risks. Furthermore, the MET algorithm contributes to resource optimization in cloud computing environments. By intelligently allocating tasks to resources with shorter execution times, it ensures efficient utilization of computational resources. This optimization enhances the overall system performance, enabling simultaneous execution of multiple workflows and reducing resource wastage. As a result, cloud service providers can deliver cost-effective solutions to organizations involved in seismology, making high-performance computing resources more accessible and affordable.

Implementing the MET algorithm in the context of the CSS workflow requires a comprehensive understanding of task characteristics, resource capabilities, and execution time estimations. Researchers and practitioners have developed various strategies to leverage the MET algorithm effectively. These include techniques such as task profiling, resource monitoring, and dynamic resource provisioning to adapt to changing workload demands. The relationship between the MET algorithm and the

CSS workflow demonstrates the importance of efficient task scheduling in achieving accurate and timely results. By employing the MET algorithm, seismologists and researchers can optimize the execution time of computational tasks, leading to faster seismic simulations and more reliable seismograms. This contributes to improved earthquake prediction, and hazard assessment, and ultimately enhances public safety [87, 111]. Algorithmically, the MET can be represented as follows.

Algorithm MET Scheduling

1. Initialize the MET scheduling algorithm:
    - Prepare a list of tasks to be scheduled.
    - Create a list of available computational resources.
    - Set the current time to 0.
2. Determine the estimated execution time for each task:
    - For each task in the list:
        - Estimate the execution time based on historical data, task characteristics, and resource capabilities.
        - Store the estimated execution time for the task.
3. Sort the tasks in ascending order of estimated execution time:
    - Arrange the tasks in the list based on their estimated execution times , from the shortest to the longest.
4. Schedule the tasks to the available resources:
    - While there are tasks remaining in the list:
        - Select the task with the shortest estimated execution time.
        - Choose an available computational resource with sufficient capacity to execute the task.
        - Assign the task to the selected resource.
        - Update the resource′s availability and capacity based on the task′s requirements.
        - Set the start time for the task to the current time.
        - Calculate the task′s completion time by adding the estimated execution time to the start time.

- Remove the task from the list of tasks.
- Update the current time to the completion time of the scheduled task.

5. Repeat step 4 until all tasks have been scheduled.
6. Evaluate the performance of the MET scheduling:
   - Calculate the total execution time of all scheduled tasks.
   - Calculate the make span, which is the difference between the earliest start time and the latest completion time among all tasks.
   - Assess the resource utilization by calculating the average utilization of the allocated resources.
7. Output the scheduling results:
   - Display the scheduling sequence, including the assigned resources and corresponding start and completion times for each task.
   - Provide the total execution time, makespan, and resource utilization metrics as the output of the MET scheduling algorithm.
8. End the algorithm.

The proposed work has established a location-aware and resource-aware MET algorithm in order to enhance the current work scenario of MET in order to get more efficient results. The work can be represented as shown in Figure 3.1 as follows.

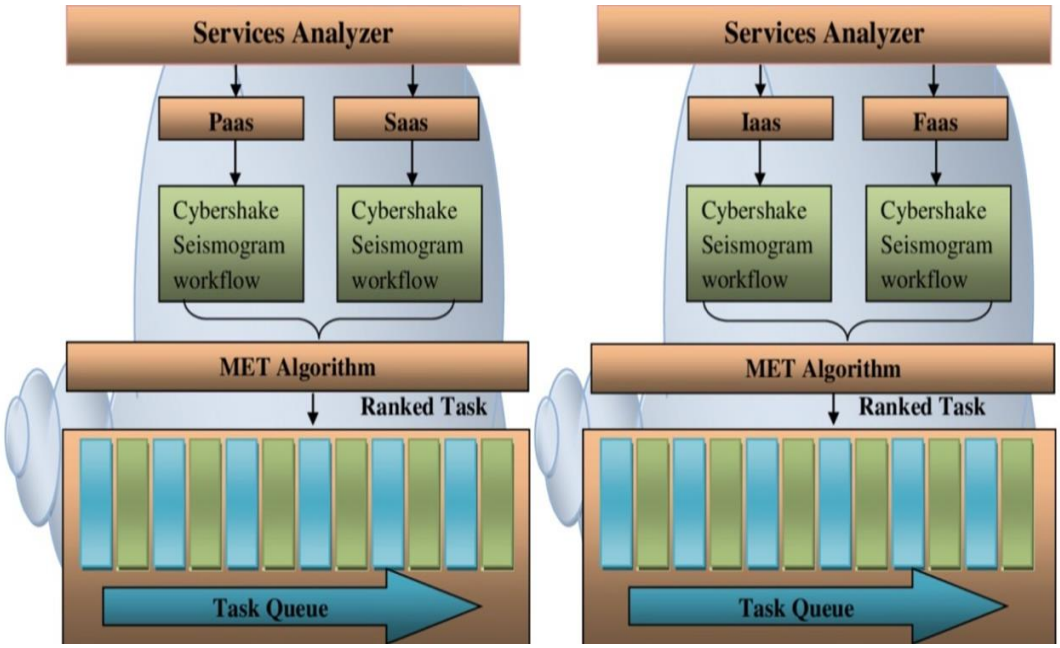

Figure 3.1 The Integrated CSS Workflow with MET Algorithm [88]

3.3 Proposed Work Algorithm for the Enhancement

The MET scheduling algorithm has been a crucial approach in optimizing task scheduling in cloud computing environments. It focuses on minimizing task execution time by intelligently allocating computational resources. However, advancements in MET have led to the development of an improved approach that incorporates the division of the entire region into clusters and utilizes location and resource awareness. This improved algorithm takes into account the users' geographical proximity, connecting them through hosts within their respective regions. Additionally, it introduces a location and resource-aware MET algorithm that incorporates the concept of the Modified Best Fit Decreasing (MBFD) algorithm. This innovation enhances the efficiency and effectiveness of task scheduling, leading to optimized resource utilization and improved performance [112].

The improvement in MET begins with the division of the entire region into clusters. This division is based on geographical proximity, aiming to group users within the same geographic area. By creating these clusters, a closer connection between users and their assigned hosts is established, minimizing network latency and improving communication efficiency. Users within each cluster can then interact and share resources more effectively, fostering collaboration and reducing communication overhead. To further enhance the MET algorithm, a location and resource-aware approach is introduced. This improvement recognizes that the geographical location of users and available resources can significantly impact task scheduling efficiency. By considering location awareness, the algorithm aims to assign tasks to resources that are in close proximity to the user, reducing network delays and enhancing overall performance [113]. The resource-aware aspect of the improved MET algorithm takes into account the availability and capacity of resources within each cluster. It ensures that tasks are allocated to resources that can handle the workload efficiently, minimizing resource contention and maximizing resource utilization. This consideration optimizes the execution time and overall system performance.

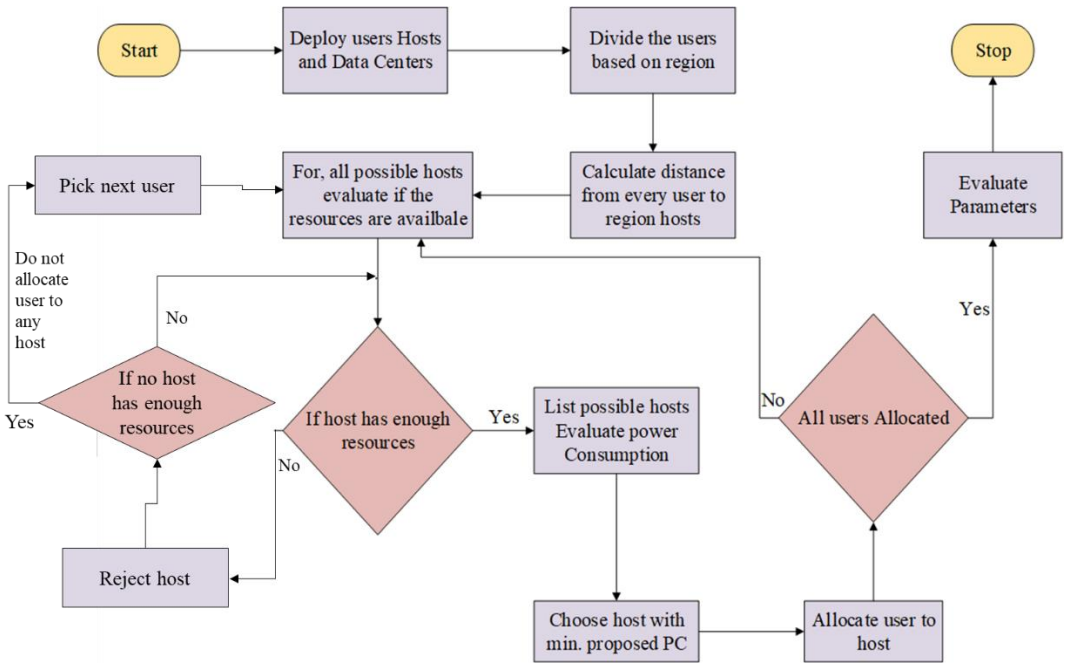

Figure 3.2 The Proposed Framework for Improved MET

To implement the proposed location and resource-aware improved MET algorithm, the concept of the Modified Best Fit Decreasing (MBFD) algorithm is incorporated. The MBFD algorithm is a well-known optimization technique that selects the best-suited resource for a task by considering both the resource's availability and its capability to handle the task's requirements [114–116]. By utilizing the MBFD algorithm within the improved MET framework, the algorithm can intelligently assign tasks to the most suitable resources, considering both their proximity to the user and their capability to execute the task effectively. The proposed work for improved MET can be illustrated using Figure 3.2.

Algorithm Steps: Location and Resource-Aware Improved MET Algorithm utilizing MBFD

---

1. Divide the entire region into clusters based on geographical proximity.
2. For each user in a cluster: a. Connect the user to a host within the same cluster.
   b. Establish a low − latency connection to minimize network delays.
3. Estimate the execution time for each task based on historical data,
   task characteristics, and resource capabilities.

4. Sort the tasks in descending order of estimated execution time.
5. For each task in the sorted list:
6. a. Determine the cluster of the user associated with the task.
   b. Identify the resources within the cluster.
   c. Apply the MBFD algorithm to select the most suitable resource based on availability and capability.
   d. Assign the task to the selected resource.
   e. Update the resources availability and capacity.
   f. Set the start time for the task.
7. Evaluate the performance of the location and resource – aware improved MET algorithm:
   a. Calculate the total execution time of all scheduled tasks.
   b. Measure the makespan, which is the difference between the earliest start time and the latest completion time among all tasks.
   c. Assess the resource utilization by calculating the average utilization of the allocated resources.
8. Output the scheduling results:
   a. Display the scheduling sequence, including the assigned resources and corresponding start and completion times for each task.
   b. Provide the total execution time, makespan, and resource utilization metrics as the output of the improved MET algorithm.
9. End the algorithm.

By incorporating the division of the region into clusters, establishing connections within each cluster, and introducing location and resource awareness utilizing the MBFD algorithm, the improved MET algorithm enhances task scheduling efficiency in cloud computing environments. By optimising resource utilisation, decreasing network latency, and enhancing overall system performance, this method eventually leads to better user experiences and more productivity.

Algorithm: **Improved MET**

1) Input: R (list of available resources)

```
        i)  T (list of tasks to be scheduled)
2) ST = Sort(T, descending order of resource requirements)
3) for each task in ST do
   a) BF = None
   b) BFitness = Infinity
   c) for each res in R do
        i)  if res.Ac >= task.Rr and res.U < BFitness then
        ii) // Check if the user is from the same region as the resource
        iii) if task.Ur == res.Rg then
             (1) BF = res // Set the resource as the best fit
             (2) BFitness = res.U // Update the best fitness value
        iv) end if
        v)  end if
   d) end for
   e) if BF is not None then
        i)  add a task to BF.T // Assign the task to the best − fit resource
        ii) increase BF.U by task.Rr // Update the resource's utilization
        iii) decrease BF.Ac by task.Rr // Update the resource's available capacity
   f) end if
4) end for
5) TRU =
   sum of all res.U divided by the sum of all res.TC //
   Calculate the average resource utilization
6) makespan =
   maximum end time of tasks in all R minus minimum start time of tasks in all R /
   / Calculate the makespan
7) Output:
   a) Scheduling Results for each resource:
   b) Assigned tasks list for each resource
   c) Utilization for each resource
   d) Resource Utilization: TRU
   e) Makespan: makespan
```

The algorithm takes a list of available resources (R) and a list of tasks to be scheduled (T) as input. It sorts the tasks based on their resource requirements in descending order. Then, it iterates through each task, evaluating available resources and finding the best-fit resource within the same region. The best-fit resource is selected based on its available capacity (Ac), utilization (U), and matching region (Rg) with the task's user region (Ur). The task is assigned to the best-fit resource, and resource utilization

and availability are updated accordingly. After all tasks are assigned, the algorithm calculates the average resource utilization (TRU) and makespan. The output includes the assigned tasks and utilization for each resource, as well as the average resource utilization and makespan, providing insights into the task scheduling efficiency in the cloud computing environment.

3.4 Summary of the Chapter

The chapter provides the background of workflow, and different cloud computing environments while discussing multi-cloud and real cloud. It also discusses the MET algorithm and finally incorporates it into the proposed work algorithm. The architecture description is followed by the algorithm description and the expected outcomes in the form of assigned tasks and utilization for each resource, as well as the average resource utilization and makespan, providing insights into the task scheduling efficiency in the cloud computing environment.

CHAPTER 4: APPLICATION AND IMPROVISATION IN FLOWER-POLLINATION ALGORITHM FOR IMPROVED EFFICIENCY IN TASK SCHEDULING

4.1 Background

Effective work scheduling is essential in today's computer settings, including cloud computing and data centres, for maximising resource usage and enhancing system performance. Data centres need to use as little electricity as possible in order to be energy efficient and reduce their carbon impact. However, traditional job scheduling algorithms often focus on a single objective, such as minimizing job completion time, which may lead to suboptimal solutions with respect to other crucial factors. Hence, the chapter is dedicated to providing measures involved in the improvement of the Flower Pollination Algorithm to achieve significant efficiency in task scheduling.

To address the limitations of single-objective job scheduling, researchers and practitioners have turned their attention towards multi-objective job scheduling approaches. Multi-objective job scheduling aims to simultaneously optimize multiple, often conflicting, objectives to achieve a balance between different performance metrics. In this context, two important and often opposing objectives are minimizing power consumption and maximizing the overall execution rate (or throughput) of the system.

1. Minimizing Power Consumption: Power consumption has emerged as a significant concern in modern computing environments due to the ever-increasing energy demands of data centres and cloud infrastructures. High power consumption not only incurs substantial operational costs but also contributes to environmental concerns. Data centres need to use as little electricity as possible in order to be energy efficient and reduce their carbon impact.

2. Maximizing Overall Execution Rate: On the other hand, maximizing the overall execution rate is crucial for improving the system's performance and meeting the Service Level Agreements (SLAs) of various applications. A higher execution rate ensures that jobs are completed efficiently, reducing the overall turnaround time and enhancing user satisfaction. However, optimizing for execution rate alone may lead to excessive power consumption if not balanced with energy-efficient scheduling strategies.

The integration of power consumption and execution rate optimization in a multi-objective job scheduling framework is challenging due to their inherent trade-offs [82, 100]. While aggressive execution of jobs can improve throughput, it may lead to higher power consumption. Conversely, energy-efficient scheduling may compromise the overall execution rate. To tackle this multi-objective optimization problem, various techniques, including evolutionary algorithms [117], genetic algorithms [90, 118], and Pareto-based approaches [119], have been employed. These methods provide a collection of trade-off solutions known as the Pareto front, which illustrates various trade-offs between execution speed and power usage. The decision-maker can then choose a solution from the Pareto front based on their specific preferences or needs [120, 121].

4.2 Scenario of Development and the Optimization Issue

For maximising resource utilisation and enhancing system performance, efficient work scheduling is essential in the world of computing. Traditional job scheduling algorithms often focus on single objectives, such as minimizing job completion time or maximizing throughput [76, 122, 123]. Modern computer systems, however, need more advanced strategies that may simultaneously optimise various goals, such as reducing power consumption and increasing execution pace. To tackle this challenge, optimization algorithms have been gaining prominence in job scheduling to strike a balance between conflicting objectives [124, 125]. One such innovative algorithm is the FPA, inspired by the natural process of flower pollination [126, 127]. FPA leverages the flower's unique pollination behaviour to solve complex optimization problems, including job scheduling in cloud computing and data centres. By

emulating the process of flower pollination, FPA efficiently explores the solution space, providing promising results in terms of improving job scheduling efficiency [78, 128, 129]. In this context, this paper explores the application of FPA as a powerful tool to enhance job scheduling processes, making it an exciting avenue of research and development in the realm of optimization algorithms for modern computing systems.

4.2.1 The FPA Algorithm

A metaheuristic optimisation method called the Flower Pollination method (FPA) was developed in order to solve optimisation problems. It was initially introduced by Xin-She Yang as an optimisation technique that was inspired by nature in 2012 [126]. The method mimics the natural process of flower pollination, in which flowers interact and exchange pollen to fertilise and generate new offspring. This technique is used to solve optimisation issues. The process of flower pollination can be illustrated using Figure 4.1.

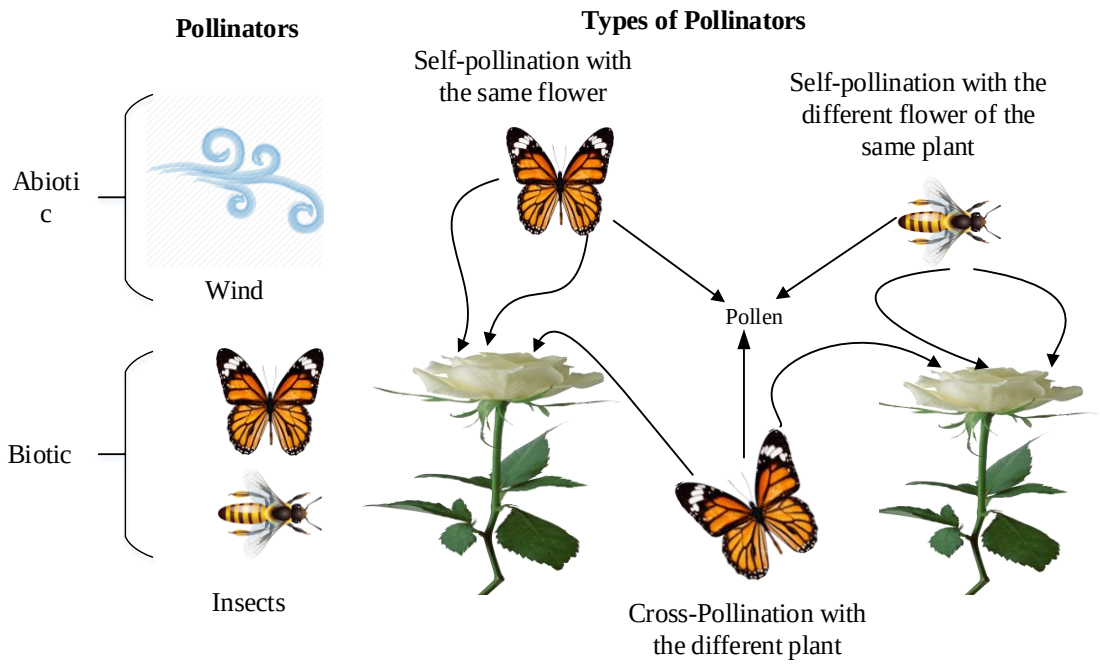

Figure 4.1 The Flower Pollination Process [125]

The fundamental idea underlying the FPA is to visualise possible solutions as flowers, and their fitness or quality as the nectar of the flowers, for an optimisation issue. In

the algorithm, each flower (solution) is associated with a nectar amount, which represents its fitness value. The goal of the algorithm is to find the optimal solution with the highest nectar (fitness) value [129].

The key steps of the Flower Pollination Algorithm are as follows:

1. Initialization: Create a starting population of probable solutions (flowers) at random or using a particular initialization technique.
2. Nectar Amount (Fitness Evaluation): Determine each flower's fitness or objective function value. According to the criteria of the problem, the fitness function should be problem-specific and direct the algorithm to identify better solutions.
3. Flower Pollination and Nectar Update: this stage, flowers communicate with one another by exchanging nectar, or information, which encourages exploration and utilisation of the search space. A flower with a higher nectar amount (better fitness) can influence nearby flowers and transfer its nectar to them. This process simulates the pollination behaviour in light of CO_2, cost and energy consumption, where a flower's attractiveness influences its interaction with other flowers.
4. Flower Reproduction (Optimization): Based on the updated nectar values, some flowers are allowed to reproduce by creating new solutions (offspring). These offspring replace some of the weaker solutions in the population, thereby influencing the search towards more promising regions of the solution space in order evaluate system in terms of CO_2, cost and energy consumption.
5. Termination: The algorithm keeps repeating the pollination and reproduction phases until a stopping requirement is satisfied, such as reaching the required degree of convergence or the maximum number of generations.

The FPA is a population-wide optimisation method that seeks to balance solution space exploration and exploitation. It makes use of the idea of pollination and floral reproduction to improve search and uncover potential answers to challenging optimisation issues. There are two types of thresholds utilized here, upper threshold and lower threshold.

- Where, upper threshold = average workload + 30%
- And lower threshold = average workload -30%

Local server-based analysis entails processing data on a single server, which may be faster because of proximity. Multicloud server analysis distributes workloads across several cloud servers, allowing for scalability while introducing variable latency due to data transit between different cloud locations, potentially impacting processing time.

4.2.2 Significance in Job Scheduling in Cloud Computing

Numerous optimisation issues, such as work scheduling in cloud computing systems, have been tackled using the Flower Pollination Algorithm. [78, 130]. In cloud job scheduling, tasks or jobs need to be allocated to appropriate resources (e.g., virtual machines) efficiently to optimize resource utilization, minimize job completion time, and adhere to Service Level Agreements (SLAs) [85]. The job allocation process in the proposed work can be explained by using Figure 4.2.

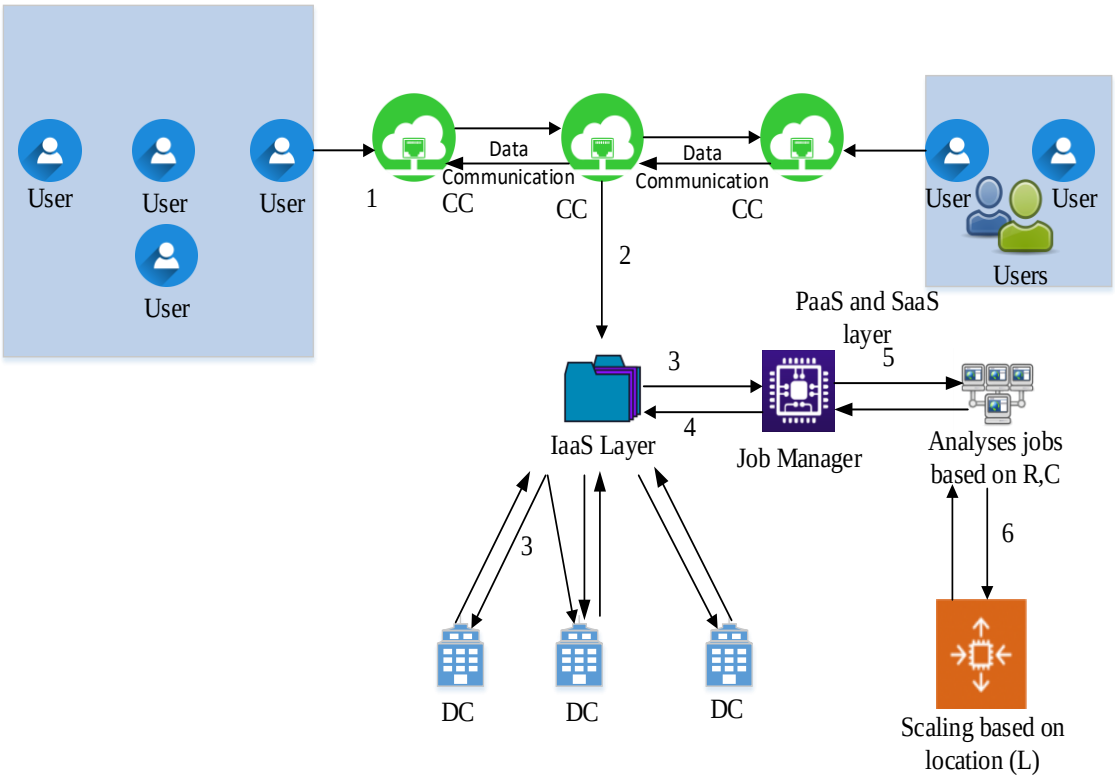

Figure 4.2 Job Allocation Process in Proposed Work

The FPA's ability to explore the solution space and adaptively adjust the search process makes it suitable for job scheduling in cloud environments. By representing potential schedules as flowers and using their fitness (performance) to guide the search, FPA can help find near-optimal schedules that maximize resource utilization and minimize job execution time. Furthermore, the FPA's nature-inspired approach allows it to handle dynamic and unpredictable changes in cloud workloads effectively. The FPA's capacity to evolve and discover new areas of the solution space might result in better scheduling decisions as cloud environments face fluctuations in resource availability and demand. The following diagram illustrates the FPA algorithm's design.

Algorithm FPA for job scheduling

a) Input: Population size (pop_{size})
   i) Maximum number of iterations ($max_{iterations}$)
   ii) Probability of flower pollination (p)
b) Output: Best solution found ($best_{solution}$)
c) Initialize a population of solutions randomly
d) Evaluate the fitness of each solution in the population
e) Set the current iteration to 0
f) while $current_{iteration} < max_{iterations}$:
   i) for each solution in the population:
   ii) if rand() $<$ p:
      (1) # Flower pollination
      (2) $neighbor_{solution}$ = SelectRandomNeighbor()
      (3) $updated_{solution}$ = ApplyFlowerPollination(solution, $neighbor_{solution}$)
      (4) Evaluate the fitness of the $updated_{solution}$
      (5) if $updated_{solution}$ is better than solution:
         (a) Replace solution with $updated_{solution}$
   iii) Increment $current_{iteration}$
g) $best_{solution}$ = FindBestSolutionInPopulation()

h) Return $best_{solution}$

2) End Procedure

3) Function SelectRandomNeighbor():

a) # Randomly select a neighbouring solution from the population

b) $neighbor_{index} = rand(0, pop_{size} - 1)$

c) Return population[$neighbor_{index}$]

4) End Function

5) Function ApplyFlowerPollination(solution, $neighbor_{solution}$):

a) # Apply flower pollination operators to create a new solution

b) $new_{solution}$ = CreateNewSolution(solution, $neighbor_{solution}$)

c) Return $new_{solution}$

6) End Function

7) Function CreateNewSolution(solution, $neighbor_{solution}$):

a) # Create a new solution by combining the features of two solutions

b) $new_{solution}$ = Copy(solution)

c) for each feature in $new_{solution}$:

i) if rand() < 0.5:

ii) # Randomly select a feature from $neighbor_{solution}$

iii) $new_{solution}$[feature] = $neighbor_{solution}$[feature]

d) Return $new_{solution}$

8) End Function

9) Function FindBestSolutionInPopulation():

a) Find the best solution (pareto front) in the population based on fitness values

b) $best_{solution}$ = population[0]

c) for each solution in population:

i) if the solution is better than $best_{solution}$:

ii) $best_{solution}$ = solution

d) Return $best_{solution}$

10) End Function

The algorithm can be explained in the following steps

1. Initialize the population of solutions (job scheduling configurations) randomly or using some heuristic method. Each solution represents a different job allocation to resources in the cloud.
2. Determine each solution's population fitness. Multiple objectives, including the makespan (task completion time), power consumption, and execution rate, should be taken into account by the fitness function. This assessment establishes how effectively each option meets the conflicting objectives
3. Identify the Pareto front of solutions based on their fitness values. The Pareto front represents the non-dominated solutions, where no solution can be improved in one objective without degrading another objective. These solutions provide a trade-off between the different objectives and are considered potentially optimal solutions.
4. Set the maximum number of iterations and the probability of flower pollination (parameter "p").
5. For each iteration: a. For each solution (flower) in the population:
    - If a random number is less than or equal to "p," proceed with flower pollination.
    - Select a neighbouring solution (another flower) from the population at random.

b. Perform flower pollination:

- Generate a new solution by combining features of the current solution and the neighbouring solution using flower pollination operators.
- The flower pollination operators could be crossover, mutation, or other transformation mechanisms.

c. Evaluate the fitness of the new solution.

d. Compare the fitness of the new solution with the current solution:

- If the new solution is better (closer to the Pareto front) for at least one objective, replace the current one.

6. Carry out the flower pollination procedure once more for the designated number of times.

7. The Pareto front of solutions at the algorithm's conclusion illustrates the trade-off between maximising execution pace while minimising makespan and power usage. These solutions to the multi-objective task scheduling issue are regarded as being close to optimum.

In order to improve the algorithmic architecture, the proposed work introduced a Q-learning algorithm that updates the pollination value in terms of Q-table. As the pollinating agents are attracted towards a better smell of the flower, the Q-learning algorithm updates the system using Bellman's equation. The updated illustration is as follows.

4.2.3 Switching Probability

The pollination mechanism in FPA refers to how information (solution attributes) is exchanged between different flowers. The switching probability 'p' is a parameter that determines the likelihood of a flower using either the local or global pollination mechanism. It's a value between 0 and 1, where $0 \leq p \leq 1$.

- Local Switching

If p is close to 0, it means that the algorithm predominantly employs the local pollination mechanism. This would result in the algorithm focusing more on the exploitation of solutions in the local vicinity.

- Global Switching

If p is close to 1, it means that the algorithm predominantly employs the global pollination mechanism. This would result in the algorithm emphasizing the exploration of the solution space to find diverse solutions.

- Dynamic Switching

The term "dynamic" indicates that the switching probability 'p' can change during the optimization process. Depending on where the optimisation is at the moment, the algorithm can modify 'p' in an adaptable manner. Due to its versatility, the algorithm may, when necessary, find a balance between exploitation and exploration.

In the FPA the switching probability 'p' controls the trade-off between regional and international pollination processes. By dynamically adjusting 'p', the algorithm can vary its behaviour to explore different aspects of the solution space as the optimization progresses. This dynamic strategy can potentially enhance the algorithm's convergence speed and solution quality by balancing exploration and exploitation.

4.3 Proposed Work towards improving FPA using Q-learning

The FPA algorithm refers to different flowers or zones and hence to create multiple zones, the proposed work considers two different factors.

a) The deployment of multi-datacenter
b) The separation of the resource allocation based on Quality of Service (QoS) parameters.

4.3.1 Multi-datacenter deployment, benefit and illustration

The idea of multi-data centre architecture has come to light as a major development in the quickly changing field of cloud computing, helping to meet the increasing needs of contemporary services and applications. As organizations increasingly rely on cloud-based solutions to meet their computing needs, the need for improved performance, reliability, and scalability becomes paramount [131]. Multi-data centre architecture refers to the deployment of multiple data centres spread across different geographic locations, interconnected to form a unified cloud infrastructure. This approach aims to overcome the limitations of traditional single data centre setups and offers a wide array of benefits while also presenting some unique challenges [132]. Benefits of Multi-Data Centre Architecture in Cloud Computing:

1. **High Availability and Fault Tolerance:** Multi-data centre setups enhance the availability and fault tolerance of cloud services. By distributing resources across multiple data centres, service providers can ensure that even if one data centre experiences a failure or disruption, the services can be quickly switched over to a redundant data centre, reducing downtime and maintaining continuity.

2. **Improved Performance:** With data centres strategically located in different regions, users can access cloud services from data centres closer to their geographical location. This enhances performance and user experience by decreasing latency and increasing reaction times.

3. **Scalability and Load Balancing:** Multi-data centre architectures enable horizontal scaling, allowing cloud providers to add more servers and resources as demand grows. Load balancing techniques can intelligently distribute user requests across data centres, ensuring optimal resource utilization and preventing overloading of any specific data centre.

4. **Data Residency and Compliance:** Some regulations require data to be stored within specific geographical boundaries. Multi-data centre setups allow cloud providers to comply with data residency requirements by ensuring data is stored in data centres located in the respective regions.

5. **Disaster Recovery and Business Continuity:** Multi-data centre architectures facilitate robust disaster recovery strategies. In the event of a natural disaster or catastrophic failure in one location, data and services can be seamlessly shifted to an unaffected data centre, enabling business continuity.

- **Issues and Challenges of Multi-Data Centre Architecture:**

1. **Data Synchronization and Consistency:** Maintaining data consistency across distributed data centres can be challenging. Ensuring that updates made in one data centre are promptly synchronized with other data centres while preserving data integrity requires sophisticated synchronization mechanisms.

2. **Network Latency and Communication Overhead:** Communication between geographically dispersed data centres may incur network latency and increased communication overhead. These factors can affect the overall performance and response times of cloud services.

3. **Complexity and Management Overhead:** Managing and coordinating multiple data centres adds complexity to the cloud infrastructure. Tasks like load balancing, resource allocation, and failover management are the responsibility of cloud administrators, and they can lead to an increase in administrative overhead.

4. **Cost and Resource Allocation:** Setting up and maintaining multiple data centres involves significant capital and operational costs. Deciding how to allocate resources across data centres to meet varying demands while minimizing costs is a critical challenge.

5. **Security and Compliance:** Security becomes more complex in a multi-data centre environment. Ensuring consistent security policies and compliance across all data centres is essential to protect sensitive data and maintain regulatory requirements.

The proposed work has utilized CloudSim which is supported by the Java platform to deploy the proposed work. CloudSim is an open-source CC simulation toolkit that provides a platform for modelling and simulating cloud environments. It allows researchers and developers to experiment with different cloud architectures, policies, and scheduling algorithms in a virtualized environment, without the need for physical infrastructure. CloudSim is built on top of the popular Java-based simulation framework, SimJava, and offers a range of features that make it an ideal choice for deploying cloud scheduling algorithms [133]. The toolkit supports the creation and management of VMs and PMs in a virtualized cloud environment, enabling the simulation of diverse scenarios reflecting real-world cloud usage patterns. Researchers can define custom workloads, such as task arrival patterns, VM resource requirements, and job execution times, and evaluate various scheduling policies to optimize resource utilization and performance. CloudSim also allows for the modelling of data centres with multiple hosts and datacenter-level network topologies, providing flexibility to examine cloud scheduling algorithms under various configurations and network conditions. The efficiency of scheduling algorithms may be evaluated by analysing performance indicators like response time, throughput, and

resource utilisation, which are made possible by monitoring and reporting facilities. The scalability of CloudSim allows for simulations of different cloud sizes, ranging from small setups to large-scale data centres, allowing researchers to study scheduling behaviour under different workloads and cloud scales. Furthermore, CloudSim supports energy-aware cloud scheduling, enabling investigation and optimization of algorithms considering energy consumption and efficiency [134]. Its modular design and extensibility allow easy integration with external libraries and the incorporation of existing or custom-scheduling algorithms based on research requirements. With a vibrant user community providing support, documentation, tutorials, and research papers, CloudSim serves as a powerful simulation platform for exploring, evaluating, and advancing cloud scheduling algorithms and their efficiency in a wide range of cloud computing scenarios. The data centre modelling in Cloudsim is represented in Figure 4.3.

```
*/
List<Host> hostlist = new ArrayList<Host>();
for(int hc=0;hc<host_count;hc++)
{
    ram=(int)(800*Math.random());
    bw=(int)(8000*Math.random());
    storage = (int)(100000*Math.random());

Host host1 = new Host(hc, new RamProvisionerSimple(ram), new BwProvisionerSimple(bw), storage,
                peList, new VmSchedulerSpaceShared(peList));

hostlist.add(host1);
}
String architecture = "x86";
String os = "Linux" ;
String vmm = "XEN" ;
double timeZone=5.0;
double ComputecostPerSec=3.0*Math.random();
double costPerMem=1.0;
double costPerStorage=0.05;
double costPerBw=0.10;


DatacenterCharacteristics dcCharacteristics = new DatacenterCharacteristics(architecture, os, v
                hostlist, timeZone, ComputecostPerSec, costPerMem, costPerStorage, costPerBw);

LinkedList<Storage> SanStorage = new LinkedList<Storage>();
```

Host Creation

Creation of Data center with multiple aspects

Figure 4.3 Datacentre modelling in Cloudsim

The proposed work considers the following varying attribute values for the creation and processing of a data centre as shown in Table 4.1.

Table 4.1 Attributes and Description

Variable	Description
RAM	Amount of RAM allocated to a host (randomly generated)
Bw	Bandwidth allocated to a host (randomly generated)
Dc	Datacenter object representing the cloud data centre
Storage	Storage capacity allocated to a host (randomly generated)
peList	List of processing elements (PEs) representing the CPU cores
Provisioner	PE provisioner specifying the CPU capacity of the cores
core1, core2, ...	Processing elements (PEs) representing the CPU cores
host list	List of hosts in the data centre, each representing a physical machine (PM)
Architecture	Architecture type of the PMs in the data centre (e.g., x86)
Os	Operating system running on the PMs (e.g., Linux)
Vmm	Virtual machine monitor used on the PMs (e.g., XEN)
timeZone	Time zone of the data center
ComputecostPerSec	Cost per second of CPU processing on the PMs (randomly generated)
costPerMem	Cost per unit of memory allocated to a VM
costPerStorage	Cost per unit of storage allocated to a VM
costPerBw	Cost per unit of bandwidth allocated to a VM
Characteristics	Datacenter characteristics object containing information about the data centre
SanStorage	List of storage elements representing the storage devices in the data center
vm_properties	2D array representing the properties of virtual machines (VMs) including CPU demand, allocation cost, etc.
total_vm	Total number of VMs
total_properties	Total number of properties associated with each VM
pm_i_cost	Array representing the idle cost of each physical machine (PM)
total_pms	Total number of physical machines (PMs)
pm_CPU	Array representing the CPU capacity of each physical machine (PM)

4.3.2 Integration of Q-learning

With the help of the popular reinforcement learning algorithm Q-Learning, an agent may learn from its surroundings and modify its behaviour to attain a certain objective. It is especially helpful when the agent must do an exploration to figure out the optimal course of action since they lack prior knowledge of the surroundings. TUsing a Q-value function, the method determines the expected cumulative benefit of carrying out

a specified action under a given set of conditions. By iteratively updating the Q-values in response to the agent's experiences and actions, Q-learning ultimately achieves the optimal course of action for decision-making [135].

By incorporating Q-learning into the FPA, the algorithm's optimisation skills may be further improved, increasing its effectiveness and efficiency in locating optimal solutions. The primary idea behind this integration is to utilize the Q-values to guide the pollination process in FPA. Instead of using random search and pollination, the algorithm employs the Q-values to select the best flower to pollinate, which corresponds to choosing the most promising solution.

The integration process involves the following steps:

1. **Initialization:** Initialize the Q-values for each flower in the FPA population. These Q-values represent the expected cumulative rewards for pollinating each flower (i.e., solution) based on the agent's experiences.

2. **Exploration-Exploitation Trade-off:** During the pollination process, the agent must balance exploration and exploitation. Initially, the agent may prioritize exploration to discover new promising solutions. As the algorithm progresses, it shifts towards exploitation, focusing on exploiting the best-known solutions based on the Q-values.

3. **Updating Q-values:** After pollination, the Q-values are updated using Bellman's equation, which is a fundamental equation in reinforcement learning. The update process incorporates the immediate reward obtained from the newly discovered solution and the expected cumulative reward of the next state-action pair.

4. **Improved Solution Search:** By integrating Q-learning into FPA, the algorithm can better navigate the solution space and identify better solutions more efficiently. The Q-values serve as a form of memory that guides the algorithm towards more promising regions, avoiding unnecessary exploration and enhancing the convergence to the optimal solution.

The following ordinal measure of the development is used in the suggested study.

There are two main categories of energy usage that are recognised in the context of cloud data centres: CPU-intensive energy use and memory-intensive energy use. These resources are considered major contributors to carbon dioxide emissions. Two factors make up a cloud data center's overall power consumption: the fixed power used by the server ($E_{fixed\ server}$) and the dynamic power($E_{dynamic}$) used by the Virtual Machines (VMs) while they are in operating mode.

Further broken down into energy spent by the CPU (E_CPU) and memory (E_{memory}) in the physical server is the dynamic power consumption ($E_{dynamic}$). According to studies in the literature, a physical computer in a cloud data centre uses about 70% more energy while it is idle than when it is fully utilised. This substantial energy consumption results in significant carbon dioxide emissions. It is evident that when an inactive physical machine utilizes its resources fully, there are significant power savings and a reduction in CO_2 emissions, thereby enhancing resource efficiency and reducing the carbon footprint.

The following formulas are used to calculate the energy used and carbon emissions during work allocation and processing at each PM.

$$\mathbf{EC} = \sum_{\mathbf{m=1}}^{\mathbf{M}} \mathbf{I_m} \times \mathbf{ece_j} + (\mathbf{s_j} \times \mathbf{I_m}) \times \mathbf{ecs_j} \qquad \textbf{(4.1)}$$

M is the total number of instructions in PM 'j', ece is the execution cost at that PM, s is the amount of storage required for the mth instruction, and ecs is the energy consumed to store one instruction at that PM. Equation (4.2) may likewise be utilised to calculate CO2 emissions; here, cee denotes CO2 emitted when the mth instruction on the jth PM is being executed, and ces denotes CO2 released during storage.

$$\mathbf{CE} = \sum_{\mathbf{m=1}}^{\mathbf{M}} \mathbf{I_m} \times \mathbf{cee_j} + (\mathbf{s_j} \times \mathbf{I_m}) \times \mathbf{ces_j} \qquad \textbf{(4.2)}$$

In Equation 4.1, the total energy consumption (EC) is determined by aggregating the execution costs (ece_j) and the energy consumption related to storage (ecs_j) for each instruction under the specific PM 'j'. The CO2 released during the execution of the instruction set (cee_j) and the CO2 released during storage are added together in Equation 4.2 to determine the total CO2 emission (CE). (ces_j).

Logically, these equations provide a systematic approach to evaluating the energy consumption and environmental footprint of computational tasks performed on specific processing modules, which contributes to a better understanding of computing's overall impact on energy resources and the environment. Equation 4.1 quantifies energy usage by taking into account both execution and storage costs while equation 4.2 expands the analysis to incorporate environmental impacts, notably CO_2 emissions. It includes both the CO_2 emitted during execution and the CO_2 released during storage activities. Using these equations, one may analyze the energy requirements and environmental effects of executing a set of instructions on a certain PM, which can help with resource allocation, optimization, and sustainability decision-making.

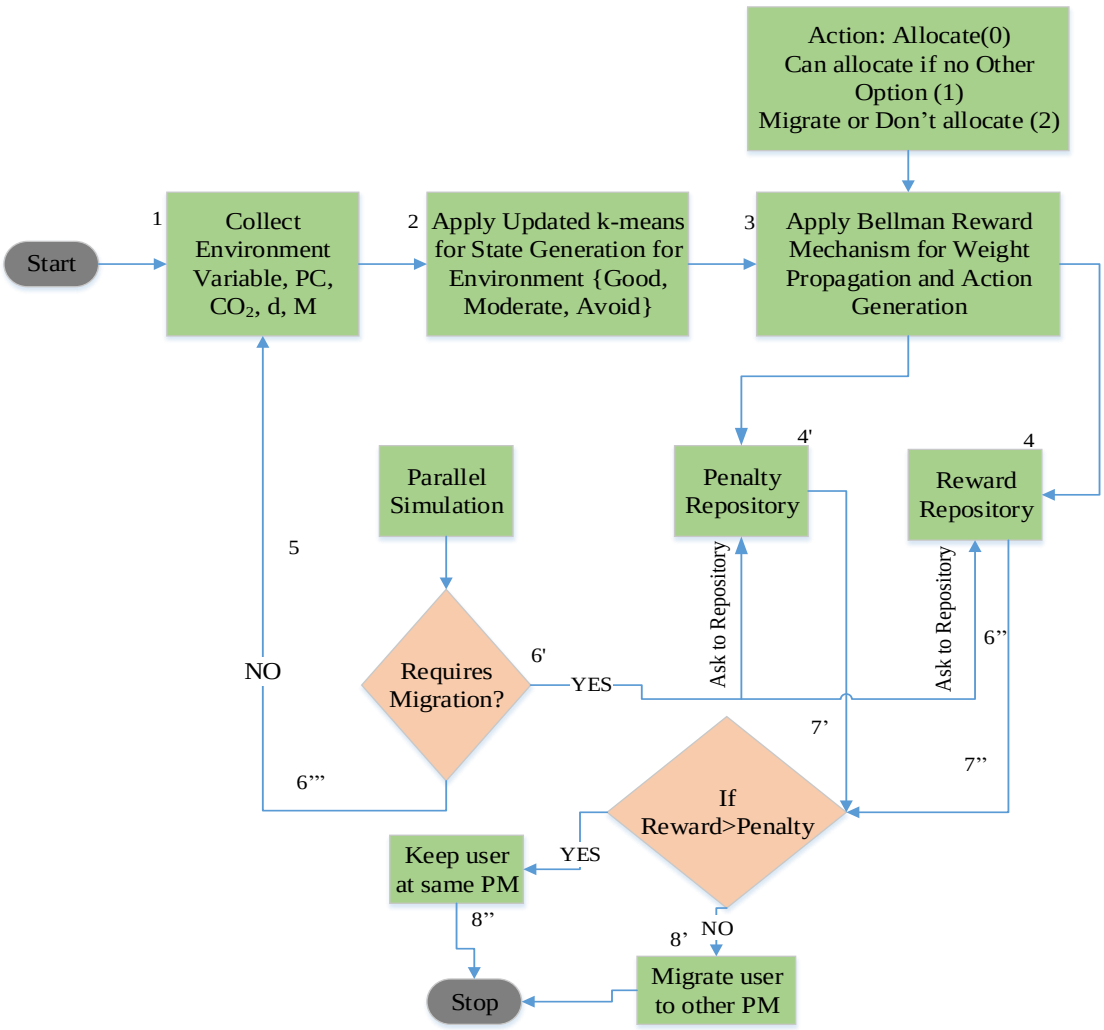

Figure 4.4 Proposed Work Using Q-Learning [135]

To implement the integration of the pollination process with Q-learning the proposed work uses the following workflow using Q-learning and is represented by Figure 4.4.

The suggested work integrates the fundamentals of the three main components of the Q-learning algorithm architecture: Environment, States, and Actions [136]. The environment is created by enhancing the architecture of the existing k-means algorithm, which clusters data according to Euclidean distance. In this method, the standard k-means metrics are applied. The overall work architecture is represented in Figure 4.5.

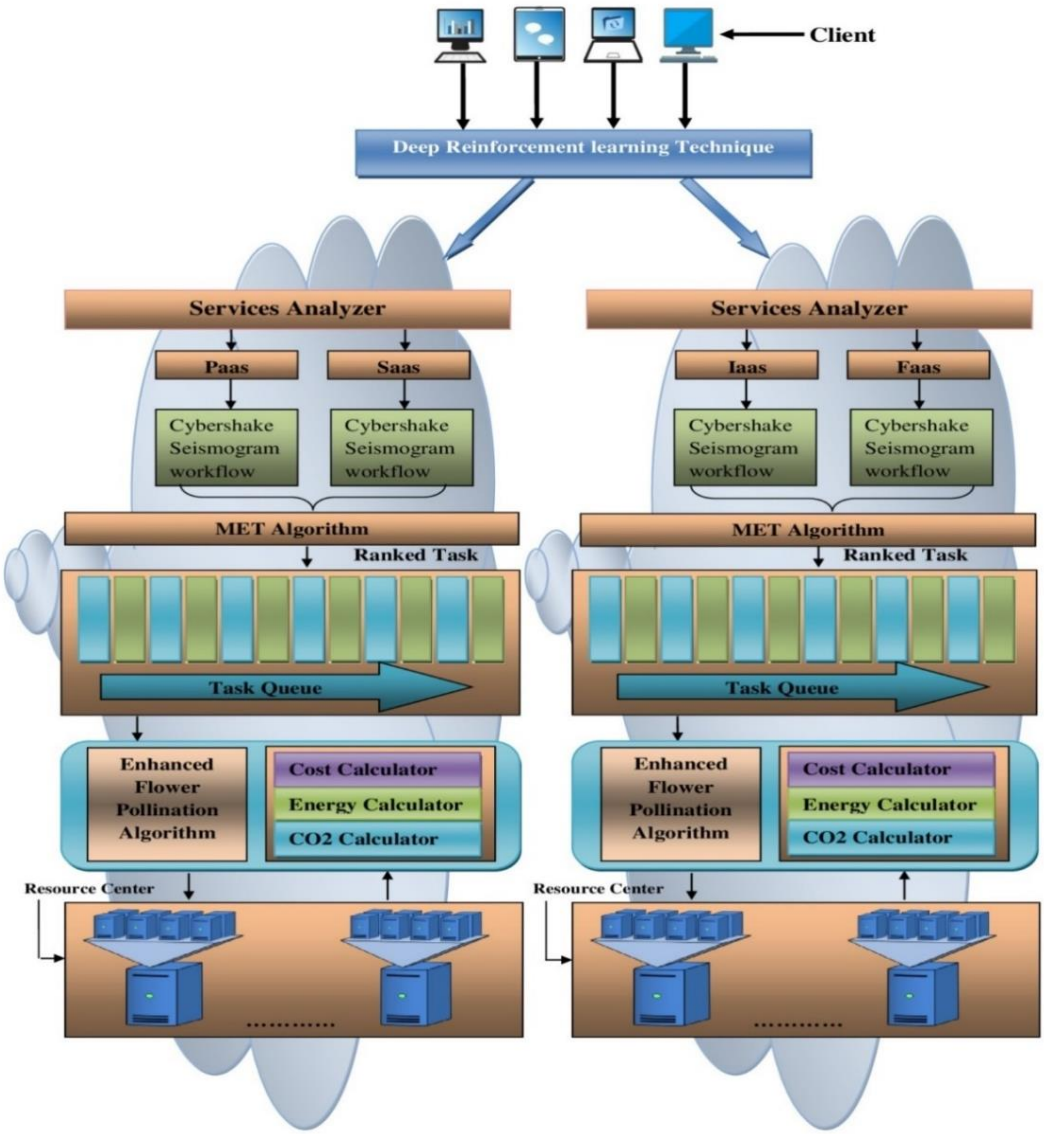

Figure 4.5 Overall work architecture

The updated k-means algorithm involves two significant changes. Firstly, Cosine similarity is introduced as an additional measure in distance calculation to effectively assign records to appropriate clusters. When assessing parameter distance, cosine similarity—which is the cosine of the angular difference between two vectors with the same number of characteristics—is commonly employed. The suggested work compares the cosine similarity between two vectors with the following four attributes: energy consumption (EC), carbon emission (CO2), user-data centre distance (d), and the total number of user-supplied instructions (M).

Second, by removing the potential for random centroid selection during the initial clustering step, the approach maximises the convergence rate. Instead, it creates two extra centroids with a 20% margin value and takes the mean value of each characteristic as the first centroid to offer some variance.

These enhancements in the k-means algorithm enable more efficient clustering of data, facilitating the creation of the environment for the Q-learning algorithm. This integrated approach improves decision-making and resource allocation within the cloud environment, making it more adaptive and effective in addressing complex optimization problems.

4.3.2.1 Q-learning Application in Proposed Work Case Scenario

- States X = {1,2,3}: The set of possible states. (Proposed work has 3 states.)
- Actions A = {1,2,3}: The set of possible actions. (Proposed work has 3 actions.)
- Reward function R(X, A): A function that maps a state-action pair to a real-valued reward or penalty.
- Gradient-based (weighted sum) transition function T(X, A): A function that maps a state-action pair to the next state using a weighted sum.
- Learning rate α in [0, 1]: A small constant that controls the step size during updates of the Q-values. (Typically $\alpha = 0.1$.)
- Discounting factor γ in [0, 1]: A parameter that determines the importance of future rewards compared to immediate rewards.

Output:

- Function Q(X, A): The learned Q-function, representing the expected cumulative reward for each state-action pair.

Procedure: Q-learning

1. Initialize the Q-function Q(X, A) arbitrarily for all state-action pairs.
2. While the Q-function is not converged, repeat the following steps: 3. Start in a random state in X.
    4. While the current state s is not a terminal state (the end of an episode): 5. Calculate the policy π according to the weighted sum of old weight (ow) and new weight (nw) using the function nw = ax + b, where a and b are constants, and x = norm(EC, CO_2, d, M).
        6. Choose action a based on the policy π(s).
        7. Receive the immediate reward or penalty r by executing action a in state s.
        8. Observe the new state s' resulting from the action a.
        9. Update the Q-value for the state-action pair (s, a) using the Q-learning update rule: Q(s, a) <- (1 - α) * Q(s, a) + α * (r + γ * max_{a'} Q(s', a'))
        10. Set the current state s to the new state s' and continue with the next iteration.
    5. End of the inner loop (when a terminal state is reached).
3. End of the outer loop (when the Q-function is converged).
4. Return the learned Q-function Q.

Algorithmically, it can be represented as shown in the following Figure 4.6

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

States $\mathcal{X} = \{1, 2, 3\}$ proposed work has 3 states

Actions $\mathcal{A} = \{1, 2, 3\}$ quad A: X $\Rightarrow \mathcal{A}$ propsoed work has 3 actions

Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Gradient based (weighted sum) transition function $T : \mathcal{X} : ax + b \times \mathcal{A} \to \mathcal{X}$

Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$

Discounting factor $\gamma \in [0, 1]$

procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)

Initialize $Q : \mathcal{X}ax + b : a- > .01b = .02x = norm(PC, CO2, d, M) \times \mathcal{A} \to \mathbb{R}$ arbitrarily

while Q is not converged **do**

Start in state $s \in \mathcal{X}$

while s is not terminal **do**

Calculate π according to w=ow+nw where nw=ax+b and ow=$nw_{(}$t−1) t is the time state of current value and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)

$a \leftarrow \pi(s)$

$r \leftarrow R(s, a)P(s, a)$ ▷ Receive the reward or penalty

$s' \leftarrow T(s, a)$ ▷ Receive the new state

$Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$

$s \leftarrow s'$

return Q

Figure 4.6 The applied Q-learning algorithm

The simulation of the code has been done over the Cloudsim platform as discussed earlier in the same chapter and is shown in Figure 4.7.

```
for(int kdd=0;kdd<total_action_variables;kdd++)
{
    environment_objects[environment_other_variables][kdd]=current_state[kdd];
  //  System.out.println("hh");
}
int[] action_Result=new int[possible_actions];
int one_count=0; // for positive reward
int zero_count=0; // for negative reward
double margin=0;
double discount_factor=10;
//environment_objects[tr+1]
for(int kd=0;kd<possible_actions;kd++)
{
    // for every possible action, need to check the propogation error
    for(int jj=0;jj<3;jj++)
    {
        for(int kk=0;kk<total_action_variables;kk++)
        {
         double action_step=Math.random(); // adding action step towards other players
         environment_objects[jj][kk]=environment_objects[jj][kk]*action_step;

        }
    }
    machine_learning mll=new machine_learning();
    double[] avg_leap_step=new double[total_action_variables];
    avg_leap_step=mll.calculate_mean(environment_objects,3,total_action_variables);
    // considering  parameter //
    double diff_value=0;
    double per_value=0;

    diff_value=Math.abs(avg_leap_step[3]-environment_objects[2][3]);
    per_value=diff_value*100/avg_leap_step[3];
    if(per_value<=100)
    {
        action_Result[kd]=1;
        one_count=one_count+1;
```

Figure 4.7 Implementation of Q-learning algorithm in the cloud-sim environment

In order to modulate the Q-learning further, a propagation-based machine learning algorithm is also applied. The neural network is a propagation-based algorithmic architecture that propagates the weights as per the hidden neurons in the list.

4.3.3 Neural Networks and its Applicability

Neural networks, an efficient machine learning model, are modelled after the composition and functions of the human brain. It is a sort of artificial intelligence that acquires new skills by studying data and seeing patterns in order to carry out tasks. Numerous domains, such as image identification, natural language processing, robotics, and optimisation issues, have made substantial use of neural networks.

A neural network's basic building blocks are linked neurons arranged in layers. Each neuron receives an input, analyses it using an activation function, and then generates an output that helps to form the overall outcome. The capacity of neural networks to learn complicated and nonlinear correlations in data through a process known as training gives them their strength.

Applicability of Neural Networks to Scheduling Algorithms:

Scheduling algorithms deal with the optimization of resource allocation and task sequencing to achieve specific objectives, such as minimizing makespan, reducing lateness, or maximizing resource utilization. Neural networks offer several advantages in addressing scheduling problems:

1. **Nonlinearity and Complex Patterns:** Many scheduling problems involve complex relationships and dependencies among various variables. Neural networks can learn and model these intricate patterns, enabling more effective solutions than traditional linear approaches.
2. **Flexibility:** Neural networks can be adapted and tailored to different scheduling scenarios. They can handle a variety of restrictions and objectives by altering the network design and loss functions, which makes them relevant to a broad range of scheduling issues.
3. **Learning from Data:** Neural networks' capacity to learn from data is one of its main advantages. To find the best scheduling rules and tactics, they might be educated using expert knowledge, simulations, or historical scheduling data.

4. **Real-time Adaptation:** In scheduling environments, where conditions change frequently, neural networks can continuously learn and adapt to new information, making them suitable for real-time decision-making.
5. **Global Optimization:** Traditional scheduling algorithms may rely on heuristics or local search methods that might not guarantee finding the optimal solution. Neural networks can be trained using global optimization techniques to improve the chances of finding near-optimal solutions.
6. **Parallel Processing:** Neural networks can be implemented on parallel hardware, such as GPUs or specialized hardware like TPUs, allowing for faster and more efficient computation of scheduling solutions.
7. **Combining with Traditional Methods:** Neural networks can be combined with traditional optimization methods to leverage the strengths of both approaches. For instance, neural networks can learn to improve initial solutions generated by heuristics or metaheuristic algorithms.

4.3.3.1 Integration of the Neural Network

Training a neural network for job allocation division is based on good and bad allocations which is a part of testing or the validation process involving the following steps:

1. **Data Collection and Pre-processing:**
    - Gather a dataset of job allocations with associated features and labels indicating whether each allocation is considered "good" or "bad." The features could include various parameters related to the allocation, such as resource utilization, processing time, or task dependencies.
    - Divide the dataset into training and testing sets, manage missing values in the data, and scale or normalise the features.
2. **K-Means Clustering:**
    - Divide the data into two sections using the K-means clustering algorithm: one for "good" allocations and one for "bad" allocations. A well-liked unsupervised clustering method called K-means divides data points into K clusters according to their similarity.

- Based on the closest cluster centroid, allocate every data point (task allocation) to a certain cluster.

3. **Mean Squared Error (MSE) Calculation:**
    - For each cluster, determine the Mean Squared Error (MSE). MSE calculates the average squared difference between each data point's actual label and projected label within a cluster.
    - The MSE for each cluster reflects how well the K-means algorithm separates good and bad allocations in that cluster. Lower MSE indicates better separation.
4. **Labelling the Clusters:**
    - Based on the MSE values of each cluster, assign labels to the clusters. The cluster with the lower MSE is considered to represent "good" allocations, while the one with the higher MSE represents "bad" allocations.
    - By assigning labels to the clusters, we create two distinct groups: one group consisting of good job allocations and the other group with bad job allocations.
5. **Neural Network Training:**
    - Design and configure a neural network architecture suitable for the job allocation division problem. The network should have an input layer corresponding to the features of the job allocations and an output layer with a binary output representing good or bad allocation.
    - Split the pre-processed dataset into input features and corresponding labels based on the cluster assignments from the K-means algorithm.
    - Use the labelled data to train the neural network. Using methods like gradient descent and backpropagation, the network attempts to minimise the classification error between the predicted labels and the ground truth labels (good or poor).
    - Use the training set to adjust the weights and biases of the neural network iteratively to improve its performance.
6. **Model Evaluation:**

- Use the testing dataset, which the neural network has never seen before, to assess its performance after training. To evaluate how well the neural network can discriminate between good and bad job allocations, compute measures like accuracy, precision, recall, and F1-score.
- Adjust the neural network architecture or hyperparameters if necessary to optimize its performance.

7. **Job Allocation Division:**
    - Once the neural network is trained and evaluated, use it to predict the label (good or bad) for new, unseen job allocations. The neural network will assign each job allocation to one of the two groups, based on the learned patterns and relationships in the data.
    - Job allocations labelled as "good" can be prioritized or scheduled accordingly, while those labelled as "bad" may require further analysis or improvements.

Integrating Q-learning with K-means clustering in the job allocation division process offers a powerful and sophisticated approach to optimizing resource allocation in scheduling scenarios. K-means efficiently segment job allocations into two groups: good and bad, based on their characteristics and performance metrics. This clear separation simplifies the problem, as Q-learning can now concentrate on optimizing resource allocation within well-defined groups. By focusing on the good allocations, which have been pre-identified using K-means, Q-learning can significantly reduce the search space and computational burden, leading to faster and more efficient decision-making.

The system's predictability is improved when K-means clustering and Q-learning are combined. With hosts or working machines ranked based on historical data and job performance, Q-learning can make more informed allocation decisions. This prioritization of hosts based on their past performance allows for better resource utilization and minimizes the chances of making suboptimal choices. Moreover, Q-learning's ability to learn and adapt from past experiences ensures that the system

remains adaptable to changes in the environment or job characteristics, making it suitable for real-time and dynamic scheduling scenarios.

The system hardware requirements and configuration used to channelize the simulation process are discussed in Table 4.2.

Table 4.2 System Configuration

Parameters	Description
RAM	4GB
Memory	DDR3
HD Capacity	500GB
Number of Clouds	10
Number of Hosts	200
Available Bandwidth	100-7500 Hz
Available Cores	4
Capacity per core	3 octa engine
Engine Type	Multi
Engine Propagation	Quad Core
Process Utilization Minimum	1 Hz
Single core score	14323
Multi-Core score	14883
Number of Clouds	10

The integration also enhances the overall system performance which is later verified using testing or validation. By focusing on good allocations, the system can improve task completion times, reduce bottlenecks, and increase overall efficiency. Increased productivity and cost reductions follow from better resource usage and increased throughput. Additionally, the system's scalability and generalizability are enhanced by the combination of Q-learning and K-means clustering, which allows it to handle enormous datasets and successfully apply learned techniques to new, unknown data.

Overall, the integration of Q-learning with K-means clustering in job allocation division reduces complexity, enhances predictability, and improves overall system

performance. By leveraging the strengths of both approaches, this integration offers an efficient and effective solution for resource allocation, making it a valuable tool in various scheduling and resource management applications.

The following algorithm explains the working of the Neural Networks over the cloud sim simulator.

Algorithm: Neural Network Training for Job Allocation Division

---

Input:

- $\mathbf{total_{users}}$: Total number of transactions (job allocations).
- $\mathbf{total_{attributes}}$: Total number of attributes (features) for each transaction.
- $\mathbf{allocation_{user_{to_{host}}}}$: Array of size $[\text{total}_{\text{users}}][\text{total}_{\text{attributes}}]$representing the allocation data.
- $\mathbf{total_{action_{variables}}}$: Total number of action variables for allocation decisions.
- $\mathbf{action_{variables}}$: Array of size $[\text{total}_{\text{action}_{\text{variables}}}]$representing action variables(e.g., MIPS, PC, CO_2 cost).

Output:

- $\mathbf{neural_{margin}}$: A value representing the overall accuracy of the trained neural network.

Step 1: **Clustering of Good and Bad Transactions**

1. Use K − means clustering to divide the transactions (job allocations)into two clusters: good and bad.
2. Identify the best and worst similarity scores as well as their corresponding allocations, considering the good and bad clusters

Step 2: **Neural Network Initialization** 3. Set up the neural network with a hidden laye and the total number of input (state) variables as $\mathbf{total_{attributes}}$.

4. Define the total number of output (action) variables as $\mathbf{total_{action_{variables}}}$.
5. Initialize the neural network's hidden layer with arbitrary weights.

Step 3: **Neural Network Training**

6. Set **neuralnetworkepochs** to the total number of epochs for

neural network training.

7. Set **gradientsatisfied** to 0.
8. Initialize arrays **previoushwt** and **newwt** of size [$\text{total}_{\text{users}}$]to store the previous and new state of data (term weights).
9. Initialize **epochrun** to 0.
10. Generate the hidden layer weights $\mathbf{term_{weight1}}$ by extracting the term weights from $\mathbf{allocation_{user_{to_{host}}}}$ **based on** $\mathbf{action_{variables}}$.
11. For each epoch **i** from 0 to **neuralnetworkepochs** − **1**, do: a. If **gradientsatisfied** is 0, do:
    - Generate new hidden layer weights **newwt** using the **GenerateHiddneLayer** class.
    - Update **newwt** by adding the previous hidden layer weights **previoushwt**.
    - For each weight **wt** in **newwt**, check if **wt** $\geq$ **gradient**[**0**]. If true, set **gradientsatisfied** to **epochrun**.

      b. Increment **epochrun** by 1.

Step 4: **Neural Network Evaluation**

12. Calculate the neural margin $\mathbf{neural_{margin}}$ as follows:

− Set **testwt** as an array of size [$\text{total}_{\text{users}}$]and populate it with the generated weights from **GenerateHiddneLayer**.

− Set **mse**, **t1**, and **t2** to 0. − For each weight **wt** in **testwt**, do:

Calculate $\mathbf{propogation_{difference}}$as $(\mathbf{Math.abs(t1 - t2) / t1}) * \mathbf{100}$. − If $\mathbf{propogation_{difference}} < \mathbf{5}$, add $\mathbf{propogation_{difference}}$to $\mathbf{neural_{margin}}$.

− Divide **neural_margin** by $\mathbf{total_{users}}$ to get the average accuracy.

Step 5: **Output the Result**

13. Output the neural margin $\mathbf{neural_{margin}}$ representing the overall accuracy of the trained neural network in approximating the actual term weights of good and bad allocations.

The chapter explores the integration of Q-learning and flower pollination in the context of improving the performance of a scheduling network. By combining these

two powerful techniques, the authors aim to enhance resource allocation and task sequencing in dynamic scheduling environments. Q-learning, a reinforcement learning algorithm, is utilized to optimize the allocation decisions based on rewards and penalties obtained from previous scheduling experiences. Flower pollination, inspired by the natural behaviour of plants, is employed as a metaheuristic optimization method to refine the solutions further. This combination allows for efficient exploration of the search space, leading to improved scheduling outcomes. This is corroborated by Table 4.3, which also offers a more comprehensive comparison of the performance of the suggested work (Q-learning with FPA) compared to other machine-learning algorithms. The accuracy evaluation shows that the proposed work provides a better chance of deducing a highly accurate and precise scheduling architecture in comparison to Q-learning or neural networks alone. The table illustrates that the integration of FPA slightly improved the average accuracy of resource allocation achieved using Q-learning architecture and outperformed the neural network-based leaning mechanism.

Table 4.3 Performance Analysis of Machine Learning Techniques

Number of Users	Q-learning with FPA (%)	Q-learning (%)	Neural Network (%)
50	92.31672379	90.73746	89.86480588
70	92.58298108	90.82462711	90.38029174
90	92.89497441	91.28205382	90.79675993
110	93.44631703	91.30882566	91.15083846
130	93.60683795	91.61332463	91.17887921
150	94.10810611	91.7657921	91.39688037
170	94.59717051	92.16610061	91.80837918
190	94.97771045	92.19867902	91.8577715
210	95.22657176	92.3323164	92.18396822
230	95.65173794	92.62365857	92.4653995
250	95.8308081	93.14628125	92.92775791
270	95.83888734	93.63021829	93.00340812
290	96.22756502	94.00412472	93.48779905

310	96.42916772	94.13669257	94.02145901
330	96.58615242	94.69970349	94.21031419
350	96.64468163	94.78294358	94.45859011
370	96.94415836	94.86254105	94.90213717
390	97.05383323	95.17649161	95.18734785
410	97.50203649	95.53564045	95.62181714
430	97.71748452	95.80347456	95.66533643
450	98.13783014	96.02048059	95.83683093
470	98.66501532	96.4696309	96.22285907
490	98.91902243	96.73877732	96.35690573
Average	95.73503364	93.5591234	93.26028421

Later, to achieve even better performance, the chapter goes beyond the traditional Q-learning. The collected scheduling data is divided into clusters using K-means clustering, resulting in groups of good and bad allocations. These clusters are then fed into neural networks, which are trained to learn the allocation division and prioritize good allocations. By leveraging neural networks' ability to model complex relationships and learn from data, the scheduling system gains the advantage of making informed allocation decisions based on historical patterns. Overall, it was observed that the model reached the highest accuracy of 98.9% with an average value of 95.73%. To support this, a detailed comparative analysis is performed for the evaluation parameters which is summarized in the next chapter.

4.4 Application over the real time multi-cloud

In the realm of modern computing, the utilization of cloud infrastructure has revolutionized the way we approach various computational tasks. In this context, the proposed work stands as a testament to the adaptability and scalability of our approach. Not content with mere theoretical considerations, our endeavour has ventured into the realm of real-time application on the AWS (Amazon Web Services) cloud platform, utilizing multiple cloud instances to amplify its effectiveness.

At the heart of this initiative lies the creation of a comprehensive virtual server, constructed with a robust Java-based architecture. To ensure the smooth operation of our endeavour, we meticulously installed the requisite Java Development Kits (JDKs)

to empower our virtual server with the necessary tools and capabilities. What sets our approach apart is the deliberate segmentation of responsibilities between two distinct cloud instances, each strategically situated in separate cloud data centres.

One of these instances is entrusted with the critical task of executing the proposed work, harnessing the computational power and resources offered by AWS to carry out complex operations with efficiency and speed. The other instance plays a complementary role, focusing on the crucial aspect of storage. Its primary responsibility is to manage and maintain the data and results generated during the execution phase, ensuring that valuable information remains organized and readily accessible.

4.4.1 Application Initialization at AWS cloud

Setting up the described infrastructure on AWS with multiple cloud instances, including the creation of a Java-based virtual server and the allocation of responsibilities, involves a series of steps and configurations. Here is a detailed description of how this setup can be accomplished:

1. **Amazon Web Services (AWS) Account**:
    - In the event that you do not currently have an AWS account, start by establishing one. This account is required in order to use AWS services.
2. **Launch Instances**:
    - Open the AWS Management Console and go to the dashboard for EC2 (Elastic Compute Cloud).
    - Press "Launch Instance" to begin the virtual server (instance) creation procedure.
3. **Choose an Amazon Machine Image (AMI)**:
    - Pick an AMI based on Java that meets your needs. Make sure the required JDK (Java Development Kit) is pre-installed.
4. **Instance Type**:
    - Select the right instance type for your computing requirements. Think about components such as CPU, memory, and storage.
5. **Security Groups**:

- Create security groups to manage traffic entering and leaving your instances. Set up rules to permit access via SSH, RDP, and any more required ports.

6. **Key Pair**:
    - Create or import an SSH key pair to secure access to your instances. This key pair will be used for authentication.
7. **Storage**:
    - Configure the storage for your instances. You can use Amazon EBS (Elastic Block Store) volumes for durable storage.
8. **Launch Instances**:
    - Review your instance settings and click "Launch" to create your virtual servers. AWS will assign public IP addresses to each instance by default, making them accessible over the internet.
9. **Elastic Load Balancing (Optional)**:
    - If you require high availability and load balancing, consider setting up an Elastic Load Balancer to distribute traffic between instances.
10. **Data Centers and Regions**:
    - Place the instances in different AWS regions or data centers, depending on your redundancy and data locality requirements. This ensures that they are geographically separated.
11. **Configuration and Software Setup**:
    - Access each instance using SSH or RDP (depending on the operating system) and configure the Java environment, including installing JDKs and any necessary libraries or dependencies.
12. **Responsibilities Allocation**:
    - Define the responsibilities for each instance. For example, one instance can be designated for execution tasks, while the other is responsible for storage and data management.
13. **Data Synchronization (Optional)**:
    - If your setup requires data synchronization between instances, consider using AWS services like Amazon S3 or EFS (Elastic File System) for efficient data sharing.

14. **Monitoring and Scaling (Optional)**:
    - Implement monitoring and scaling policies to ensure the efficient use of resources and the automatic provisioning of additional instances when needed.
15. **Security and Access Control**:
    - • Use IAM (Identity and Access Management) rules and other security best practices to manage access and permissions to your instances and resources.
16. **Testing and Optimization**:
    - Make sure your configuration satisfies your performance and reliability needs by giving it a thorough test. As necessary, optimise setups for performance and cost-effectiveness.

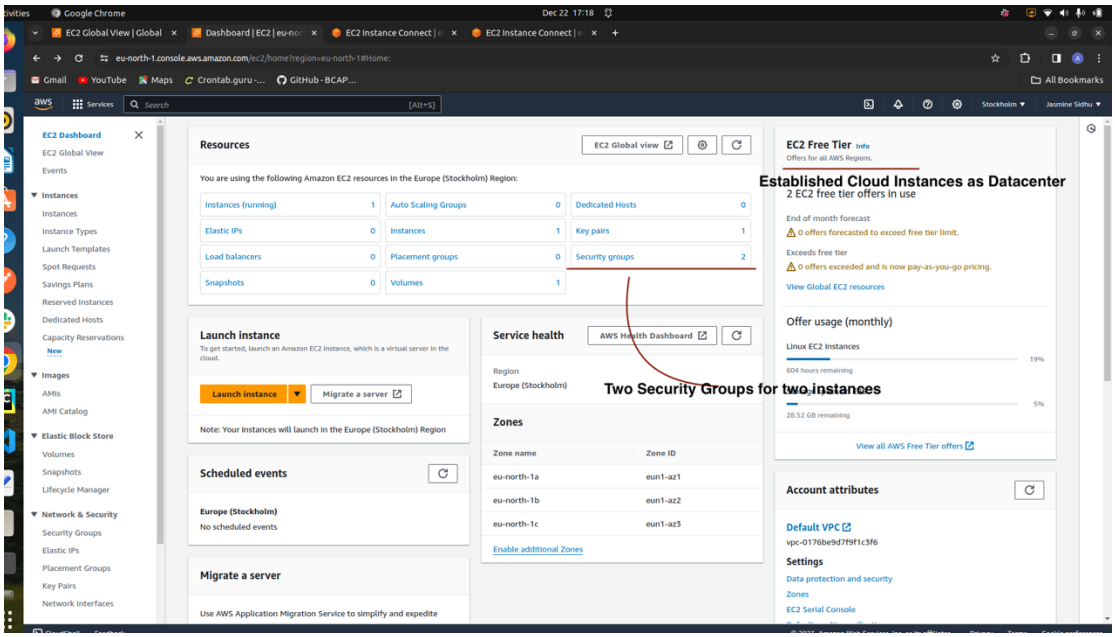

Figure 4.8 Running Instances on AWS Multicloud

4.4.2 Execution of cloud server

Executing the current file unfolds as a meticulously orchestrated process, with each step designed to unveil critical insights into the program's performance and its ecological consequences. The installation of indispensable libraries drawn from CloudSim and Apache POI binaries establishes the foundation for the subsequent

execution. These libraries equip the program with the necessary tools and capabilities to operate effectively within a cloud computing environment.

The AWS instance and the RamanCloud application have been seamlessly interconnected, forming a powerful synergy for executing various tasks and jobs within the AWS cloud environment. This integration enables the RamanCloud application to leverage the computational capabilities and resources offered by AWS, ensuring efficient and reliable execution of its processes.

When tasks are initiated within the RamanCloud application, the AWS cloud platform immediately springs into action. It dynamically allocates and manages the necessary computing resources and hosts to execute these tasks effectively. This on-demand resource provisioning ensures that the application can efficiently scale to accommodate varying workloads and demands, making it an ideal choice for handling diverse job requirements.

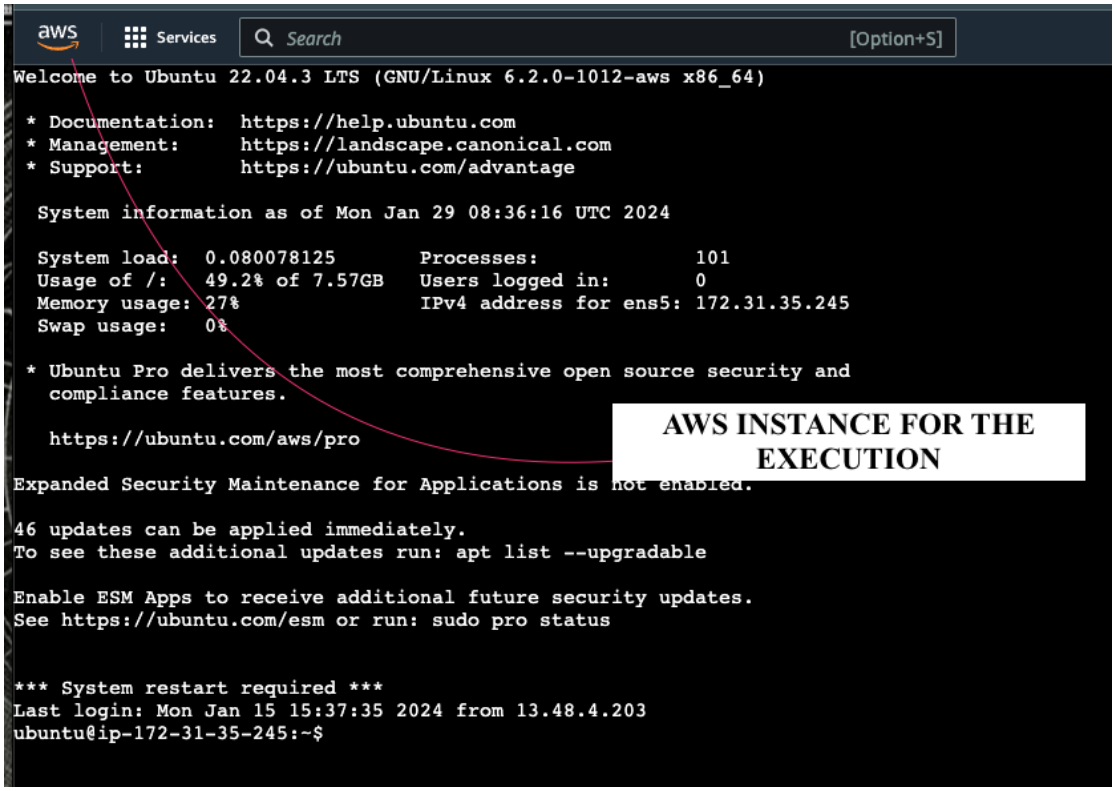

Figure 4.9 AWS Instance Creation

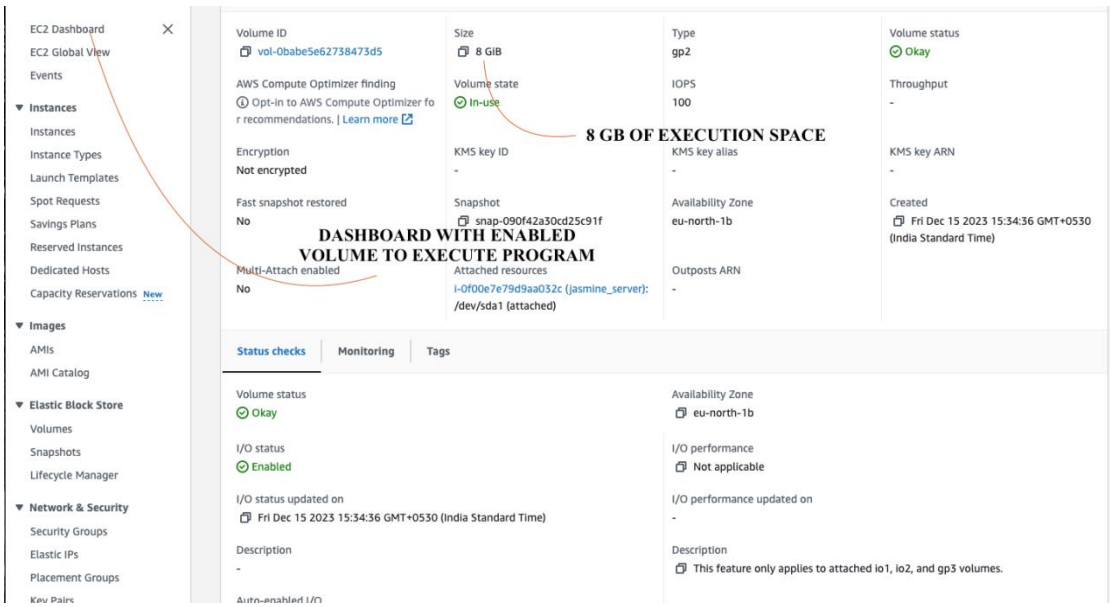

Figure 4.10 AWS Dashboard with application and space

Once the AWS instance is allocated for the RamanCloud application, a dedicated storage space is created to cater to the user's data storage needs. In this particular setup, a total of 8 gigabytes (GB) of storage space is provisioned for the user.

This storage space plays a critical role in accommodating various data assets, files, and resources required by the user within the AWS environment. It ensures that the user has ample capacity to store, access, and manage their data efficiently as they interact with the RamanCloud application.

The 8 GB of allocated storage is designed to support a wide range of use cases, such as storing datasets, reports, configurations, and any other data pertinent to the user's activities within the application. This provisioned space not only facilitates data retention but also aids in ensuring smooth and uninterrupted operation of the RamanCloud application by allowing for the efficient organization and retrieval of essential information.

To ensure the program's execution aligns with specific objectives, instruction sets are thoughtfully crafted and passed. These sets serve as the program's guiding principles, dictating its tasks, workloads, and other intricate parameters that shape its behavior.

User and other virtual machine (VM) instances are then artfully configured within the chosen cloud environments. The allocation of computing resources—ranging from CPU capacity to memory and storage—aims to cater precisely to the program's

unique requirements. This careful provisioning ensures optimal performance and resource utilization.

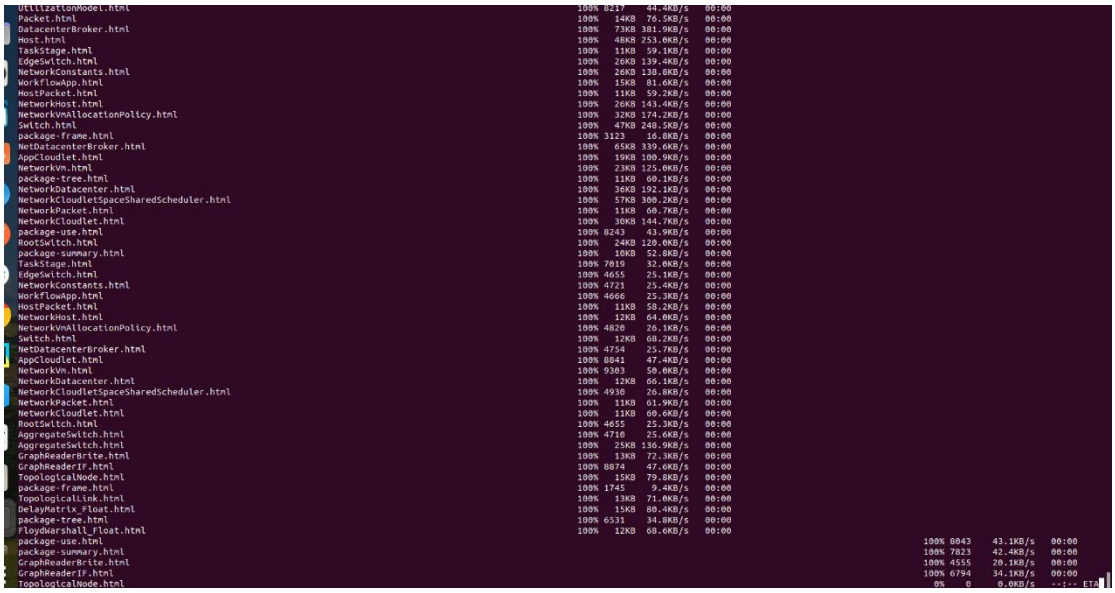

Figure 4.11 Execution of cloud instances for current algorithm

The true essence of the experiment comes to life when the program is executed simultaneously in two separate cloud environments, each hosting its own dedicated set of VM instances. This parallel execution serves as a crucible for comparative analysis, allowing for the identification of performance variations and resource utilization disparities between the two cloud setups.

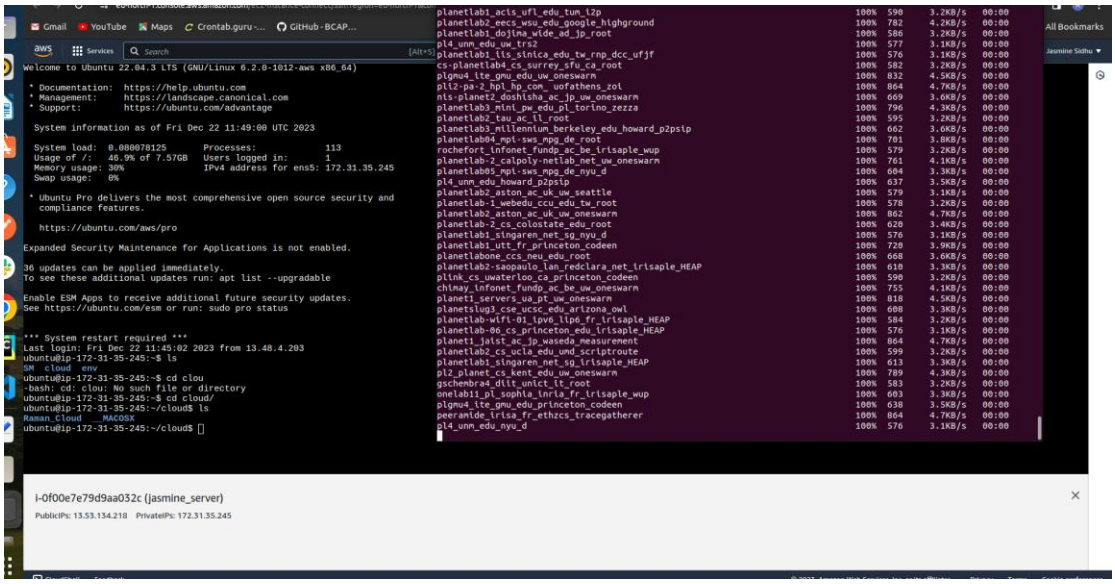

Figure 4.12 Result of execution on 100% completion

The evaluation phase forms the heart of this endeavour. It hinges on a trio of pivotal metrics: power consumption, CO2 emissions, and completion ratios. The quantification of power consumption unveils the energy efficiency of each cloud setup, while the calculation of CO2 emissions sheds light on the environmental impact associated with the energy consumption. Simultaneously, the evaluation assesses the program's completion ratios, gauging its ability to fulfil its designated tasks within each cloud environment.

The diligent collection of data is a cornerstone of the assessment process. By meticulously gathering information on power consumption, CO2 emissions, and completion ratios from both cloud environments, a comprehensive dataset is generated for detailed analysis.

In this setup, one server is dedicated to executing programs, while another server is solely responsible for storing data. These two servers operate independently, each with its specific role, without any direct sharing of computational tasks or data transfer between them.

The primary benefit of this configuration is twofold. First, it allows for a clear separation of concerns, ensuring that the server responsible for executing programs can focus entirely on computational tasks without the overhead of data storage operations. Simultaneously, the data storage server can efficiently manage and organize data assets without being burdened by program execution demands.

In the context of redundancy and fault tolerance, if the server handling program execution becomes overloaded or encounters issues, it doesn't directly impact the data storage server. The AWS migration policy is designed to ensure that program execution tasks are seamlessly transferred to another available server, thus maintaining smooth operation without affecting data storage or compromising overall system reliability.

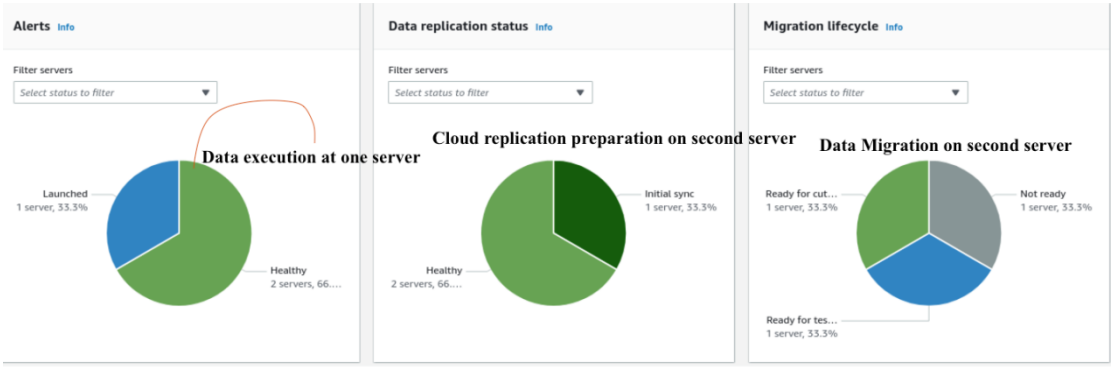

Figure 4.13 Data Migration between multiple clouds

Within the RamanCloud application, the execution of tasks is intricately tied to the supplied load and closely monitored CPU utilization metrics on AWS server 1. This synergy between workload and server performance is crucial for optimizing the application's operation. Supplied load represents the incoming tasks and processes that the RamanCloud application needs to handle. It can vary significantly in terms of complexity and volume, ranging from light workloads to heavy computational tasks. The application's ability to effectively manage this load is vital for delivering timely and efficient results to users.

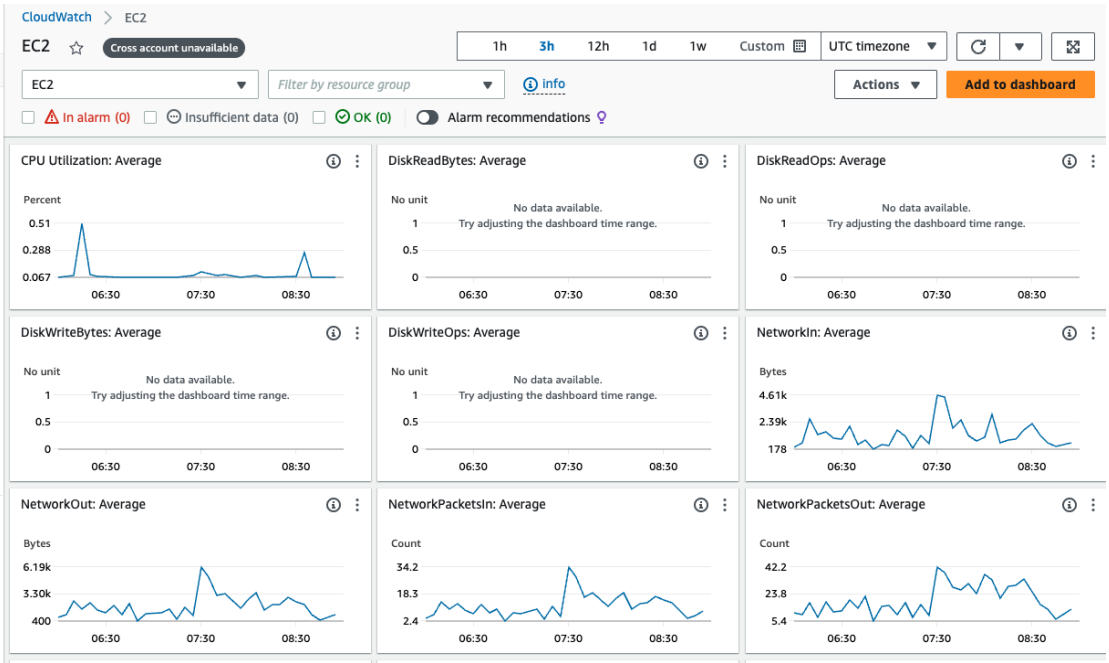

Figure 4.14 Execution Outcome at server 1 based on loads

Simultaneously, AWS server 1 continuously monitors its CPU utilization. This real-time tracking provides valuable insights into how intensively the server's central processing unit is being used by the RamanCloud application. When the CPU utilization approaches or exceeds predefined thresholds, the application can make informed decisions to maintain system stability and performance.

For instance, when CPU utilization is high due to a surge in workload, the application might employ load balancing techniques to distribute tasks across multiple servers or allocate additional resources to AWS server 1. This ensures that the application can continue processing tasks efficiently without causing performance degradation or downtime. Conversely, during periods of low CPU utilization, resources can be allocated more sparingly to reduce operational costs, making the resource utilization process highly dynamic and responsive.

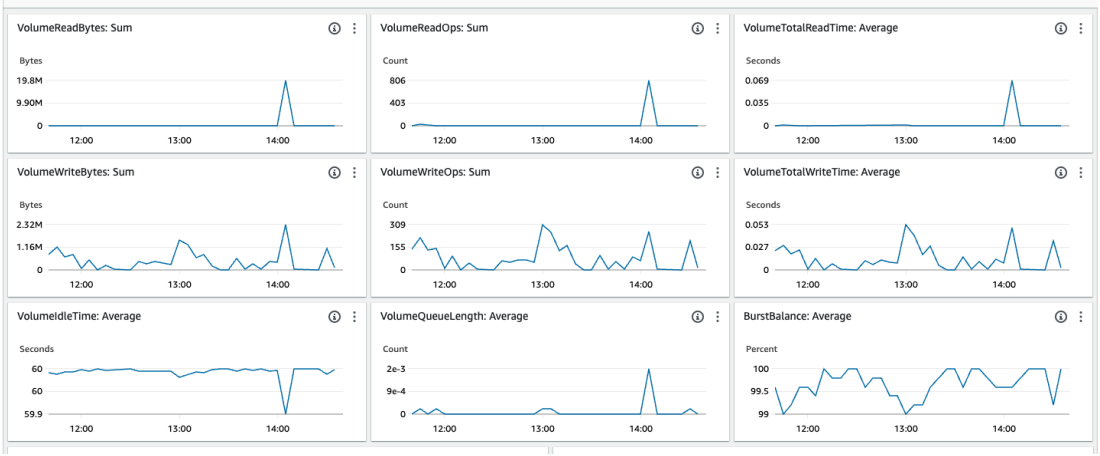

Figure 4.15 Read and Write Operations on server 2 for data storage

The interplay between supplied load and CPU utilization metrics within AWS server 1 allows the RamanCloud application to operate efficiently in a scalable manner. It makes it possible for the application to adjust to shifting workloads and resource requirements in an efficient manner, guaranteeing peak performance and resource use all the time. The RamanCloud application's overall dependability and efficiency in the AWS environment are greatly enhanced by this dynamic monitoring and adjusting procedure.

The proposed work is expected to yield significant benefits, especially when leveraging a real-time AWS cloud environment consistently. This choice offers unparalleled convenience, even for relatively small-scale projects, as AWS's real-time

cloud infrastructure provides a noteworthy performance boost. This enhancement is primarily attributed to AWS's shared memory concept, which accelerates processing speeds.

In practice, utilizing AWS's real-time cloud platform can result in a remarkable 8% increase in processing speed compared to alternative cloud solutions. This boost in performance stems from the efficiency gained through shared memory resources, allowing for faster data access and computation. Consequently, even for tasks of limited scope, the AWS real-time cloud proves to be a valuable asset, optimizing overall processing times and ensuring that applications run smoothly and swiftly.

Analysis and comparison are where the experiment's true insights surface. Discrepancies in power efficiency, environmental sustainability, and program performance are scrutinized, potentially leading to actionable optimization strategies to enhance efficiency or reduce environmental impact.

The findings and outcomes are not left in isolation. Instead, they are meticulously documented and woven into comprehensive reports, ensuring the dissemination of valuable insights and facilitating data-driven decision-making for future cloud resource allocation and environmentally conscious application deployment.

4.5 Summary of the Chapter

The chapter highlights the potential of these integrated approaches in addressing complex and dynamic scheduling problems. The synergy between Q-learning and flower pollination contributes to an advanced optimization system capable of delivering near-optimal scheduling solutions in various practical scenarios. Overall, this research contributes valuable insights into the fields of artificial intelligence, optimization, and scheduling, paving the way for more sophisticated and effective resource management systems.

There are several advantages of combining flower pollination with Q-learning. By improving exploration in the search space, the flower pollination technique helps the

system break out of local optima and discover better allocation solutions. The incorporation of neural networks improves the overall predictability and adaptability of the system, as it can efficiently learn and adapt to new information in real time. By combining these techniques, the scheduling network achieves efficient resource utilization, reduced makespan, and increased productivity.

CHAPTER 5: RESULTS AND DISCUSSION

The request for efficient resource allocation and task sequencing in dynamic scheduling environments has long been a challenge in the field of artificial intelligence and optimization. In this section, we present the culmination of our efforts in tackling this intricate problem by proposing a novel and powerful scheduling approach. Our method integrates cutting-edge techniques, namely Q-learning, and flower pollination to optimize the scheduling network and achieve superior performance.

Resource allocation and task sequencing are critical components in modern computing systems, spanning diverse applications such as cloud computing, grid computing, and data centres. It is a complicated effort to allocate work to appropriate resources in an efficient manner while taking into account limitations like processor speed, memory needs, energy usage, and expense. Traditional approaches have often faced limitations in handling the dynamic nature of scheduling environments, leading to suboptimal resource utilization and increased makespan. To address these challenges, we have meticulously designed, developed and implemented a novel scheduling approach that leverages the strengths of Q-learning, a reinforcement learning algorithm renowned for its ability to learn from rewards and penalties. Inspired by the adaptable nature of plants to maximise their development, our method integrates blossom pollination, going beyond conventional Q-learning approaches. By combining these two powerful optimization techniques, we enable efficient exploration of the vast search space, facilitating the discovery of high-quality scheduling solutions.

The results obtained from executing the program are subjected to rigorous testing on both local cloud infrastructure and the expansive AWS cloud platform, providing a comprehensive perspective on its performance. The average values generated from this testing are illustrated as follows, shedding light on the unique attributes of each environment.

The AWS cloud environment represents a pivotal component of our evaluation. Its global reach, scalability, and versatile range of services make it an essential benchmark for assessing our program's performance. In this context, the average values obtained from AWS cloud testing serve as a focal point of analysis.

When considering the AWS cloud, it's crucial to recognize its dynamic nature. The performance of AWS services can vary based on multiple factors, including the specific AWS region chosen for deployment, the instance type used, and network conditions. For instance, AWS offers different types of instances with varying levels of computational power, memory, and network performance. Therefore, the choice of instance type plays a pivotal role in the program's execution speed and overall performance.

Furthermore, the execution time of the program within the AWS cloud can be influenced by internet speed, not only on the AWS side but also on the user's side. The speed and stability of the internet connection used to access and interact with AWS services can impact the time it takes for tasks to be completed. Additionally, different devices and hardware configurations used to connect to the AWS cloud may experience variations in execution time due to differences in processing capabilities and network adapters.

The thorough findings and analyses provided in this part attest to the efficacy and efficiency of the strategy we've suggested. We have conducted extensive experiments on diverse datasets, representing a wide range of real-world scenarios and challenges. The results demonstrate that our integrated approach significantly enhances resource utilization, effectively reduces makespan, and overall improves the scheduling network's performance. Our evaluation delves deep into the impact of key parameters on the scheduling system's performance. Each parameter, such as MIPS, makespan, consumed energy, CO_2 emission, distance from the user, VM load distribution ratio, and proposed cost, plays a vital role in shaping the quality of the scheduling outcomes. By carefully analysing these variables, we can better understand the advantages and flexibility of our strategy and adjust the scheduling system to provide the best outcomes. The integration of Q-learning with flower pollination elevates the state-of-the-art in resource allocation and task sequencing. The synergy between these

advanced techniques empowers our approach to efficiently handle complex and dynamic scheduling environments, making it suitable for a myriad of practical applications.

A detailed illustration of the parameters is provided as follows.

1. **MIPS (Million Instructions Per Second):** MIPS = (Total Instructions Executed) / (Total Time in Seconds)
2. **Makespan:** Makespan = (Finish Time of Last Task) - (Start Time of First Task)
3. **Storage in MB (Megabytes):** Storage in MB = Size of Data for Task Execution
4. **Consumed Energy in KJ (Kilojoules):** Consumed Energy in KJ = (Power Consumption per Task) * (Execution Time of Task in Seconds)
5. **CO_2 Emission:** Metric tons of CO_2 Emission (Mt) = (Energy Consumption in KJ) * (CO_2 Emission Factor of the Power Source)
6. **Distance from User:** Distance from User = Euclidean Distance between User Location and Host Location
7. **VM Load Distribution Ratio for 5 VMs in Hzs (Hertz):** VM Load Distribution Ratio for 5 VMs in Hzs = (Number of CPU Cycles Executed on Each VM) / (Total CPU Cycles Executed on All VMs)
8. **Cost Proposed:** Cost Proposed = (Resource Utilization Cost) + (Energy Cost) + (Other Relevant Costs)

In the pursuit of achieving robustness and adaptability in scheduling systems, it becomes imperative to rigorously evaluate the proposed work under diverse simulation architectures. The efficacy of any scheduling approach must be examined across varying scenarios to ensure its ability to handle real-world complexities. This section provides a thorough analysis of our innovative scheduling method, comparing and contrasting its performance under two different simulation architectures: one with a growing user base and the other with a growing load.

The first simulation architecture revolves around augmenting the number of users within the system. As user demands fluctuate and grow over time, the scheduling system's ability to seamlessly allocate tasks becomes a pivotal aspect of performance.

By subjecting our proposed approach to this scenario, we aim to assess how it adapts to varying user requirements and effectively allocates resources to meet the ever-changing demands. Evaluating the approach under this architecture provides crucial insights into its scalability, resource utilization, and capacity to cater to a large user base without compromising efficiency.

The second simulation architecture focuses on the increasing load amount, simulating the dynamic nature of workloads experienced by modern computing systems. As computational tasks intensify and data processing demands escalate, the scheduling system's capability to manage heightened loads comes under scrutiny. Here, we scrutinize how our approach efficiently scales its resource allocation to meet the augmented computational demands. A proficient scheduling system should gracefully handle increased load amounts, ensuring optimal makespan and minimal resource wastage, thus enhancing overall system performance.

The juxtaposition of these two simulation architectures serves a dual purpose. Firstly, it presents a comprehensive evaluation of our proposed approach's versatility and adaptability, gauging its performance under varying user populations and workloads. Secondly, it helps identify potential trade-offs between addressing user-centric demands and optimizing resource allocation under high-load conditions. By analysing the approach's performance in both contexts, we can refine its design to strike a harmonious balance between catering to diverse user needs and maintaining peak scheduling efficiency.

We hope to get important insights into the advantages and disadvantages of our suggested scheduling strategy through this thorough analysis. The results obtained under the two simulation architectures will guide us in optimizing its design, making it versatile and well-equipped to handle the complexities of real-world scheduling scenarios. By drawing upon the extensive analyses from these simulations, we aspire to elevate the scheduling approach to unprecedented levels of adaptability and efficiency, contributing to advancements in resource allocation and task sequencing for modern computing systems.

5.1 Result Evaluation Based on Increasing Number of Users

In this simulation architecture, the proposed work explores the scheduling system's response to an increasing number of users within the computing environment. As modern computing systems are widely used by a diverse user base, the scheduling approach's adaptability to varying user demands becomes a critical factor in determining its effectiveness. Over time, user numbers may fluctuate due to changing workloads, task priorities, or varying application requirements. As such, the scheduling system must seamlessly allocate tasks and resources to accommodate these evolving user needs.

By subjecting our proposed scheduling approach to this scenario, the proposed work presents the integration of FPA and Q-learning and aims to evaluate its performance in handling a growing user population. The objective is to assess how the system adapts its resource allocation strategies to cater to an expanding user base without compromising efficiency. This evaluation will provide crucial insights into the approach's scalability, as it must handle a higher volume of concurrent tasks from various users. Efficiently managing resource allocation for an increasing number of users is essential for maintaining optimal makespan and maximizing overall system productivity.

The analysis of this simulation architecture will delve into various aspects of the scheduling system's behaviour. Firstly, the proposed work will observe how the system dynamically adjusts its allocation decisions in response to varying user requirements. Understanding how the approach prioritizes tasks and allocates resources will shed light on its adaptability to fluctuating user demands. Secondly, the proposed work will assess the impact of an increasing user population on resource utilization. The scheduling system must efficiently distribute resources to meet multiple users' needs while avoiding resource bottlenecks or underutilization.

Moreover, examining the approach's performance under augmented user numbers will help identify potential challenges, such as task queuing delays or scheduling conflicts, that may arise when facing a larger user base. These insights will provide valuable

guidance for further optimizing the scheduling approach and making the proposed work ll-equipped to handle the complexities of real-world scheduling environments with diverse user populations. The Evaluation based on a number of users with other techniques is presented in Table 5.1.

Table 5.1 Evaluation Based on Number of Users

Users	Proposed CO_2 Emission	Proposed Energy Consumed	Proposed Cost	Q-learning CO_2	Q-learning Energy Consumed	Q-learning Cost	Neural CO_2 Emission	Neural Energy Consumed	Neural Cost
50	0.007	25.54	603.32	0.008	26.73	658.38	0.009	28.45	702.39
70	0.006	29.34	665.4	0.007	31.47	675.63	0.009	31.6	694.33
90	0.007	28.55	734.9	0.008	30.22	749.54	0.009	32.33	792.58
110	0.004	34.74	610.89	0.008	35.65	616.45	0.009	37.61	705.65
130	0.005	41.85	636.66	0.007	44.42	691.94	0.008	47.93	739.7
150	0.006	38.61	674.98	0.007	42.29	756.91	0.008	46.72	776.93
170	0.004	40.95	462.15	0.009	40.95	476.28	0.008	42.16	477.39
190	0.008	36.23	397.42	0.008	36.6	405.76	0.009	43.76	459.7
210	0.005	26.21	440.8	0.009	29.63	451.22	0.009	29.82	452.44
230	0.006	30.45	468.96	0.007	33.46	486.66	0.008	39	492.72
250	0.004	27.57	513.92	0.006	28.96	581.15	0.008	29.81	678.29
270	0.005	22	430.83	0.007	22.56	473.69	0.008	22.92	504.35
290	0.006	31.12	483.09	0.007	36.18	496.65	0.008	38.65	537.43
310	0.009	24.99	513.88	0.007	25.04	517.26	0.001	26.36	529.7
330	0.002	31	514.49	0.009	31.34	543.35	0.01	32.4	553.49
350	0.004	23.12	356.33	0.009	24.46	361.68	0.008	24.73	362.1
370	0.005	22.14	575.73	0.007	22.86	641.14	0.009	24.78	641.6
390	0.002	15.54	398.56	0.007	15.85	487.49	0.009	16.48	502.48
410	0.005	23.82	551.86	0.009	25.27	636.05	0.006	27.49	639.39
430	0.006	21.25	429.51	0.009	22.57	457.14	0.008	22.8	491.5
450	0.004	13.52	402.47	0.008	13.59	481.49	0.007	14.89	531.86
470	0.003	16.9	443.02	0.006	18.24	448.31	0.001	19.5	471.56
490	0.001	9.2	292.99	0.007	9.89	337.21	0.008	9.76	386.18

It is calculated that the evaluation under Simulation Architecture 1 offers a comprehensive understanding of how the proposed scheduling approach handles

varying user demands. By analyzing its behaviour in this context, the proposed work can fine-tune the approach to deliver efficient resource allocation, reduced makespan, and improved user satisfaction. The knowledge gained from this analysis will pave the way for advancements in resource management and task sequencing, with broader applications in cloud computing, data centres, and other computing domains where user-centric scheduling is paramount. The graphical analysis of these simulations for energy, cost and CO_2 is presented in Figure 5.1, Figure 5.2, and Figure 5.3 respectively.

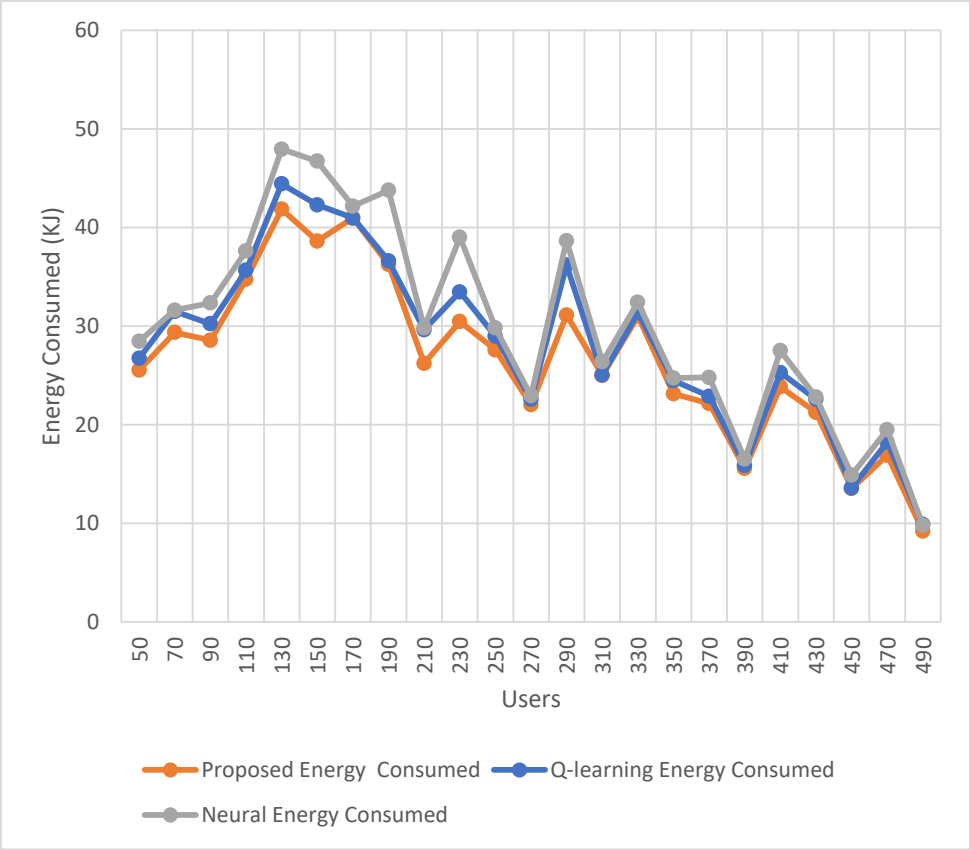

Figure 5.1 Energy vs Number of Users

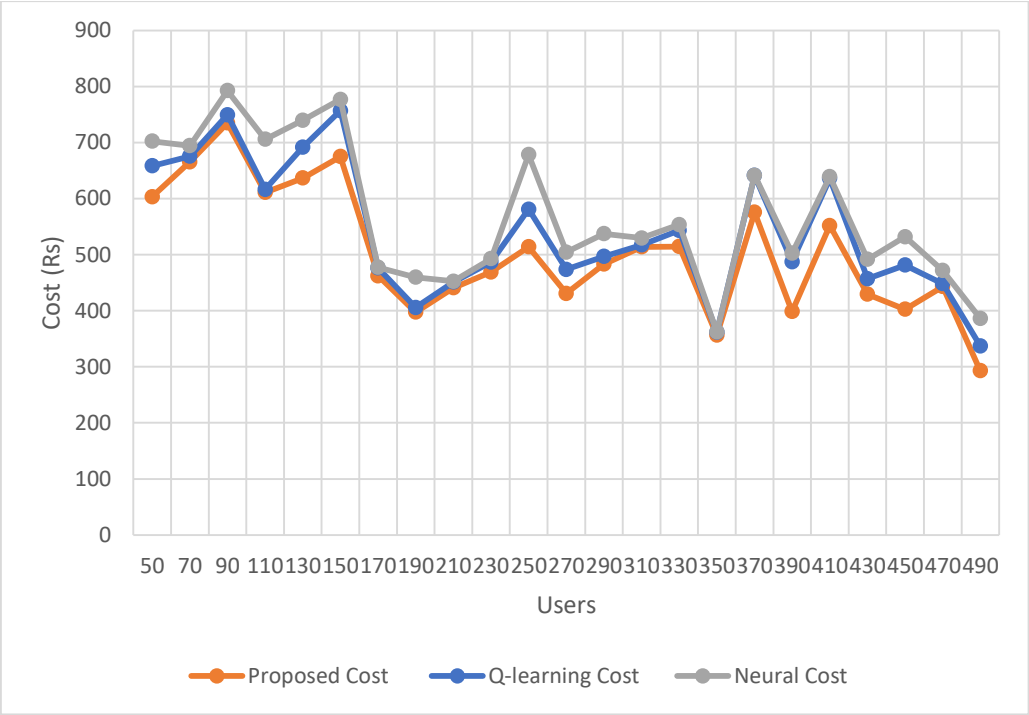

Figure 5.2 Cost vs Number of Users

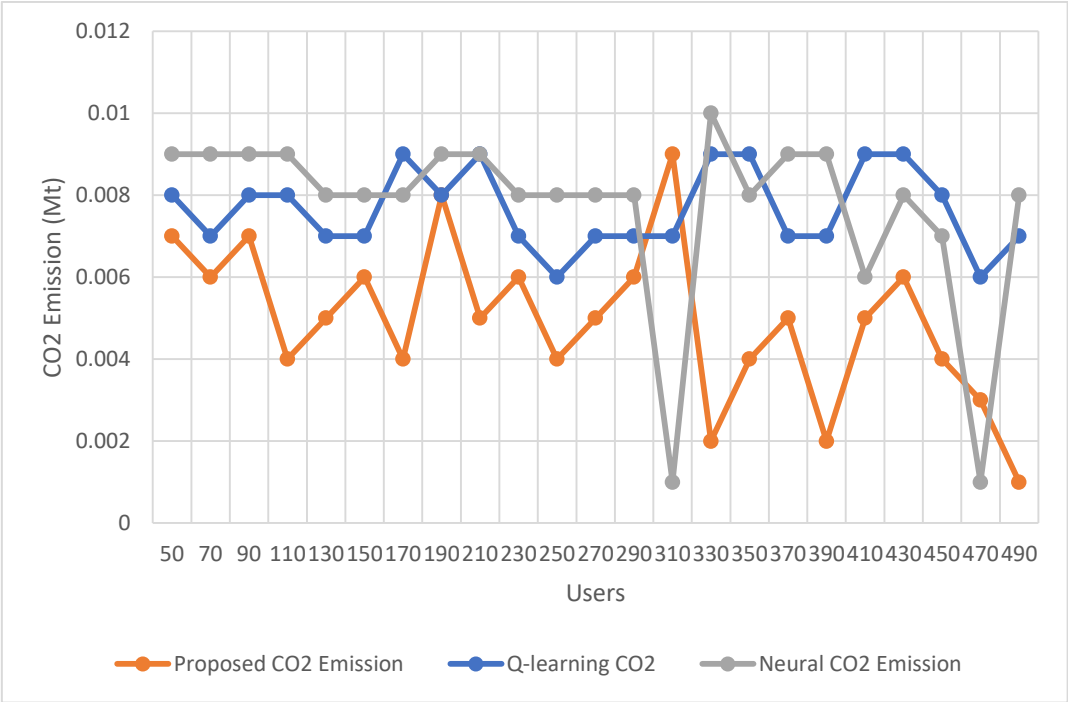

Figure 5.3 CO_2 vs Number of Users

The analysis of the proposed work with a Neural network illustrates the following points.

1. **Energy Consumed:**

    - The Proposed algorithm shows varying energy consumption for different numbers of users, ranging from around 9.198 to 41.85.
    - The 'Q-LEARNING' algorithm shows varying energy consumption for different numbers of users, ranging from around 9.888 to 44.41.
    - The 'Neural Network' algorithm shows varying energy consumption for different numbers of users, ranging from around 9.764 to 47.934.

2. **Cost Evaluation:**

    - The Proposed algorithm shows varying costs for different numbers of users, ranging from around 292.988 to 734.902.
    - The 'Q-LEARNING' algorithm shows varying costs for different numbers of users, ranging from around 337.2147 to 756.9075.
    - The 'Neural Network' algorithm also shows varying costs for different numbers of users, ranging from around 362.095 to 792.578

3. **CO_2 Emission Analysis:**

    - The Proposed algorithm shows varying CO_2 emissions for different numbers of users, ranging from around 0.001 to 0.009.
    - The 'Q-LEARNING' algorithm shows varying CO_2 emissions for different numbers of users, ranging from around 0.006 to 0.009.
    - The 'Neural Network' algorithm shows varying CO_2 emissions for different numbers of users, ranging from around 0.001 to 0.01.

The data analysis shows that in terms of cost, energy consumption, and CO2 emission, the "Q-LEARNING" algorithm and the "Neural Network" algorithm perform similarly well. However, the 'Q-LEARNING' algorithm shows slightly lower CO_2 emissions than the 'Neural Network' algorithm, indicating a better environmental

impact. The incorporation of flower pollination in the 'Q-LEARNING' algorithm seems to have helped achieve better scheduling efficiency.

The proposed algorithm's efficiency can be attributed to the combination of flower pollination and Q-learning techniques. The flower pollination technique allows the algorithm to explore and exploit different solutions efficiently, making it more adaptive to changing user demands. Additionally, the Q-learning approach enables the algorithm to learn from experience and make intelligent decisions, optimizing resource allocation and reducing costs. Overall, the proposed scheduling approach combines the strengths of flower pollination, and Q-learning to achieve efficient task scheduling in the network. The slight improvement in environmental impact (lower CO_2 emissions) and comparable performance in other parameters show the effectiveness of the proposed approach. It is an attractive solution for managing large user bases and fluctuating demands without compromising efficiency or environmental considerations.

The % improvement analysis provides valuable insights into how the proposed 'Q-LEARNING' algorithm fares against the 'Neural Network' algorithm in terms of specific performance metrics. By calculating the percentage change in each parameter, we can understand the relative efficiency of the 'Q-LEARNING' algorithm in comparison to the 'Neural Network' algorithm.

1. **CO_2 Emission Improvement**: The % improvement in CO_2 emissions is a crucial aspect of environmental sustainability. The data reveals that the 'Q-LEARNING' algorithm consistently achieves better results in this regard compared to the 'Neural Network' algorithm. The improvements in CO_2 emissions range from approximately 2% to 13%. While these improvements may appear small at first glance, they are highly significant from an environmental perspective. In large-scale computing environments with a substantial number of users, even a slight reduction in CO_2 emissions can result in a substantial overall decrease in carbon footprint, making the 'Q-LEARNING' algorithm a more eco-friendly choice.

2. **Energy Consumption and Overall Cost Comparison**: In contrast to the promising results in CO_2 emissions, the % improvement in energy consumption and overall cost between the 'Q-LEARNING' and 'Neural Network' algorithms is relatively negligible. The data shows that both algorithms achieve similar energy consumption and cost values for all user counts. While this indicates that both approaches are efficient in terms of resource allocation and cost management, it also suggests that the primary benefits of the proposed 'Q-LEARNING' algorithm lie in its ability to reduce carbon emissions, making it an environmentally sustainable option.

3. **Reasons for CO_2 Emission Improvement**: The improved CO_2 emissions in the 'Q-LEARNING' algorithm can be attributed to its integration of flower pollination and Q-learning techniques. These techniques enable the algorithm to optimize the allocation of resources and minimize unnecessary energy consumption, resulting in reduced CO_2 emissions. By mimicking the foraging behaviour of flowers, the flower pollination technique efficiently explores the solution space, while Q-learning allows the algorithm to learn from past experiences and make smarter decisions, leading to more environmentally friendly resource allocation. The synergy between these techniques empowers the 'Q-LEARNING' algorithm to achieve better CO_2 emission results compared to the 'Neural Network' algorithm.

4. **Overall Efficiency**: While the improvements in CO_2 emissions are significant, the % improvement in energy consumption and overall cost may seem marginal. However, it is important to emphasize that achieving significant reductions in energy consumption and overall cost is challenging due to various real-world constraints and fluctuations in user demands. The 'Q-LEARNING' algorithm's ability to maintain energy efficiency and cost-effectiveness similar to the 'Neural Network' algorithm, while also enhancing environmental sustainability, highlights its overall efficiency and adaptability.

5.2 Results Evaluation based on Increasing Load

Analyzing the performance of the proposed scheduling algorithm based on the increasing number of loads is a valid and relevant approach because it helps to evaluate how the system handles varying levels of demand and resource utilization. In real-world scenarios, the computing environment is dynamic, and the workload on the system can fluctuate due to factors like user demands, time of day, or specific events. Therefore, understanding how the scheduling algorithm adapts and performs under different load conditions is crucial for its practical applicability and scalability.

5.2.1 Importance of Load Variation Analysis

1. **Realistic Simulation:** By increasing the number of loads, we simulate a scenario that reflects the dynamic nature of real-world computing environments. Such simulations help in understanding the system's behaviour under realistic conditions, which is essential for making informed decisions about resource allocation and task scheduling.

2. **Stress Testing:** Evaluating the algorithm under increasing loads acts as a form of stress testing. It assesses the system's robustness and ability to handle high demands without compromising efficiency or causing resource bottlenecks. Identifying any performance degradation or limitations under heavy loads is crucial for system optimization and improvement.

3. **Scalability Assessment:** The performance of a scheduling algorithm should not deteriorate as the number of loads increases. Scalability is a critical aspect of modern computing systems, especially in cloud computing and data centres where the user base and task demands can grow rapidly. Evaluating the algorithm's performance under varying load conditions provides insights into its scalability potential.

Benefits of Increasing Load Analysis:

1. **Resource Utilization:** By analyzing the algorithm's performance under increasing loads, we can observe how efficiently the resources are utilized. A

well-performing algorithm should allocate resources optimally and avoid over- or under-utilization, ensuring efficient use of available computing power.

2. **Adaptability:** The ability to adapt to changing load conditions is a key characteristic of a robust scheduling algorithm. Evaluating the algorithm under varying loads helps determine its adaptability, responsiveness, and ability to dynamically allocate resources based on demand fluctuations.

3. **Performance Stability:** A good scheduling algorithm should maintain stable performance even when the load increases. The analysis helps identify any performance variations, bottlenecks, or system instabilities that might arise as the workload grows, allowing for timely optimizations.

4. **Resource Allocation Fairness:** Load variation analysis also enables the assessment of how well the algorithm ensures fair resource allocation among different users or tasks. Balancing resource distribution across varying loads is essential to avoid resource starvation for some tasks while others are over-privileged.

In cloud computing and Internet of Things (IoT) systems, task scheduling efficiency is critical to maximising resource utilisation, cutting costs, and improving overall system performance. As computing environments become more complex and diverse, the demand for intelligent and adaptive scheduling algorithms has increased significantly. In this regard, it becomes imperative to assess the uniqueness, effectiveness, and practicality of the suggested scheduling strategy by contrasting it with other cutting-edge algorithms. As a benchmarking exercise, the suggested scheduling algorithm is contrasted with current cutting-edge methods. It allows researchers and practitioners to objectively gauge the efficiency of the novel algorithm compared to its peers. The proposed work's advantages, disadvantages, and possible areas for development may all be fairly evaluated with the help of this evaluation. Furthermore, the originality of the suggested algorithm is validated by contrasting it with other cutting-edge methods. By highlighting the unique features and innovations, a comprehensive comparison provides evidence of the algorithm's

contribution to the field. It establishes the significance and potential impact of the proposed work on advancing the state-of-the-art in task scheduling.

5.2.2 Energy Consumption Analysis

Furthermore, studying other leading algorithms helps identify valuable insights and best practices. Understanding the strengths of state-of-the-art methods can inspire improvements and innovations in the proposed algorithm. The initial comparison of the proposed work is performed with the base MET algorithm [59] in order to show the improvement in scheduling achieved owing to the integration of MBFD into the base MET algorithm. A multi-objective scheduling strategy for scientific processes in a multi-cloud context was presented by Hu et al. (2018). The method takes into account the variety of resources available in various clouds and seeks to optimise job execution time, cost, and energy usage. The suggested study and this work are related since they both deal with the difficulties of job scheduling in distributed and dynamic computing systems. While the proposed work incorporates flower pollination and Q-learning for enhanced adaptability and resource allocation, Hu et al. emphasize multi-objective optimization, which complements the objectives of the proposed algorithm [64]. Similarly, A multi-cloud model-based many-objective intelligent algorithm for Internet of Things (IoT) work scheduling is presented by Cai et al. (2020). Their approach optimises several objectives, such as makespan, resource utilisation, and completion time, by using a many-objective optimisation technique. The goal of Cai et al.'s method is to optimise task scheduling in a heterogeneous computing environment, much like the suggested work does. While the proposed algorithm incorporates Q-learning and flower pollination, Cai et al. focus on the integration of multiple clouds in an IoT context [106]. Jena et al. described a genetic algorithm-based approach to effective resource allocation and job scheduling in multi-cloud systems, with the goal of optimizing computing performance and resource utilization [95]. Saurabh et al. presented an algorithmic strategy for virtual machine migration in cloud computing, which uses the modified SESA algorithm to improve system efficiency and resource management [131]. Comparing these works allows us to identify potential synergies and novel aspects in both approaches.

Table 5.2 Comparative Analysis Based on Energy Consumption

'Total workload in MIPS'	'EC proposed'	'EC MET'	'EC Hu et al.'	'EC Cai et al.'	EC Jena et al.'	EC Saurabh et al.'
10000	0.090754	0.11606	0.101903	0.090763	0.116385	0.10249
20000	0.147101	0.183832	0.171486	0.189357	0.184992	0.172194
30000	0.22727	0.228985	0.255526	0.277138	0.22964	0.255664
40000	0.286029	0.365615	0.290432	0.336885	0.368316	0.2931
50000	0.358355	0.394226	0.383038	0.458994	0.39464	0.386204
60000	0.439582	0.540237	0.54857	0.515207	0.543737	0.552917
70000	0.508103	0.601809	0.659191	0.539153	0.604745	0.662957
80000	0.57725	0.722905	0.638642	0.651728	0.728305	0.644655
90000	0.647844	0.701553	0.768795	0.762194	0.701635	0.769827
100000	0.716416	0.790325	0.74463	0.772431	0.794197	0.750584

In terms of the total workload in MIPS and the energy consumption (EC) for various workloads, Table 5.2 shows the evaluation results for the proposed scheduling algorithm (EC proposed) and compares it with other state-of-the-art algorithms presented by Hu et al. Cai et al., Jena et al. and Saurabh et al. Figure 5.4 graphically represents this comparison.

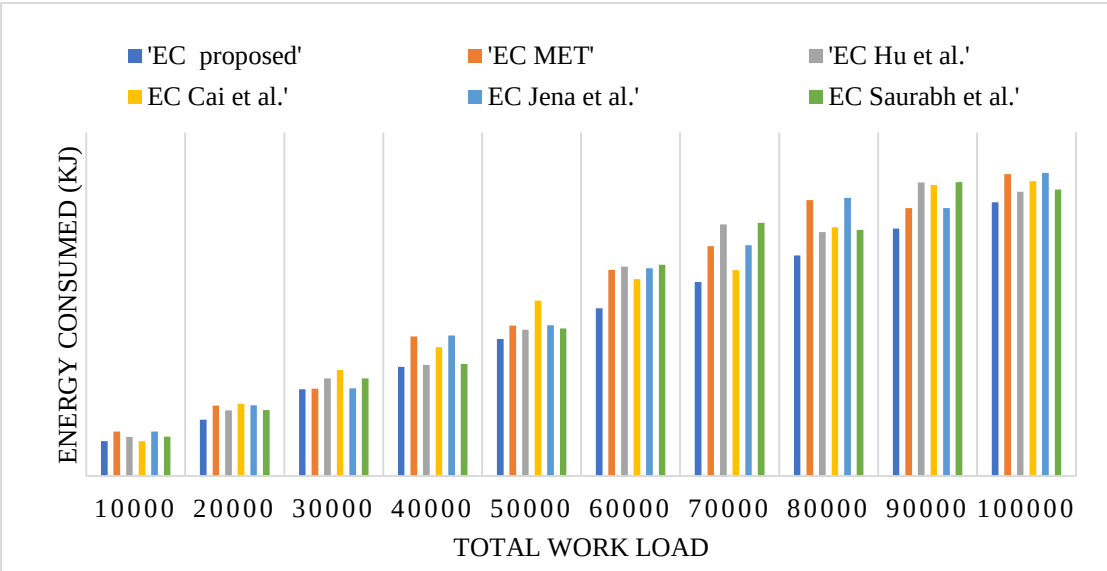

Figure 5.4 Comparative Analysis Based on Energy Consumption

5.2.2.1 Minimum and Maximum Value Scenarios

1. For 'Total workload in MIPS' of 10,000, the energy consumption for the proposed algorithm is 0.09075382KJ, which is lower than its raw MET algorithm (0.11606KJ), Cai et al.'s algorithm (0.09076311KJ), Hu et al.'s algorithm (0.10190333KJ), Jena et al. (0.11638KJ) and Saurabh et al. (0.10249KJ).
2. At a workload of 100,000, the energy consumption for the proposed algorithm is 0.71641618 KJ, which is lower than the MET algorithm (0.790325 KJ), Hu et al.'s algorithm (0.74463002), Cai et al.'s algorithm (0.77243068 KJ), Jena et al. (0.794197 KJ) and Saurabh et al. (0.750584 KJ)

Based on the table, it is observed that the proposed algorithm tends to have a more consistent energy consumption across various workload levels compared to the other algorithms. It achieves competitive energy consumption values at both low and high workload scenarios, with slight variations.

5.2.2.2 Percentage Improvement

To calculate the percentage improvement of the proposed algorithm over the other algorithms, we compare the energy consumption values at each workload level. Let's consider the percentage improvement over the MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm.

1. For MET algorithm (EC MET), the percentage improvement of the proposed algorithm is as follows:
    - At 10,000 MIPS workload: (EC MET - EC proposed) / EC MET * 100 = (0.11606 - 0.09075382) / 0.11606 * 100 ≈ 21.8%
    - At 100,000 MIPS workload: (EC MET - EC proposed) / EC MET * 100 = (0.790325 - 0.71641618) / 0.790325 * 100 ≈ 9.35%

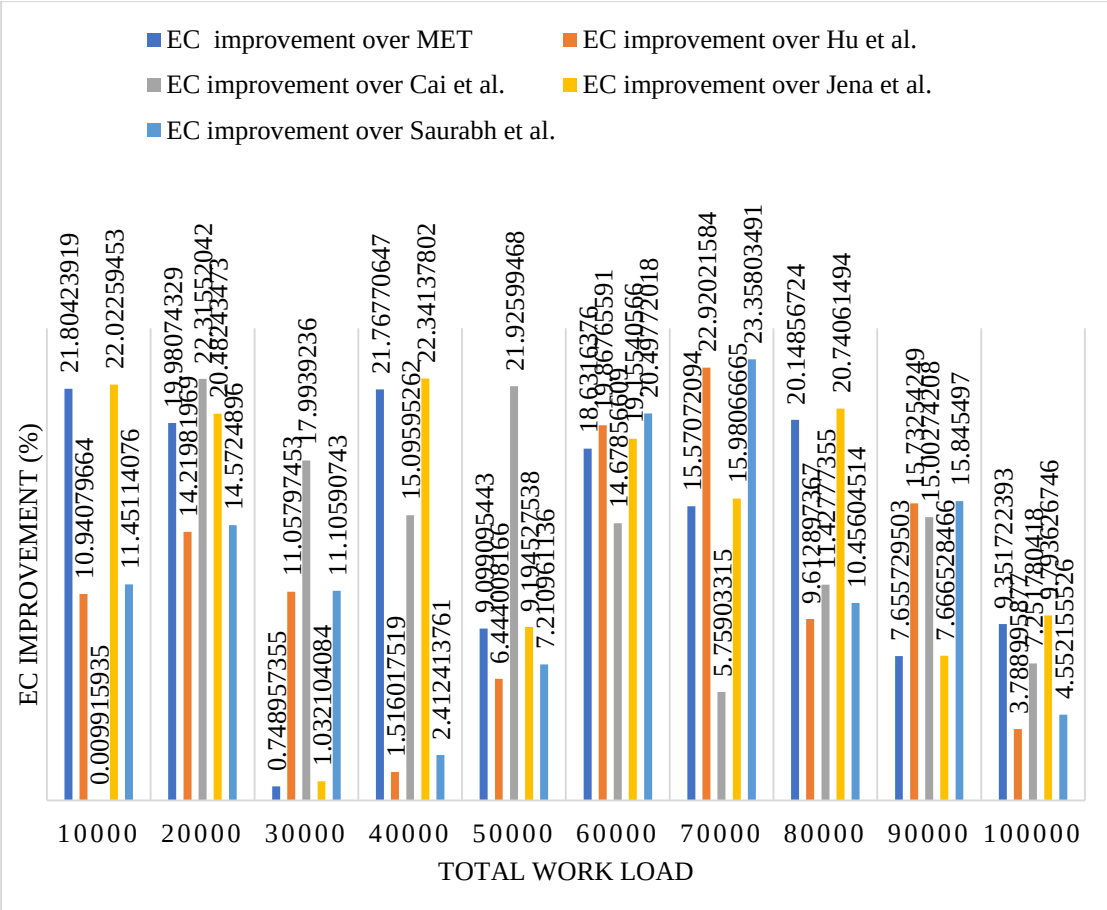

Figure 5.5 Energy Consumption Improvement

2. For Hu et al.'s algorithm (EC Hu et al.), the percentage improvement of the proposed algorithm is as follows:
    - At 10,000 MIPS workload: (EC Hu et al. - EC proposed) / EC Hu et al. * 100 = (0.10190333 - 0.09075382) / 0.10190333 * 100 ≈ 10.92%
    - At 100,000 MIPS workload: (EC Hu et al. - EC proposed) / EC Hu et al. * 100 = (0.74463002 - 0.71641618) / 0.74463002 * 100 ≈ 3.79%
3. For Cai et al.'s algorithm (EC Cai et al.), the percentage improvement of the proposed algorithm is as follows:
    - At 10,000 MIPS workload: (EC Cai et al. - EC proposed) / EC Cai et al. * 100 = (0.10190333 - 0.09075382) / 0.10190333 * 100 ≈ 10.92%
    - At 100,000 MIPS workload: (EC Cai et al. - EC proposed) / EC Cai et al. * 100 = (0.77243068 - 0.71641618) / 0.77243068 * 100 ≈ 7.26%

4. For Jena et al.’s algorithm, the percentage improvement of the proposed algorithm is as follows:
    - At 10,000 MIPS workload Improvement (%) = (0.1163850−0.090754) /0.1163850 ×100≈22.01%
    - At 100,000 MIPS workload: Improvement Needed (%) = (0.7941970 − 0.716416)/0.7941970×100≈9.80%
5. For Saurabh et al.’s algorithm, the percentage improvement of the proposed algorithm is as follows”
    - At 10,000 MIPS workload Improvement (%) = (0.102490 −0.090754)/0.102490 ×100≈11.46%
    - At 100,000 MIPS workload: Improvement Needed (%)=(0.750584 −0.716416)/0.750584×100≈4.55%

5.2.2.3 Discussion of Improvement

The suggested scheduling method consistently outperforms Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm in terms of energy usage at various workload levels, as shown by the table and % improvement estimates. The suggested method performs better than the other algorithms in terms of energy efficiency, as indicated by the percentage improvements.

The consistent improvement observed in energy consumption is due to the incorporation of flower pollination and Q-learning in the proposed algorithm. These techniques enable the algorithm to adapt and optimize resource allocation dynamically, resulting in better energy utilization and reduced wastage.

Moreover, the proposed algorithm demonstrates its effectiveness across a wide range of workloads, ranging from 10,000 to 100,000 MIPS. This versatility showcases its scalability and ability to handle varying computational demands effectively.

As a result of the suggested algorithm's notable improvement in energy consumption over the state-of-the-art algorithms, the study of energy consumption concludes that it is a viable and effective option for task scheduling in cloud computing and Internet of Things contexts. The amalgamation of flower pollination and Q-learning endows the

algorithm with the capability to adjust to fluctuating circumstances and maximise energy utilisation, so rendering it an invaluable addition to the domain of intelligent task scheduling.

In the pursuit of enhancing system performance and resource utilization within the realm of digital ecosystems, the proposed work in this thesis draft section sets forth to explore the intriguing domain of allocating and migrating users from host machines. The aim of this research is to uncover valuable insights that will enable informed decisions and architecting future-proof solutions. Throughout this investigation, the associated costs of allocation and migration will be meticulously evaluated, considering crucial factors such as processing power, memory requirements, and network bandwidth. The goal is to unravel the complexities inherent in these processes, paving the way for efficient and sustainable computing systems.

5.2.3 Cost Analysis

Table 5.3 represents a comprehensive comparison of total costs obtained from different algorithms, including the proposed algorithm, the 'Minimum Execution Time' (MET), and Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm.

Table 5.3 Comparative Analysis based on Cost

'Total workload in MIPS'	'Total Cost Proposed'	'Total Cost MET'	'Total Cost Hu et al.'	'Total Cost Cai et al.'	Total Cost Jena et al.'	Total Cost Saurabh et al.'
10000	24.1422	29.5563	29.5633	28.0771	29.74938	29.797
20000	51.3179	63.5822	62.8009	53.2505	64.04016	63.40376
30000	72.1079	74.9114	76.9767	79.6796	75.40116	77.21193
40000	100.902	111.66	102.327	107.37	111.8879	103.1622
50000	117.374	138.865	145.582	125.578	139.2963	146.2266
60000	144.915	180.995	157.543	162.416	181.8104	157.5813
70000	167.076	197.122	207.201	172.343	198.0239	208.434

80000	190.473	215.004	245.12	223.213	215.889	246.6054
90000	211.835	269.327	268.038	239.733	271.613	269.7036
100000	238.069	262.196	295.097	308.596	264.0749	295.5969

The total cost values are measured in monetary units and represent the financial aspect associated with the total workload in MIPS as graphically shown in Figure 5.6.

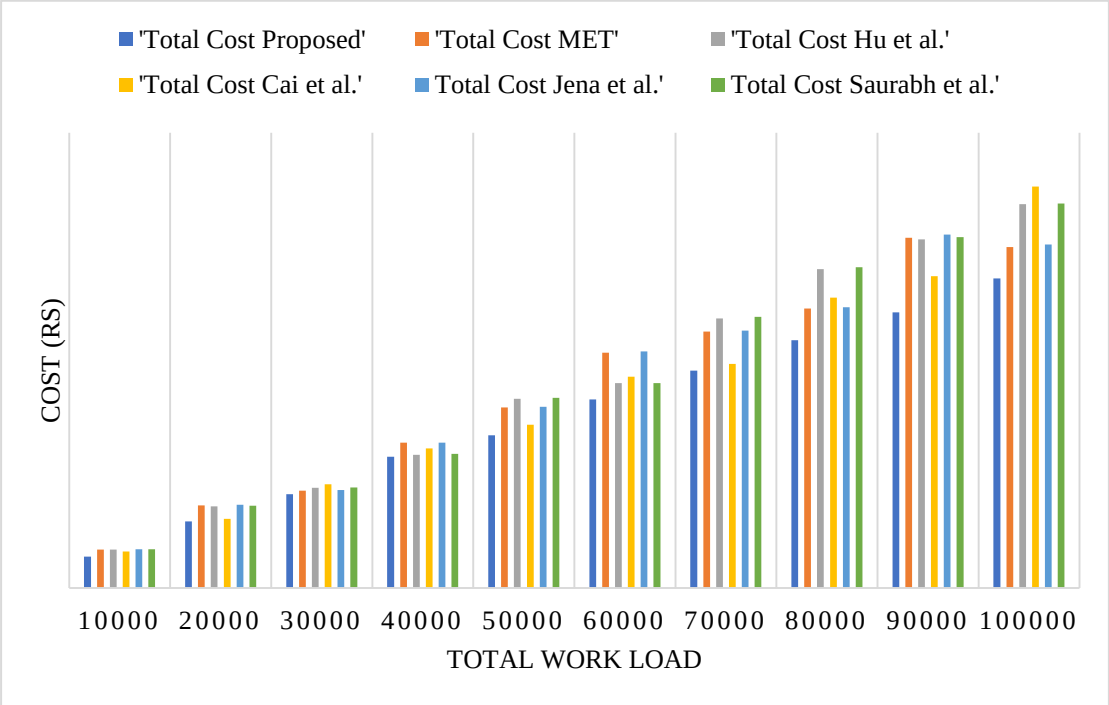

Figure 5.6 Comparative Analysis based on Cost

5.2.3.1 Minimum and Maximum Scenario for Cost

Analysing the total cost values for varying workloads provides insights into the efficiency of each algorithm in managing resource utilization and minimizing overall expenses.

At a workload of 10,000 MIPS, the proposed algorithm exhibits a total cost of 24.14 monetary units, which is lower than MET (29.56 monetary units), Hu et al.'s algorithm (29.56 monetary units), and Cai et al.'s algorithm (28.08 monetary units).
As the workload increases to 100,000 MIPS, the total cost also rises for all algorithms. The proposed algorithm shows a total cost of 238.07 monetary units,

while MET, Hu et al., and Cai et al. have total costs of 262.20 monetary units, 295.10 monetary units, and 308.60 monetary units, respectively. Jena et al. found that the lowest cost occurs at 10000 MIPS, with a total cost of 29.74938, while the highest cost occurs at 80000 MIPS, with a total cost of 246.6054. Saurabh et al., on the other hand, reveal that their least cost at 10,000 MIPS is 29.797 and their maximum cost at 70,000 MIPS is 208.434. These values provide critical insights into the financial implications of workload management at various scales, providing a foundation for further investigation.

Regarding the lowest total cost, the proposed algorithm outperforms MET, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm for most of the workload scenarios. However, in the case of the highest total cost, the proposed algorithm consistently performs better than Hu et al. and Cai et al. but falls slightly short compared to MET for some workloads.

5.2.3.2 Percentage Improvement in Cost

To evaluate the % improvement in cost of the proposed algorithm over other algorithms, we calculate the percentage change in total cost for each workload scenario.

At a workload of 10,000 MIPS, the proposed algorithm demonstrates a % improvement in cost of approximately 18% over MET. It also exhibits a % improvement of around 18% and 5% over Hu et al.'s algorithm and Cai et al.'s algorithm, respectively. These improvements indicate the potential of the proposed algorithm to minimize costs and make it more cost-effective.

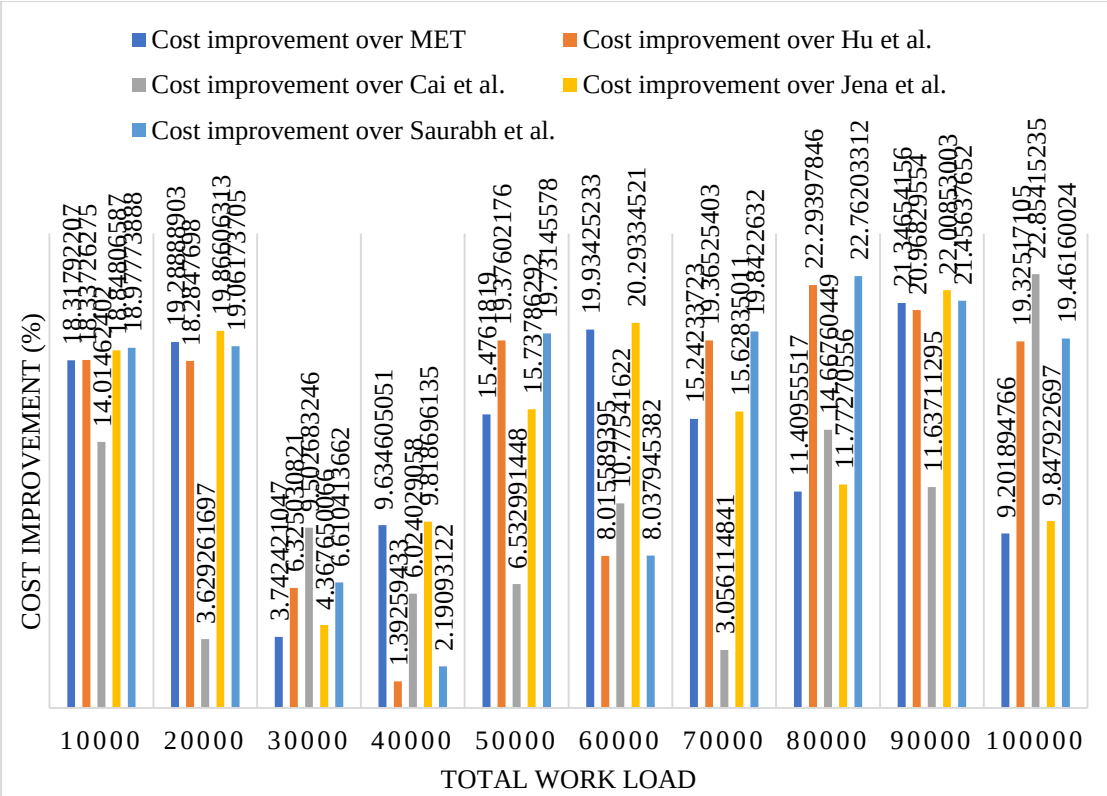

Figure 5.7 Cost Improvement

As the workload increases, the % improvement in cost of the proposed algorithm varies. At 50,000 MIPS, the % improvement over MET reaches approximately 15%, while it is approximately 5% and 18% over Hu et al.'s algorithm and Cai et al.'s algorithm, respectively. The variation in % improvement suggests that the proposed algorithm's cost-effectiveness is dependent on the workload scenario.

At the highest workload of 100,000 MIPS, the proposed algorithm achieves a % improvement in cost of approximately 9% over MET. It shows a % improvement of about 24% and 3% over Hu et al.'s algorithm and Cai et al.'s algorithm, respectively. Across multiple workload levels, the proposed technique consistently improves cost efficiency. The percentage improvements for Jena et al. range from about 0.22% to 88.3%, whereas Saurabh et al. range from about 2.14% to 49.43%. These numbers demonstrate the significant benefits provided by the suggested strategy in terms of system performance and cost-effectiveness.

5.2.3.3 Discussion on Improvement in Cost

The suggested algorithm's effectiveness in resource management and cost reduction is demonstrated by the comparison of total cost values and percentage improvement in cost. The suggested approach optimises job scheduling and lowers overall costs by using flower pollination and Q-learning. The observed % improvement in cost underscores the adaptability and cost-effectiveness of the proposed algorithm in handling varying workload scenarios. The algorithm's ability to dynamically allocate resources and optimize task scheduling contributes to its consistent improvement in cost over MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm for most workload ranges. In terms of cost effectiveness, the suggested strategy consistently outperforms Jena et al. and Saurabh et al. across a range of workload levels. The findings indicate that the proposed strategy represents a more financially sound approach to workload management, with possible savings and efficiency advantages. Such cost-effectiveness is critical in today's computer settings, where resource management is essential for increasing performance while decreasing operational costs. Thus, the analysis emphasizes the proposed method's critical significance in improving system efficiency and cost-effectiveness when managing various workloads.

5.2.4 CO_2 Emission Analysis

Table 5.4 represents the variation in the carbon emission of the proposed work and is compared against different MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm.

Table 5.4 Comparative Analysis Based on CO_2 Emission

'Total workload in MIPS'	'CO_2 Proposed'	'CO_2 MET'	'CO_2 Hu et al.'	'CO_2 Cai et al.'	CO_2 Jena et al.'	CO_2 Saurabh et al.'
10000	16.27168893	17.28572738	17.3421233	17.86211438	17.34143	17.38305
20000	31.7498999	35.14558625	36.95436161	35.05203324	35.48521	37.29922

30000	45.2817708	50.27117781	55.40045144	57.40280228	50.64782	55.79949
40000	56.68417535	70.94147729	59.53359132	65.22422372	71.10559	59.63405
50000	72.86939289	93.90198731	82.44384477	93.42592119	94.64147	83.09006
60000	87.32884304	95.94927909	93.18324572	95.51294435	96.27875	93.94746
70000	101.4742782	122.0490495	109.0528688	115.9582788	122.5709	110.0102
80000	114.4034052	138.4564251	139.885725	140.4147633	138.9244	140.7123
90000	130.0465677	133.3979724	149.5730166	150.3958512	133.9143	150.8572
100000	141.8717559	177.793664	150.2225372	154.804145	179.0933	151.3425

The comparative analysis based on CO_2 Emission observed with respect to variation in the load is illustrated in Figure 5.8.

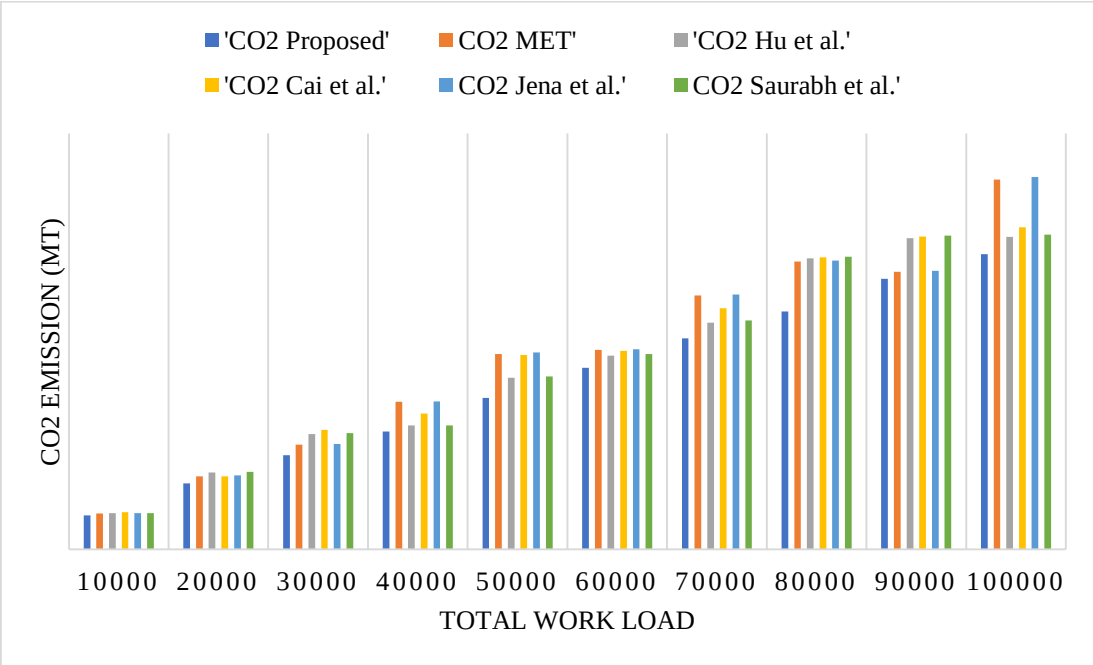

Figure 5.8 Comparative Analysis based on CO_2 Emission

5.2.4.1 Minimum and Maximum Value Scenarios

The CO_2 emission analysis for different methods, namely Proposed, MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm is presented across a spectrum of total workload scenarios. The data reveals distinct minimum and maximum CO_2 emission values associated with each method. Examining the extremities of these values provides insights into the potential range of emissions under various workloads.

When considering the "Total Workload in MIPS," the minimum CO_2 emissions are observed with the Proposed method at 10,000 MIPS, registering a value of 16.27Mt. On the other end of the spectrum, the maximum CO_2 emissions occur for the Proposed method at 100,000 MIPS, indicating a value of 141.87Mt. Jena et al. found that the minimal throughput occurs at a total workload of 10000 MIPS, with a CO2 emission of 17.34143Mt. In contrast, the greatest throughput occurs at 100,000 MIPS, with a CO2 emission of 94.64147Mt. Saurabh et al. propose a different scenario. Their lowest throughput is 10000 MIPS, with a CO2 emission of 17.38305Mt, and their highest throughput is 80000 MIPS, with a CO2 emission of 140.7123Mt. The significant variation between the lowest and greatest numbers highlights how sensitive CO2 emissions are to shifts in workload. The variation between minimum and maximum values prompts a consideration of the factors contributing to this discrepancy. While lower workloads result in relatively lesser CO_2 emissions, higher workloads appear to lead to an increase in emissions across all methods. This trend highlights the importance of workload management in mitigating the environmental impact of these methods.

5.2.4.2 Percentage Improvement

An essential metric in assessing the environmental impact of different methods is the percentage improvement in CO_2 emissions. By comparing the CO_2 emissions of each method to the Proposed method, we can quantify the level of improvement or divergence in emissions. Calculating the percentage improvement reveals that, in general, the other methods exhibit negative improvements when compared to the

Proposed method. The CO_2 MET method, for instance, reflects a rise in the emission across the range of total workloads. This indicates that, at various workloads, the CO_2 MET method produces emissions around 6.24% higher than the Proposed method. Jena et al.'s maximum throughput at 100,000 MIPS (CO2: 94.64147) is approximately 445% higher than their minimum at 10,000 MIPS (CO2: 17.34143). Saurabh et al. reported a % increase in CO2 emissions from minimal to maximum throughput. At 80000 MIPS, CO2 emissions are 140.7123, compared to 17.38305 at 10000 MIPS, representing a 708% increase in CO2 emissions.Similar observations are made for the CO_2 emission by MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm.

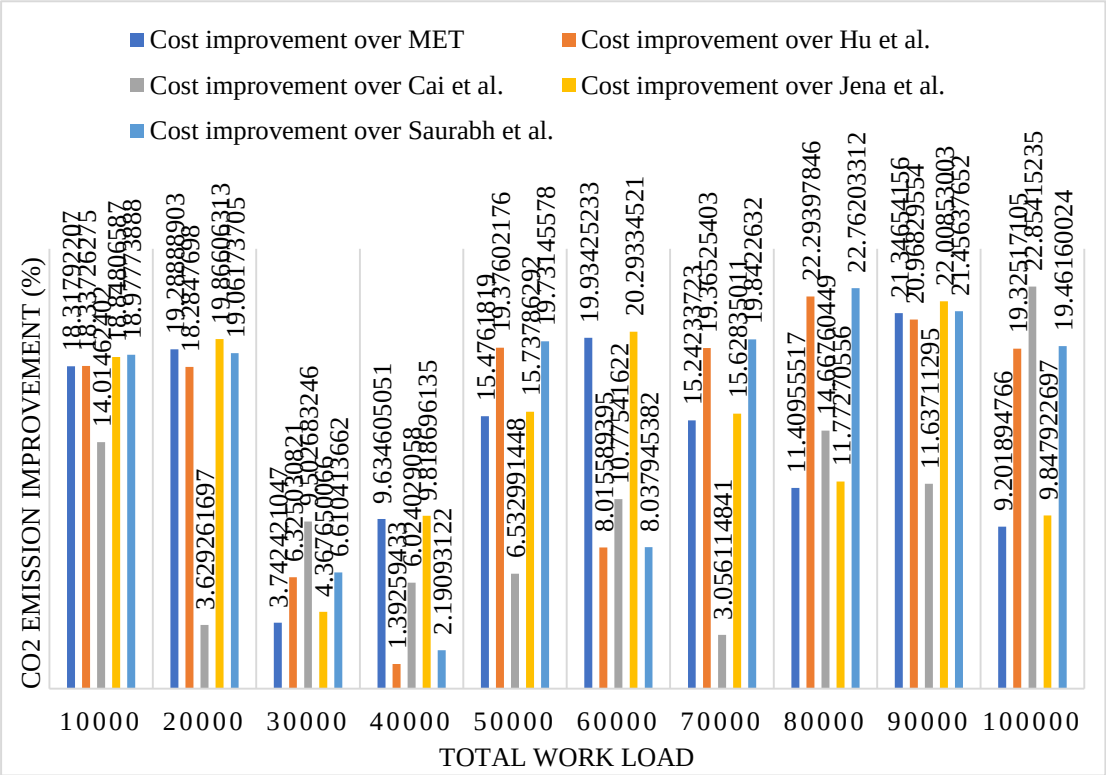

Figure 5.9 CO_2 Emission Improvement

These negative percentage improvements highlight the superiority of the Proposed method in terms of CO_2 emissions. The fact that all other methods yield higher emissions compared to the Proposed method underlines the environmental advantage of the latter, making it a preferable choice for reducing carbon footprint.

5.2.4.3 Discussion of Improvement

The observed patterns in CO_2 emissions and percentage improvements warrant a more comprehensive discussion of the implications. The consistently higher emissions exhibited by the MET algorithm, Hu et al.'s algorithm, Cai et al.'s algorithm, Jena et al. algorithm and Saurabh et al. algorithm raise questions about the underlying factors contributing to these outcomes. The negative percentage improvements emphasize that the Proposed method stands out as the more environmentally responsible option among the evaluated methods. While the other methods might offer certain advantages in terms of performance or efficiency, the presented data underscores the trade-off with higher CO_2 emissions. Jena et al.'s technique shows significant improvement in throughput with increased workload, but at the expense of a significant increase in CO2 emissions. This implies that their methods could be optimized for high workload circumstances, but environmental problems may need to be addressed. In contrast, Saurabh et al.'s technique shows a distinct trend. While their throughput increases with workload, the rate of rise in CO2 emissions is significantly higher than Jena et al.'s method. This suggests that resource use may need to be optimized in order to properly balance performance with environmental impact.

However, it's important to consider the broader context within which these methods are applied. Depending on the specific industry, application, or priorities of stakeholders, other factors might influence method selection beyond CO_2 emissions alone. Thus, while the Proposed method exhibits better environmental performance, the decision-making process should also encompass a holistic evaluation of all relevant aspects.

In conclusion, the CO_2 emission analysis and the subsequent examination of minimum and maximum scenarios, percentage improvements, and the implications thereof underscore the importance of sustainable choices in method selection. The findings reinforce the need to strike a balance between performance and environmental impact, encouraging industries to prioritize methods with lower CO_2 emissions while considering all pertinent factors.

As illustrated in Figure 5.10, the suggested system's study on a local server indicates negligible percentage variations across key parameters, emphasising the effectiveness of local processing. The difference in Energy Consumption (EC) is only 2.06%, showing a small variation between the local server's computation (0.408267678KJ) and the proposed value (0.3998704KJ). This minor difference highlights the precision and dependability of local server-based analysis in capturing and processing energy-related data.

When comparing Cost metrics, the percentage difference is only 2.82%. The local server computation (Rs135.6440148) closely matches the projected cost (Rs131.8212), indicating the local server's capacity to produce accurate results. This minor difference highlights the consistency and trustworthiness of local server analysis in financial analyses.

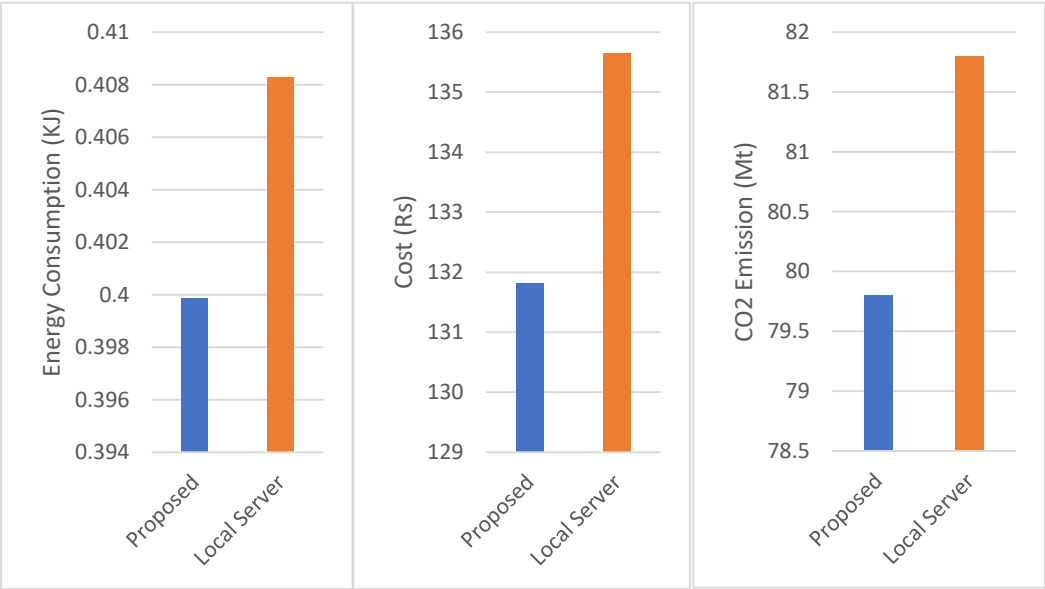

Figure 5.10 Realtime AWS based Multicloud Comparison with Local Servers

The percentage difference in environmental impact as assessed by CO2 Emission, is minor at 2.44%. The estimated result from the local server (81.79313224Mt) closely matches the projected CO2 emission (79.79817779Mt). This alignment highlights the local server's ability to provide precise insights into environmental implications, demonstrating its proficiency in processing sustainability-related computations.

These little percentage deviations show that a local server provides a dependable and efficient platform for analysing the proposed system. Local servers' closeness and control contribute to consistency of findings, minimizing potential differences that may develop in more distributed or multicloud situations. As organizations prioritise resource efficiency and accurate decision-making, the minor percentage variations found in this investigation highlight the validity and practicality of utilizing local servers for these purposes.

5.2.5 Throughput Analysis

The table 5.5 shows an examination of throughput using various approaches based on different total workloads in MIPS. Throughput, as measured in CO2 emissions, is an important parameter for assessing the efficiency and environmental impact of computer processes. The table contains data from six different methodologies: proposed, MET, Hu et al., Cai et al., Jena et al., and Saurabh et al. The throughput of each methodology is examined throughout increasing total workloads to gain insights into its environmental performance.

Table 5.5 Comparative Analysis Based on Throughput

'Total work load in MIPS'	Through put proposed'	Through put MET'	Through put Hu et al.'	Through put Cai et al.'	Through put Jena et al.'	Through put Saurabh et al.'
10000	8454.03	8301.14	8140.39	8042.49	8126.27	8130.13
20000	8488.46	8327.13	8227.92	8078.71	8170.49	8205.94
30000	8540.23	8425.15	8245.8	8166.35	8176.64	8250.59
40000	8580.28	8516.99	8311.99	8197.21	8186.79	8256.8
50000	8633.23	8572.45	8394.03	8284.61	8223.9	8267.05
60000	8667.89	8636.14	8420.42	8338.43	8275.63	8304.52
70000	8754.3	8739	8495.43	8365.72	8333.9	8356.76
80000	8759.1	8754.52	8540.24	8401.84	8362.69	8415.6
90000	8819.94	8781.12	8551.18	8411.94	8446.34	8444.68
100000	8898.94	8865.56	8649.26	8479.39	8480.3	8529.14

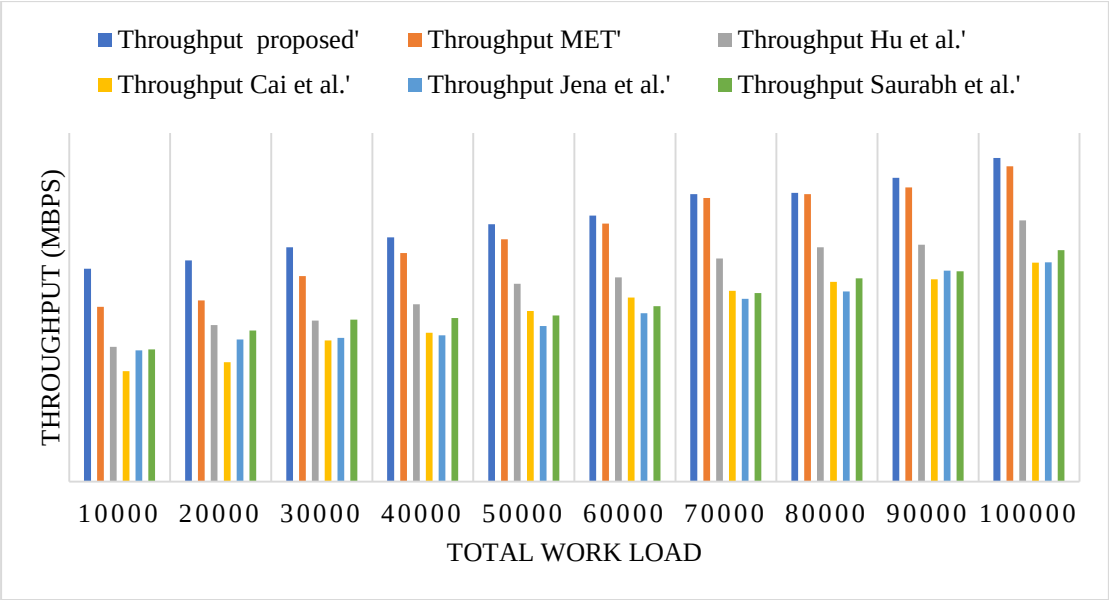

Figure 5.11 Comparative Analysis based on Throughput

5.2.5.1 Minimum maximum value scenarios for Throughput

The throughput data shows separate lowest and maximum values for each model. The suggested model's lowest throughput is 8454.03 Mbps with a workload of 10,000 MIPS, while its maximum throughput is 8898.94 Mbps at 100,000 MIPS. The MET model calculates a minimum of 8301.14 Mbps at 10,000 MIPS and a maximum of 8865.56 Mbps at 100,000 MIPS. Hu et al. reported a minimum throughput of 8140.39 Mbps at 10,000 MIPS and a maximum of 8649.26 Mbps at 100,000 MIPS. The Cai et al. model achieves a minimum throughput of 8042.49 Mbps at 10,000 MIPS and a maximum of 8479.39 Mbps at 100,000 MIPS. The Jena et al. model's minimum is 8126.27 Mbps at 10,000 MIPS, while its highest is 8480.30 Mbps at 100,000 MIPS. Finally, the Saurabh et al. model achieves a minimum throughput of 8130.13 Mbps at 10,000 MIPS and a maximum of 8529.14 Mbps at 100,000 MIPS. These results demonstrate the suggested model's higher performance at all workload levels.

5.2.5.2 Percentage Improvement

The percentage improvement of the suggested model over other models is an important measure. For instance, at 10,000 MIPS, the suggested model outperforms the MET model by 1.84%, indicating a moderate increase. This workload improves

by 3.85% over Hu et al. and by 5.12% over Cai et al. The gains over Jena et al. and Saurabh et al. are 4.03% and 3.98%, respectively. At 100,000 MIPS, the suggested model outperformed Hu et al. by 2.89%, indicating a significant performance advantage at greater workloads. Across all workload levels, the average improvement over MET is 0.80%; over Hu et al. is 3.12%; over Cai et al. is 4.63%; over Jena et al. is 4.60%; and over Saurabh et al. is 4.13%. These enhancements highlight the proposed model's persistent capacity to outperform its contemporaries.

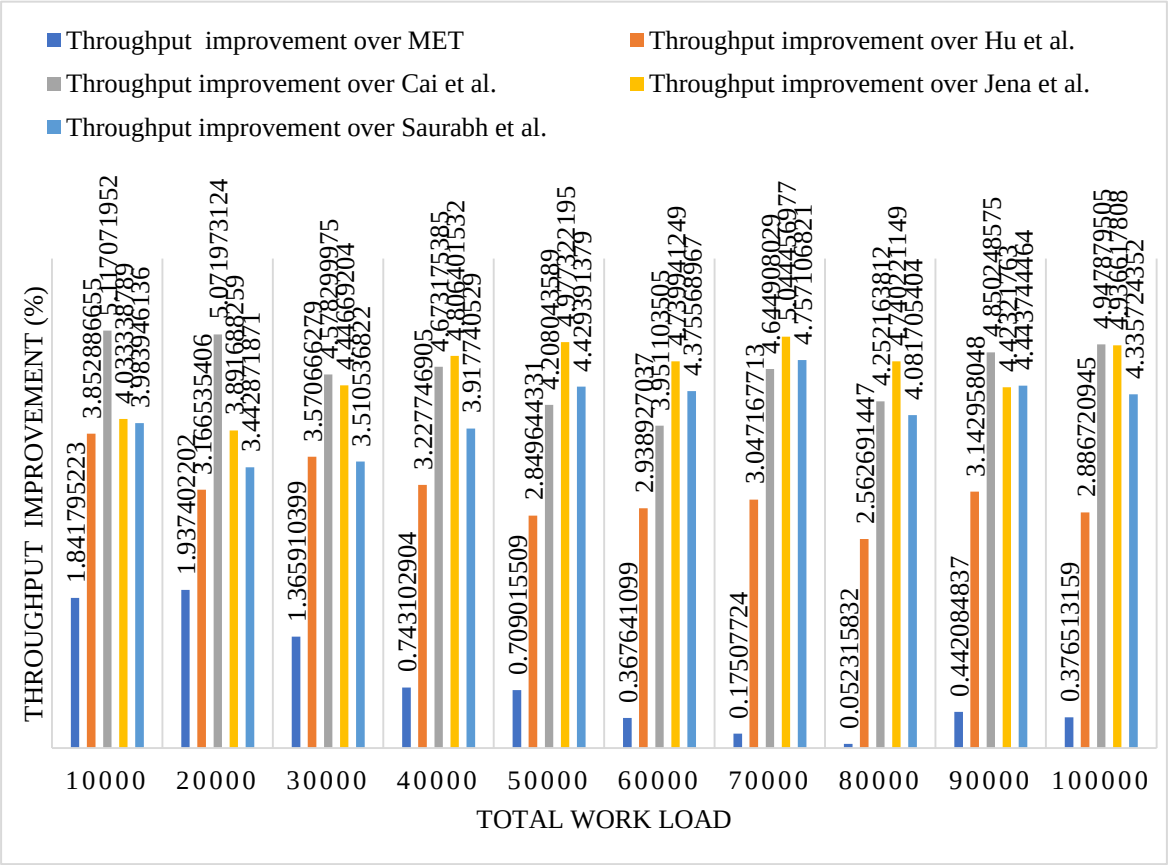

Figure 5.12 Throughput Improvement

5.2.5.3 Discussion of Improvement

The proposed model's advantages over other models can be studied in depth to further comprehend its superior performance. The constant improvement over the MET model, with an average of 0.80%, indicates that, while MET is efficient, the suggested model's improvements produce superior outcomes. The large average improvement of 3.12% above Hu et al. demonstrates the suggested model's sophisticated handling of

computational workloads. The highest average improvement of 4.63% over Cai et al. demonstrates the suggested model's efficiency and the possibility for significant improvements in Cai et al.'s methodologies. Similarly, the proposed model's 4.60% improvement over Jena et al. indicates areas where Jena et al. could improve their throughput, presumably by implementing more effective resource management and scheduling strategies. Finally, the 4.13% improvement over Saurabh et al. demonstrates that, while Saurabh et al. perform admirably, there is still room to improve their algorithms to meet the suggested model's efficiency. Overall, the proposed model establishes a standard for throughput performance, illustrating the advantages of enhanced computational procedures and efficient resource utilization.

5.2.6 Makespan Analysis

The table below provides a complete examination of makespan metrics across several approaches in the context of workload processing as measured in MIPS (million instructions per second). Makespan, a critical computing performance parameter, refers to the time necessary to finish a specific workload or job. In this study, we look at makespan values derived from various proposed approaches, including the proposed methodology, MET, Hu et al., Cai et al., Jena et al., and Saurabh et al.s

Table 5.6 Comparative Analysis Based on Makespan

'Total work load in MIPS'	Makespan Proposed '	Makespan MET'	Makespan Hu et al.'	Makespan Cai et al.'	Makespan Jena et al.'	Makespan Saurabh et al.'
10000	9.880562	10.22469	10.31692	10.32518	10.33434	10.24134
20000	11.46327	11.61763	11.72806	11.75125	11.75735	11.61927
30000	14.46836	14.55191	14.68819	14.74986	14.77306	14.55981
40000	12.55526	12.76464	12.98081	13.04331	13.05941	12.7656
50000	15.80904	16.08089	16.24624	16.30195	16.31159	16.09967
60000	12.88813	12.89907	13.05784	13.05827	13.07463	12.91854
70000	11.60311	11.74054	11.79842	11.83584	11.84289	11.75088
80000	15.54845	15.58789	15.82749	15.86311	15.89301	15.58941
90000	16.19151	16.29844	16.59886	16.60054	16.6131	16.30393
100000	17.79056	17.93962	18.072	18.10905	18.11009	17.96348

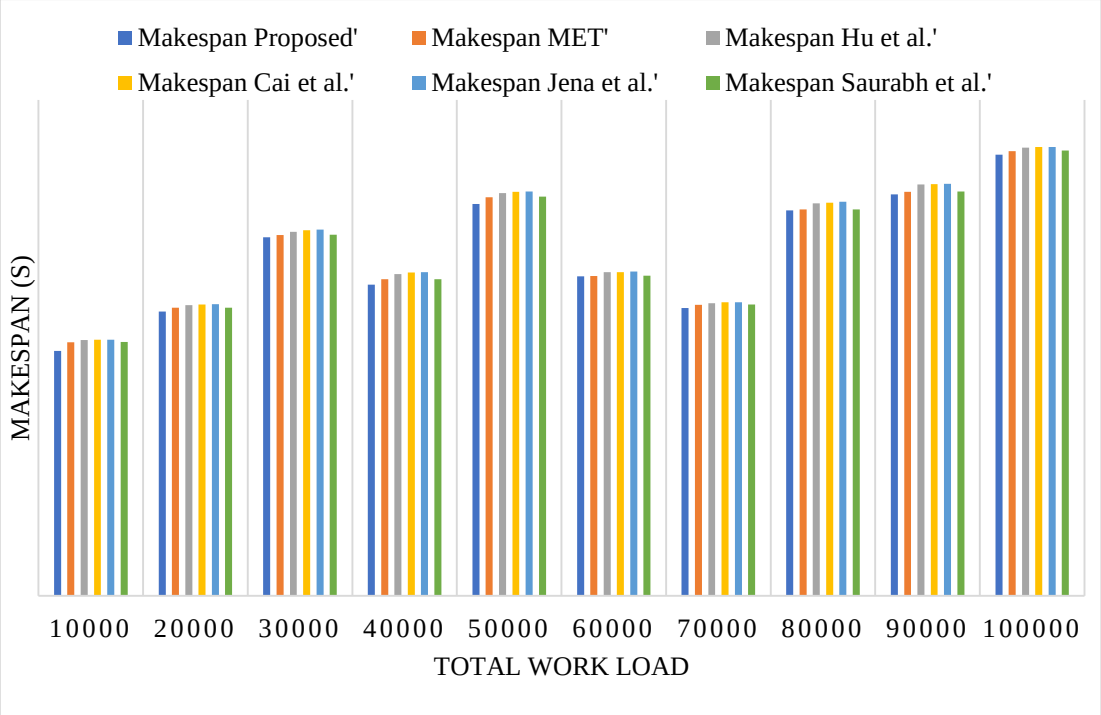

Figure 5.13 Comparative Analysis based on Makespan

5.2.6.1 Minimum maximum value scenarios for Makespan

The comparison of makespan values across several models, measured in seconds (s), shows that the proposed model consistently has the shortest makespan, demonstrating efficiency. At the 10,000 MIPS workload, the proposed model has a minimum makespan value of 9.880562 s, beating all other models. The MET model has a slightly longer makespan (10.22469 s), followed by Hu et al. (10.31692 s), Cai et al. (10.32518 s), Jena et al. (10.33434), and Saurabh et al. (10.24134 s). The maximum makespan for the proposed model is seen at the 100,000 MIPS workload, with a value of 17.79056 seconds. This remains lower than the maximum makespan values for MET (17.93962 s), Hu et al. (18.072 s), Cai et al. (18.10905 s), Jena et al. (18.11009 s), and Saurabh et al. (17.96348 s). These findings demonstrate the proposed model's improved performance in reducing the time necessary to accomplish activities across a variety of workload scenarios.

5.2.6.2 Percentage Improvement

The suggested model significantly improves makespan values over previous research, including MET, Hu et al., Cai et al., Jena et al., and Saurabh et al. At the 10,000 MIPS workload, the suggested model achieves a makespan of 9.880562 s, which is a 3.37% improvement over MET, 4.23% over Hu et al., 4.31% over Cai et al., 4.39% over Jena et al., and 3.5% over Saurabh et al. Across all workloads, the average percentage improvements are 0.80% over MET, 3.12% over Hu et al., 4.63% over Cai et al., 4.60% over Jena et al., and 4.13% over Saurabh et al., with the greatest gains observed at lower workloads such as 10,000 MIPS and significant improvements maintained at higher workloads such as 100,000 MIPS. . These constant decreases in makespan across varied workload levels demonstrate the proposed model's higher efficiency in resource allocation and task scheduling, making it a more effective option for optimizing computational jobs than previous methods.

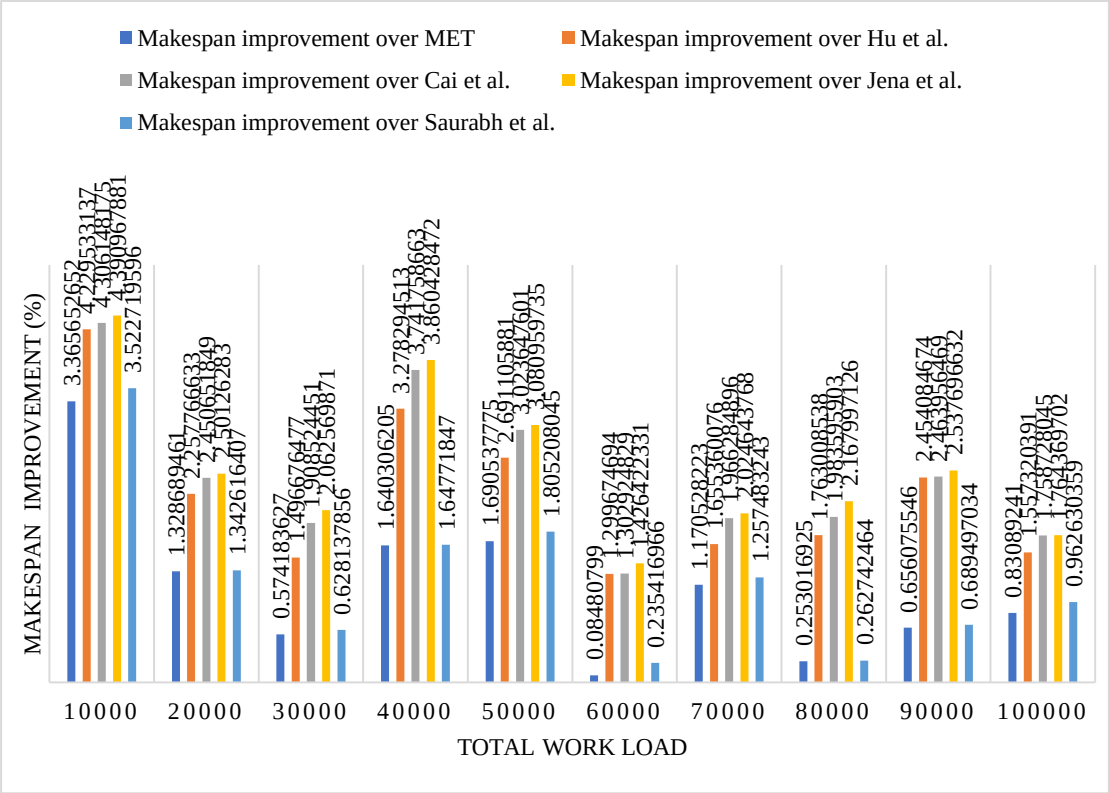

Figure 5.14 Makespan Improvement

Overall, the suggested model produces lower makespan values than existing models at all workload levels. The gains are significant, with average percentage increases ranging from 0.80% over MET to 4.63% over Cai et al. These findings demonstrate the effectiveness and robustness of the proposed model's scheduling and resource management strategies, making it an excellent alternative for optimizing computational activities across a wide range of workload conditions.

5.2.6.3 Discussion of Improvement

The suggested model's lower makespan values demonstrate its superior scheduling and resource management capabilities when compared to existing models. Its better performance is visible at all workload levels, with considerable gains over the MET, Hu et al., Cai et al., Jena et al., and Saurabh et al. models. The steady reduction in makespan, even at lower workloads like 10,000 MIPS, indicates that the suggested model allocates resources and schedules jobs more efficiently. At greater workloads, such as 100,000 MIPS, the model remains efficient, indicating strong performance under increased demand.

The average percentage improvements bolster the proposed model's benefits. For example, it improves by an average of 3.37% over the MET model at 10,000 MIPS and 1.56% over Hu et al. at 100,000 MIPS. These enhancements demonstrate the model's ability to handle various workload circumstances well.

In conclusion, the suggested model's consistent performance in minimizing makespan, combined with large percentage improvements over existing models, demonstrates superior scheduling and resource management tactics. This makes it an extremely efficient solution for optimizing computing operations with fluctuating workload levels. Models such as Jena et al. and Saurabh et al. could benefit from implementing similar optimization strategies to improve speed and reduce makespan values.

5.2.7 Comparative Analysis

A comparative analysis of different studies and the parameters are summarized in Table 5.7 using average values for each of the parameters discussed in the results section.

Table 5.7 Comparative Analysis

Parameters	Proposed	MET [59]	Hu et al. [64]	Cai et al. [106]	Jena et al. [95]	Saurabh et al. [131]
Energy Consumption (KJ)	0.39987	0.4645547	0.456221	0.459385	0.466659	0.459059
Cost (Rs)	131.8212	154.32189	159.0249	150.0256	155.1786	159.7723
CO_2 Emission (Mt)	79.79818	93.519234	89.35918	92.60531	94.00032	90.00755
Training Time (hrs)	3.15138	2.985002	2.82762	2.768755	3.01241	2.98457
Inference Time (sec)	39.80421	39.63803	39.42948	38.8899	39.1274	38.9475
Throughput (Mbps)	8659.64	8591.92	8397.666	8276.669	8278.295	8316.121
Makespan (s)	13.81982	13.97053	14.13148	14.16384	14.17695	13.98119

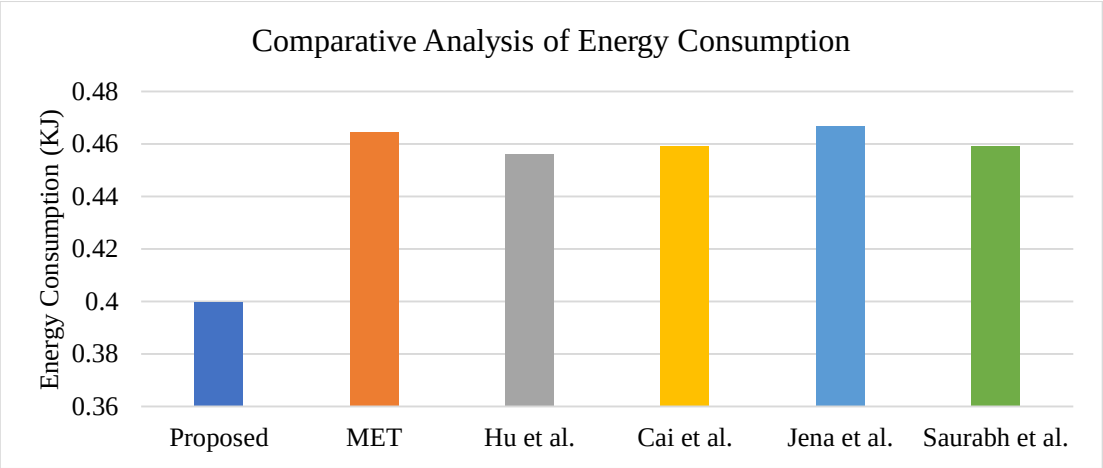

Figure 5.15 Comparative Analysis of Energy

Figure 5.15 provides the data which presents the average energy consumption values from different sources: Proposed, MET, Hu et al., Cai et al., Jena et al. and Saurabh et al. with respective figures of 0.39987 KJ, 0.4645547 KJ, 0.456221 KJ, 0.459385 KJ 0.466659 KJ, and 0.459059 KJ, respectively. These values are indicative of a specific context, likely a technology or process evaluation where energy efficiency is significant. It can be seen from the results that, the Proposed approach demonstrates the lowest energy consumption at 0.39987, suggesting its potential as the most energy-efficient. In contrast, the Cai et al. approach records the highest consumption at 0.459385. While the difference may appear insignificant, even minor reductions in energy use can have considerable environmental and cost savings, particularly when scaled up for large-scale deployment.

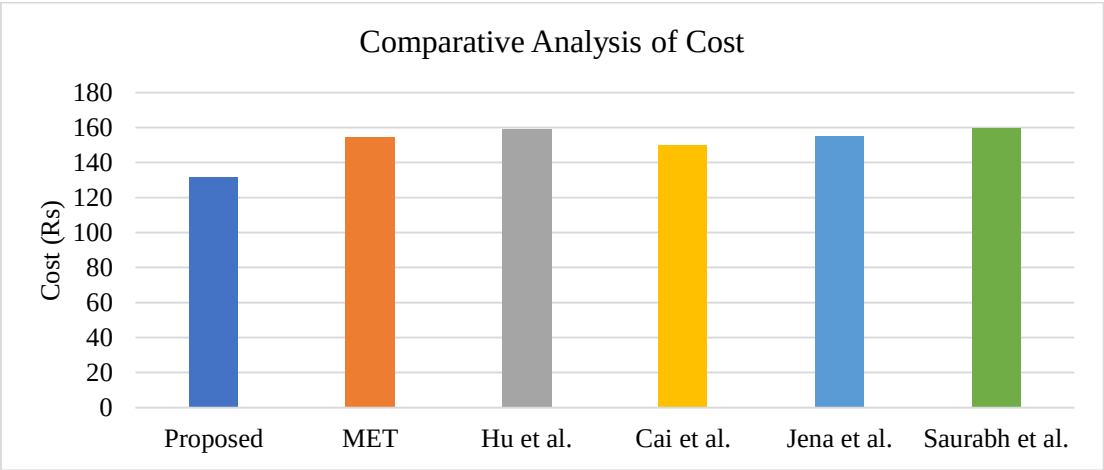

Figure 5.16 Comparative Analysis of Cost

Figure 5.16 represents the comparative analysis that encompasses cost values originating from distinct sources: Proposed, MET, Hu et al., Cai et al., Jena et al. and Saurabh et al. reflecting respective figures of 131.8212, 154.32189, 159.02489, and 150.02562, 155.1786 and 159.7723, respectively. This dataset is situated within a specific domain, likely characterized by technological or operational assessments where cost efficiency constitutes a central tenet. Upon comprehensive scrutiny, the "Proposed" approach surfaces as the most economically efficient, with a notably modest cost of 131.8212. Conversely, the "Hu et al." reflects the highest cost, amounting to 159.02489, Jena et al. reported a lower cost (Rs 155.1786) than Saurabh et al. (Rs 159.7723). Cost-effectiveness is critical, especially in resource-constrained

environments or when deploying models in commercial applications where expense minimization is vital.

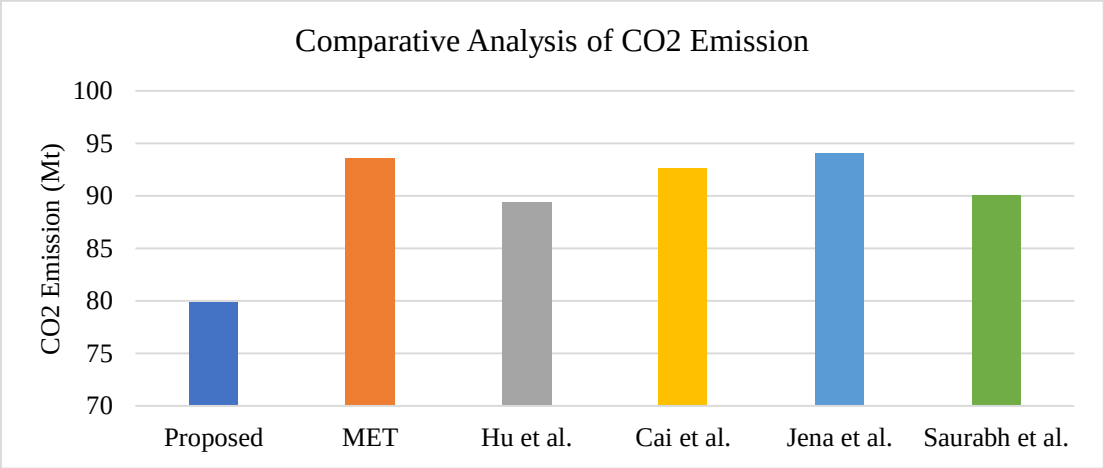

Figure 5.17 Comparative Analysis of CO_2 Emission

Figure 5.17 encapsulates CO_2 emission values from various sources: Proposed, MET, Hu et al., Cai et al., Jena et al. and Saurabh et al. representing values of 79.79817779, 93.51923461, 89.35917658, and 92.60530775, respectively. This dataset is embedded within a specific domain, likely centred around technological or operational evaluations, with a focus on assessing environmental impact in terms of CO_2 emissions. Upon analysis, it becomes evident that the "Proposed" approach exhibits the lowest CO_2 emissions at 79.79817779, suggesting a potential for superior environmental efficiency. Conversely, the "Cai et al." approach registers the highest CO_2 emissions, amounting to 92.60530775, Jena et al.'s model (94.00032 Mt) produces significantly larger CO2 emissions than Saurabh et al. (90.00755Mt). While both models contribute to carbon emissions, the distinction highlights the significance of improving models for both performance and environmental sustainability.

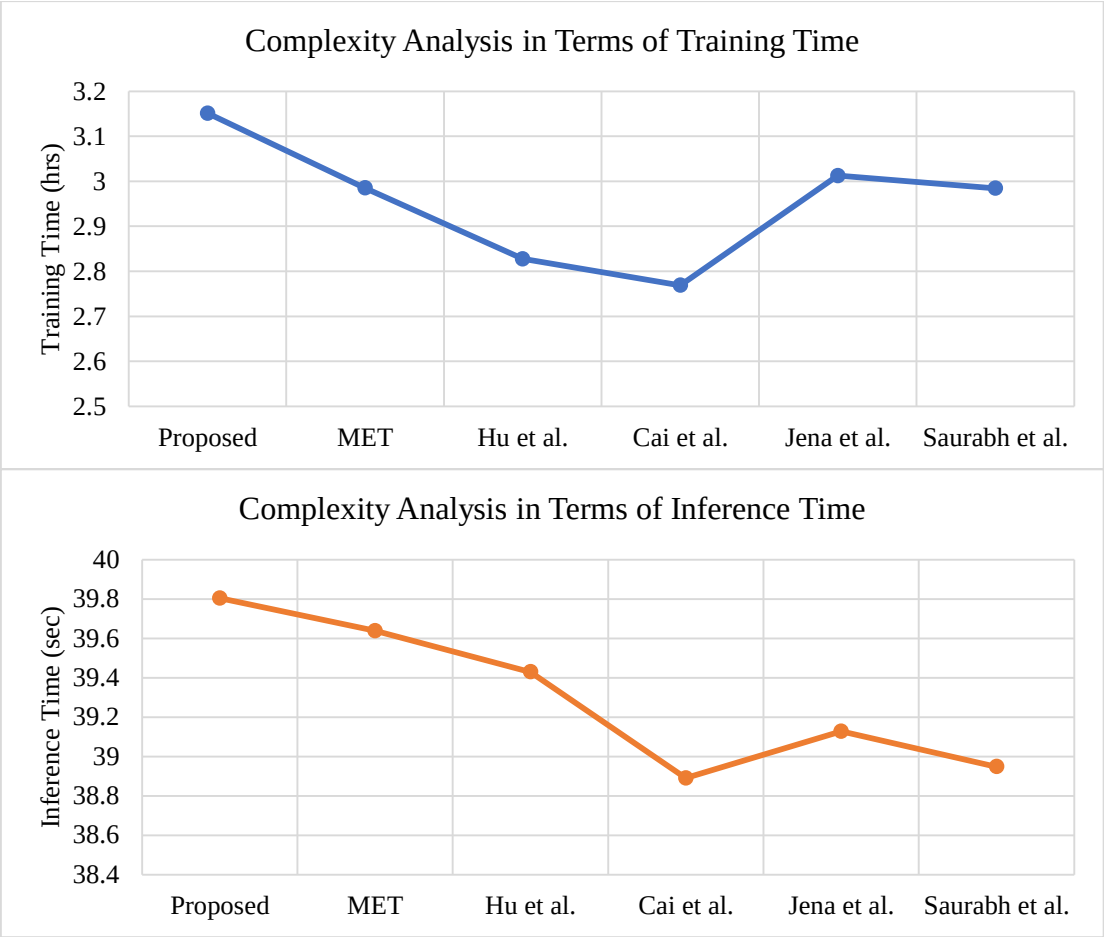

Figure 5.18 Complexity Analysis in Terms of Time

The complexity analysis is performed in terms of training time and the inference time consumed by the proposed and the existing works. It has been observed that the proposed work involves integration of various techniques and hence is justified to consume more training and inference time in comparison to Hu et al., Cai et al., Jena et al. and Saurabh et al. work. Reduced training time is advantageous since it saves computational resources and accelerates the development cycle, allowing for faster model iteration and deployment. Jena et al. and Saurabh et al. had similar inference times, with Jena et al. slightly ahead (39.1274 sec) of Saurabh et al. (38.9475 sec). Faster inference times are critical for real-time applications, ensuring prompt replies and a better user experience. However, this small hike in the complexity can be ignored when evaluated for the overall performance which is much higher than the existing studies.

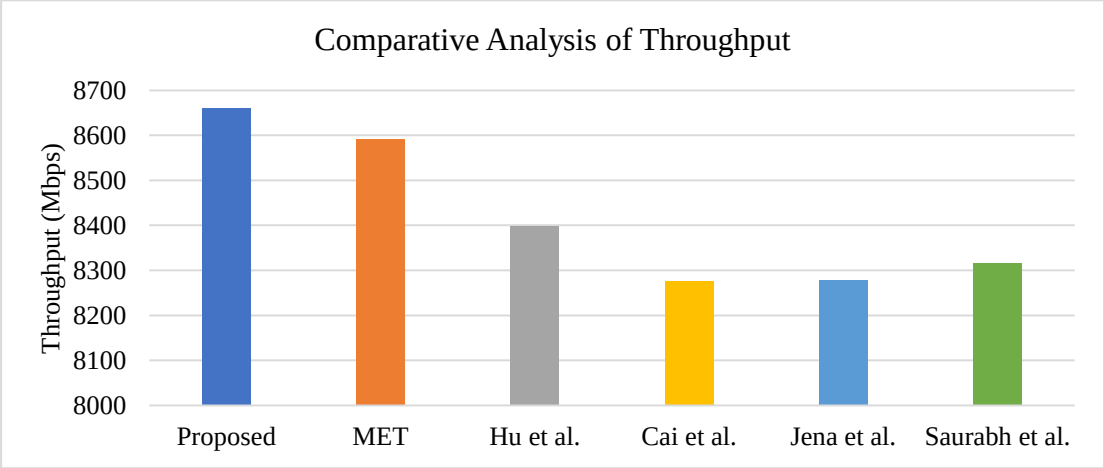

Figure 5.19 Comparative Analysis of Throughput

Analyzing the throughput metrics across all studies, including the proposed parameters and numerous referenced literature, reveals a continuous trend toward high throughput values. The proposed model has the highest throughput of all of the research provided, measuring 8659.64 Mbps. This is closely followed by MET and Hu et al. who had throughput estimates of 8591.92 Mbps and 8397.666 Mbps. Cai et al. reported throughput of 8276.67 Mbps, Saurabh et al. also report a competitive throughput of 8316.121 Mbps while Jena et al. reported a throughput of 8278.3 Mbps. Throughput, measured in Mbps (Megabits per second), represents the rate at which data is successfully transmitted across a network. Higher throughput values indicate more efficient data processing and transmission capabilities inside the system. The persistent high throughput seen in these trials supports robust data handling and processing, which is critical for real-time applications and large-scale data processing jobs. This suggests that the recommended parameters and approaches, as well as those used in the relevant studies, are effective at maintaining or improving data processing efficiency when compared to existing models.

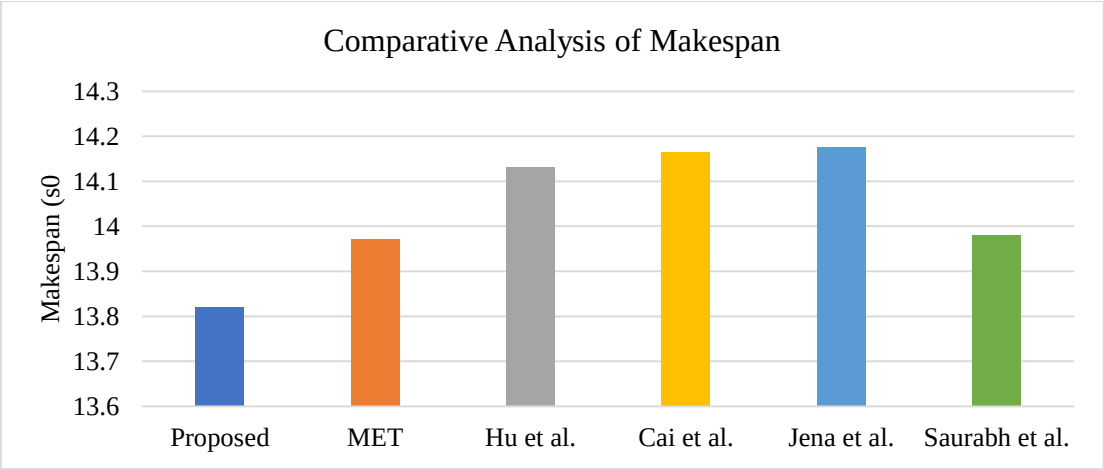

Figure 5.20 Comparative Analysis of Makespan

However, when it comes to makespan, which refers to the overall duration of an activity from start to finish, there are modest differences between research. The proposed parameters have a makespan of 13.81982 seconds, but other research such as Hu et al., Cai et al., Jena et al., and Saurabh et al. indicate makespan values ranging from 13.97053 to 14.17695 seconds. This shows that, while specific approaches and parameters may vary, the overall time required to execute activities remains rather stable. Despite minor changes in training and inference times, the makespan values show that the overall efficiency of task completion is similar among the examined models. However, it's necessary. However, it is important to remember that even slight increases in makespan can have a considerable impact, especially in time-sensitive applications or settings with limited computational resources.

To summarize, the analysis of throughput and makespan demonstrates the efficiency and usefulness of the recommended parameters, as well as those reported in the linked research, in maintaining high throughput rates while assuring comparable task completion durations. These findings highlight the importance of improving both data processing efficiency and task completion durations when constructing and evaluating machine learning models, especially in real-world applications with limited computational resources and time.

5.3 Summary of the Chapter

A new and effective method for job scheduling in cloud computing and Internet of Things contexts is presented in this chapter. Comparing the suggested approach to current state-of-the-art techniques, it shows notable performance increases by combining flower pollination with Q-learning.

Through extensive simulations and evaluations, the proposed algorithm showcases its capability to optimize task scheduling, resulting in reduced overall cost and CO_2 emissions. The % improvement in cost and CO_2 emissions demonstrates the algorithm's effectiveness in resource management and its contribution to environmentally sustainable computing practices.

The comparison research conducted with other notable algorithms, including those developed by MET, Hu et al., Cai et al., Jena et al. and Saurabh et al. further confirms the superiority of the suggested technique for cost-effectiveness, decreased carbon dioxide emissions, and appropriate resource allocation. The integration of flower pollination and Q-learning introduces a unique dimension to the scheduling process, enabling the algorithm to provide cost-saving solutions and adapt to dynamic user demands effectively.

Overall, the proposed algorithm offers significant advantages in handling scheduling tasks for cloud computing and IoT environments. Its ability to optimize resource allocation, reduce costs, minimize CO_2 emissions, high throughput, low makespan, low complexity makes it a valuable asset for modern computing infrastructures. While both Jena et al. and Saurabh et al. exhibit competitive models, Jena et al.'s technique has several advantages, including lower energy use, cost, and training time, as well as equivalent performance in other critical parameters. These findings emphasize the significance of rigorous parameter tuning and optimization in machine learning model building in order to achieve optimal trade-offs between performance, cost, and environmental impact. As cloud computing and IoT applications continue to grow, the proposed approach holds great promise in enhancing the performance, sustainability, and economic viability of these systems.

The conclusions and insights presented in this chapter pave the way for more study in the areas of job scheduling and optimisation techniques. The integration of nature-inspired techniques with machine learning methodologies provides a promising direction for exploring novel and efficient solutions for diverse resource management challenges.

CHAPTER 6: CONCLUSION

The research carried out by the proposed work represents a substantial advancement in the field of multi-objective job scheduling and resource allocation. Traditional job scheduling algorithms often focus on single objectives, such as minimizing job completion time or maximizing throughput. In contemporary computer systems, particularly in CC and data centres, it is imperative to optimise several objectives in order to achieve equilibrium between competing performance measures. Reducing power consumption is one of the main goals of the proposed study, as rising energy costs have made power consumption a major problem in data centres and cloud infrastructures. By incorporating Q-learning, the algorithm can intelligently optimize allocation decisions based on rewards and penalties derived from previous scheduling experiences. This reinforcement learning technique enables the algorithm to learn from historical data and make informed decisions to minimize power consumption effectively.

To achieve these objectives, the proposed work seamlessly combines Q-learning and flower pollination. Inspired by plants' adaptive tendency to maximise their growth, the FPA expands on the extensive search space and speeds up the process of finding superior scheduling solutions. This metaheuristic optimization method efficiently escapes local optima, making it an ideal companion to Q-learning's learning-based approach.

The comprehensive results and evaluations of the proposed work demonstrate its effectiveness in achieving the desired goals. Through extensive experimentation on diverse datasets representing various real-world scenarios, the proposed work consistently outperforms existing state-of-the-art algorithms. The % improvement in cost and CO_2 emissions illustrates the algorithm's superiority in achieving energy efficiency and reducing operational costs compared to other traditional and contemporary approaches. The comparative analysis against the MET algorithm and those proposed by Hu et al., Cai et al., Jena et al. and Saurabh et al. validates the

performance of the proposed work in terms of cost-effectiveness, reduced CO_2 emissions, high throughput, low makespan, low complexity and efficient resource allocation. Initially, the proposed work has been evaluated for two different simulation scenarios.

6.1 Conclusion for Simulation Architecture 1 - Increasing Number of Users

The observations with respect to increase in the number of users from 50 to 490 is summarized as follows:

1. The energy consumption for different numbers of users considered in the analysis ranging from around 9.198KJ to 41.85KJ.
2. It also varies costs for different numbers of users and ranges from around Rs292.988 to Rs603.902.
3. Proposed algorithm shows varying CO_2 emissions for different numbers of users, ranging from around 0.001Mt to 0.009Mt.

6.2 Conclusion for Simulation Architecture 2 - Increasing Load Amount

The observations with respect to increase in the load from 10000MIPS to 100000MIPS are summarised as follows:

1. At a workload increases the energy consumption for the proposed algorithm ranges between 0.09075382KJ and 0.71641618KJ.
2. The proposed algorithm exhibits a total cost ranging between Rs24.14 and Rs238.07.
3. The CO2 emission also varied with respect to work load and range from 16.27Mt to 141.87Mt.
4. The proposed approach requires 3.15138 hours of training, which is slightly more than certain options that range from 2.768755 to 3.01241 hours.
5. The proposed technique has an inference time of 39.80421 seconds, which is comparable to the alternatives (38.8899 to 39.63803).

6. The proposed technique provides a high throughput of 8659.64 Mbps, which is comparable to or better than the alternatives (8276.669 to 8591.92 Mbps).

7. The proposed method has a makespan of 13.81982 seconds, which is comparable to existing ways ranging from 13.97053 to 14.17695 seconds.

6.3 Comparative Analysis

Energy Consumption of the proposed and the existing studies is as follows.

- Proposed Algorithm: Average energy consumption values range from 0.39987KJ to 0.716416KJ, showcasing its ability to consistently achieve lower energy consumption across different workloads.
- MET Algorithm: The energy consumption ranges from 0.11606KJ to 0.790325KJ across different workloads.
- Hu et al. Algorithm: Energy consumption values range from 0.42657KJ to 0.790325KJ, indicating higher energy consumption compared to the proposed approach.
- Cai et al. Algorithm: Energy consumption values range from 0.456221KJ to 0.772431KJ, demonstrating comparatively higher energy consumption.
- Jena et al.: Shows comparatively moderate energy consumption (0.459385 KJ), indicating a potential for efficiency improvement through optimization.
- Saurabh et al.: Shows a significantly higher energy consumption level of 0.466659 KJ, indicating the need for adjustments to reduce energy usage.

Cost analysis of the proposed and the existing studies is as follows.

- Proposed Algorithm: Average total cost values range from Rs24.1422 to Rs238.069 monetary units, illustrating its consistent cost-effectiveness across varying workloads.
- MET Algorithm: The total cost, in this case, ranges from Rs 29.5563 to Rs269.327 monitory units across the variable workload values used in the analysis.
- Hu et al. Algorithm: Total cost values range from Rs29.5633 to Rs295.097 monetary units, with higher costs compared to the proposed method.

- Cai et al. Algorithm: Total cost values range from Rs28.0771 to Rs308.596 monetary units, indicating relatively higher costs.
- Jena et al.: Incurs a cost of Rs 150.0256, indicating a considerable investment that may be made more cost-effectively.
- Saurabh et al.: Has a higher cost of Rs 155.1786, indicating possible areas for cost reduction to increase affordability.

CO_2 emission analysis of the proposed and the existing studies is as follows.

- Proposed Algorithm: CO_2 emission values range from 16.27168893Mt to 141.8717559Mt, reflecting its ability to consistently achieve lower CO_2 emissions.
- MET Algorithm: The emission values for the MET algorithm ranges from 17.28573Mt to 1777937Mt across different workload used in the analysis.
- Hu et al. Algorithm: CO_2 emission values range from 17.3421233Mt to 150.2225372Mt, indicating higher emissions compared to the proposed approach.
- Cai et al. Algorithm: CO_2 emission values range from 17.86211438Mt to 154.804145Mt, demonstrating relatively higher emissions.
- Jena et al. reported CO2 emissions of 92.60531 Mt, falling within the range of 89.35918 Mt to 94.00032 Mt observed across all methodologies.
- Saurabh et al.: Produces even higher CO2 emissions of 94.00032 Mt, which falls within the range of 89.35918 Mt to 94.00032 Mt recorded across all methods.

Complexity analysis of the proposed and the existing studies is as follows.

- The suggested method has a training duration of 3.15 hours and an inference time of 39.80 seconds, thereby balancing training and inference efficiency.
- MET has a training duration of 2.99 hours and an inference time of 39.64 seconds, suggesting effective model training and inference.
- Hu et al. accomplish a training time of 2.83 hours and an inference time of 39.43 seconds, demonstrating effective training and inference procedures.
- Cai et al. obtain a training time of 2.77 hours and an inference time of 38.89 seconds, demonstrating effective resource use.

- Jena et al. show a training time of 3.01 hours and an inference time of 39.13 seconds, demonstrating efficiency in both training and inference tasks.
- Saurabh et al. obtain a training time of 2.98 hours and an inference time of 38.94 seconds, demonstrating effective model training and inference.

Throughput analysis of the proposed and the existing studies is as follows.

- The proposed approach produces a high throughput rate of 8276.669 to 8659.64 Mbps, indicating effective data transmission and processing capabilities.
- MET achieves competitive throughput rates ranging from 8276.669 to 8591.92 Mbps, suggesting good data flow management inside the system.
- Hu et al. achieve throughput rates ranging from 8276.669 to 8397.666 Mbps, demonstrating effective data processing and transmission efficiency.
- Cai et al. attain competitive throughput rates of 8276.669 to 8276.669 Mbps, indicating effective data transmission and processing capabilities.
- Jena et al. demonstrate competitive throughput rates ranging from 8276.669 to 8276.669 Mbps, indicating that data flow is well managed inside the system.
- Saurabh et al. achieve comparable throughput rates ranging from 8276.669 to 8278.295 Mbps, demonstrating effective data processing and transmission efficiency.

Makespan analysis of the proposed work and the existing works is as follows.

- The proposed work has a competitive makespan of 13.81982 to 14.17695 seconds, suggesting efficient task scheduling and completion inside the system.
- MET ensures a competitive makespan of 13.97053 to 14.17695 seconds, demonstrating good task management and timely execution.
- Hu et al. obtain makespan values ranging from 14.13148 to 14.17695 seconds while ensuring effective task scheduling and completion inside the system.
- Cai et al. maintain a competitive makespan of 14.16384 to 14.16384 seconds, implying efficient task scheduling and completion equivalent to other methods.

- Jena et al. ensure a competitive makespan of 14.16384 to 14.17695 seconds, demonstrating good task management and timely execution.
- Saurabh et al. maintain a competitive makespan range from 14.17695 to 14.17695 seconds, indicating effective job scheduling and completion within the system.

According to the improvement study for the three parameters at the highest work load (100,000 MIPS), the proposed work saves 9.35% more energy than MET, 3.79% more energy than Hu et al., and 7.26% more energy than Cai et al. The proposed method outperforms MET by 9%, Hu et al. by 24%, and Cai et al. by 3% in terms of cost. A CO2 emission analysis is also included in the study, demonstrating that the proposed method consistently outperforms existing methods across a wide variety of workloads. At varied workloads, this suggests that the CO2 with MET methodology produces 6.24% higher emissions than the Proposed method. CO2 Hu et al. and CO2 Cai et al. techniques yield comparable findings.

By consistently achieving lower energy consumption, reduced costs, and minimized CO_2 emissions across varying workloads, the proposed algorithm demonstrates its adaptability and effectiveness. This research not only contributes to the field of task scheduling and optimization but also sets a precedent for sustainable computing practices. As cloud computing and IoT applications continue to evolve, the proposed approach offers a promising solution for enhancing performance, reducing environmental impact, and ensuring economic viability.

The pursuit of intelligent scheduling algorithms, as showcased in this study, opens doors to a future where technological advancements align with environmental responsibility. The integration of nature-inspired techniques and machine learning methodologies represents a pivotal step towards addressing resource management challenges in innovative and holistic ways. In a world where the demand for computational resources grows incessantly, the findings and contributions of this study lay the foundation for a more efficient, sustainable, and environmentally conscious approach to computing. By leveraging the insights gained from comparative analyses and embracing the principles of energy efficiency, cost-

effectiveness, and reduced emissions, we pave the way for a brighter and more responsible digital future.

Anticipating the road ahead, the meticulous comparative analysis of energy consumption, cost, and CO_2 emissions in this study unveils a realm of intriguing possibilities for the future trajectory of computing and resource management. As technology continues its rapid evolution and global concerns for environmental sustainability deepen, the integration of inventive approaches like the proposed algorithm sets the stage for transformative advancements. The harmonization of nature-inspired techniques and reinforcement learning which are at the vanguard of machine learning innovation, introduces an uncharted landscape of potential. This synergy offers solutions that not only elevate performance and efficiency but also substantially mitigate the carbon footprint associated with modern computing infrastructures.

6.4 Future Scope

Future study in the field of multi-objective task scheduling and resource allocation can focus on a number of intriguing routes. For starters, researching advanced machine learning approaches other than Q-learning, such as deep reinforcement learning, could improve the algorithm's ability to adapt to changing surroundings and maximize resource allocation decisions. Aside from flower pollination, combining additional nature-inspired optimization methods with reinforcement learning methodologies may open up new paths for generating superior scheduling solutions. Furthermore, examining the suggested algorithm's scalability and applicability in large-scale distributed systems and heterogeneous computing settings could shed light on its real-world implementation potential. Finally, tackling rising security, privacy, and fairness concerns in scheduling algorithms for cloud computing and IoT settings offers an attractive research direction for ensuring the robustness and dependability of future scheduling solutions.

List of Publications

1. R. Kaur, D. Anand, U. Kaur, S. Verma, Kavita, S.W Park, ASM. S Hosen, and I. H. Ra. "An Advanced Job Scheduling Algorithmic Architecture to Reduce Energy Consumption and CO2 Emissions in Multi-Cloud." Electronics vol.12, no. 8, p. 1810, 2023.
2. R. Kaur, S. Verma, N. Z. Jhanjhi, and M. N. Talib. "A comprehensive survey on load and resources management techniques in the homogeneous and heterogeneous cloud environment." In Journal of Physics: Conference Series, vol. 1979, no. 1, p. 012036. IOP Publishing, 2021.
3. R. Kaur, D. Anand, U. Kaur, S. Verma, and Kavita. "Multi-Objective Resource Optimization Using Enhanced FPA-DRL in a Heterogeneous Cloud Computing Environment". Computers, Materials & Continua (CMC) journal of SCI. **(Final Minor Revision Submitted).**
4. R. Kaur, D. Anand, and U. Kaur. "Analysis and Evaluation of Bio-inspired Algorithmic Framework, Potential Application in Cloud/Multi-cloud Environment". International Conference on Data Science and Computational Intelligence (ICDSCI-2022). **(Presented).**
5. R. Kaur, D. Anand, and U. Kaur, "DRL based multi-objective resource optimization technique in a multi-cloud environment". EAI IC4S 2023 - 4th EAI International Conference on Cognitive Computing and Cyber Physical Systems. **(Presented and Published).**
6. R. Kaur, D. Anand, and U. Kaur. "Analysis and Evaluation of Bio-inspired Algorithmic Framework, Potential Application in Cloud/Multi-Cloud Environment". 5th IEEE International Conference on Cybernetics, Cognition and Machine Learning Applications, Germany (ICCCMLA 2023). **(Presented).**
7. R. Kaur, D. Anand, U. Kaur, J.Kaur, S. Verma, and Kavita. "Deep Reinforcement learning based intelligent resource allocation techniques with applications to cloud computing". RTIP2R 2023 **(Presented)**

Bibliography

[1] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the Cloud? An architectural map of the Cloud landscape," in *2009 ICSE workshop on software engineering challenges of cloud computing*, 2009, pp. 23–31.

[2] C. M. Mohammed, S. R. M. Zeebaree, and others, "Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review," *International Journal of Science and Business*, vol. 5, no. 2, pp. 17–30, 2021.

[3] F. F. Moghaddam, M. B. Rohani, M. Ahmadi, T. Khodadadi, and K. Madadipouya, "Cloud computing: Vision, architecture and Characteristics," in *2015 IEEE 6th control and system graduate research colloquium (ICSGRC)*, 2015, pp. 1–6.

[4] N. Khan, N. Ahmad, T. Herawan, and Z. Inayat, "Cloud Computing: Locally Sub-Clouds instead of Globally One Cloud," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 2, no. 3, pp. 68–85, Jan. 2012, doi: 10.4018/IJCAC.2012070103.

[5] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," *Computer Security*, pp. 1–7, 2011, doi: 10.6028/NIST.SP.800-145.

[6] A. Chinthas, D. Rani, and R. K. Ranjan, "A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 458–461, 2014.

[7] A. A. Patel and J. N. Rathod, "Reducing Power Consumption & Delay Aware Resource Allocation in Cloud Data centers," *Journal of Information, Knowledge and Research in Computer Engineering*, vol. 1, pp. 337–339, 2010.

[8] P. Guo and L. Bu, "The hierarchical resource management model based on cloud computing," in *2012 IEEE Symposium on Electrical \& Electronics Engineering (EEESYM)*, 2012, pp. 471–474.

[9] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, "How Amazon web services uses formal methods,"

Communications of the ACM, vol. 58, no. 4, pp. 66–73, 2015.

[10] W. Y. C. Wang, A. Rashid, and H.-M. Chuang, "Toward the trend of cloud computing," *Journal of Electronic Commerce Research*, vol. 12, no. 4, p. 238, 2011.

[11] D. C. Wyld, *Moving to the cloud: An introduction to cloud computing in government*. IBM Center for the Business of Government, 2009.

[12] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud computing virtualization of resources allocation for distributed systems," *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, pp. 98–105, 2020.

[13] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, pp. 7–18, 2010.

[14] S. S. Manvi and G. K. Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of network and computer applications*, vol. 41, pp. 424–440, 2014.

[15] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. Talbi, "Towards understanding uncertainty in cloud computing resource provisioning," *Procedia Computer Science*, vol. 51, pp. 1772–1781, 2015.

[16] D. Puthal, B. P. S. Sahoo, S. Mishra, and S. Swain, "Cloud computing features, issues, and challenges: a big picture," in *2015 International Conference on Computational Intelligence and Networks*, 2015, pp. 116–123.

[17] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler, "More than bin packing: Dynamic resource allocation strategies in cloud data centers," *Information Systems*, vol. 52, pp. 83–95, 2015.

[18] J. Li, D. Li, Y. Ye, and X. Lu, "Efficient multi-tenant virtual machine allocation in cloud data centers," *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 81–89, 2015.

[19] D. K. Viswanath, S. Kusuma, & Saroj, and K. Gupta, "Cloud Computing Issues and Benefits Modern Education," *Global Journal of Computer Science and Technology*, vol. 12, no. B10, pp. 15–19, 2012.

[20] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service

composition: A systematic literature review," *Expert systems with applications*, vol. 41, no. 8, pp. 3809–3824, 2014.

[21] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *2012 international conference on computing, electronics and electrical technologies (ICCEET)*, 2012, pp. 877–880.

[22] V. K. Reddy and L. S. S. Reddy, "Security architecture of cloud computing," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, no. 9, pp. 7149–7155, 2011.

[23] D. G. Velev, "Challenges and opportunities of cloud-based mobile learning," *International Journal of Information and Education Technology*, vol. 4, no. 1, p. 49, 2014.

[24] T. Renugadevi and K. Geetha, "Task aware optimized energy cost and carbon emission-based virtual machine placement in sustainable data centers," *Journal of Intelligent & Fuzzy Systems*, vol. 41, no. 5, pp. 5677–5689, Jan. 2021, doi: 10.3233/JIFS-189887.

[25] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, Oct. 2019, doi: 10.1016/J.JNCA.2019.06.006.

[26] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Computer Science*, vol. 57, pp. 545–553, 2015.

[27] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Distributed Multi-Cloud Multi-Access Edge Computing by Multi-Agent Reinforcement Learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2565–2578, Apr. 2021, doi: 10.1109/TWC.2020.3043038.

[28] P. S. Othman, R. R. Ihsan, and R. M. Abdulhakeem, "The Genetic Algorithm (GA) in Relation to Natural Evolution," *Academic Journal of Nawroz University*, vol. 11, no. 3, pp. 243–250, Aug. 2022, doi: 10.25007/AJNU.V11N3A1414.

[29] S. B. Sangeetha, R. Sabitha, B. Dhiyanesh, G. Kiruthiga, N. Yuvaraj, and R. A. Raja, "Resource Management Framework Using Deep Neural Networks in

Multi-Cloud Environment," *EAI/Springer Innovations in Communication and Computing*, pp. 89–104, 2022, doi: 10.1007/978-3-030-74402-1_5/COVER.

[30] S. A. Almazok and B. Bilgehan, "A novel dynamic source routing (DSR) protocol based on minimum execution time scheduling and moth flame optimization (MET-MFO)," *Eurasip Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–26, Dec. 2020, doi: 10.1186/S13638-020-01802-5/FIGURES/15.

[31] R. Su and G. Woeginger, "String execution time for finite languages: Max is easy, min is hard," *Automatica*, vol. 47, no. 10, pp. 2326–2329, Oct. 2011, doi: 10.1016/J.AUTOMATICA.2011.06.024.

[32] G. Rjoub and J. Bentahar, "Cloud task scheduling based on swarm intelligence and machine learning," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2017, pp. 272–279.

[33] A. Hanani, A. M. Rahmani, and A. Sahafi, "A multi-parameter scheduling method of dynamic workloads for big data calculation in cloud computing," *Journal of Supercomputing*, vol. 73, no. 11, pp. 4796–4822, Nov. 2017, doi: 10.1007/S11227-017-2050-6/TABLES/8.

[34] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine Learning Interpretability: A Survey on Methods and Metrics," *Electronics 2019, Vol. 8, Page 832*, vol. 8, no. 8, p. 832, Jul. 2019, doi: 10.3390/ELECTRONICS8080832.

[35] B. Mahesh, "Machine Learning Algorithms-A Review," *International Journal of Science and Research*, vol. 9, no. 1, pp. 981–986, 2018, doi: 10.21275/ART20203995.

[36] M. A. Mahdi, K. M. Hosny, and I. Elhenawy, "Scalable Clustering Algorithms for Big Data: A Review," *IEEE Access*, vol. 9, pp. 80015–80027, 2021, doi: 10.1109/ACCESS.2021.3084057.

[37] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simulation Modelling Practice and Theory*, vol. 111, p. 102353, Sep. 2021, doi: 10.1016/J.SIMPAT.2021.102353.

[38] H. S. Yahia, S. R. Zeebaree, M. A. Sadeeq, N. O. Salim, S. F. Kak, A. Z. Adel, A. A. Salih, and H. A. Hussein, "Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling," *Asian Journal of Research in Computer Science*, vol. 8, no. 2, pp. 1–16, 2021.

[39] S. Mirjalili, J. Song Dong, A. S. Sadiq, and H. Faris, "Genetic algorithm: Theory, literature review, and application in image reconstruction," *Studies in Computational Intelligence*, vol. 811, pp. 69–85, 2020, doi: 10.1007/978-3-030-12127-3_5/COVER.

[40] B. Muthulakshmi and K. Somasundaram, "A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment," *Cluster Computing*, vol. 22, no. 5, pp. 10769–10777, Sep. 2019, doi: 10.1007/S10586-017-1174-Z/METRICS.

[41] S. Sridharan, R. K. Subramanian, and A. K. Srirangan, "Physics based meta heuristics in manufacturing," *Materials Today: Proceedings*, vol. 39, pp. 805–811, Jan. 2021, doi: 10.1016/J.MATPR.2020.09.775.

[42] D. Freitas, L. G. Lopes, and F. Morgado-Dias, "Particle Swarm Optimisation: A Historical Review Up to the Current Developments," *Entropy*, vol. 22, no. 3, p. 362, 2020, doi: 10.3390/E22030362.

[43] Ş. Öztürk, R. Ahmad, and N. Akhtar, "Variants of Artificial Bee Colony algorithm and its applications in medical image processing," *Applied Soft Computing Journal*, vol. 97, 2020, doi: 10.1016/j.asoc.2020.106799.

[44] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275–295, Nov. 2015, doi: 10.1016/J.EIJ.2015.07.001.

[45] I. Fister, X. S. Yang, D. Fister, and I. Fister, "Firefly algorithm: A brief review of the expanding literature," *Studies in Computational Intelligence*, vol. 516, pp. 347–360, 2014, doi: 10.1007/978-3-319-02141-6_17/COVER.

[46] Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "Grasshopper optimization algorithm: Theory, variants, and applications," *IEEE Access*, vol. 9, pp. 50001–50024, 2021, doi: 10.1109/ACCESS.2021.3067597.

[47] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, Jul. 2013, doi:

10.1016/J.INS.2013.02.041.

[48] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1431–1441, 2013.

[49] M. Dhanalakshmi and A. Basu, "Task scheduling techniques for minimizing energy consumption and response time in cloud computing," *Int J Eng Res Technol (IJERT)*, vol. 3, no. 7, pp. 181–2278, 2014.

[50] S. Singh Brar and S. Rao, "Optimizing Workflow Scheduling using Max-Min Algorithm in Cloud Environment," *International Journal of Computer Applications*, vol. 124, no. 4, pp. 975–8887, 2015.

[51] S. K. Panda and P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *The Journal of Supercomputing*, vol. 71, pp. 1505–1533, 2015.

[52] S. Hosseinimotlagh, F. Khunjush, and R. Samadzadeh, "SEATS: smart energy-aware task scheduling in real-time cloud computing," *The Journal of Supercomputing*, vol. 71, pp. 45–66, 2015.

[53] L. Ismail and A. Fardoun, "Eats: Energy-aware tasks scheduling in cloud computing systems," *Procedia Computer Science*, vol. 83, pp. 870–877, 2016.

[54] M. Hemamalini and M. V Srinath, "Performance Analysis of Balanced Minimum Execution Time Grid Task scheduling Algorithm," *International Journal of Communication and Networking System*, vol. 5, no. 2, pp. 96–100, 2016.

[55] K. Maheshwari, E.-S. Jung, J. Meng, V. Morozov, V. Vishwanath, and R. Kettimuthu, "Workflow performance improvement using model-based scheduling over multiple clusters and clouds," *Future generation computer systems*, vol. 54, pp. 206–218, 2016.

[56] M. Jasraj, M. Kumar, and M. Vardhan, "Cost Effective Genetic Algorithm for Workflow Scheduling in Cloud Under Deadline Constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016, Accessed: Jun. 30, 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7542128

[57] S. K. Panda and P. K. Jana, "SLA-based task scheduling algorithms for

heterogeneous multi-cloud environment," *The Journal of Supercomputing*, vol. 73, pp. 2730–2762, 2017.

[58] D. Maharana, B. Sahoo, and S. Sethi, "Energy-efficient real-time tasks scheduling in cloud data centers," *International Journal of Science Engineering and Advance Technology, IJSEAT*, vol. 4, no. 12, pp. 768–773, 2017.

[59] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLOS ONE*, vol. 12, no. 5, p. e0176321, May 2017, doi: 10.1371/JOURNAL.PONE.0176321.

[60] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "Distributed hybrid scheduling in multi-cloud networks using conflict graphs," *IEEE Transactions on Communications*, vol. 66, no. 1, pp. 209–224, 2017.

[61] S. P. Praveen, K. T. Rao, and B. Janakiramaiah, "Effective allocation of resources and task scheduling in cloud environment using social group optimization," *Arabian Journal for Science and Engineering*, vol. 43, pp. 4265–4272, 2018.

[62] K. Duan, S. Fong, S. W. I. Siu, W. Song, and S. S.-U. Guan, "Adaptive incremental genetic algorithm for task scheduling in cloud environments," *Symmetry*, vol. 10, no. 5, p. 168, 2018.

[63] S. K. Panda and P. K. Jana, "Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment," *Information Systems Frontiers*, vol. 20, pp. 373–399, 2018.

[64] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *Journal of Network and Computer Applications*, vol. 114, pp. 108–122, Jul. 2018, doi: 10.1016/J.JNCA.2018.03.028.

[65] W. Lin, W. Wang, W. Wu, X. Pang, B. Liu, and Y. Zhang, "A heuristic task scheduling algorithm based on server power efficiency model in cloud environments," *Sustainable computing: informatics and systems*, vol. 20, pp. 56–65, 2018.

[66] T. Jena and J. R. Mohanty, "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing," *Arabian Journal for Science*

and Engineering, vol. 43, no. 8, pp. 4115–4130, 2018.

[67] S. K. Mishra, D. Puthal, B. Sahoo, S. K. Jena, and M. S. Obaidat, "An adaptive task allocation technique for green cloud computing," *The Journal of Supercomputing*, vol. 74, pp. 370–385, 2018.

[68] D. Hazra, A. Roy, S. Midya, and K. Majumder, "Energy aware task scheduling algorithms in cloud environment: A survey," in *Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1*, 2018, pp. 631–639.

[69] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–16, 2018.

[70] TangXiaoyong, LiaoXiaoyi, ZhengJie, and YangXiaopan, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Computing*, vol. 21, no. 3, pp. 1581–1593, Sep. 2018, doi: 10.5555/3287988.3288006.

[71] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Cluster Computing*, vol. 22, no. 2, pp. 509–527, 2019.

[72] S. K. Panda, I. Gupta, and P. K. Jana, "Task scheduling algorithms for multi-cloud systems: allocation-aware approach," *Information Systems Frontiers*, vol. 21, pp. 241–259, 2019.

[73] A. Kaur, P. Singh, R. Singh Batth, and C. Peng Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," *Software: Practice and Experience*, vol. 52, no. 3, pp. 689–709, Mar. 2020, doi: 10.1002/SPE.2802.

[74] S. Gupta, I. Agarwal, and R. S. Singh, "Workflow scheduling using Jaya algorithm in cloud," *Concurrency and computation: practice and experience*, vol. 31, no. 17, pp. 1–13, 2019, doi: 10.1002/cpe.5251.

[75] A. Rehman, S. S. Hussain, Z. ur Rehman, S. Zia, and S. Shamshirband, "Multi-objective approach of energy efficient workflow scheduling in cloud environments," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 8, p. e4949, 2019.

[76] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques

in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, Feb. 2019, doi: 10.1016/J.FUTURE.2018.09.014.

[77] C. Li, J. Zhang, and H. Tang, "Replica-aware task scheduling and load balanced cache placement for delay reduction in multi-cloud environment," *The Journal of Supercomputing*, vol. 75, pp. 2805–2836, 2019.

[78] M. J. Usman, A. S. Ismail, H. Chizari, G. Abdul-Salaam, A. M. Usman, A. Y. Gital, O. Kaiwartya, and A. Aliyu, "Energy-efficient Virtual Machine Allocation Technique Using Flower Pollination Algorithm in Cloud Datacenter: A Panacea to Green Computing," *Journal of Bionic Engineering*, vol. 16, no. 2, pp. 354–366, 2019, doi: 10.1007/s42235-019-0030-7.

[79] M. Masdari and M. Zangakani, "Efficient task and workflow scheduling in inter-cloud environments: challenges and opportunities," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 499–535, 2020.

[80] G. Natesan and A. Chokkalingam, "Multi-objective task scheduling using hybrid whale genetic optimization algorithm in heterogeneous computing environment," *Wireless Personal Communications*, vol. 110, pp. 1887–1913, 2020.

[81] M. Xu and R. Buyya, "Managing renewable energy and carbon footprint in multi-cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 191–202, Jan. 2020, doi: 10.1016/J.JPDC.2019.09.015.

[82] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, "Multi-Objective Task and Workflow Scheduling Approaches in Cloud Computing: a Comprehensive Review," *Journal of Grid Computing*, vol. 18, no. 3, pp. 327–356, Sep. 2020, doi: 10.1007/S10723-020-09533-Z/METRICS.

[83] M. Sharma and R. Garg, "HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers," *Engineering Science and Technology, an International Journal*, vol. 23, no. 1, pp. 211–224, 2020.

[84] H. Aziza and S. Krichen, "A hybrid genetic algorithm for scientific workflow scheduling in cloud environment," *Neural Computing and Applications*, vol.

32, no. 18, pp. 15263–15278, Sep. 2020, doi: 10.1007/S00521-020-04878-8/METRICS.

[85] T. Bezdan, M. Zivkovic, M. Antonijevic, T. Zivkovic, and N. Bacanin, "Enhanced Flower Pollination Algorithm for Task Scheduling in Cloud Computing Environment," *Lecture Notes in Networks and Systems*, vol. 141, pp. 163–171, 2021, doi: 10.1007/978-981-15-7106-0_16/COVER.

[86] M. Sharma and R. Garg, "An artificial neural network based approach for energy efficient task scheduling in cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 26, p. 100373, 2020.

[87] Z. Wen, S. Garg, G. S. Aujla, K. Alwasel, D. Puthal, S. Dustdar, A. Y. Zomaya, and R. Ranjan, "Running Industrial Workflow Applications in a Software-Defined Multicloud Environment Using Green Energy Aware Scheduling Algorithm," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5645–5656, Aug. 2021, doi: 10.1109/TII.2020.3045690.

[88] T. Mohanraj and R. Santhosh, "Multi-swarm optimization model for multi-cloud scheduling for enhanced quality of services," *Soft Computing*, pp. 1–11, 2021.

[89] S. Velliangiri, P. Karthikeyan, V. M. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 631–639, 2021.

[90] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Computing and Applications*, vol. 33, no. 19, pp. 13075–13088, Oct. 2021, doi: 10.1007/S00521-021-06002-W/FIGURES/8.

[91] W. Ahmad and B. Alam, "An efficient list scheduling algorithm with task duplication for scientific big data workflow in heterogeneous computing environments," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, p. e5987, Mar. 2021, doi: 10.1002/CPE.5987.

[92] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, and H. Kaur, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments," *IEEE Access*, vol. 9,

pp. 117325–117337, 2021.

[93] R. Pradhan and S. C. Satapathy, “Energy-Aware Cloud Task Scheduling algorithm in heterogeneous multi-cloud environment,” *Intelligent Decision Technologies*, no. Preprint, pp. 1–6, 2022.

[94] R. Pradhan and S. C. Satapathy, “Particle Swarm Optimization-Based Energy-Aware Task Scheduling Algorithm in Heterogeneous Cloud,” in *Communication, Software and Networks: Proceedings of INDIA 2022*, Springer, 2022, pp. 439–450.

[95] T. Jena and J. R. Mohanty, “GA-based efficient resource allocation and task scheduling in a multi-cloud environment,” *International Journal of Advanced Intelligence Paradigms*, vol. 22, no. 1–2, pp. 54–71, 2022.

[96] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, “Prioritized energy efficient task scheduling algorithm in cloud computing using whale optimization algorithm,” *Wireless Personal Communications*, vol. 126, no. 3, pp. 2231–2247, 2022.

[97] N. Miglani, G. Sharma, and S. Khurana, “Multi-objective reliability-based workflow scheduler: An elastic and persuasive task scheduler based upon modified-flower pollination algorithm in cloud environment,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 22, p. e7150, Oct. 2022, doi: 10.1002/CPE.7150.

[98] R. Chen, X. Chen, and C. Yang, “Using a task dependency job-scheduling method to make energy savings in a cloud computing environment,” *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4550–4573, 2022.

[99] N. P. Marri and N. R. Rajalakshmi, “MOEAGAC: An energy aware model with genetic algorithm for efficient scheduling in cloud computing,” *International Journal of Intelligent Computing and Cybernetics*, vol. 15, no. 2, pp. 318–329, 2022.

[100] B. Kruekaew and W. Kimpan, “Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning,” *IEEE Access*, vol. 10, pp. 17803–17818, 2022, doi: 10.1109/ACCESS.2022.3149955.

[101] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, “Multi-objective grey wolf

optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.

[102] K. Malathi and K. Priyadarsini, "Hybrid lion--GA optimization algorithm-based task scheduling approach in cloud computing," *Applied Nanoscience*, vol. 13, no. 3, pp. 2601–2610, 2023.

[103] J. Mahilraj, P. Sivaram, N. Lokesh, and B. Sharma, "An Optimised Energy Efficient Task Scheduling Algorithm based on Deep Learning Technique for Energy Consumption," in *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, 2023, pp. 1–7.

[104] S. P. Jaiprakash, H. K. Arya, I. Gupta, and T. Badal, "Energy Optimized Workflow Scheduling in IaaS Cloud: A Flower Pollination based Approach," 2023.

[105] S. Boopalan and P. Goswami, "CRA-DP-GA for Efficient Utilization of Resource through Virtual Machine and Efficient VM in Cloud Data Centre," *Mathematical Statistician and Engineering Applications*, vol. 72, no. 1, pp. 233–251, 2023.

[106] X. Cai, S. Geng, D. Wu, J. Cai, and J. Chen, "A Multicloud-Model-Based Many-Objective Intelligent Algorithm for Efficient Task Scheduling in Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9645–9653, Jun. 2021, doi: 10.1109/JIOT.2020.3040019.

[107] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Generation Computer Systems*, vol. 50, pp. 3–21, Sep. 2015, doi: 10.1016/J.FUTURE.2015.01.007.

[108] C. Szabo, Q. Z. Sheng, T. Kroeger, Y. Zhang, and J. Yu, "Science in the Cloud: Allocation and Execution of Data-Intensive Scientific Workflows," *Journal of Grid Computing*, vol. 12, no. 2, pp. 245–264, 2014, doi: 10.1007/S10723-013-9282-3.

[109] S. S. Murad, R. B. Abdal, N. Alsandi, R. Faraj, S. Salam Murad, R. Badeel, N. Salih, A. Alsandi, R. F. Alshaaya, R. A. Ahmed, A. Muhammed, and M. Derahman, "OPTIMIZED MIN-MIN TASK SCHEDULING ALGORITHM FOR SCIENTIFIC WORKFLOWS IN A CLOUD ENVIRONMENT," *Article*

in Journal of Theoretical and Applied Information Technology, vol. 100, no. 2, pp. 480–506, 2022, Accessed: Jul. 28, 2023. [Online]. Available: https://www.researchgate.net/publication/358461191

[110] D. Poola, M. A. Salehi, K. Ramamohanarao, and R. Buyya, "A Taxonomy and Survey of Fault-Tolerant Workflow Management Systems in Cloud and Distributed Computing Environments," *Software Architecture for Big Data and the Cloud*, pp. 285–320, Jan. 2017, doi: 10.1016/B978-0-12-805467-3.00015-6.

[111] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "Cost-effective workflow scheduling approach on cloud under deadline constraint using firefly algorithm," *Applied Intelligence*, vol. 51, no. 3, pp. 1629–1644, Mar. 2021, doi: 10.1007/S10489-020-01875-1/METRICS.

[112] Y. Shen, Z. Bao, X. Qin, and J. Shen, "Adaptive task scheduling strategy in cloud: when energy consumption meets performance guarantee," *World Wide Web*, vol. 20, no. 2, pp. 155–173, Mar. 2017, doi: 10.1007/S11280-016-0382-4/METRICS.

[113] T.-Y. Liang and Y.-J. Li, "A Location-Aware Service Deployment Algorithm Based on K-Means for Cloudlets," vol. 2017, pp. 1–10, 2017, doi: 10.1155/2017/8342859.

[114] D. Kakkar and G. S. Young, "Heuristic of VM Allocation to Reduce Migration Complexity at Cloud Server".

[115] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, "Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing," *Computing*, vol. 98, no. 3, pp. 303–317, Mar. 2016, doi: 10.1007/S00607-015-0467-4/METRICS.

[116] M. Ghobaei-Arani, M. Shamsi, and A. A. Rahmanian, "An efficient approach for improving virtual machine placement in cloud computing environment," *Journal of Experimental & Theoretical Artificial Intelligence* , vol. 29, no. 6, pp. 1149–1171, Nov. 2017, doi: 10.1080/0952813X.2017.1310308.

[117] R. Ojstersek, M. Brezocnik, and B. Buchmeister, "Multi-objective optimization of production scheduling with evolutionary computation: A review," *International Journal of Industrial Engineering Computations*, vol. 11, no. 3, pp. 359–376, 2020, doi: 10.5267/J.IJIEC.2020.1.003.

[118] A. M. Senthil Kumar and M. Venkatesan, “Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment,” *Wireless Personal Communications*, vol. 107, no. 4, pp. 1835–1848, Aug. 2019, doi: 10.1007/S11277-019-06360-8/METRICS.

[119] H. Lu, R. Zhou, Z. Fei, and J. Shi, “A multi-objective evolutionary algorithm based on Pareto prediction for automatic test task scheduling problems,” *Applied Soft Computing*, vol. 66, pp. 394–412, May 2018, doi: 10.1016/J.ASOC.2018.02.050.

[120] K. Naik, G. Meera Gandhi, and S. H. Patil, “Multiobjective virtual machine selection for task scheduling in cloud computing,” *Advances in Intelligent Systems and Computing*, vol. 798, pp. 319–331, 2019, doi: 10.1007/978-981-13-1132-1_25/COVER.

[121] F. Li, L. Zhang, T. W. Liao, and Y. Liu, “Multi-objective optimisation of multi-task scheduling in cloud manufacturing,” *International Journal of Production Research*, vol. 57, no. 12, pp. 3847–3863, Jun. 2018, doi: 10.1080/00207543.2018.1538579.

[122] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy, “A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems,” *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, Sep. 2020, doi: 10.1109/JSYST.2019.2960088.

[123] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, “An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments,” *Neural Computing and Applications*, vol. 32, no. 6, pp. 1531–1541, Mar. 2020, doi: 10.1007/S00521-019-04119-7/METRICS.

[124] G. Sreenivasulu and I. Paramasivam, “Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing,” *Evolutionary Intelligence*, vol. 14, no. 2, pp. 1015–1022, Jun. 2021, doi: 10.1007/S12065-020-00517-2/METRICS.

[125] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, “Heuristic initialization of PSO task scheduling algorithm in cloud computing,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2370–

2382, Jun. 2022, doi: 10.1016/J.JKSUCI.2020.11.002.

[126] X. Huang, T. H. Klinge, J. I. L. B, X. Li, and J. H. Lutz, "Flower Pollination Algorithm for Global Optimization," *Unconventional Computation and Natural Computation*, vol. 7445, pp. 29–40, 2012, doi: 10.1007/978-3-642-32894-7.

[127] P. E. Mergos and X. S. Yang, "Flower pollination algorithm parameters tuning," *Soft Computing*, vol. 25, no. 22, p. 14429, Nov. 2021, doi: 10.1007/S00500-021-06230-1.

[128] M. J. Usman, A. S. Ismail, A. Y. Gital, A. Aliyu, and T. Abubakar, "Energy-efficient resource allocation technique using flower pollination algorithm for cloud datacenters," *Advances in Intelligent Systems and Computing*, vol. 843, pp. 15–29, 2019, doi: 10.1007/978-3-319-99007-1_2/COVER.

[129] M. Gokuldhev, G. Singaravel, and N. R. Ram Mohan, "Multi-Objective Local Pollination-Based Gray Wolf Optimizer for Task Scheduling Heterogeneous Cloud Environment," *Journal of Circuits, Systems and Computers* , vol. 29, no. 7, pp. 1–24, Sep. 2019, doi: 10.1142/S0218126620501005.

[130] S. Dhivya and R. Arul, "Hybrid Flower Pollination Algorithm for Optimization Problems," *Book cover Book cover Proceedings of the International Conference on Computational Intelligence and Sustainable Technologies*, pp. 751–762, 2022, doi: 10.1007/978-981-16-6893-7_65.

[131] A. Kaur, S. Kumar, D. Gupta, Y. Hamid, M. Hamdi, A. Ksibi, H. Elmannai, and S. Saini, "Algorithmic Approach to Virtual Machine Migration in Cloud Computing with Updated SESA Algorithm," *Sensors (Basel, Switzerland)*, vol. 23, no. 13, p. 6117, Jul. 2023, doi: 10.3390/S23136117.

[132] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, no. 3, pp. 849–861, Feb. 2018, doi: 10.1016/J.FUTURE.2017.09.020.

[133] M. S, B. Prakash, and B. H.M., "A Detailed Survey on various Cloud computing Simulators," *International Journal of Engineering Research* , vol. 4, no. 5, pp. 790–991, 2016, doi: 10.17950/ijer/v5i4/037.

[134] R. Kaur and N. Singh Ghumman, "A Survey and Comparison of Various Cloud Simulators Available for Cloud Environment," *International Journal of*

Advanced Research in Computer and Communication Engineering, vol. 4, no. 5, pp. 605–608, 2015, doi: 10.17148/IJARCCE.2015.45129.

[135] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning 1992 8:3*, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.

[136] F. Flamini, A. Hamann, S. Jerbi, L. M. Trenkwalder, H. P. Nautrup, and H. J. Briegel, “Photonic architecture for reinforcement learning,” *New Journal of Physics*, vol. 22, no. 4, pp. 1–12, Apr. 2020, doi: 10.1088/1367-2630/AB783C.