Bi-Objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing

Ketema Deresa Biru

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

Bi-Objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing

Ketema Deresa Biru.

Advisor: Dr. -Ing Frezewd Lemma

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

Declaration

I declare that this thesis, entitled "**Bi-objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing,**" is my work and has not been submitted to any university for a similar purpose. The references used in this proposal are duly recognized by proper citations.

Ketema Deresa Biru.	_____	_____
Name of student	Signature	Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Bi-objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing,**" prepared under my guidance by **Ketema Deresa Biru** and submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. -Ing Frezewd Lemma _____ _____

Major Advisor. Signature Date

Approval Page of M.Sc. Thesis

I, the advisor of the thesis entitled "**Bi-objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing,**" developed by **Ketema Deresa Biru,** hereby certify that the recommendations and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. -Ing Frezewd Lemma	_____	_____
Major Advisor	Signature	Date

Approval of Board of Reviewers

We, the undersigned, members of the Board of Examiners of the thesis by **Ketema Deresa Biru,** have read and evaluated the thesis entitled "**Bi-objective Optimization of Bandwidth Resources and Energy Consumption for Efficient Virtual Machine Placement in Cloud Computing**" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon the submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

TABLE OF CONTENTS

LIST OF TABLES i
LIST OF FIGURES ii
LIST OF EQUATION iii
LIST OF ABBREVIATIONS AND ACRONYMS iv
ACKNOWLEDGEMENT v
ABSTRACT vi
CHAPTER ONE 1
INTRODUCTION 1
1.1. Background of the Study 1
1.2. Motivation 3
1.3. Statement of the Problem 3
1.4. Research Questions 5
1.5. The Objective of the Research 5
1.5.1. General Objectives 5
1.5.2. Specific Objectives 5
1.6. Significance and Beneficiaries of the Study 5
1.6.1. Significance of the Study 5
1.6.2. Beneficiaries of the study 6
1.7. The expected outcome of the study 6
1.8. Scope and Limitations 7
1.9. Organization of the Thesis 7
CHAPTER TWO 9
LITERATURE REVIEW AND RELATED WORKS 9
2.1. Literature Review 9
2.2. Cloud Computing 9
2.2.1. Cloud Deployment Models 10
2.2.2. Cloud Service Model 11
2.3. Virtualization 14

2.3.1. Types of Virtualizations 16
2.4. Virtual machine placement algorithms 17
2.5. Data center network topology 25
2.6. Related Works 26
CHAPTER THREE 34
PROPOSED METHODOLOGY 34
3.1. Research Design 34
3.2. Data Collections 35
3.3. Simulation Tool 35
3.4. Setup of the simulation environment 38
3.5. Implementation and Coding 41
CHAPTER FOUR 42
PROPOSED SOLUTION 42
4.1. Proposed solution architecture 42
4.2. Proposed solution process 44
4.2.1. Formulation For Bandwidth Resource and Energy Consumption model 44
4.2.2. Optimization Formulation for Bandwidth and Energy consumption 47
4.3. Genetic Algorithm 48
4.3.1. Bi-objective Hybrid Genetic Optimization Algorithm 51
4.3.2. Integration of GA with ACO 52
4.3.3. Virtual Machine Placement Considering Bandwidth Resource and Energy . 55
4.3.4. Flowchart of the Proposed BOHGOA 64
CHAPTER FIVE 66
EXPERIMENTAL RESULTS AND DISCUSSIONS 66
5.1. Performance Evaluation 66
5.1.1. Bandwidth Resource Consumption Evaluation of BOHGOA 66
5.1.2. Energy Consumption Evaluation of BOHGOA 68
5.2. Complexity analysis of BOHGOA 70

CHAPTER SIX 72
CONCLUSION AND FUTURE WORKS 72
6.1. Conclusion 72
6.2. Contribution 73
6.3. Future Works 74
REFERENCE 76
APPENDICES 83
Appendix I: Snapshots of Some Results. 83
Appendix II: Sample Source Code Listing. 83

LIST OF TABLES

Table 2.1: Summary of Related Work Focuses and Variations of Current Work 30
Table 3.2. Configuration Setup of CloudSim Tools and test problems. 38
Table 3.3. Energy consumption of experimental host (watt). 39
Table 3.4. virtual machine's requirements instances 40
Table 3.5. BOHGOA Algorithm Parameters 41
Table 4.1: Main Notations and their definitions 46
Table 4.2. Chromosome encoding representation in GA 49
Table 5.1. Time complexity 70
Table 5.2. Time complexity 71

LIST OF FIGURES

Figure 2.1. Types of Cloud deployment models. 11
Figure 2.2: Cloud Computing Service model 12
Figure 2.3. The Essential Characteristics of Cloud Computing 14
Figure 2.4: Virtualization Mechanism 15
Figure 2.5. virtual machine placement schema. 21
Figure 3.1. Research process 34
Figure 3.2. CloudSim Class Design Diagram 36
Figure 3.3. Host type-1 configuration 39
Figure 3.4. Host type-2 configuration. 40
Figure 3.5. Host and VM instance requirements setup. 41
Figure 4.1. proposed solution architecture 43
Figure 4.2: the population of generated possible solutions representation in GA 49
Figure 4.3. Chromosome representation in GA. 49
Figure 4.4: Flowchart of classical GA 51
Figure 4.5. Flowchart of proposed BOHGOA. 65
Figure 5.1. Bandwidth resource consumption of PM of 240 VMs Placement. 67
Figure 5.2. Bandwidth resource consumption of PM of 500 VMs Placement. 67
Figure 5.3. Energy consumption of PM of 240 VMs Placement 68
Figure 5.4. Energy consumption of PM of 500 VMs Placement. 68

LIST OF EQUATION

Eq 4.1. Energy consumption of each physical machine (PM)..... 45

Eq 4.2. Total Energy consumption of all physical machines (PMs). 45

Eq 4.3. Formulation of Bandwidth Resource Consumption..... 45

Eq 4.4. Minimization formulation for Energy consumption. 47

Eq 4.5 Minimization formulation for Bandwidth resource. 47

Eq 4.6 -4.10 Resource constraint consideration equation----------48

Eq 4.11. Transformation or integration of GA to ACO equations 53

Eq 4.12. Solution Construction 53

Eq 4.13. Path Selection..... 54

Eq 4.14. Heuristic information 54

Eq 4.15. Pheromone Updating..... 54

LIST OF ABBREVIATIONS AND ACRONYMS

ACO	Ant Colony Optimization
ACS-VMC	Ant Colony system virtual machine consolidation
BFD	Best Fit Decreasing
BOHGOA	Bi-objective Hybrid Genetic optimization algorithm
CSP	Cloud Service Providers
DCNs	Data center Networks.
DLMM-VMC	Dynamic Load Mean Multi Virtual Machine Consolidation.
EC2	Elastic Cloud Computing
FFD	First Fit Decreasing.
GA	Genetic algorithm.
IBMMT	Imbalance VM with Minimum Migration Time.
IGA-POP	Improved Permutation-Based Genetic Algorithm.
IoT	Internet of Things.
MBFD	Modified Best Fit Decreasing.
MDF-PC	Modified Discrete Firefly Algorithm power consumption.
NFV	Network Function Virtualization.
OM-FNN	Online Multi-Resource Feed-forward neural network.
PM	Physical Machine
PSO	Particle Swarm Optimization.
QoS	Quality of Service.
SCA	Sine-Cosine Algorithm.
VMP	Virtual Machine Placement.
VMPACS	Virtual machine placement ant colony system.

ACKNOWLEDGEMENT

First and foremost, I would like to express My gratitude to the Almighty for all of his mercies and kindness in providing me with the knowledge, health, and wisdom necessary to work on this thesis and shepherding me to successfully complete it. I now comprehend that every aspect of my life is entirely subject to God's authority, and I genuinely value his benevolence. Baba God, it’s been you all the way.

I would like to express my heartfelt gratitude and profound appreciation to my advisor, Dr.-Ing Frezewd Lemma, for his keen interest, inspiring guidance, and constant encouragement of my work during all stages of bringing this thesis to fruition. The success of this thesis would be improbable without his great supervision and proper follow-up throughout the thesis's progress.

Also, I extend my deepest thanks to all the staff members in the School of Electrical Engineering and Computing at ASTU for their unwavering support and for cultivating a conducive environment I am truly thankful for an opportunities and experiences they shared with me.

Finally, to all my family, my parents, I say thank you for all your support, love, prayer and patience. **Addatti haadhakoo Kuulii Kumsaa, sirraa dhalachuukootti hedduun gammada. Umurii naaf dheeradhu. Sin jaalladha.**

ABSTRACT

Cloud computing has revolutionized IT organizations through its on-demand computing model, with Infrastructure as a Service (IaaS) being one of the leading services. Cloud service providers maintain a large number of physical devices, including servers and networking equipment, to meet user demands. However, these devices consume a significant amount of energy and require sufficient bandwidth to handle data traffic during virtual machine placement. Balancing energy consumption and bandwidth resource utilization is crucial for CSP, and user experience. To address this trade-off, this thesis introduced BOHGOA integrated with ACO for efficient virtual machine placement in cloud computing. It aims to optimize the placement process by considering both bandwidth resource utilization, and energy consumption simultaneously. The proposed approach generates a set of Pareto-optimal solutions, providing a systematic approach of non-dominated solution selection for the bandwidth resource-energy trade-off. The performance of BOHGOA was evaluated using the CloudSim toolkit, comparing it with GA, ACO, and FFD algorithms. The results demonstrated the effectiveness of BOHGOA, showing significant reductions in bandwidth resource consumption by 54.14%, 32.11%, and 57.47% for 240 VM placement and 38.12%, 22.76%, and 47.05% for 500 VM placement when compared with GA, ACO, and FFD, respectively. Additionally, the proposed algorithm achieved notable reductions in physical machine energy consumption by 37.70%, 34.01%, and 40.14% for 240 VM placement, and 27.50%, 22.28%, and 30.52% for 500 VM placement when compared with GA, ACO, and FFD, respectively. These findings highlight the effectiveness and potential benefits of employing BOHGOA as a solution for optimizing virtual machine placement in cloud data centers when compared with these mentioned three algorithms.

***Keywords**: Cloud Computing, Bandwidth resource, Data Center, Energy consumption, efficient Virtual Machine Placement, Physical Machine. Virtual Machine*

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

Before the recent development of cloud computing technology, computing services were thought of as parts that were available for purchase from outside sources. Now, however, computing is considered a service that can be received online through the cloud. Big data analysis, the Internet of Things (IoT), and machine learning are just a few of the applications that have come to realize the utilization of cloud computing to be an interesting platform. It provides new paradigms for the provision of flexible computing resources through the Internet, such as processing power, Internet applications, networks, and storage(Hashem et al., 2015)

Sharing of resources and services has become more common since the pay-as-you-go method gets popularity. A lot of commercial applications now rely heavily on the cloud, which has evolved into an innovative business model. To provide continuous and effective access to resources, cloud computing must deliver more adaptable and high-performance network and service infrastructures (hardware and software). Many service providers, such as Microsoft, Amazon, Google, IBM, and many others, provide various services for hardware, platform, and software, and they are referred to as infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS), respectively. Users of cloud services can access them from anywhere, regardless of time, device, or geography restrictions(Arnold et al., 2020)

Energy consumption has significantly increased as a result of users and service providers' increasing use of cloud infrastructure resources. The data centers that power the cloud infrastructure must run at full capacity in order to fulfill users and cloud service provider needs, which consumes a lot of energy. The high energy consumption and carbon emissions associated with cloud computing are caused by the continual power needs of data center equipment, which are further impacted by the number of active computing components in a data center. Particularly, energy consumption of the host in a data center is computed using the maximum power used when the host is fully utilized, the power consumed when the host is idle, and CPU utilization of the host, which is also a function of time (Niranjan & Pillai, 2019).

Additionally, virtual machines exchange data among themselves. This data exchange is fulfilled by communication among the physical machines (servers) hosting them. If the two virtual machines are placed on different physical machines or placed far apart from each other, links that provide communication between these VMs could suffer congestion. Network switches and network links are required for the transmission of network traffic between communicating virtual machines. Data transmission between virtual machines is influenced by the network resources between physical machines, leading to severe performance deterioration(Yu et al., 2018).

Cloud computing relies on virtualization, which is a new paradigm that abstracts the actual cloud computing resources. Virtualization technology, which is the building block of cloud computing, is now one of the main challenges to achieving green clouds. Accordingly, virtualization can improve cloud energy consumption and become a crucial demand in the operation of modern large cloud DCs. This technology enables cloud data centers to offer virtual machines. Virtualization split the hardware resources of one or more machines into separate environments known as virtual machines(Rawas & Zekri, 2018)

As virtualization becoming more widely spread, virtual machine placement which addresses the problem of how to effectively deploy virtual machines on hosts, has received a lot of interest in cloud computing. Various approaches have been developed to improve virtual machine placement, considering different metrics and objectives, including single, bi-objective, or multiple objective optimization functions that may or may not be related to each other. But, bi-objective minimization of bandwidth resources and energy has not been thoroughly investigated in the deployment of virtual machines in the cloud data center. As the problem size increases, the computational effort required to find the optimal solution becomes substantial(Fatima et al., 2018).

To solve these kinds of problems, we used an evolutionary algorithm that uses bi-objectives to direct the search for solutions in optimization. It works by balancing a set of conflicting objectives and then iteratively generating new solution until a solution is found (Sloss & Gustafson, 2019). Therefore, we proposed bi-objective optimization of bandwidth resources and energy consumption for efficient virtual machine placement in cloud computing using a bi-objective hybrid genetic optimization algorithm to find a near-optimal solution. In this case, we balanced the conflict of both objectives while optimizing them simultaneously.

1.2. Motivation

A cloud data center is equipped with a substantial number of physical machines, typically servers. Data center power management not only has a severe impact on pricing, but can also greatly impact IT deployments. Thus, powering off as many physical machines as possible will improve data center energy savings. Although many large-scale applications or systems can be supported by virtualization, including multi-tier web applications, Map Reduce, and NFV, A task of this size might require running on multiple VMs. Different layers typically use different VMs and have various methods for accessing memory (Moualla, 2019).

For instance, a typical multi-tier web application has three tiers: the web tier for presentation, the app tier for business logic, and the data access tier. Internal traffic, including inter-VM traffic, contributes up to about 80% of the whole traffic of the data center(Noormohammadpour & Raghavendra, 2018). Consequently, this excessive inter-VM traffic causes link congestion and increases infrastructure's network resource utilization in data, which could potentially impact the performance degradation of other applications and services running within the data center. This problem is due to the improper placement of the virtual machine on the server(El Yadari et al., 2022).

Thus, this issue leads to link congestion due to the limited availability of bandwidth resources in the data center. Depending on these two problems, we are motivated to solve them bi-objectively rather than solving a single objective. Bi-objective optimization aims to balance bandwidth resources and energy consumption for virtual machine placement efficiently in data centers or cloud computing services. Hence, balancing the two conflicting objectives motivates us to solve the problems using bi-objective optimization techniques. We used a bi-objective hybrid genetic optimization algorithm and efficiently identified near-optimal solutions that minimize both bandwidth resources and energy consumption simultaneously for deploying virtual machine in cloud data environment.

1.3. Statement of the Problem

Data centers are an essential part of the modern digital infrastructure, providing the necessary computing power to support various online services and applications. However, one of the great challenges facing data centers is their high energy consumption. The Data center servers need enormous amount of power which is leading to high energy costs and carbon emissions. As such, it is essential to find ways to reduce their energy consumption without compromising their functionality. The virtual machines placement in cloud

computing is a critical decision that affects the performance of infrastructure. The objective is to place virtual machine on available hosts. There are several factors to consider when deciding where to place VMs, including bandwidth resources and energy consumption.

Additionally, numerous applications from sizable corporations like those based on Map Reduce, are frequently communicate, so they are too big to be handled by a single virtual machine. As a result, when these communicating virtual machines are installed on physical machine, they use a lot of bandwidth resources like links and switch. The Fat-Tree topology is used in the cloud computing, and heterogeneous switches and physical machines are deployed(Mellette et al., 2016).

The computing resources are consumed due to improper deployment of virtual machines in data centers. To address the issue of placing virtual machines, Various solutions are proposed and still better solutions are proposed in the arena. Many researchers have studied virtual machine placement problems considering different metrics using different optimization techniques. Achieving single-objective optimization is not always good for virtual machine deployment efficiently. Because considering one objective and ignoring other objectives will not a give better optimal solution for the problem. Therefore, due to this problem, some researchers considered multi-objective optimization techniques to achieve multiple matrices(Ganesan & Ganesan, 2021)(Hariharan et al., 2021).

The weighted sum approach has been used to simultaneously optimize multiple objectives, which is traditional method, and provides a single optimal solution based on the assigned weights. However, it's important to note that while the weighted sum approach facilitates simultaneous optimization, it does not capture the balanced trade-offs between the objectives in a comprehensive manner (Bazgan et al., 2022).

Without dealing with bandwidth resources, virtual machines may be placed on servers that have heavily utilized the network resource capacity. This can cause network congestion. Optimizing only for bandwidth resources without considering energy usage, might lead to inefficient energy use. VMs may be deployed on power-hungry servers. This may result in increased energy use. As a result, we focused on identifying the best and most efficient VMP strategy with the dual goals of optimizing bandwidth resources and energy consumption simultaneously. This mechanism is proposed to trade off both objectives by properly placing VMs in the cloud environment.

1.4. Research Questions

This thesis intends to answer the following four research questions:

Q1. How to formulate both bandwidth resources and energy consumption for efficient VMP?

Q2. How to integrate GA (Genetic Algorithms) with ACO (Ant Colony Optimization) for efficient VMP in Cloud Computing?

Q3. How to place Virtual Machine considering bandwidth resource and energy consumption simultaneously using a Bi-objective Hybrid Genetic optimization Algorithm (BOHGOA)?

Q4. How can the BOHGOA performance be evaluated?

1.5. The Objective of the Research

1.5.1. General Objectives

The overall objective of this thesis was Bi-objective optimization of bandwidth resource and energy consumption for efficient virtual machine placement in cloud computing.

1.5.2. Specific Objectives

To achieve the overall objective, the following specific objectives were formulated.

- To formulate optimization for both bandwidth resource and energy consumption model for efficient VMP
- To propose Bi-objective hybrid Genetic optimization algorithm for efficient VMP.
- To use CloudSim on which experiments can be made to implement the proposed VMP algorithm.
- To evaluate and compare bandwidth resource and energy consumption for efficient VMP algorithm with other algorithms

1.6. Significance and Beneficiaries of the Study

1.6.1. Significance of the Study

In basically, virtualization enables dynamic resource sharing across the cloud infrastructure's physical resources, such as CPUs, memory, and storage. Mapping physical machine to virtual machines is a challenging problem in cloud computing. Users request resources from the data center (like CPU, memory, bandwidth, etc.), and if a virtual machine is placed correctly on a physically accessible machine with adequate resources, the user will obtain the requested resources. Therefore, this can improve user experience.

In another instance, when numerous virtual machines request resources at once, the demand for a cloud data center increase, which has led to a rise in energy consumption and bandwidth resource usage. By minimizing both of these factors simultaneously, the study provides a solution that addresses the sustainability and economic concerns of cloud computing. The overloading of physical machines and frequent virtual machine migration that results from too many PMs being turned off or put into hibernation during this time can then harm QoS. Thus, this study for VMP is a critical solution to minimize the probability of excessive bandwidth resource, and energy consumption. Hence, properly allocating resources of a physical host to virtual machines reduces energy consumption and inter-virtual machine bandwidth resource consumption.

1.6.2. **Beneficiaries of the study**

The beneficiaries of the study are given below:

Cloud End-users

End users of cloud services refer to individuals who can access various resources of cloud on the internet, subscribe to cloud services, and use services. Cloud users benefit significantly from the bio-objective optimization of bandwidth resources and energy usage through virtual machine deployment in cloud computing. This optimization enables cloud users to have better access to cloud resources at reduced efficiency. By minimizing energy consumption and bandwidth resources, cloud users can benefit from the improved quality of service offered by cloud service providers.

Cloud service providers

For cloud service providers, the bio-objective bandwidth resource and energy consumption optimization VM placement algorithm provides a way to improve bandwidth resource utilization and energy efficiency while maintaining the quality of service for their customers. By optimizing the VMP, providers of cloud service can make better use of their available resources, reducing the number of Overutilized servers and the need for additional infrastructure.

1.7. The expected outcome of the study

The intended outcome of this study is to find out a solution that provides a balance between two simultaneous objectives, namely:

- Minimizing bandwidth resource utilization: This objective aims to reduce the amount of bandwidth resources used by the virtual machines to communicate with each other,

and this is achieved by traffic minimization among virtual machines in the cloud data center.

- Minimizing energy use: The objective is to reduce the energy consumption of the physical machine. This may be done by deploying virtual machines on servers that use less energy and lowering communication overhead between virtual machines.

In general, the expected outcome of the bi-objective optimization of bandwidth resource and energy consumption for VMP efficiently in cloud computing is a set of solutions that balance the trade-off between bandwidth resource and energy consumption, such that no single solution is optimal in both objectives but the set of solutions that offers trade-offs between the two objectives.

1.8. Scope and Limitations

This research aims to achieve a bi-objective optimization of bandwidth resources and energy consumption during the placement of VMs in data centers. By focusing on these two aspects, the study seeks to enhance the efficiency of VMP. However, it is important to note that other factors such as, security, data center fault tolerance, reliability, and SLA were not within the scope of study. When placing VMs, three dimensions have been taken under consideration: CPU, memory, and bandwidth. These dimensions represent the requirements or specifications of the VMs.

It is worth emphasizing that this study solely focuses on the bandwidth resources and energy consumption aspects of the data centers' hosts. By examining and optimizing these dimensions, the focus is to minimize bandwidth resources and energy consumption. Therefore, this research provides insights into improving the efficiency of VMP in terms of bandwidth and energy.

1.9. Organization of the Thesis

The organization of the thesis describes how the thesis's content and structure are ordered and presented. It provides a roadmap for the reader to navigate through the study by outlining the chapters that compose the complete entire document. Therefore, this thesis's general arrangement is as the following:

The first chapter of this study describes the background of cloud computing, motivation, the statements of the problem, the research questions, Objective of the study, the expected outcome of the study, the scope of the study, the limitations of the study. The second chapter

presents about Cloud Computing as overview, literature review of VMP in cloud computing in detail. It describes different optimization techniques and discusses related work on VMP for cloud computing.

The third chapter discusses the proposed methodology which include: - data collection, Simulation tools, and simulation environment setup. Fourth chapter discussed about the proposed Solutions for the bi-objective optimization of the study which includes: proposed solution architecture, and process. The fifth chapter is all about evaluation and results discussion. In this chapter, the study tried to show the analysis of the experimentation in line with the discussion of the results obtained from the CloudSim simulation tool. The sixth chapter is about the conclusion and future works or recommendations.

In general, we have briefly covered the study's background, motivation for the study, statements of the problems, the research questions that have been raised, objectives, scope, and limitations. We presented about the literature review and related works in the following chapter. To resolve the VMP problem, we also identified gaps in this paper on the related work topic.

CHAPTER TWO

LITERATURE REVIEW AND RELATED WORKS

2.1. Literature Review

This chapter deals with a review of important concepts concerning the research work. It provides a detailed definition of cloud computing, its main characteristics, the cloud computing service model and cloud computing deployment model, virtualization, and virtual machine placement.

2.2. Cloud Computing

In recent years, computing and data processing have shifted from personal computers or supercomputers located in businesses or homes to vast, geographically scattered data centers throughout the world (Furht, 2010). The way that computers and computing resources are utilized has changed significantly. Computing is currently being offered and used as a commercial resource, and consumers are required to pay the provider(s) on a pay-per-use basis, just like they would for other utilities like electricity, water, and gas (Buyya et al., 2011).

Cloud computing is an innovative paradigm and service model for business computing, following the footsteps of parallel computing, distributed computing, and grid computing. A data center is considered a server farm or computer house used for cloud computing. Multiple computer systems are networked together to create a server farm, enabling communication between internal and external systems. Cloud service providers (CSP) manage the computing resources in data centers (DC) and combine them to meet user demand. Additionally, businesses can use cloud computing hardware and software for personal use. The client can use computing resources in a convenient and on-demand way, just like the water and electricity we use daily (Panda & Jana, 2019).

Cloud is defined as a parallel and distributed computing system comprised of interconnected and virtualized computers, dynamically delivered to customers as a unified computing resource in accordance with predetermined service level agreements (SLAs) (Buyya et al., 2009). It serves as a framework that facilitates efficient and adaptable access to a shared collection of computing resources, including networks, servers, storage, applications, and services. This framework allows for rapid provisioning and deprovisioning with minimal management efforts or reliance on service provider involvement (Kaur Sidhu et al., 2013).

This provisioning and un-provisioning process is accomplished swiftly with minimal management effort or dependency on service providers (Kaur Sidhu et al., 2013). Described as a transformative technology, it possesses the capability to revolutionize collaboration, agility, scalability, and availability while offering avenues for cost reduction through streamlined and effective computing (Shahzad, 2014). Cloud computing has the capability to revolutionize the world by offering a technology that enables an on-demand, utility-based approach for allocating and utilizing computer resources. In this transformative paradigm, components can be rapidly provisioned, decommissioned, scaled up, or scaled down, ushering in a new era of computing (LeeJeongkeun et al., 2014).

2.2.1. Cloud Deployment Models

The cloud deployment model illustrates how cloud computing services are used and delivered to customers. Virtualizing the processing power of physical equipment and applications is a common strategy used in all cloud deployment models. The cloud has four deployment models: private model, public model, hybrid model, and Community cloud (Faheem et al., 2017).

A. **Public Cloud**

This is a form of cloud deployment where the service provider maintains control and an agreed-upon SLA is established between the customer and the resource provider. Examples of cloud service providers include Microsoft, Google, Amazon, Vmware, IBM, Sun, and Rackspace. The platform has been created in the style of generalized computing and meets a broad range of client demands. Public access to and easy accessibility to the resources are provided (Bazgan et al., 2022).

B**. Private cloud**

In the private cloud, multiple customers are served by an individual company that manages and maintains the cloud infrastructure. Resources would only be available internally if a company created its own private cloud and recently created its servers using physical hardware servers and a virtualization layer on top of them. Since they don’t need to use Microsoft or Amazon servers, their program can be deployed to their physical control server(Davidovic et al., 2015).

C**. Hybrid Cloud**

A hybrid cloud combines more than one cloud deployment, which can be either public, private, or community clouds but remain separate, independent entities. When a customer

has a large demand for resources, hybrid clouds are important because they typically permit the migration of some computation tasks from private to public clouds. Because it provides more secure control over the apps and data, it is well-organized and enables access to data by various entities via the internet. It has advantages over other deployment techniques and can be hosted both internally and externally(Soni & Malik, 2021).

D. **Community cloud.**

This is a cloud service that offers services to a group of customers or businesses that have similar goals or issues. Organizations adopting this cloud service share the same goals, rules for governance, and security standards. Cloud services may be hosted at one provider, at the area of the client organization, at the area of the peer organization, or a combination of these. This community cloud concept is commonly used in marketing to describe the service's intended customers (Marinos & Briscoe, 2009).

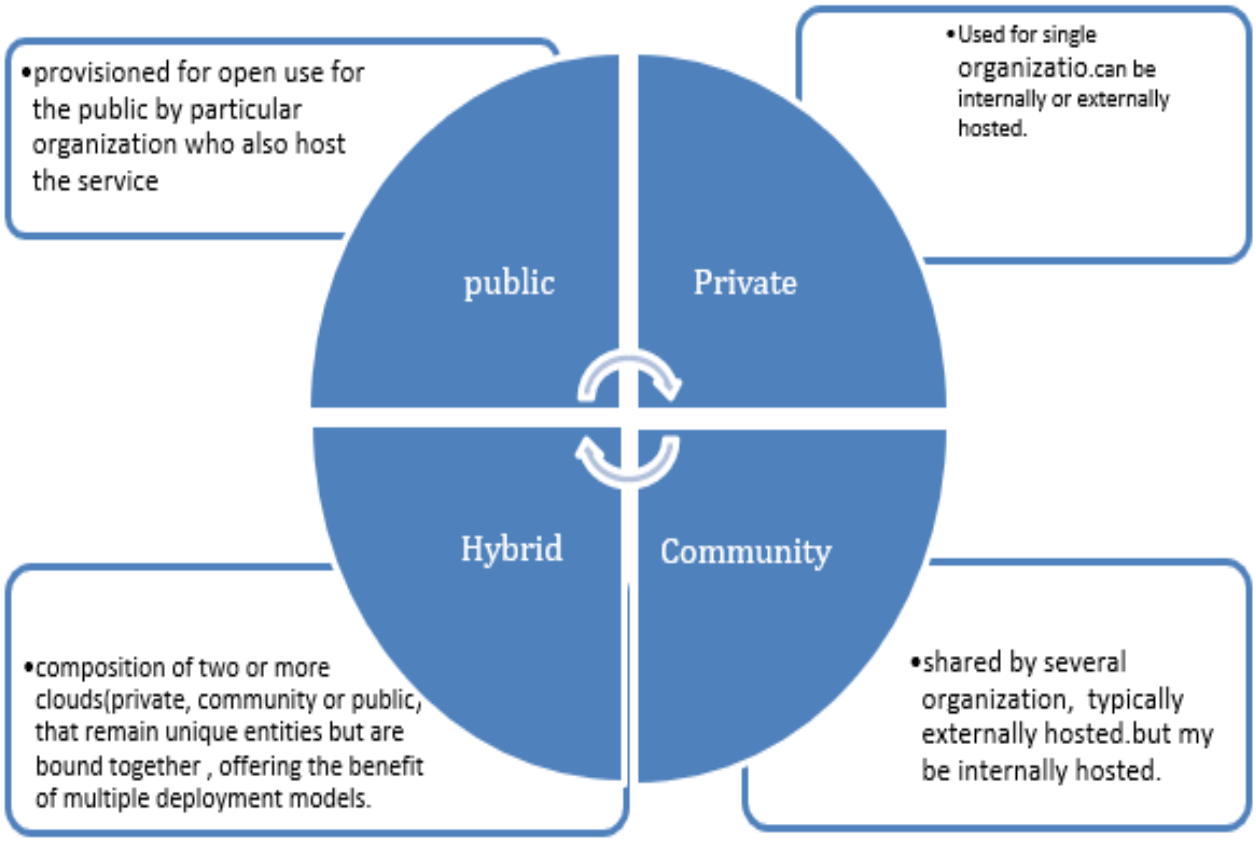

Figure 2.1. Types of Cloud deployment models.

2.2.2. Cloud Service Model

The three primary service models that cloud service providers offer to their clients are, PaaS, IaaS, and SaaS (Q. Zhang et al., 2010). The infrastructure layer controls both the hardware

layer and the IaaS layer. In virtual form, these layers house computational resources including CPU, memory, bandwidth, and disk storage. IaaS is, in other words, a service layer that offers virtualized computing services.The following layer, PaaS, houses resources including software frameworks, databases, and file storage, among others. Microsoft Azure, Google App Engine, Amazon DB, and other platform-oriented services are available(Al-Sayyed et al., 2019).

Thirdly, the SaaS layer hosts and administers many of business applications, online services, and multimedia applications for cloud customers. For instance, business applications and online services such as Google Apps, Facebook, YouTube, etc. are handled at the application layer as software services. Out of all the services offered by cloud computing, managing and supplying IaaS is quite expensive for cloud providers. Data centers are the fundamental resources for cloud service providers in hosting all of the cloud-based services(Al-Sayyed et al., 2019).

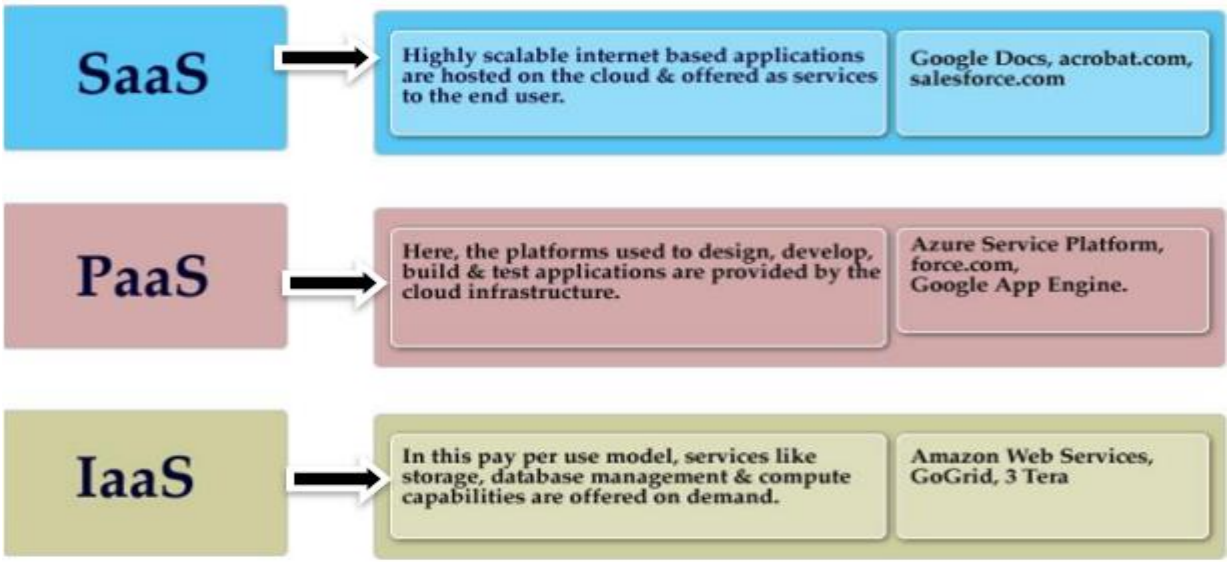

Figure 2.2. Cloud Computing Service model

2.2.3. Essential Characteristics of cloud computing

The NIST states that the following five features are necessary for cloud computing (B. Patel & Kansara, 2021).

The on-demand self-service.

This is defined in terms of users, who can independently give computer capabilities that are required automatically without the supervision or contact of a human from each service provider. The computational power may come from network storage or server time(Butt et al., 2012).

The broad network access.

This is explained by utilizing computational resources that can be accessed through a controlled and standard process and is made available via the internet or network. This mechanism is now in place to encourage the usage of heterogeneous platforms, which can be either extremely thin or very large. Smartphones, tablets, laptops, and workstation PCs are a few examples of platforms(Khalil Aljwari & Aljwari, 2022).

The resource pooling.

In use of a multi-tenant strategy, the provider's computing resources are combined to support a select number of users, with unique host machine and virtual machines are being dynamically assigned and reallocated in response to user demand. They might be able to give a more general description of the place (such a nation, state, or data center). Typically, the client has no control over or information over the precise location of the resources provided. This gives the customer a perception of location independence(Chaturvedi & Rashid, 2017).

The rapid elasticity.

In this circumstance, the elasticity which is provided and released is used to demonstrate computing capabilities. To scale internally and externally, the release may be automatic in particular circumstances. This scaling is also utilized to correspond with client demand. The capabilities that are made available frequently appear to the user as though they are limitless and can be used in both quantity and duration.

The measure service.

In this illustration, a cloud system is managed and optimized automatically using a resource consumed. This measuring feature is used as an abstraction when it is determined to be appropriate for a particular kind of service. As it further increases transparency for the supplier and user concerning the services being utilized, the resource may also be used for much more, such as tracking, controlling, and reporting.

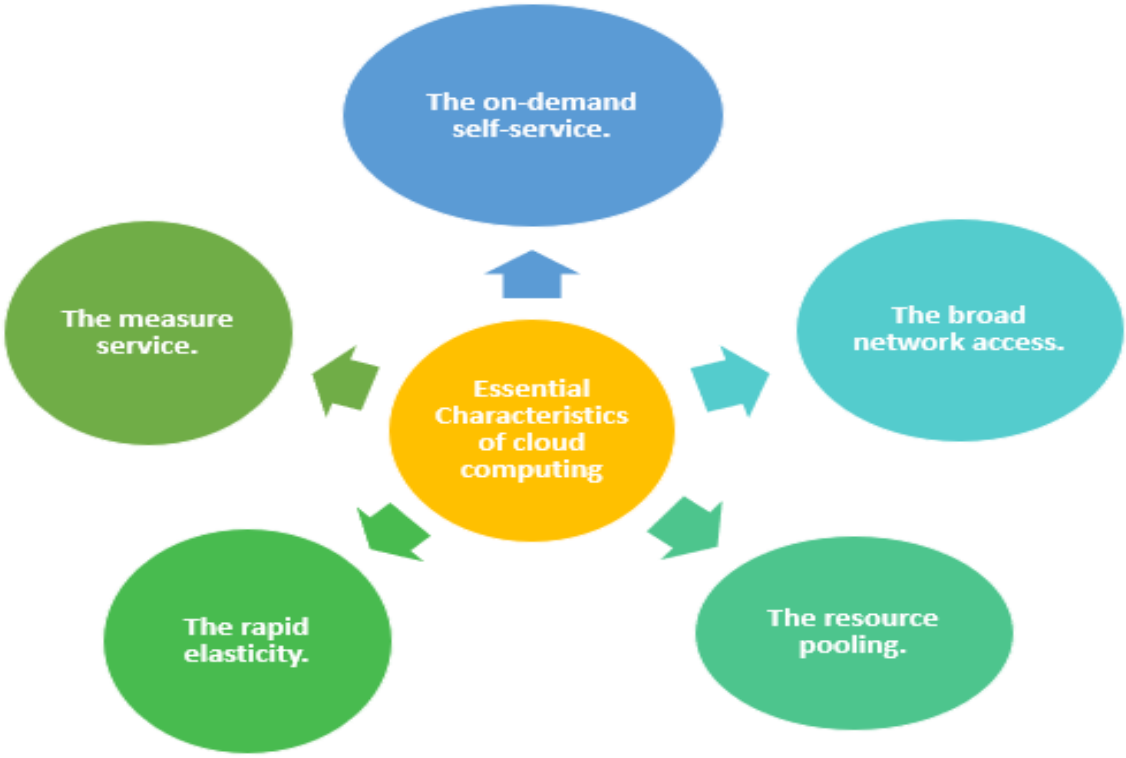

Figure 2.3. The Essential Characteristics of Cloud Computing

2.3. Virtualization

The most important technologies in data centers, virtualization abstracts the flexible use of physical resources as logical resources. Creating a virtual environment of resources to use, the idea of virtualization is the storage of data on virtual storage or the access to data from virtual servers. On a single host, numerous VMs can be found using the virtualization technique. The virtual world is not visible on physical hardware. Less hardware and save money on maintenance; this process is known as virtualization. In order to effectively utilize the expensive hardware available at the time, IBM initially developed virtualization in the late 1960s, and it was mostly used in the desktop industry (Dasgupta et al., 2011).

A large, idle mainframe's hardware was logically divided into slots to enable time-sharing by multiple users. The virtualization approach is still used to distribute a single computer's hardware resources among various distinct computing environments, even though modern computer hardware is less expensive than it once was. As part of IaaS, virtualized computing resources are offered as a service(Jadon, 2018).

Their use is expected to adhere to a Service Level Objective (SLO), which requires the CSP to provide several group metrics, including security metrics and quality of service metrics (performances, availability, and reliability). The IaaS model's main pillar is virtualization

technology. This technology allows a physical computing system to be divided into separate environments known as virtual machines," where each virtual machine can perform computing tasks (Dillon et al., 2010).

Applications can run on virtual machines or containers. A physical machine can handle several virtual machines by effectively sharing hardware resources like CPU, RAM, and I/O ports. Accordingly, based on the inspected and modeled user's request, the amount of host machine resources required to host and finish the requested job are calculated. The VM is appropriately mapped to the host machine during this time.

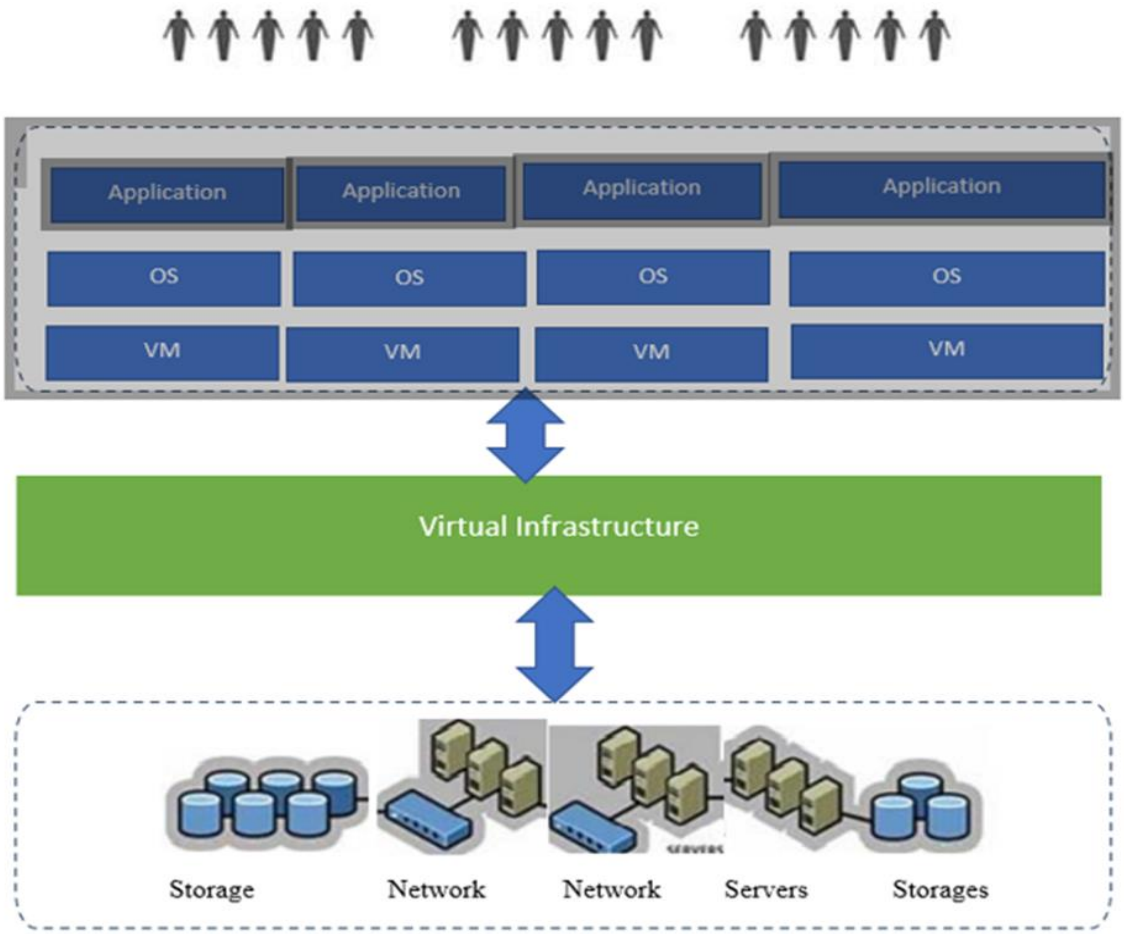

Figure 2.4. Virtualization Mechanism

Advantages of Virtualization is not limited or restricted like cloud computing; it has great benefits over budgetary constraints, lowering the risk factor, and flexible resource allocation. Through scalability and remote access, virtualization improves productivity. They can address the issues of on-demand services and disaster recovery (Postolache et al., 2010).

2.3.1. Types of Virtualizations

There are many types and uses for virtualization, such as hardware virtualization, operating system virtualization, storage virtualization, application virtualization, desktop virtualization, and network virtualization. Below are explanations of each virtualization concept in detail.

Hardware Virtualization

Hardware virtualization is done by deploying the VM on the physical device. Various types of software and OS can be installed through hardware virtualization. However, the Vm manager or hypervisor should take resource allocation into account in addition to carefully tracking and taking care of the VM. Numerous VM can be placed on a physical machine; they are then grouped as a cluster and handled as a single server(Kaur & Chopra, 2020).

Operating System Virtualization

Operating device virtualization is the technique of putting in software program or any software at the running device of a digital device or host device. The virtual machine operating system is mostly used to test newly developed applications on many operating system platforms, including Linux, Windows, and others (Laadan & Nieh, 2010).

Storage Virtualization

Virtualization of storage is nothing but the creation of numerous small virtual storage systems on the physical storage devices, which are then grouped and referred to as single storage. Putting the physical storage from many network-connected storage devices into the host is another way to virtualize the storage. This type of virtualization is used to prevent issues with the disaster recovery principle and to do regular backups(Dong et al., 2019).

Application Virtualization

This is the process of remotely accessing any application from a cloud server. All the personal information that the client or end-user registers will be saved on the program that is deployed on the virtual operating system platform under application virtualization(Swathi et al., 2014).

Desktop Virtualization

In desktop virtualization, a user's Computer will virtually connect to a data center’s server from a distance. The consumer can now obtain data access permissions from any location. The key benefit of virtualization of desktop is that users don't need to think about program upgrades. Anyone who uses desktop virtualization will have access to use a variety of operating systems (Charul Jagtap et al., 2020).

Network Virtualization

It is nothing, but adding high-security virtual switches, and routers overlying private networks over the existing network. Typically, a physical network allows for the creation of numerous separate, smaller networks that the network administrator monitor. One benefit of virtualizing networks is that the network connection will not be lost when one of the networks fails (Baroncelli et al., 2010).

2.4. Virtual machine placement algorithms

The virtualization technique is important in the cloud computing environment because it divides the actual computing resources into numerous isolated execution environments called virtual machines. The placement of a VM is typically thought of as an optimization problem with particular resource restrictions. As VMP is an NP-hard problem, it is expensive to find the optimal solutions for algorithms.

The purpose of VMP is to effectively schedule and distribute virtual machines across real machines in a way that benefits both the user and the service provider (Pires & Benjamán Barán, 2015). Therefore, some popular VMP algorithms are discussed in the paragraphs that follow.

Heuristic Algorithm

A heuristic algorithm is a group of constraints that are designed to solve a problem. Compared to meta-heuristic algorithms, heuristic algorithms are more straightforward to implement.

First Fit Decreasing: The VMs are arranged in descending order according to the demand projected for implementation. Next, after removing each VM from the list, an effort is made to place it on the first PM where it fits, which means that after its placement, the entire sum of all the VMs' resource demands on that PM does not exceed the PM's capacity (Xiao et al., 2015).

First Fit Decreasing algorithm is a greedy heuristic. Every virtual machine is put into "the first bin that will fit it." This strategy is more effective when the list of elements is first sorted in decreasing order (Usmani & Singh, 2016). Both physical and VM are sorted from high-capacity to low-capacity. Once both are arranged, the first fit process takes place (Kumaraswamy & Nair, 2019).

Best Fit Decreasing: The best-fit decreasing algorithm first arranges the VMs in decreasing order of their level of virtual machine use. If a host in the host list has sufficient resources

for the VM, the host is then chosen as a destination; otherwise, nothing is done (Remesh Babu & Samuel, 2014).

Deterministic algorithms

Deterministic techniques seek to look at the problem's diagnostic characteristics to identify a set of points that will lead to the global optimal solution. Deterministic optimization methods for solving linear, nonlinear, and mixed integer linear programming problems have seen many advances. The convexity of the goal function utilized and the contributions of the continuous and discrete variables are the major criteria used to classify deterministic techniques.

Deterministic algorithms for handling optimization problems with several constraints and objectives and approximation algorithms based on mathematics are widely used (Van et al., 2010). The suggested approach also addresses dynamic VM migrations and placement. The primary goals are to decrease energy usage and improve performance. Another constraint-aware VM placement framework is proposed to increase the availability of cloud services. They used a hierarchical process to support four alternative approximation methods that satisfied a range of requirements, including those related to demand, communication, and availability (Jayasinghe et al., 2011).

Hybrid Optimization Techniques

A hybrid optimization algorithm combines more than one algorithm by introducing some changes to connect the algorithms during its execution. Hybridization is introduced in any of these two forms: where one algorithm’s output is given as input to the next algorithm, or where a few parameters of an algorithm are determined using another algorithm. The key objective of hybridization is to enhance the algorithm's overall performance.

Metaheuristic algorithm

Meta-heuristic algorithms are different from heuristic algorithms in that they are primarily intended for general-purpose problems. Meta-heuristic algorithms follow a set of dependable procedures while developing and resolving problems.

Common meta-heuristic algorithms, such as genetic algorithms, ant colony optimization, particle swarm optimization, and honeybee foraging algorithms, are derived from nature. These algorithms are based on population evolutions, gaining the best population in each evolution and maintaining it in subsequent evolutions (M. Xu et al., 2010).

Genetic algorithm (GA)

GA is a branch of stochastic optimization algorithms that adopts Darwin's theory. GA has been used to address issues with modeling and optimization, autonomous programming, and machine learning (Goldberg & Holland, 1979). Six essential GA issues need consideration: the representation of the chromosome (genome), the selection function, mutation operators and crossover operators, the creation of the initial population, termination criteria, and the evaluation function (Ebrahimzadeh & Ghazalian, 2010).

GA searches start with the initialization of the population. Chromosomes make up the population. A solution to a problem can be found on every chromosome. Populations will produce new offspring, and these offspring should be better than the parent population. A healthy natural population will have the chance to keep growing so that it can continue to generate healthy children. Finding a variety of diverse solutions that are not dominated in a single run may be readily done with a standard single-objective GA. In non-convex, discontinuous, and multifunctional problem spaces, GA's capability to simultaneously search many regions of a solution space enables the discovery of a broad range of solutions for challenging problems.

The GA crossover operator may use the structures of successful solutions for various objectives to generate novel non-dominated solutions in undiscovered regions of the Pareto front. As a consequence, GA has been the most commonly utilized heuristic for solving problems involving multi-objective design and optimization (Konak et al., 2006).

Ant Colony algorithm

An innovative metaheuristic method for solving challenging stochastic optimization problems is called ant colony optimization. The concept for ACO came from actual ants, which communicate using pheromones. because of the way they leave and follow pheromone trails. ACO is based on the comparable indirect communication of a colony of simple agents, known as artificial ants, using artificial pheromone trails (Dorigo & Stützle, 2018).

In ACO, the pheromone trails serve as a dispersed, numerical data set that the ants utilize to probabilistically develop solutions to the current problem. They modify the algorithm while it is being run to adjust for their search experience. The ants in make decisions based on input data for the problem at hand, synthesized pheromone trails, and maybe accessible heuristic knowledge. As a result, for many combinatorial optimization problems, ACO might be considered superior to the widely-used building heuristics.

What ants do in an ACO algorithm is explained below. An ant colony can move between phases of the problem concurrently and asynchronously by establishing routes. Heuristic data and pheromone trails are used in a stochastic local decision process that they use to move around. To solve the optimization challenge, ants move and create iterative solutions. The ant examines the solution and leaves pheromone trails on the components or interactions it utilized. It is possible for this to occur both while and after the solution has been created. Knowing more about these pheromones will help future ants hunt more effectively (Tamura et al., 2021).

Virtual machine placement

Virtual machines are used to provide cloud's resources as an on-demand approach. A physical machine can host numerous instances of virtual machines. The virtual machine monitor known as a hypervisor distributes physical computing resources to each virtual machine (Popek & Goldberg, 1974). The term " A lightweight layer of software known as a "hypervisor" runs on top of real hardware to manage virtual machines. Cloud computing relies on the concept of VMP to configure and manage virtualized applications. The way of assigning or mapping a VM to a physical host is known as VMP. It refers to the technique of distributing a list of VM over several physical machines to fully utilize the hosting physical machine. Cloud service providers use virtual machine deployment techniques to increase the number of VM deployed within each physical machine and increase the number of VM that a data DC can deploy (Gupta & Pateriya, 2014).

VMP is a challenging computational problem that belongs to the class of NP-hard combinatorial problem-solving (J. Xu & Fortes, 2011). Server energy is wasted when they are not in use, and the consumption of idle servers is what causes data centers to be inefficient. According to a previous computation, the annual usage of DC power in the United States just would rise to 140 billion kWh by 2020, costing US companies 13 billion USD in electricity bills and producing around 100 million tons of CO2 annually (Mishra & Sahoo, 2011).

In order to choose the best physical machine (PM) to host the VM, VMP is a vital method in cloud computing. It directly impacts the efficiency, resource use, and usage power of the cloud data centers and can lower the cost of maintenance for cloud providers (Masdari et al., 2016). As a result, one of the most prevalent techniques for using the physical device in a

data center is to place VMs while taking bandwidth and energy usage into consideration. This method is regarded as the cornerstone of cloud computing.

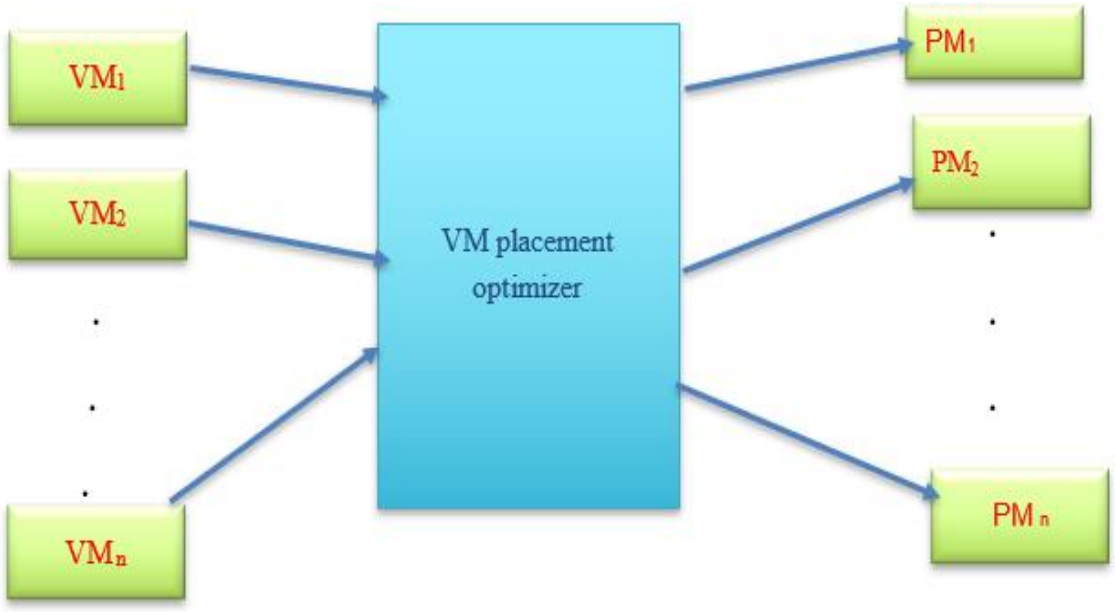

Figure 2.5. virtual machine placement schema.

(Alharbi et al., 2019). presented the offloading of VM services from the cloud to the fog, considering the British Telecom (BT) network topology has been investigated. They presented a framework for energy-efficient VMP over a cloud-fog architecture. They developed a MILP model taking into account several tiers of cloud and fog resources to identify the best placement of various VM types. Accordingly, the BT core network topology consists of 20 nodes and 68 bidirectional links. Each core node in the design under evaluation is linked to two XGPON access networks via a metro network made up of one Ethernet switch and two edge routers. According to (Liu et al., 2020) study, a resolution-aware and energy-efficient multi-objective virtual cluster allocation approach is proposed. Formally, the availability model, energy consumption model, average resource utilization model, and load balance model are described.

In order to identify a trade-off between those goals, an enhanced population-based multi-objective optimization technique is presented to address the multi-objective virtual cluster allocation issue as a constraint. The biogeography-based optimization (BBO) algorithm is then proposed and leveraged in both theoretical demonstration and experimental evaluation in a different setting. However, this study did not take bandwidth resources into account as an objective.

(Feizollahibarough & Ashtiani, 2020) have proposed a new approach for allocating virtual machines to physical machines while considering the security risk of co-allocation. They assumed four different attributes, such as the vulnerability of the VM, its importance level, the commutative vulnerability level of the physical machine, and its capacity. Due to the uncertainty of all these factors in real-world scenarios, this uncertainty should be reflected in the decision-making process. This goal is accomplished by applying a fuzzy set to these four qualities in order to make security-aware allocation decisions. Another worthy point of this paper is that it is possible to extend these attributes or even add more attributes to the schema. Nevertheless, it is assumed that the cloud provider can assign the attribute weight according to its needs and requirements. So, such an assignment may not be optimal.

((Ammar et al., 2019)). A balanced resource utilization algorithm called BRC-IBMMT has been proposed. The algorithm effectively balances resource consumption, and finds an acceptable balance between the contrary correlation objectives of power consumption and SLA violation. Two algorithms for VMP and selection, called Balanced Resource Consumption (BRC) and Imbalance VM with Minimum Migration Time (IBMMT), are proposed. Both algorithms consider CPU, RAM, and bandwidth (BW). The VM deployment algorithm (BRC) leverages the efficient of resource utilization and minimizes resource fragmentation, while the VM selection policy (IBMMT) enhances the balance of resource consumption and reduces the migration cost.

(Jiang et al., 2012) formulated an offline algorithm to address a static issue in the context of a network snapshot and an online response for a dynamic setting with fluctuating traffic. However, they neglected to account for bandwidth and energy use when positioning the virtual machines. Combinational Ordering First-Fit Genetic Algorithm (COFFGA) and Combinational Ordering Next-Fit Genetic Algorithm (CONFGA), two already-existing GAs, were coupled to form a hybrid model by (Hallawi et al., 2017).

The results show that this model performs better than the earlier GAs. (Yi & Singh, 2014) have minimized energy consumption for fat-tree data center networks. The direct method for the advancement of green data centers is to narrow their energy consumption. The fundamental idea is to use power management strategies in data center information and communication technology (ICT) equipment in case they did not consider bandwidth utilization during the investigation. Cloud-based user-side integrated energy management system is recommended by (Tian et al., 2018) to improve energy efficiency in a smart

community. They provided a two-level design that makes use of a load aggregator and an energy hub. Additionally, a mechanism for information masking is created based on linear mapping functions to protect the privacy of the cloud service. A proactive strategy that adapts to demand changes and moves VM between physical hosts has been studied for managing SLA violations and offers probabilistic SLA guarantees according to Bubnoff et al. (2007).

(Sharma & Sharma, 2012) state that a replacement virtual machine load-balancing algorithm has been proposed following a review of several virtual machine load-balancing methods. After carefully examining numerous VM load-balancing algorithms, a novel virtual machine load-balancing technique has been developed and put into practice in the virtual machine environment of cloud computing to reduce response time and cost.

(Mueen Uddin, 2012)created a tool to boost data centers' functionality and energy effectiveness. Depending on various parameters, they divided up the components of the data center into different resource pools. The framework emphasizes the importance of incorporating green practices to maximize data center efficiency, including power usage effectiveness (PUE), carbon emission calculators, and data center effectiveness.

(Ajith Singh & Hemalatha, 2014) suggested BASIP as a VMP method that addresses deadlock problem by combining a banker algorithm and stochastic integer programming. In addition, the suggested approach is replicated using thousands of VM running on hundreds of servers. To lessen overall inter-cluster traffic, (Gao & Rouskas, 2018) examine a network-aware variation of the problem, where a virtual request is divided into smaller pieces. To deal with the difficulty's inherent complexity, they developed a partitioning technique based on spectral clustering.

(Yang et al., 2017) have developed a joint optimization issue for data centers that takes both communication demand and energy usage into account. This research for data center introduces Job-Aware VMP, and Route Scheduling, a combined optimization method that is energy-efficient. Contrary to earlier tactics. Energy-Efficient Virtual Machine Placement for Effective Resource Utilization Using Modified Binary.

The VMP techniques suggested for cloud computing data centers are thoroughly surveyed and analyzed in (Masdari et al., 2016). To do this, the description and goals of virtual machine placement are first given, and a list of virtual machine placement schemes is then

categorized according to the kind of placement algorithm. The advantages, characteristics, and constraints of each recommended virtual machine placement scheme are then clarified, followed by a description of the numerous suggested virtual machine placement schemes and an examination of the virtual machine placement elements in each scheme.

(Takouna et al., 2013) discused the peer virtual machine aggregation algorithm for communication-aware placement of parallel applications. Their approach aggregates communicative virtual machines into a server using migration. This assists in localizing the traffic and minimizing the communication delay among communicative virtual machines. The approach requires virtual machines to be involved with the Migration Manager (MM) to determine the communication patterns among virtual machines. Accordingly, they compare their approach with the CPU approach, which shows a random placement of virtual machines due to a lack of awareness about communication among virtual machines.

Han et al. (2018) investigated the detailed security assessment of VMP strategies and provided a method to comprehensively quantify cloud security risks based on the vulnerabilities caused by a variety of factors, including the network, the physical machines, the virtual machines, and the co-residence of virtual machines. With the available resources and other constraints, we have developed a novel method to generate virtual machine placement that is based on multiple-goal optimization. Their suggested approach attempts Pareto-optimal placement while taking various optimization goals and restrictions into account. However, their study and experiment did not evaluate the real-world environment.

According to (Karmakar et al., 2022) paper, a multi-objective optimization technique to reduce both the communication cost, and the resource wastage of the active hosts has been presented. In this paper, an ant colony system-based metaheuristic technique for determining near-optimal solutions in real time is proposed. The simulations are performed on both the VL2 and Fat-Tree network topologies.

(Su et al., 2016) developed a multi-objective optimization technique for VMP based on a discrete firefly algorithm. They considered the resource dynamics and power consumption of the data center as objective functions and ensured QoS. According to this study, the total resource wastage and total incremental power drawn by the server pool are simultaneously minimized without compromising the performance objectives. However, this study did not consider network resource consumption.

As per (Son & Buyya, 2019) the host group with higher processing, and resources of networking is where PAVA seeks to install virtual machines of high-priority applications. The first fit decreasing (FFD) technique is employed by the authors for a low-priority application. Additionally, they suggest using a bandwidth allocation (BWA) mechanism to assign the necessary network bandwidth to a high-priority application in a cloud network controlled by SDN. Accordingly, the virtual machine allocation method places the critical application on the closer hosts so that it reduces both the energy consumption and the average response time for the critical application.

They give high consideration to fine-grained values rather than binary values. So, the application of different priorities was not considered. A task-based virtual machine placement method (ETVMC) has apparently been provided by incorporating heterogeneous tasks, virtual machines, and hosts in the cloud system, according to (S. K. Mishra et al,.2018). In order to reduce energy usage and task rejection rates, it is important to effectively assign tasks to virtual machines (VMs), then to hosts. FCFS, Round-Robin, and EERACC algorithms are used to compare its solution performance. However, they did not take into account the data center topology's bandwidth resource.

According to (Fatima et al., 2019), a novel meta-heuristic approach based on the GWO algorithm was created to address the problems with virtual machine placement in cloud data centers. As more users request services, there are an increasing number of cloud data centers. Cloud data centers house physical machines that host virtual machines. The millions of virtual machines reduce the number of physical machines. Given this expansion, managing the cloud data center requires a clever strategy. The biggest issue with cloud computing is how to deploy virtual machines. The multi-objective strategy for this study has been presented as the gray wolf optimization algorithm. The suggested approach has been developed to be more successful while using less energy.

2.5. Data center network topology

Cloud service providers (CSPs) manage massive infrastructure in their data centers, which are dispersed throughout the world, to meet the growing demand for services. Service providers typically construct their network using a two- or three-tier architecture to control hundreds of servers in each data center, based on the dimensions of the data centers (Al-Fares et al., 2008). Despite the fact that a small data center may be handled by a two-tier

design with Edge and Core switches, large-scale data centers are structured as three tiers with Edge, Aggregation, and Core switches. The physical servers, on which virtual machines act as the users' processing units, are connected by the edge switches.

Cloud service providers adopt hierarchical topologies like Fat-Tree (Petrini & Vanneschi, 1997) for designing their networks due to the drawbacks associated with the conventional three-tier design. The traditional three-tier design, which relies on a hierarchical tree structure, suffers from various issues such as a single point of failure, limited scalability, and inadequate bandwidth availability. Hence, cloud service providers opt for hierarchical topologies like Fat-Tree to overcome these limitations. The fat tree is a comprehensive binary tree-based indirect connectivity network, and unlike a traditional tree, fat trees resemble genuine trees due to their root-like thickening.

VL2 (virtual layer 2) is a flexible three-tier hierarchical model that scales up by increasing the number of ports of switches (node degrees) at different levels. The highest level, the core layer, contains intermediate switches that are connected to D aggregation switches. Top of Rack (ToR) switches in the lowest level, the access layer, interconnect their underlying racks (Greenberg et al., 2011). In the B-Cube interconnection structure, each server is equipped with a few number of ports (typically no more than four). The B-Cube network architecture takes a server-centric approach rather than a switch-oriented one. It utilizes commodity switches and places intelligence on MDC servers (Guo et al., 2009).

2.6. Related Works

To explore the gap, an extensive literature review based on the previous domain of studies like books, journal articles, unpublished materials, research, reports, and other documents from the internet is explored. Following the collection of all relevant materials, a comparison, and contrast of studies on the subject is undertaken. So, in this related work, algorithms, methods, and different procedures have been investigated based on recent studies and journals.

(Balaji et al., 2022). formulate the virtual machine placement problem as a minimization objective to reduce the power consumption in a data center. They have proposed a modified discrete firefly algorithm (MDF-PC) to reduce the power consumption of the data center by efficiently allocating the virtual machine to the appropriate physical machine such that the waste is very minimal. The fundamental aim of the proposed work is to schedule the virtual

machine as densely as possible on a minimal number of servers using the proposed modified discrete firefly algorithm for power consumption. An objective function is modeled to estimate the power consumption of the data center. The proposed discrete firefly algorithm is compared with the native genetic algorithm (GA) and particle swarm optimization (PSO) algorithms in terms of minimizing energy consumption, showing that the proposed algorithm works better. However, bandwidth resources were not considered.

(Li & Cao, 2022) proposed a virtual machine consolidation algorithm based on the dynamic load mean and multi-objective optimization in cloud computing, in which the host load status is comprehensively measured based on multidimensional resources and the dynamic characteristics of the system load are considered. The optimized ant colony algorithm is then used to obtain the optimal mapping between the virtual machines and the hosts to achieve multiple objectives such as resource utilization, energy consumption, migration, and communication overhead optimization. They compared the dynamic load mean virtual machine consolidation (DLMMVMC) with the Ant Colony system virtual machine consolidation (ACS-VMC) in terms of convergence speed.

Finally, the outcomes of the experiment demonstrate that the DLMM-VMC is successful in reducing energy consumption, resource utilization, and migration overhead when compared with the ACS-VMC. This paper ignores the energy consumption generated by other devices in the data center and its impact on the system's performance. As well as bandwidth resources usage due to traffic demand between virtual machines, is not properly studied.

(W. Zhang et al., 2021). developed an initial virtual machine that was placed fault-tolerantly using a multifaceted optimization strategy. To manage the virtual machine fault-tolerant placement process for virtual machine static placement in star topological data centers, a virtual machine fault-tolerant management system is established in this work at the unified resource layer. The multi-objective optimization model of initial virtual machine fault-tolerant placement, which is based on several criteria, is suggested to be solved using a heuristic ant colony algorithm. The service-providing VMs are placed by the ant colony algorithms, and the redundant VMs are placed by the conventional heuristic algorithms. According to experimental results from simulation, the proposed approach produces a better VM fault-tolerant placement solution than the conventional first fit (FF) or best fit descending (BFD) approach. But multiple VM failures and bandwidth resources were not taken into account.

According to (Regaieg et al., 2021). a multi-objective linear integer programming approach has been presented that aims to simultaneously optimize the number of virtual machines hosted, resource usage, and active servers to lower power consumption. Weighted-sum approach which is traditional multi-objective optimization is used, and implemented in the Julia package MultiJuMP, to obtain non-dominated solutions to the multi-objective problem. They claimed that by setting these new objectives, it would be possible to increase client satisfaction while cutting down on operational costs for data centers. Lastly, they note that the heterogeneous DCs perform better in terms of the number of hosted VMs, the amount of resource wastage, the number of used PMs, and the amount of power consumption. However, they failed to provide a good compromise between the conflicting objects and failed to consider the data centers' bandwidth resource usage.

(Xing et al., 2022). The proposed ACO algorithm aims to minimize energy and networking components during VM placement using a weighted sum approach, which is suitable for a single optimization approach. The approach aggregates two objectives into one objective.

A mathematical model to calculate the power consumption of networking devices based on inter-VM data transfer was proposed. However, this algorithm suffers from issues such as unstable convergence speed, the significant impact of parameter settings on results, and unstable results due to randomness. This study did not properly consider the tradeoff between bandwidth resources and energy consumption, where one objective is not dominating another but rather getting a non-dominated optimal solution. They used an approach, which is a valuable method for single-objective optimization by considering the relative importance of objectives, and they overlooked important trade-offs between the objectives and failed to capture the true nature of the problem.

(Saxena & Singh, 2021). Suggested an energy-efficient virtual machine allocation framework to address the energy consumption, resource wastage, and QoS degradation of cloud data centers. An online multi-resource feed-forward neural network (OM-FNN) was used to predict the different resource demands. Based on the prediction result, virtual machine allocation was done on energy-efficient PMs. A novel OM-FNN predictor is developed to forecast the utilization of multiple resources simultaneously with lesser time and space complexity as compared to multiple conventional neural networks. Future applications are grouped into clusters based on their predicted resource requirements, and the number and type of virtual machines suitable for a cluster are scaled automatically. The

selected auto-scaled virtual machines are placed on energy-efficient servers by applying the proposed multi-objective virtual machine placement algorithm. The result shows that the Online Multi-Resource Feed-forward Neural Network (OM-FNN) predictor shows better accuracy and lesser time and space complexity over a traditional single-input and single-output feed-forward neural network (SISO-FNN) predictor. However, bandwidth resources of VMs according to their inter-dependency while considering resource management are not considered.

(Abohamama & Hamouda, 2020). study, the energy consumption of the data centers is minimized by minimizing the number of physical servers by using the hybrid virtual machine placement algorithm. An improved permutation-based GA (IGA-POP) and resource-aware best-fit strategy virtual machine placement algorithm are proposed. The number of active servers is reduced through efficient, balanced usage of the multidimensional resources of active servers. The experimental outcomes demonstrated that the suggested VMP algorithm is superior to the best fit (BF), genetic algorithm (GA), sine-cosine algorithm (SCA), and best fit (BF). However, bandwidth resources in the data center were not considered.

(Arivuselvi et al., 2021).have formulated the problem of virtual machine positioning as just an NP-hard problem to obtain effective use of total memory resources and the full use of treatment resources. This problem formulation was addressed by the ACO algorithm. The proposed algorithm is built to efficiently deal with ample solution space and obtain a collection of non-dominated solutions. The main focus of this study is to put forward a virtual machine migration technique in the cloud environment that improves CPU, memory utilization and traffic minimization in the cloud data center using the ACO algorithm. However, in this study, the server's energy consumption was not taken into account properly.

The following Table 2.1. shows a summary of the related works, including their strengths, weaknesses, and ideas

Table 2.1. Summary of Related Work Focuses and Variations of Current Work

Author with published year	**Main ideas**	**Metrics**	**Finding**	**Limitation**
(Balaji et al., 2022)	Power-aware virtual machine placement in IaaS cloud using discrete firefly algorithm	Reduces power consumption of the server	The proposed discrete firefly algorithm is compared with the native genetic algorithm (GA) and particle swarm optimization (PSO) algorithm show the proposed algorithm works better.	Bandwidth resource consumption were not taken into account.
(Li & Cao, 2022)	A Virtual Machine Consolidation Algorithm Based on Dynamic Load Mean and Multi-Objective Optimization in Cloud Computing	Resource utilization, energy consumption, and migration time are considered.	Experimental outcomes demonstrate that the DLMM-VMC is successful in reducing energy consumption, improving resource utilization, and reducing migration overhead when compared with ACS-VMC	They failed to consider bandwidth resource

(W. Zhang et al., 2021)	A Multi-Objective Optimization Method of Initial Virtual Machine Fault-Tolerant Placement for Star Topological Data Centers of Cloud Systems	single VM failure minimization.	The heuristic ACO produces a better VM fault-tolerant placement solution than the conventional FF and BFD approach	multiple VM failures, Energy consumption and bandwidth resources consumption were not considered.
(Regaieg et al., 2021)	Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers	Maximize hosted VM, minimize resource wastage, and active PM	Finally, they note that the heterogeneous DCs perform better than homogeneous DCs	The inter-VM bandwidth resource is not considered. weighted-sum method which is Traditional multi-objective optimization is used
(Xing et al., 2022).	ant colony optimization algorithm aims to minimize energy and networking components during VM placement.	Energy consumption of network equipment's and bandwidth resource	ACO achieves better performance than MBF, PEBFD, FF, FFD, MBFD, and PABFD algorithm and obtain better performance when compared with them.	trade-off between the two objectives were not considered. Weighted sum approach is used for single optimization which traditional approach. experiment is done by synthetic data which is not real data trace.

(Saxena & Singh, 2021b)	Proactive autoscaling and energy-efficient VM allocation framework using an online multi-resource neural network for cloud data center.	Energy consumption, resource wastage, and QoS degradation.	The result shows that the OM-FNN predictor shows better accuracy and lesser time and space complexity over a traditional SISO-FNN predictor.	They didn't consider Bandwidth resources of VMs according to their inter-dependency while considering resource management decisions.
(Abohamama & Hamouda, 2020)	A Hybrid Energy-Aware Virtual Machine Placement Algorithm for Cloud Environments.	Energy Consumption of PM.	Experimental outcomes demonstrated that the IGA-POP algorithm is superior to BF, GA and SCA.	bandwidth resource is not considered.
(Arivuselvi et al., 2021)	An Effective Ant Colony Optimization Methodology for Virtual Machine Placement in Cloud Data Centre	CPU, Memory utilization, and traffic.	The main focus of this study is to put forward a virtual machine migration technique in the cloud environment, which improves the CPU, Memory utilization and traffic in the cloud data center using the ACO algorithm.	Physical machines Energy consumption was not studied.

As summarized and shown in Table 2.1 above, it highlights a significant gap in the existing research on VM placement. None of the approaches considered in the studies effectively addressed the simultaneous optimization and trade-off between bandwidth resource and energy consumption objectives. Instead, most studies focused on addressing one objective while ignoring the other.

Furthermore, these studies employed different optimization techniques, including single optimization, bi-objective optimization, and multi-objective optimization, targeting different objectives independently. The weighted sum approach was commonly used in an attempt to achieve simultaneous optimization. However, it fails to provide a comprehensive understanding of the trade-offs between bandwidth and energy consumption. By assigning weights to the objectives and combining them into a single value. This approach overlooks the true nature of the problem and the relative importance of each objective. Consequently, it may yield biased or suboptimal results that do not accurately capture the desired trade-offs of the both objectives.

Recognizing this limitation, we have proposed an alternative approach for efficient VM placement in cloud computing. Our approach focuses on minimizing both metrics, bandwidth resource and energy consumption, simultaneously. To achieve this, we employed bi-objective optimizations, aiming to optimize both objectives simultaneously and obtain a balanced and compromised solution. This approach recognizes the interdependence and trade-offs between bandwidth resources and energy consumption.

In contrast to the weighted sum approach, our proposed approach leverages the Pareto-set optimal approach. The Pareto set optimal approach is better suited for finding non-dominated solutions that represent the optimal trade-offs between bandwidth resources and energy consumption in VM placement. It generates a set of solutions, known as the Pareto set, where no solution is superior to others in all objectives. This allows decision-makers to select the most suitable solution based on their preferences, considering the trade-offs between both objectives.

CHAPTER THREE

PROPOSED METHODOLOGY

In this chapter, we discussed the proposed methodology for bi-objective optimization of bandwidth and energy consumption for efficient virtual machine placement in cloud computing, which includes research design, data collection, simulation tools, and experimental environment setup.

3.1. Research Design

Research is referred to as an approach to advancing knowledge. To conduct this study, we went through the knowledge of prior research. To gather knowledge of previously researched concepts and methods for efficient virtual machine placement in cloud computing, book reviews, journal articles, research papers, subject reports, and other online publications were examined. Therefore, we used the methods and strategies listed below to fulfill the study's objectives and respond to the research questions.

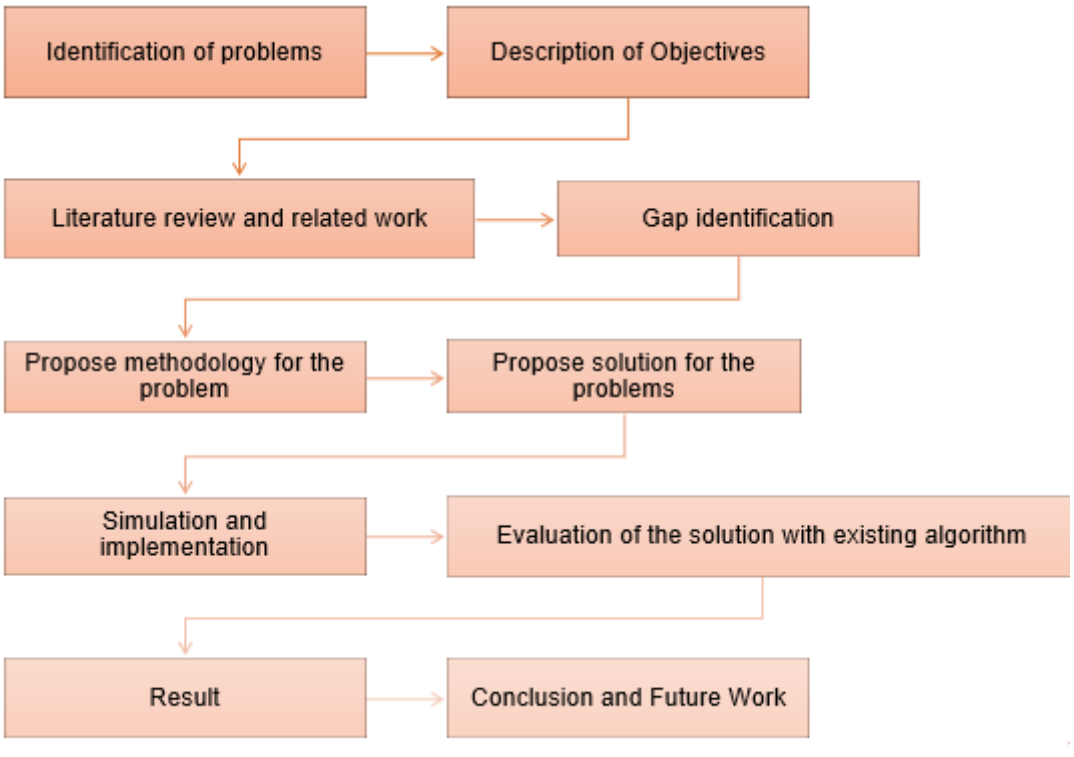

Figure 3.1. Research process

3.2. Data Collections

We mainly used secondary data sources for this study, such as articles, journals, research papers, dissertations, web resources, and other topic-related documentation and guidelines. To ensure the realism of the results, it is crucial to carry out experiments using actual workload traces obtained from a real system. Therefore, to evaluate the proposed algorithm on actual workload data and to be able to confirm its genuine outcomes, we used the workload available at Delft University: The Grid Workloads Archive (GWA), Bitbrains workload[1]. It is conducted with a set of different numbers of VMs from three random dates that were selected from the obtained traffic traces. The workload consists of one file per VM showing the VM's dynamic workload, which is trialed every 5 minutes. CloudSim analyzed this workload file to produce several cloudlets. These data consisted of the VM's CPU and main memory used for a specific time (Iosup et al., 2008).

Bitbrains is a cloud service provider that specializes in managed hosting and business computation for enterprises(Feleke, 2018). The dataset collected contains resource utilization by 1,750 VMs from a distributed data-center and is available online from the Grid Workloads Archive. It is organized into two folders: fastStorage traces consists of 1,250 VMs and Rnd traces consists of 500 VMs. In this work, we used the fastStorage traces. The dataset is organized as one file per a VM, each file containing 30 days of data sampled every 5 minutes.

3.3. Simulation Tool

The main aim of the simulation tool is to test the implementation work in the absence of the required environment (Shaw & Singh, 2015). As it is highly difficult and expensive to undertake repeatability experiments in the real-time cloud environment that provides the infrastructure of the large-scale virtualized data center, Because of its broad features to support the evaluation of bi-objective optimization of bandwidth resources and energy consumption for efficient virtual machine placement in cloud computing, Cloudsim 3.03 toolkit was chosen for modeling the suggested algorithm for this study.

CloudSim: CloudSim is a framework developed by the GRIDS laboratory of the University of Melbourne that enables seamless modeling, simulation, and experimentation with designing cloud computing infrastructures (Masoumzadeh & Hlavacs, 2013).

[1] http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

CloudSim is a self-contained platform that can be used to model data centers, service brokers, and the scheduling and allocation policies of a large-scale cloud platform. It provides a virtualization engine with extensive features for modeling the creation and lifecycle management of virtual engines in a data center. The CloudSim framework is built on top of the GridSim framework, also developed by the GRIDS laboratory (Rekha Yashwantrao, 2017).

It offers (i) support for modeling and simulating large-scale cloud computing systems, (ii) a platform for simulating clouds, service brokers, and resource allocation strategies, and (iii) support for simulating network connections between the simulated system. As depicted in Figure 3.2, CloudSim consists Java classes for simulating the various cloud components, including classes for the data center, host, and virtual machine.

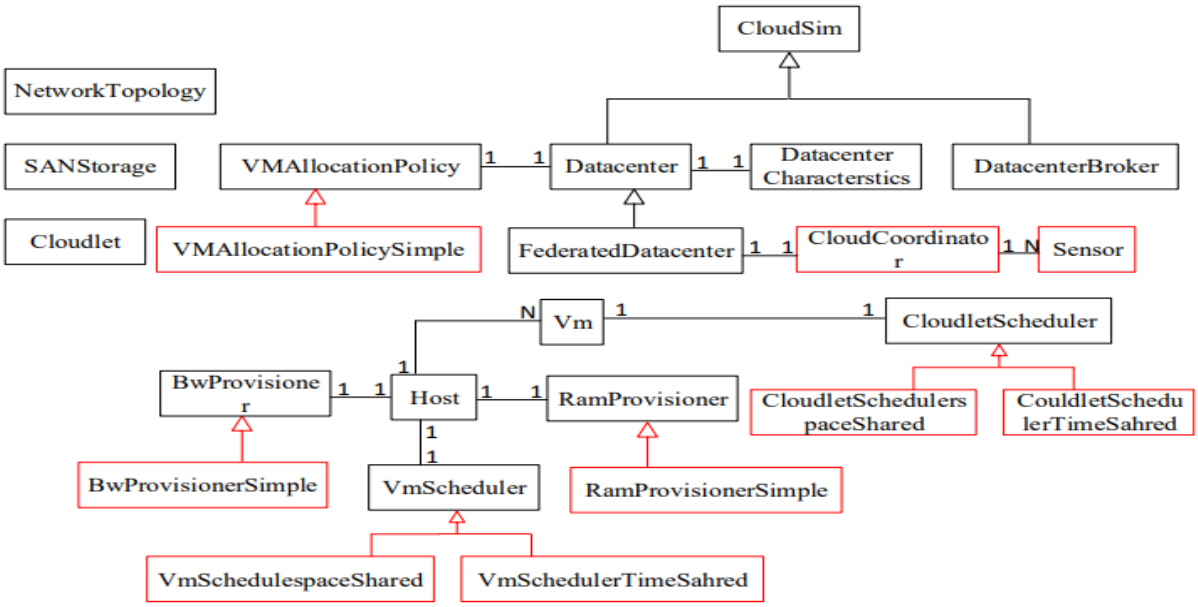

Figure 3.2. CloudSim Class Design Diagram (Masoumzadeh & Hlavacs, 2013).

CloudSim Classes Description

CloudSim is the primary class in responsible for regulating the sequential execution of simulation events and handling event queues.

Datacenter has a number of compute hosts and employs a number of policies that allocate storage, processing, and memory among hosts and VMs.

Datacenter Broker This class models broker that negotiate on behalf of cloud and SaaS providers.

Datacenter Characteristics contains configuration information of data-center resources.

Host class models a real resource like a server for computing or storage. It contains crucial details like the amount of memory and storage, a list and description of the processing cores (to simulate a multi-core system), and a strategy for allocating the processing power among the VM.

Vm This class models the behavior of a virtual machine (VM), which is controlled and hosted by a cloud host component.

Bw Provisioner, RAM Provisioner and VmScheduler These three classes offer elements for setting policies of bandwidth control, memory management, and virtual machine management for a server. They are all abstract classes.

Vm Allocation Policy This abstract class represents a provisioning method that a VM Monitor use to assign VMs to hosts.

VmScheduler This is an abstract class that simulates the rules (space-shared, time-shared) necessary for assigning processor cores to virtual machines. It is implemented by a Host component.

Cloudlet This class serves as a representation of cloud-based application services.

Cloudlet Scheduler The application of various policies that control how much processing power is distributed across Cloudlets in a VM extends this abstract class.

NetBeans is the most effective tool that has been selected for developing and carrying out the proposed solution. It is more than just an editor. It is what is known as an Integrated Development Environment (IDE, as it is generally called), which is a group of applications that come packaged as a single entity and enable programmers to develop application software. Java's NetBeans IDE is an open-source and integrated development environment that contains the Java Development Kit (JDK)(Bai, 2011).

Additionally, we used the visual paradigm (version 16.0). A software program called a "visual paradigm" is made for software development teams to manage development processes and model information technology systems. Unified Modeling Language, BPMN, and XMI are just a few of the important modeling languages and standards that Visual Paradigm supports. The system architecture and flowchart for the suggested method are constructed in this work using the Visual Paradigm (Visual Paradigm Community Circle, 2020).

Hardware Requirements.

The hardware requirements for conducting this research are listed below.

Processor: Intel(R) Core (TM) i3-3120M CPU @ 2.50GHz.

- RAM: Installed memory (RAM) 8.00GB.
- System Type: x64-based processor.
- Hard Disk: 1 TB.

3.4. Setup of the simulation environment

We considered and configured two heterogeneous types of physical machines, namely Type-1 physical machines with 40000 MIPS CPU capacity, 32 GB of memory, and 1 Gbit of bandwidth, and Type-2 physical machines with 24000 MIPS CPU capacity, 16 GB of memory, and 1 Gbit of bandwidth, respectively.

To properly evaluate the performance of the proposed BOHGOA, we used 10 instances for tests. The maximum bandwidth consumption of the link (B_{ei}^{max}) is set to 100 MB/s in the experiment. Also, we considered the Linux x86 operating systems and Xen VMM as the system architecture of our data center.

Table 3.2. Configuration Setup of CloudSim Tools and test problems.

Simulator		CloudSim
Version		3.0.3
Datacenter Characteristics	PM-type-1	MIPS: 40000, Storage:1TB, RAM: 32GB, BW: 1 Gbit. Power busy pj is set to 550 W OS: Linux. System Architecture: x86. VMM: Xen
	PM-type-2	MIPS: 24000, Storage:1TB, RAM: 16GB, BW: 1 Gbit. Power busy pj is set to 113 W. OS: Linux.System Architecture: x86. VMM: KVM
Test problems	Number of VM	40 – 500
	Number of PM	20-80

In This experimental simulation, a cloud data center with two types of hosts is used. Thus, the physical machines' resource characteristics and energy properties are obtained from the published results of a SPECPower based on real data and industry-standard benchmarks.

The Standard Performance Evaluation Corporation (SPEC) was founded as a non-profit organization with the goal of creating, maintaining, and endorsing standardized benchmarks and tools to assess the performance and energy efficiency of the most recent generation of computer systems. The benchmark suites that SPEC develops are reviewed and published together with the findings that their member organizations and other benchmark licensees submit. Power and performance standards are required as a result of the rising need for energy-efficient IT equipment. In response, the SPEC community started the SPECpower Committee, a project to add a power/energy measurement to current or future industry-standard benchmarks and tools.

The following Table 3.3, is a tabular representation of the energy consumption of two different types of experimental hosts (type-1 and type-2) at various utilization levels (ranging from 0% to 100%). The energy consumption is measured in watts.[2]

Table 3.3. Energy consumption of experimental host (watt).

Host-Type	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Type-1	79.1	82.49	85.88	89.27	92.66	96.05	99.44	102.83	106.22	109.61	113
Type-2	385	401.5	418	434.5	451	467.5	484	500.5	517	533.5	550

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package org.cloudbus.cloudsim.power.models;

/**
 *
 * @author Ketema Deresa
 */
public class PowerModelSpecPowerHost_Type_1 extends PowerModelSpecPower {

        /** The power. */
        private final double[] power = {79.1, 82.49, 85.88, 89.27, 92.66, 96.05, 99.44, 102.83, 106.22, 109.61, 113};

        /*420, 390, 400, 407, 414, 424, 431, 438, 448, 458, 468, 550
         * (non-Javadoc)
         * @see org.cloudbus.cloudsim.power.models.PowerModelSpecPower#getPowerData(int)
         */
        @Override
        protected double getPowerData(int index) {
                return power[index];
        }

}
```

Figure 3.3. Host type-1 configuration

[2] https://www.spec.org/power_ssj2008/

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package org.cloudbus.cloudsim.power.models;

/**
 *
 * @author Ketema Deresa
 */
public class PowerModelSpecPowerHost_Type_2 extends PowerModelSpecPower {

        /** The power. */
        private final double[] power = { 385, 401.5, 418, 434.5, 451, 467.5, 484, 500.5, 517, 533.5, 550 };

        /*
         * (non-Javadoc)
         * @see org.cloudbus.cloudsim.power.models.PowerModelSpecPower#getPowerData(int)
         */
        @Override
        protected double getPowerData(int index) {
                return power[index];
        }

}
```

Figure 3.4. Host type-2 configuration.

Virtual machine types and their requirements

For this research work, the data of the virtual machine's requirements were obtained from global cloud computing providers called Amazon EC2 to better reflect the real demands of VMs.[3]. Amazon EC2 offers a diverse range of instance types that are designed to cater to different use cases. Researchers have the opportunity to choose the best resource mix for their applications because to the flexibility these instance types offer in terms of configurations for CPU, memory, storage, and networking capabilities.

Within each instance type, there are multiple instance sizes available. This enables users to scale the resources allotted in accordance with the particular demands of their task.

Table 3.4. virtual machine's requirements instances

Parameters	VMType			
	High-CPU Medium	Extra Large	Small Instance	Micro
No-of Cores	1	1	1	1
MIPS	3000	2500	2000	1000
RAM	8500	3750	2000	800
BW	1MB	1MB	1MB	1MB
Storage	3.85GB	2GB	1.75GB	613MB

[3] https://aws.amazon.com/ec2/instance-types/

```
public class MyInstance {

        public final static boolean ENABLE_OUTPUT = true;
        public final static boolean OUTPUT_CSV    = false;

        public final static double SCHEDULING_INTERVAL = 300;
        public final static double SIMULATION_LIMIT = 24 * 60 * 60;

        public final static int CLOUDLET_LENGTH = 2500 * (int) SIMULATION_LIMIT;
        public final static int CLOUDLET_PES    = 1;

        public final static int VM_TYPES         = 4;
        public final static int[] VM_MIPS        = { 3000, 2500, 2000, 1000  };
        public final static int[] VM_PES         = { 1,1,1,1};
        public final static int[] VM_RAM         = { 8500,  3750, 2000, 800 };
        public final static int VM_BW=1000; // 1 MB/s
        public final static int VM_SIZE          = 2500; // 2.5 GB
        public final static int[] Storage = {3580,2,1750,613};

        public final static int HOST_TYPES        = 2;
        public final static int[] HOST_MIPS       = { 40000, 24000 };
        public final static int[] HOST_PES        = { 2, 2 };
        public final static int[] HOST_RAM        = { 16000, 32000 };
        public final static int HOST_BW           = 1000000; // 1 Gbit/s
        public final static int HOST_STORAGE = 1000000; // 1 GB

        public final static PowerModel[] HOST_POWER = {
                new PowerModelSpecPowerHost_Type_2(),
                 new PowerModelSpecPowerHost_Type_1()

        };

}
```

Figure 3.5. Host and VM instance requirements setup.

In this study, we consider a data center built based on the commonly used Fat-Tree topology. Assuming that each switch owns N ports and there are M pods. Each pod contains N/2 aggregation switches, N/2 edge switches, and N2/4 physical machines. Besides, each pod is connected to N^2 or 4 core switches.

3.5. Implementation and Coding

Implementation was done using CloudSim and NetBeans IDE. Snapshots of some results and code listings are shown in Appendices I and II, respectively.

BOHGOA Algorithm Parameter setting.

Based on exploratory experiments, appropriate parameter values were identified. In our experiments, we used the following parameters for BOHGOA that have been summarized in the Table 3.5, below.

Table 3.5. BOHGOA Algorithm Parameters

	Parameters	**Values**	**Parameters**	**Values**	**Parameters**	**Values**
GA	Max-generation	100	Crossover rate	0.8	Max-Pareto set	20
	Population-size	50	Mutation-rate	0.1		
ACO	Number of ants	100	α	1	ϵ	0.1
	Maximum iteration	50	β	2	Evaporation rate	0.1

CHAPTER FOUR

PROPOSED SOLUTION

In this chapter, we discussed the proposed solution for bi-objective optimization of bandwidth and energy consumption for efficient virtual machine placement in cloud computing, which includes the proposed solution architecture, solution process, formulation of both objectives, proposed algorithm, integration of GA with ACO, and specific implementation of the suggested algorithm.

4.1. Proposed solution architecture

As discussed earlier, this study provides a detailed discussion of the proposed solution architecture for the minimization of bandwidth and energy consumption for efficient virtual machine placement in cloud computing. The system model considered in this work consists of big data centers that contain different heterogeneous physical machines, and each physical machine is categorized by its CPU performance in terms of MIPS (millions of instructions per second), its total network bandwidth, and its total RAM occupied. These physical machines also consist of several virtual machines to run user applications and fulfill the customer's demands. They are also characterized by their total MIPS, network bandwidth, and RAM so that several users can request the provisioning of virtual machines along with their particular characteristics. For this study, we have used a fixed number of physical machines that are heterogeneous.

Many IaaS providers like Google Compute Engine and Amazon AWS today provide specialized infrastructure as virtual machine instances to their customers. They offer many types of virtual machine instances with different configurations and prices. A big challenge is how to best place these virtual machines (VMs) in cloud data centers to maximize the cloud provider's profit while meeting users' needs. Because many users are simultaneously requesting virtual machine instances. During these requests, bandwidth resources and energy are excessively consumed, which has an impact on data centers' performance. To address this problem, we provided our proposed solution architecture for managing virtual machines in a cloud computing data center, as depicted in the Figure 4.1 below. It has four main modules. Namely: VMQueue, VM placement, Resource monitor, and Migrator module.

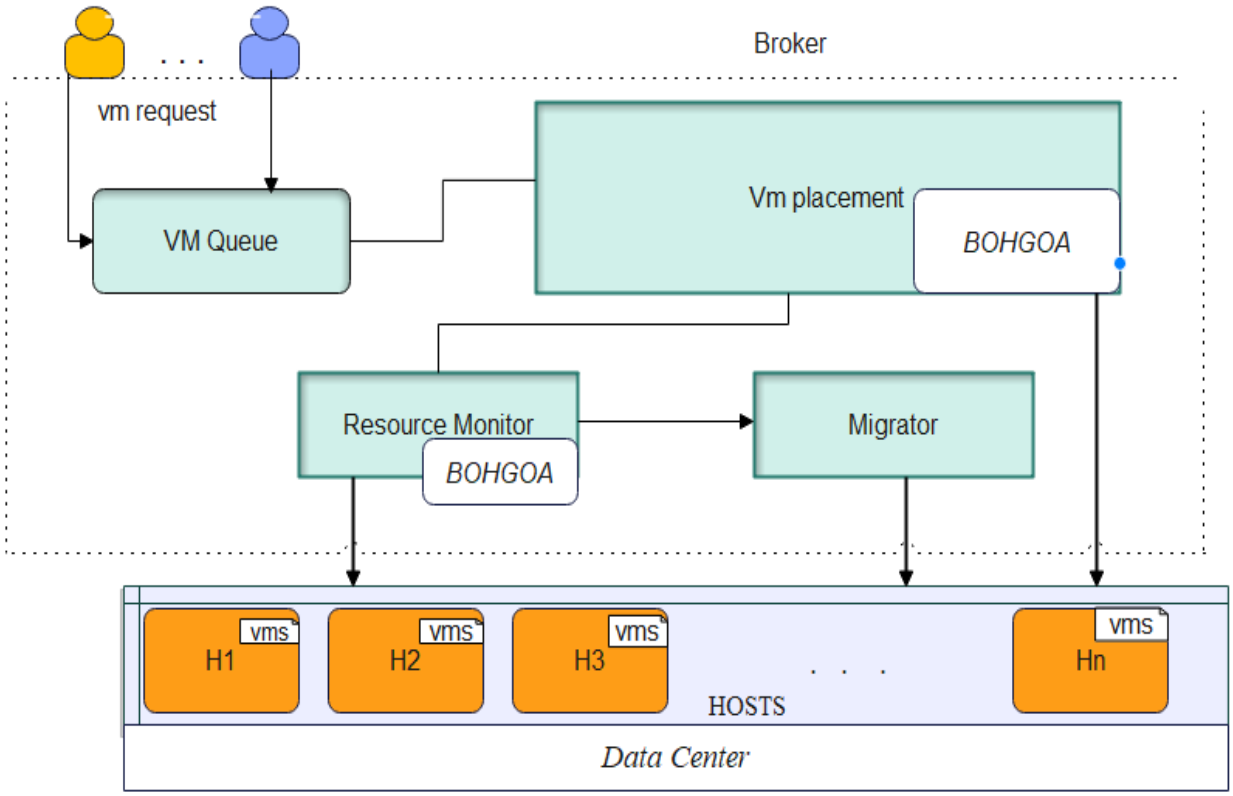

Figure 4.1. proposed solution architecture

Main Components of the proposed solution architecture.

User: Services offered by clouds have increased in popularity due to the wide variety of services and application types that have been developed. Customers may send their applications as virtual machine deployment requests to the cloud provider. This application requests that types communicate with each other. Means, there will be traffic between these requested virtual machines. Physical resource indicators like CPU cores, memory capacity, and network bandwidth are used to establish each virtual machine's configuration.

Virtual machine queue. The virtual machine requests from users for virtual machines are stored in the virtual machine request queue based on their traffic demand. In this case, in particular, the VMs are stored in a list called vmList, which is sorted in descending order based on their level of traffic demand. By placing the high-demand VMs together, the amount of network traffic between them can be reduced, leading to a decrease in the overall use of bandwidth resources. This is very important for the proper placement of the requested virtual machine by the users and for reducing link congestion in the cloud data center.

Resource monitoring and migration modules: After a virtual machine (VM) is placed on a physical machine (PM), the resource monitoring module regularly checks the status of the PM by collecting metrics such as CPU usage, memory usage, and bandwidth resource

consumed. If the monitoring module get that a physical machine (host) is overloaded, meaning that its resource usage has reached or exceeded a predefined threshold, it will trigger the migrator module to move one or more VMs from the overloaded physical machine to another physical machine that has more available resources.

It used BOHGOA to check the overflow of the physical machine by investigating whether a solution was feasible or not. If not, it removes virtual machines from physical machines to lower their utilization below a set upper threshold. The proposed algorithm's goal is to identify a set of Pareto-optimal solutions that minimize both objectives of energy usage and bandwidth resource utilization simultaneously.

Virtual machine placement module This module takes the virtual machine list from the queue and optimally places the virtual machine on the physical machine. This module is integrated with the BOHGOA algorithm. BOHGOA is a bi-objective optimization algorithm that aims to optimize two objectives simultaneously, namely, minimizing the bandwidth resource and energy consumption.

4.2. Proposed solution process

We have used BOHGOA in our proposed approach to achieve the optimal mapping of virtual machines to physical hosts. Since we are optimizing two solutions (bandwidth resources and energy consumption), we used a bi-objective function. This approach is used to optimize bandwidth resources and energy consumption simultaneously. To strengthen the performance of the BOHGOA, it has been integrated with ACO.

This integration aims to enhance the algorithm's capabilities and improve its effectiveness in solving optimization problems. Population-based approaches are typically used by these bi-objective optimization techniques to find the best solutions. Additionally, to tradeoff between the two objectives described above, we used the Pareto-dominance concept during the selection process. For the bi-objective optimization function, Pareto optimal solutions, or Pareto Set (PS), are formed, in which there is no solution dominated by any other solution.

4.2.1. Formulation For Bandwidth Resource and Energy Consumption model

This subsection presents an energy consumption model since power management is this paper's primary goal and one of the evaluation criteria. This energy model is commensurate with server processing utilization because the primary amount of server energy usage corresponds to its processor utilization. Numerous studies prove that the server's processing utilization and power consumption are linearly related (Zhao et al., 2019).

Energy consumption of each physical machine (PM).

In this study, we merely focused on physical machine energy consumption in data centers. The power consumption of the server's fully utilized state and its idle state are both taken into account when calculating the CPU utilization. Based on this model, energy consumption has been examined in this work.

$$P_{(pj)} = y_{pj} . (p_j^{idle} + (1 - k). p_j^{full} . u_{pj}^{cpu}) \qquad eq(4.1)$$

In this model, k shows a fraction of the energy consumed when the server is idle. In this work, we take an idle server that consumes 70% of the maximum power of the server based on the study. P_j^{full} is the power of the physical machine p_j is at the status of full load. The running physical machine p_j consumes energy even when it carries no load, and P_j^{idle} is the power of the non-load physical machine p_j . P_j ($U_{p\,j}^{cpu}$) is the power of physical machine p_j when the CPU resource utilization is Up_j^{cpu}. If pj does not host any VM, it is switched off for energy-saving purposes, and y_{pj} is set to 0; otherwise, y_{pj} is set to 1.

If pj is idle, its power consumption is fixed as Power (pj idle) and u_{pj}^{cpu} is set to 0; if pj is hosting VMs, the corresponding power consumption is in proportion to u_{pj}^{cpu}

Total Energy consumption of all physical machines (PMs).

Therefore, the total Energy consumption of all physical machines (PMs), EC(PM), is defined according to the following equation (4,2).

$$EC(PM) = \sum_{j=1}^{|p|} p(p_j) \qquad eq\ (4.2)$$

Where EC represents energy consumption, PM represents all physical machines, represents power consumption. In our current research work for energy consumption, the server gets turned off when they enter into an idle state.

Formulation of Bandwidth Resource Consumption

The communication traffic between virtual machines has a drastic impact on the overall performance of a cloud computing data center. Data exchange between virtual machines is achieved through the communication of the physical machine hosting them. Inter-virtual machine communication data is exchanged through communication links. If virtual machines are not properly placed, the link that supports communication among them may become saturated.

Hence, it may lead to performance degradation for applications, and it may also seriously lead to DC's overall performance degradation. Consequently, network performance should also be considered when minimizing the energy consumption of the data center's servers. Providing bandwidth guarantees to specific applications is becoming increasingly important as applications compete for shared cloud network resources (LeeJeongkeun et al., 2014).

The bandwidth resource usage for two arbitrary virtual machines depends on the locations of their hosts and the volume of traffic between them. There is no bandwidth resource usage if both virtual machines are placed on the same physical machines; otherwise, there is bandwidth resource consumption along the communication line between the two PMs that are hosting the two virtual machines. The bandwidth resource consumption by all virtual machines is determined by the number of virtual switches that are traversed in the communication path between physical machines.

In a virtualized environment, when virtual machines communicate with each other, they do so through virtual switches. The more switches that a communication path passes through, the higher the consumption of bandwidth resources will be. Therefore, Bandwidth Resource Consumption is calculated as the follow equation (4,3):

$$R_{B_{total}} = \sum_{vi \in V} \sum_{vj \in V} N_{p(vi),(vj} . T_{vi,vj} \qquad eq(4.3).$$

where, ($N_{p(vi)}$ $N_{p(vj)}$ represents the number of switches along the communication path between two PMs, $p_{(vi)}$ and $p_{(vj)}$. $T_{vi,vj}$ is the traffic amount between two VMs, vi and vj.

Table 4.1: Main Notations and their definitions

Notations	Definition
V	The set of VMs for placement,V = {v1,…,v|V|}, where |V| is the cardinality of V.
P	A set of PMs hosting VMs in V, P ={p1,..pn|P|}, where |P| is the cardinality of P.
p_j	The j th PM
B_{ei}^{Csm}	The already consumed bandwidth of link ei,
B_e^{max}	The maximum bandwidth of link ei
C_{pj}^{cpu}	The CPU capacity of PM pj.
R_{vi}^{cpu}	The CPU resource needed for hosting VM vi.
R_{vi}^{mem}	The memory resource needed for hosting VM vi.
C_{pj}^{mem}	The memory capacity of PM pj.

Vm_i	The i th VM
V_{pi}	The set of VMs that are hosted by PM pj,
U_{pj}^{cpu}	The CPU resource utilization of the PM pj
P_j^{full}	Power of PM in idle state
P_j^{idle}	Power of PM in idle state
EC(PM)	Energy consumption of all PM
$p(PM)$	The total amount of power consumed by all PMs.
K	The ratio of pj idle to pj full. It is set to 0.7 for this experiment
p_j	The $_j$ th PM
y_{pj}	A binary variable indicating if PM pj is switched on or off. y_{pj}=1 if pj is switched on, and y_{pk}= 0, otherwise
$R^B{}_{total}$	Total amount of bandwidth consumed by the communication among all VMs
Tvi , vj	Amount of traffic between VMs Vi and Vj
Npi , pj	The number of switches along the communication path between PMs, p_i and p_j. If the two PMs are connected by the same edge switch, $N_{pi\ ,pj}$ =1; if they belong to the same pod and connected to different edge switches, $N_{pi\ ,\ pj}$ = 3; if their communication involves a core switch, then $N_{pi\ ,\ pj}$ = 5

4.2.2. Optimization Formulation for Bandwidth and Energy consumption

For any bi-objective optimization problem, there are set of two objectives that are need to be minimized. Here, we have formalized the minimization of both bandwidth resource consumption and minimization of energy consumption for efficient VMP problems as the following Equation (4.4) and (4.5).

Minimization formulation for Energy consumption.

$$\text{Minimize} \sum \text{EC(PM)} = \sum P_{(pj)} = y_{pj} \,.\, (p_j^{idle} + (1-k) \,.\, p_j^{full} \,.\, u_{pj}^{cpu}) \qquad \text{eq (4.4)}$$

Minimization formulation for Bandwidth resource.

$$\text{Minimize} \sum R_{B_{total}} = \sum_{vi \in V} \sum_{vj \in V} N_{p_{(vi)\,,(vj}} \,.\, T_{vi,vj} \qquad \text{eq (4.5)}$$

Subject to:

$$\bigcup_{pj \in p} V\,p_j = V \qquad \text{eq (4.6)}$$

$$V_{pi} \bigcap V\,p_j = \emptyset, \quad \forall p_i, \forall p_j \in P, p_i \neq p_j \qquad \text{eq (4.7)}$$

$$\sum_{vi \in Vp_j} R_{vi}^{cpu} \leq C_{pj}^{cpu} \qquad \text{eq (4.8)}$$

$$\sum_{vi \in Vp_j} R_{vi}^{mem} \leq C_{pj}^{mem} \qquad \text{eq (4.9)}$$

$$B_{ei}^{Csm} < B_e^{max} \qquad \text{eq (4.10)}$$

Where Constraint in eq (4.6) ensures that all VMs in V must be successfully placed. Constraint in eq (4.7) ensures that each VM is placed on a single PM. Constraints in eq (4.8) - (4.9) define that the CPU and memory resources occupied by all VMs on PM pj cannot exceed the maximum CPU and memory capacities of pj, respectively. Constraint in eq (4.10) ensures that the bandwidth consumption of link ei cannot exceed a threshold value, B_e^{max}.

4.3. Genetic Algorithm

The genetic algorithm is one of the best-known evolutionary algorithms (GA). A classical Genetic algorithm was originally designed to optimize one objective function. It uses the law of survival of the fittest from the generations after the first population is formed to evolve better and better approximations. It works by, encoding the candidate solutions as chromosomes, applying genetic operators such as selection, crossover, and mutation to generate new solutions, and evaluating the solutions using objective functions(Tang & Pan, 2015).

Population initialization

In a classical genetic algorithm, population initialization is a key step since it can impact convergence speeds and the quality of the final result. Random initialization is the most commonly employed method to generate candidate solutions when information about the solution is not available. Once the initial population is generated, an optimization algorithm can be applied to improve the quality of the solutions (Victer Paul et al., 2015).The GA starts by randomly generating a population of chromosomes or potential solutions.

The number of possible solutions, or the population size, is denoted by M. A matrix of chromosomes of size MxN represents the population of possible solutions generated.

$$Population = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_j \\ \vdots \\ X_M \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,i} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,i} & \cdots & x_{2,N} \\ & & & \vdots & & \\ x_{j,1} & x_{j,2} & \cdots & x_{j,i} & \cdots & x_{j,N} \\ & & & \vdots & & \\ x_{M,1} & x_{M,2} & \cdots & x_{M,i} & \cdots & x_{M,N} \end{bmatrix}$$

Figure 4.2: the population of generated possible solutions representation in GA

Every $x_{j,j}$ decision variable may be represented as a floating point number (real values) or as a predefined set of values for discrete problems. To reproduce a new generation, some of the initial generation's potential solutions are chosen as parents. Each gene (chromosome) describes a certain aspect of the potential cure. The algorithm implementer encodes the targeted problem in a way that will allow the genetic algorithm to try to solve it, and the result of this process is the definition of the chromosomal structure. For instance, in the following example, the chromosome represents VM1 and VM5 on host 1, VM2 and VM4 on host 2.VM3 on host 3, and VM4 on host 4. Note that each column has at most one "1" value, ensuring that each VM is placed on only one PM.

Table 4.2. Chromosome encoding representation in GA

Host/VM	Vm1	Vm2	Vm3	Vm4	Vm5
H1	1	0	0	0	1
H2	0	1	0	1	0
H3	0	0	1	0	0
H4	0	0	0	1	0

$$\text{Solution P} = \begin{matrix} [1, 0, 0, 0, 1], \\ [0, 1, 0, 1, 0], \\ [0, 0, 1, 0, 0], \\ [0, 0, 0, 1, 0] \end{matrix}$$

Figure 4.3. Chromosome representation in GA.

In this optimization algorithm, a chromosome is made up of |Cr| genes, each of which stands for the VM id (VMj) to be positioned at PM id (PMk). A gene's value is a positive number that falls between 1 and PMmax and represents the physical machine where the virtual machine will be placed(Mosa & Paton, 2016).

Selection strategy

Selection helps continuously improve the population's overall fitness. It raises the possibility that a fitter population will be chosen for the following generation. The primary objective of the select operator is to generate a new chromosome for the next generation to maintain population variety. According to their fitness values, the selection operator selects mating chromosomes. Several selection techniques may be used with GA, including random selection, roulette wheel selection, stochastic universal sampling, and elitism (Wang, 2022)

Crossover operator

The crossover process, which is based on the idea of biological interchange, illustrates how to naturally produce new children. The crossover operator exchanges some genes on two chosen chromosomes to create two new chromosomes for the following generation.

Mutation

Another crucial GA operator that initiates additional variability to produce and preserve diversity is mutation. The goal of mutation in GA is to diversify the population under study. By preventing the population of chromosomes from becoming too similar to one another, slowing or even stopping convergence to the local optimum. With the exploration of the search space and a decreased risk of local optimal stagnation, this strategy provides the probability of an entire gene mutation with a very low chance (but greater than zero). After the chosen chromosomes have been rearranged, the mutation operation will be terminated.

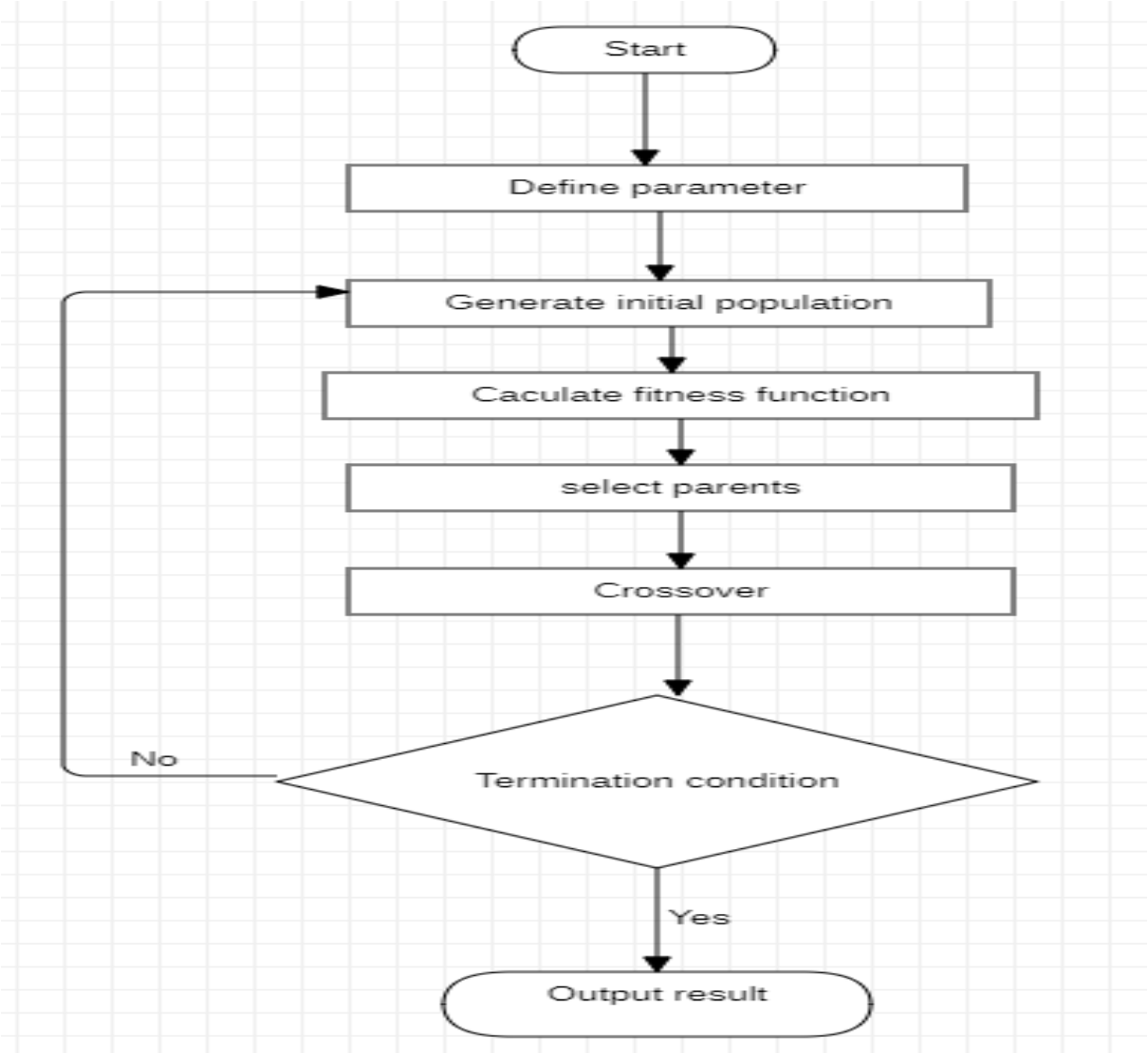

Figure 4.4: Flowchart of classical GA

4.3.1. Bi-objective Hybrid Genetic Optimization Algorithm

A genetic algorithm is a powerful global optimization method that can effectively seek optimum solutions in a large research space. However, it has some limitations that can impact its efficiency.

The limitation of genetic algorithms is, that they often require numerous iterations to converge on the optimal solution. This is because genetic algorithms rely on the principles of evolution, which means that they required to repeatedly generate and evaluate new candidate solutions to find the best possible solution. This repeated process of generating new solutions can result in many redundant iterations, which can lead to inefficiencies in the search process. Additionally, genetic algorithms can sometimes get trapped in local optima, which can limit their ability to search the global optimum.

Another limitation of GA is that, it has no feedback mechanism to guide the search process. The algorithm generates a set of candidate solutions, evaluates their fitness, and selects the best solutions for further evolution. However, there is no mechanism to provide feedback on the quality of the solutions or guide the search toward more promising areas of the search space.

By combining the strengths of both genetic and ACO, it is possible to improve the efficiency and effectiveness of virtual machine placement in cloud computing. The genetic algorithm provides a powerful global search capability to explore the search space and find high-quality solutions, while ACO helps to keep from fall in local optima and quickly converge to the optimal solution.

Ant Colony Optimization (ACO) Algorithm

ACO algorithm is a metaheuristic inspired by the behavior of real ants in their search for the shortest path to food sources. Ants tend to choose the paths marked by the strongest pheromone concentration. The ACO algorithm is an essential system based on agents that simulates the natural behavior of ants, including the mechanisms of cooperation and adaptation. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem(Duan & AI, 2016).

4.3.2. Integration of GA with ACO

GA is a powerful optimization algorithm known for its ability to generate diverse solutions for VM placement. It is used to generate a diverse set of solutions, while the ACO algorithm is used to refine these solutions and ensure that the algorithm does not get stuck in local optima. The GA algorithm generates a population of individuals that represent candidate solutions. Fitness function is used to evaluate the quality of these solutions considering factors such as bandwidth resource utilization and energy usage.

The best-fit individual with the highest fitness value is then selected to be used as the initial pheromone for the ACO algorithm. ACO works by simulating the behavior of ants that deposit pheromones on the ground as they search for food. The pheromones represent the solutions' quality found by the ants. In the ACO algorithm for VMP, ants construct their paths by selecting physical machines for each VM based on the available heuristic information and the pheromone trails left by other ants. These paths represent potential configurations for VM placement. As the ants build their paths and update the pheromone

trail, they incorporate information about the bandwidth resource utilization and energy usage of the selected placements

Once the ants have constructed their paths and updated the pheromone trail, the crossover operator is applied. This operator combines the paths of two ants to create a new path, allowing exploration of different placement possibilities. The selection of paths for crossover and mutation is influenced by the pheromone trail values associated with each path.

Paths with higher pheromone trail values, indicating better utilization of bandwidth resources and energy efficiency, have a higher probability of being chosen for crossover and mutation. The crossover operation can be implemented by exchanging the physical machines assigned to VM that are in the same ant's path. This exchange allows for the exploration of alternative physical machine assignments while considering the constraints of the problem.

Transformation or integration of GA to ACO equations

The transformation of the genetic algorithm solution into the initial pheromone is shown in the following Equation (4.11).

$$I_i^S(0) = \varepsilon . G_n \qquad eq(4.11)$$

Here, $I_i^S(0)$ represents the initial pheromone, ε is a constant and for this work, we have chosen, $\varepsilon = 0.1$, and it determines the balance between exploration and exploitation of the solution space G_n is the GA's optimal solution that is selected based on its fitness value.

Solution Construction

After initialization, ACO moves on to build solutions iteratively so that it may discover the best ones that are feasible and use fewer servers. during this time, the pheromone is deposited between VMi-PMj pairs.

The historical experience of packing virtual machines VMi on the same server pj is given by the preference value as Equation(4.12) below.

$$\tau(i,j) = \begin{cases} \frac{1}{|Vj|} \sum_{j \in pj} \eta(i,j) \text{ if } |Vj| \neq 0 \\ t_o \qquad \text{else} \end{cases} \qquad eq(4.12)$$

Where, $\tau(i,j)$ represent the historic desirability of placing VMi on Pj. $\eta(i,j)$ represents the pheromone trail value between VMi and Pj, |Vj| is the number of VMs that are already on Pj. the value of t_o is typically a small positive constant that is used to ensure that even

servers with no VMs running on them have some level of preference. For this work, we have given $t_o = 0.001$.

Path Selection: During each iteration of the ant colony optimization (ACO) algorithm, each ant selects the next physical machine based on a node transition rule. The node transition rule is based on the pheromone level and heuristic information of the available physical machines, as well as the current state of the ant. The use of heuristics and the pheromone trail will guide ants to specific solutions for specific problems. The probability of assigning an unassigned VMi to server pj using pheromone and heuristic information is calculated by:

$$p_{(i,j)} = \frac{(\tau(i,j)^{\alpha}.(\eta_{(i,j)})^{\beta}}{\sum_{j\in pj}[\tau(i,j)]^{\alpha}.[\eta_{(i)}]^{\beta}} \qquad eq(4.13)$$

Where, $p_{(i,j)}$ is the probability of assigning the unassigned VM j to server i. $\tau(i,j)$ is the heuristic information that represents the desirability of server pj for assigning VMi related to minimum bandwidth resource and energy consumption. $\eta(i,j)$ is the pheromone on the trail between server pi and VMj. α and β are parameters that control the relative importance of the pheromone and heuristic information, respectively. According to preliminary experiments, the value of both α and β we used for the experiment is, α =1 and β=2.

Heuristic information

The heuristic information indicates the desirability of assigning VM v to PM p.

$$\tau_{(i,j)} = \frac{1}{Bei^{csm} + p(pj)} \qquad eq(4.14)$$

Where $\tau_{(i,j)}$ is heuristic information about the resource, Bei^{csm} and $p(pj)$ are bandwidth consumed by link e and power consumption of the j-th physical machine respectively. As heuristic information is larger, the corresponding probability of the host being selected is also larger.

Pheromone Updating: Pheromone information is used as an indirect way for the exchange of information for ants. The purpose of these rules is to reinforce the paths taken by good solutions and discourage the paths taken by poor solutions, thus guiding the search for a better solution. In the iterative part, the pheromone trail should be updated before reiterating the process. The pheromone on each VMi-Pj is updated by:

$$\eta(i,j) = (1-\varepsilon).\eta(i,j) + \varepsilon.\Delta t \qquad eq(4.15)$$

Where ε is the pheromone volatilization factor, which indicates the degree of pheromone evaporation per unit of time. The value is given in the range of (0,1]. Δt(i,j) is the amount of pheromone deposited on the VMi-Pj pair by the ant that finds the solution currently. It is calculated using the following formula of equation (4.16):

The amount of pheromone deposited on the VMi-Pj pair by the ant is represented as follows:

$$\Delta t = \frac{1}{[C_{pj}^{cpu} - R_{vi}^{cpu}] + [C_{pj}^{mem} - R_{vi}^{mem}] + [B_e^{max} - B_{ei}^{Csm}]} \qquad eq(4.16)$$

Where, $[C_{pj}^{cpu} - R_{vi}^{cpu}] + [C_{pj}^{mem} - R_{vi}^{mem}] + [B_e^{max} - B_{ei}^{Csm}]$ are the corresponding normalized remaining CPU and RAM and Bandwidth on server j (the ratio of remaining resources to the resource capacity).

4.3.3. Virtual Machine Placement Considering Bandwidth Resource and Energy

To place Virtual Machines (VMs) considering both bandwidth resource and energy consumption simultaneously, a Bi-objective Hybrid Genetic Optimization Algorithm (BOHGOA) can be employed. This problem can be framed as a bi-objective optimization problem where the objectives are to trade off both bandwidth resource and energy consumption problems. The encoding, fitness function, selection method, and genetic operators are the same as in the original Genetic algorithm(Mo et al., 2015).

The placement of VMs involves allocating them to physical servers in a data center while minimizing both bandwidth resource utilization and energy consumption. Bandwidth resource refers to the network capacity required for VM communication, while energy consumption relates to the power consumed by the physical servers. The BOHGOA combines the benefits of ACO with the Pareto optimality concept.

GA employs natural evolution-inspired concepts, such as maintaining a population of candidate solutions and employing genetic operators such as selection, crossover, and mutation. Fitness functions are defined to evaluate the quality of solutions based on bandwidth resource and energy consumption objectives. Pareto's optimality ensures that the BOHGOA generates a set of optimal solutions that represent the trade-off between bandwidth resources and energy usage. The Pareto set comprises solutions in which achieving one aim leads to a worsening of the other. These concepts offer the best compromise between the two objectives(Mattson et al., 2007).

Pareto optimal solutions

When dealing with more than-objective optimization problems, it's necessary to take into account several objectives at the same time. However, it's important to note that these objectives can conflict with each other or be incomparable, meaning that the best solution for one objective may not be the best for another. In practice, there isn't a single best solution that can be found, instead, there's a set of solutions that are superior when considering all objectives simultaneously. The solution of the more than one objective problem can be classified into two namely the Pareto method and scalarization. The scalarization method makes more than one-objective function to create a single solution and the weight is selected before the optimization process. Scalarization is the process of converting more than one objective into a manageable single optimization issue (Erzurum Cicek & Kamisli Ozturk, 2020).

weighted sum method and ε – Constraint method are the main scalarization methods used in the solution of more than one optimization problem. The weighted sum method combines all objective functions by multiplying them by a predetermined weighting factor, wi, and then minimizes the resulting single function. It is the most popular classical technique. All the objectives for mixed optimization problems (min-max) must be converted into a single type. The ε - constraint approach minimizes the most crucial objective function, fs(x) while treating all other objective functions as constraints. It focuses on a single objective with limits on others. It consistently offers a weakly Pareto optimum location (Marler & Arora, 2010).

Bi-objective optimization considers the objectives as a vector of bi-objectives rather than a single objective function, and the solution is not unique but a set of solutions which are also known as Pareto optimal solutions. Hence, we used the Pareto approach for this bi-objective optimization problem instead of the above traditional formula approach. The Pareto front approach provides the decision-maker with a set of alternative solutions that can be selected based on their trade-off between the bandwidth resource and energy consumption objectives(Yeung et al., 2023).

In conclusion, using a BOHGOA we minimized the bandwidth resource and energy consumption for virtual machine placement in a data center to provide a set of alternative solutions that trade-off between the conflicting objectives, allowing the decision-maker to make an informed decision based on their specific requirements.

The specific implementation algorithm steps for the Bi-objective hybrid genetic optimization algorithm are as follows:

Input: Pop (population size), max-generation, VM (the set of virtual machines), PM(the set of physical machines).

Output: Pareto set of non-dominated solutions.

Begin

1. Obtain the set of sorted virtual machines in descending order of their traffic demand, PM
2. Initialize the Population with the Pop individual.
3. Evaluate the fitness of each individual in the population according to Eq (4.6) – Eq (4.10).
4. Create a Pareto set from the population regarding bi-objective to generate non-dominated solutions using Eq (4.4) & Eq (4.5).
5. While the termination condition is not true do the following.
6. Generate new population pop.
7. Apply a selection operator to choose individuals for the next generation. Individuals with higher fitness values have a greater chance of being selected.
8. Transform selected individuals into ACO using eq (4.11).
9. Initialize pheromone for each VM-PM pair
10. Calculate the desirability of $\tau(i,j)$ according to Eq. (4.12)
11. Select the physical machine for the virtual machine using Eq (4.13)
12. Update the pheromone of each path using Eq. (4.15)
13. Select the best ant
14. Apply crossover operator.
15. Apply a mutation operator to randomly modify some individuals in the population.
16. Evaluate the fitness of each individual in the population according to Eq. (4.6)–Eq (4.10).
17. Apply a feasibility repair algorithm
18. apply a local optimization algorithm
19. Return non-dominated solution

End

Pseudo-Code for the Bi-objective hybrid genetic optimization algorithm:

Input: PopSize (population size), max-generation, VM(the set of virtual machines), PM(the set of physical machines).

Output: Pareto set of non-dominated solutions.

1. for i = 1 to popSize do
2. population(i, l:n) [i][l..n]← VM and PM.
3. end for
4. for i = 1 to popSize do
5. evaluate fitness(population) using equations (4.6) - (4.10)
6. end for
7. Pareto-set(**PS**)← non-dominated(population) using equations (4.4) and (4.5)
8. while termination criteria are not met do
9. for i = 1 to popSize do
10. use roulette selection to choose_ population.
11. Initialize pheromone, (number of VMs, number of PMs, "Ants,α, β).
12. For every pj ∈ PM.
13. Calculate $\tau(i,j)$ according to Eq. (4.12)
14. Select the physical machine for the virtual machine using Eq (4.13)
15. Update the pheromone of each path using Eq. (4.16)
16. Calculate the value of the objective for every ant.
17. Select the best ant
18. end for
19. probabilistically_use_crossover_operator_to_produce_new_offspring
20. end for
21. for i = 1 to popSize do
22. apply-mutation to(population) with-probability
23. evaluate fitness(population) using equations (4.6) - (4.10)
24. end for
25. repair infeasible solutions(population) using equations (4.6) - (4.10)
26. apply local optimization to solutions of(population)
27. **PS**← update non-dominated solution(population) using equations (4.4) and (4.5)
28. end while
29. return Pareto-set as a non-dominated solution

A roulette wheel selection operator is chosen as the selection method for the proposed multi-objective hybrid genetic algorithm. With this technique, some chromosomes with higher fitness are copied into the new population. In this manner, chromosomes with different fitness levels have a chance to be picked. Then, parents will be selected at random based on how many crossovers we intend to carry out in the following population. For each chromosome in the population, the fitness value is calculated. Then, based on their relative fitness value, two chromosomes are selected to create the new generation(T. et al., 2009).

To increase the chance of producing offspring with higher fitness values, a single-point crossover is selected for the exchange between a particular set of chromosomes. The descendant population's members replace the ascending population's chosen individuals(Fernandez et al., 2021).

Algorithm of Crossover operator

Input: Parent Solution PM1 = [a1 a2...an], PM2 = [b1 b2...bn] with n as the number of hosts. Objective functions: obj_bandwidth, obj_energy.
Output: New Solution NS = [c1 c2...cn] as the new solution where n is the number of hosts.

1. fit1_BW &En = Fitness(PM1) based on objective function obj_bandwidth,& obj_energy
2. fit2_ BW &En= Fitness(PM2) based on objective function obj_bandwidth, obj_energy
3. for j = 1 to n do
4. randomly generate a real value between 0 and 1, r, so that, $0 \leq 1 \leq r$.
5. Compute the total fitness value for bandwidth and energy.
   Total_fit = fit1_BW & En + fit2_BW & En.
6. if r < fit1_BW &En / fit1_BW &En + fit2_ BW &En)
7. Set NS[j] = PM1[j]
8. End if.
9. Else
10. - Set NS[j] = PM2[j]
11. End else.
12. End for.

Return the new solution: NS

Where, PM1 & PM2 represents set of host1 and host1 in population respectively.

Algorithm of Mutation operator.

Input: already {VMs mapped to PMs}. m: number of PMs, n: number of VMs as chromosome Chr

Output: {VMs mapped to PMs} as new chromosome NewChr

Begin

Put NewChr = Already {VMs mapped to PMs}.;

// Select 2 host machines i and j such that

Draw two random numbers i and j, where i is a pi and pj is a PM such that $Pi.CPUUtil <$

$C_{pi}^{cpu}$and Pj.$CPUUtil < C_{pj}^{cpu}$and pi^{bcsm}< pj^{bcsm};

//From host I and j select one of the best as b (s for select) and other as remaining (r for reject) such that

Pb .avg$CPUUtilization <$ threshold && Pb $.CPUUtilization > Pr\ .CPUtilization$

Assign VMs from Pr to Pb in NewChr.

Return {VMs mapped to PMs} as NewChr.

End

The infeasible solution repairing Algorithm.

An individual produced by the crossover operators and mutation operators throughout the evolution or produced in the initial population may not be possible for VM placement, which means that some of the VM placement constraints in eq (4.6)– (4.10) are violated.

On the other hand, each virtual machine in the chromosome can be randomly allocated to any physical machine. However, such randomly generated chromosomes may become infeasible due to constraint violations. Hence, when there are infeasible solutions that exceed the capacity of the resources, BOHGOA uses a repair procedure to resolve these violations. When the CPU, memory, and bandwidth resource limitations on the physical machine are violated, it re-allocates(migrates) the virtual machines that are currently allocated to that physical machine one by one to another physical machine. Thus, turning infeasible solutions into feasible solutions(Zhang et al., 2020).

Therefore, we extended the Genetic algorithm by incorporating an infeasible solution repairing procedure and local optimization.

Pseudo-Code of infeasible solution repairing algorithm

1. for j = 1 to N do // where N is the number of PMs
2. if V[j].violation = 1 then
3. select = V[j].pointer
4. V[j].pointer = V[j].pointer.next
5. fixed = false
6. j' = j + 1
7. $avail^{CPU}$ = Cpj'^{cpu} - R_{vi}^{cpu} of j'
8. $avail^{mem}$ = Cpj'mem - Rvimem of j'
9. $avil^{Bw}$ = B^{max}_{ei} - B^{csm}_{ei} of e connecting j and j'
10. while j' ≠ j do
11. if $avail^{CPU}$ ≥ select.v[j] and $avail^{mem}$ ≥ select.v[j] and $avil^{Bw}$ ≥ select.v[j].trafficdem then
12. select.next = V[j'].pointer
13. V[j'].pointer = select // update the selected VM of j' PM
14. Cpj'^{cpu} += select.v[j]
15. Cpj'^{mem} += select.v[j]
16. B^{max}_{ei} += select.v[j].traffic-dem
17. fixed = true
18. end if
19. j' = j' + 1
20. $avail^{CPU}$ = Cpj'^{cpu} - R_{vi}^{cpu} of j'
21. $avail^{mem}$ = Cpj'^{mem} - R_{vi}^{mem} of j'
22. $avil^{Bw}$ = B^{max}_{ei} - B^{csm}_{ei} of e connecting j and j'
23. end while
24. end if
25. end for

The local optimization Algorithm

The local optimization technique employs a heuristic algorithm to decrease the number of PMs in the VM placement, thereby lowering the overall energy used by the PMs.

It uses a heuristic algorithm to minimize the number of active servers; after that, it looks for a physical machine with a capacity greater than or equal to all of the other co-hosted virtual machines on the current physical machines. If it finds one, it offloads all of the co-hosted virtual machines from the server (PMi) to the server (PMj); after that, it changes PMi's status to hibernate mode. By doing this, it is possible to decrease the amount of bandwidth resources consumed as well as power consumption.

Pseudo-Code of the local optimization algorithm.

Input: Pop as a population with PopSize individuals, n: number of VMs, m: number of PMs

Output: The optimal population with the minimum number of used PMs

```
1. for i = 1 to PopSize do
2.     for j = 1 to the number of PM do
3.         if PM[i][j].Status = Active then
4.             j' = j + 1;
5.             while (j' <= m) do
6.                 if PM[i][j'].Status = Active then
7.                     if PM[i][j'].Cpj'^cpu - PM[i][j'].Upj'^cpu >= PM[i][j].Upj^cpu and
8.                         PM[i][j'].Cpj'^mem - PM[i][j'].Upj'^mem >= PM[i][j].Upj^mem and
9.                         B^csm_ei + PM[i][j'].Upj'^bw - PM[i][j].Upj^bw <= B^max_ei then // migrate vms
10.                        PM[i][j'].Cpj'^cpu = PM[i][j'].Upj'^cpu + PM[i][j].Upj^cpu;
11.                        PM[i][j'].Cpj'^mem = PM[i][j'].Upj'^mem + PM[i][j].Upj^mem;
12.                        PM[i][j'].B^max_ei = PM[i][j'].Upj'bw + PM[i][j].B^csm_ei;
                       /* Remove(drop) VMj from PMj and hibernate it */
13.                        PM[i][j].Status = "Off hibernate";
14.                 end if
15.                 j' = j' + 1;
16.             end while
17.         end if
```

18. end for

19. end for

Algorithm of Metaheuristic search for Pareto set Ranking

Input: pareto size, Set of Hosts and VMs with Initial Solution.

Output: Set of Pareto non-dominated solutions

Begin.

1. For each new solution:
2. If a new solution is not dominated by any solution in the Pareto set
3. Add a new solution to the Pareto set
4. End if
5. For each solution in the Pareto set.
6. If the solution is dominated by a new solution:
7. Delete the solution from the Pareto set,
8. End if
9. End for
10. if pareto set size exceeds pareto size
11. apply ranking of solution in increasing order based on bandwidth & energy
12. Select the top Pareto set solutions with the lowest values of bandwidth resource and energy consumption.
13. End for
14. Output: Set of Pareto non-dominated solutions

End

Pseudo-code of a greedy heuristic algorithm that orders VMs in descending order based on their traffic demands

Here is a possible pseudo-code of a heuristic algorithm that orders VMs in descending order based on their traffic demand and stores them in a list called vmsList.

Input: a set of virtual machines with their traffic demands

Output: a list of virtual machines ordered by their traffic demand

1. vmList = {};
2. Ttemp = Ttraffic
3. i = 1

4. Ttemp = sort(Ttemp, by traffic_demand, order descending)
5. while i <= size(Ttraffic);
6. (vmi, vmj) = Ttraffic[k]
7. if vmi not in vmList:
8. add vmi to vmList
9. if vmj not in vmList:
10. add vmj to vmLis;
11. i = i + 1
12. end of while
13. output vm as vmList for virtual machine placement
14. end

4.3.4. **Flowchart of the Proposed BOHGOA**

A flowchart is merely a graphical depiction of steps. It is frequently used to depict the flow of algorithms, workflows, or processes and displays steps in sequential order.

The proposed algorithm's flowchart initiates by acquiring the virtual machines (VMs) and storing them in a queue called vmList. The order in the queue is determined by traffic demands, with higher demands appearing first. The VMs are then randomly distributed among the servers, taking into account the capacity of each server or host. This distribution process is performed by the Virtual Machine Placement Module. Regular checks are conducted to ensure the availability of each server. For determining the threshold value of host resources, we utilize (Riahi & Krichen, 2018).

Once all the VMs have been distributed among the hosts, the optimal placement for each VM is assessed based on its fitness value. The fitness values that demonstrate the highest degree of suitability are stored in the Pareto Set. From the Pareto Set, the best solution is selected as the non-dominated solution. If the maximum generation value of the algorithm has not been reached, new VMs are generated and redistributed among the servers. Furthermore, the VMs are swapped to achieve a more optimal placement, considering the objectives.

Following each placement, the utilization of each host is evaluated. If the utilization of a host surpasses its threshold value for resources, the VMs are migrated to other hosts. Hosts that do not have any VMs placed on them are switched off.

Therefore, the following Figure 4.5, represents the flowchart of our approach called the Bi-objective Hybrid Genetic Optimization Algorithm (BOHGOA).

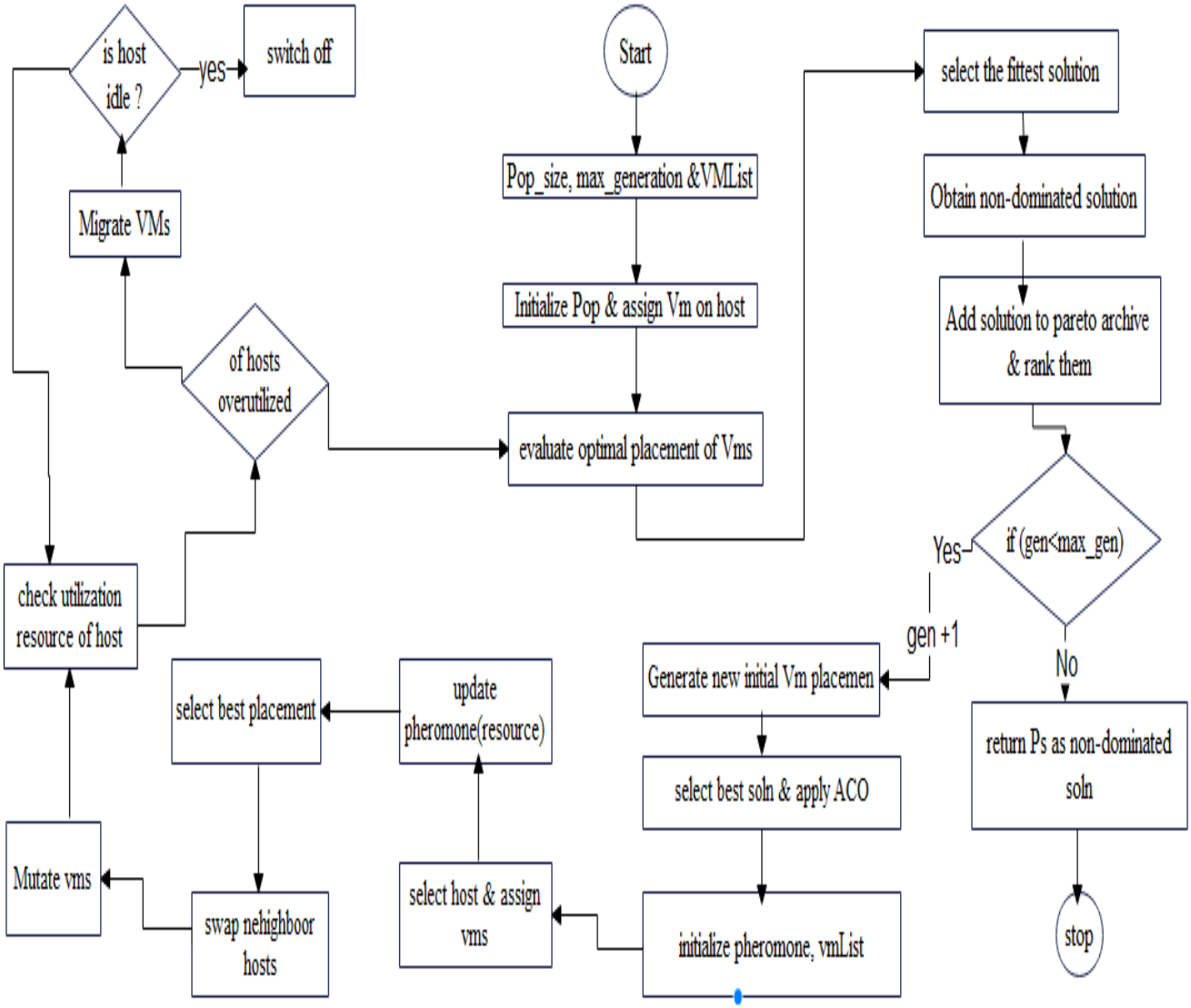

Figure 4.5. Flowchart of proposed BOHGOA.

CHAPTER FIVE

EXPERIMENTAL RESULTS AND DISCUSSIONS

In this chapter, we discussed the algorithm parameter setup and the outcome evaluation metrics, which are bandwidth resource and energy consumption reduction. With a bi-objective hybrid genetic optimization algorithm, we were compared to other algorithms, including the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and First Fit Decreasing (FFD) algorithm.

5.1. Performance Evaluation

In this section, we evaluated the performance and efficiency of the proposed bi-objective optimization technique called BOHGOA.

To evaluate our proposed BOHGOA, we compared it with GA, ACO, and, FFD in terms of total bandwidth resource consumption and total energy consumption. For the simulation of the cloud environment, we used 10 different test cases that were run with varying numbers of VMs and different workload conditions. We have chosen the policy of minimum migration time (MMT) for VM selection with the value 1.2 as a safety parameter.

5.1.1. Bandwidth Resource Consumption Evaluation of BOHGOA

In this section, we evaluated the proposed algorithm's bandwidth resource consumption compared to existing algorithms. We have compared our proposed algorithm's efficiency and performance by measuring its bandwidth resource usage and energy consumption during VMP in cloud computing with each of the existing algorithm.

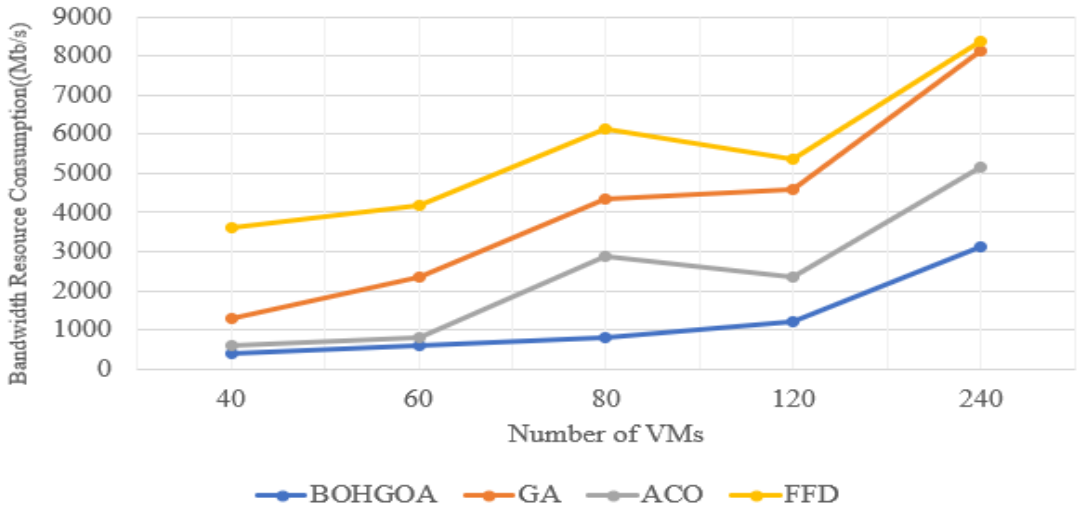

Figure 5.1. Bandwidth resource consumption of PM of 240 VMs Placement.

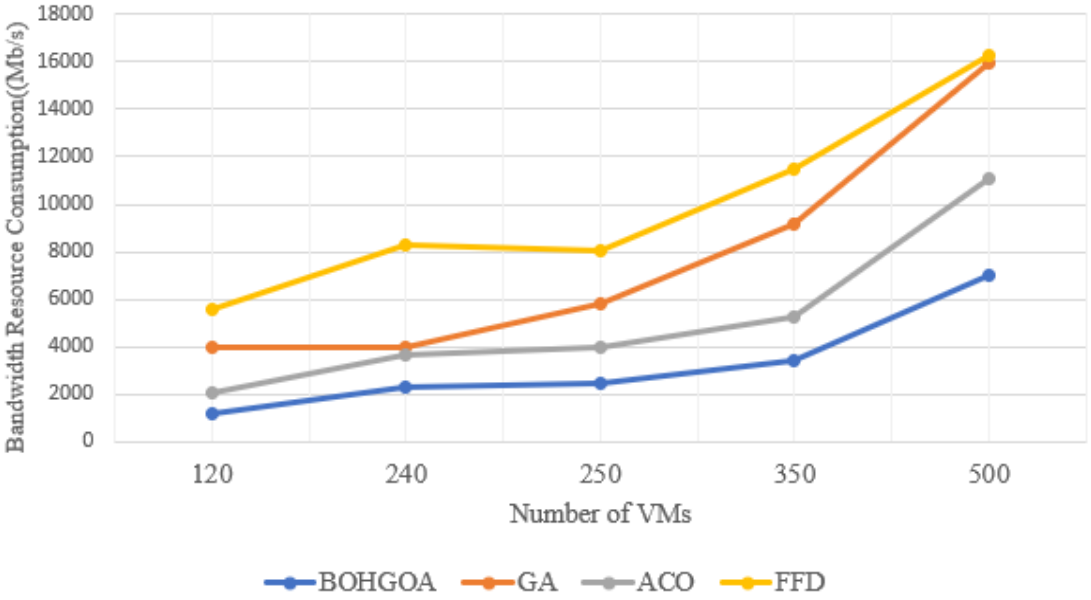

Figure 5.2. Bandwidth resource consumption of PM of 500 VMs Placement.

As shown in Figure 5.1. above, BOHGOA decreased Bandwidth resource consumption by 54.14%, 32.11%, and 57.47% when compared to GA, ACO, and FFD for 240 VMs placement respectively. Also, as depicted in Figure 5.2, it reduced by 38.12%, 22.76%, and 47.05% when compared to the GA, ACO, and FFD for 500 VMs placement respectively. These reductions exhibit BOHGOA's efficiency to minimize bandwidth resource usage more successfully than the other methods. Therefore, BOHGOA reduces link congestion and improves overall network performance by efficiently deploying virtual machines.

5.1.2. Energy Consumption Evaluation of BOHGOA

In this section, we focused on evaluating the efficiency of BOHGOA in reducing physical machine energy consumption. The primary goal of the investigation was to evaluate how effectively BOHGOA manages and optimizes physical machine energy usage in comparison with the other algorithms. Energy consumption is an important consideration in data centers because it has a direct influence on environmental sustainability. BOHGOA outperformed the three other algorithms.

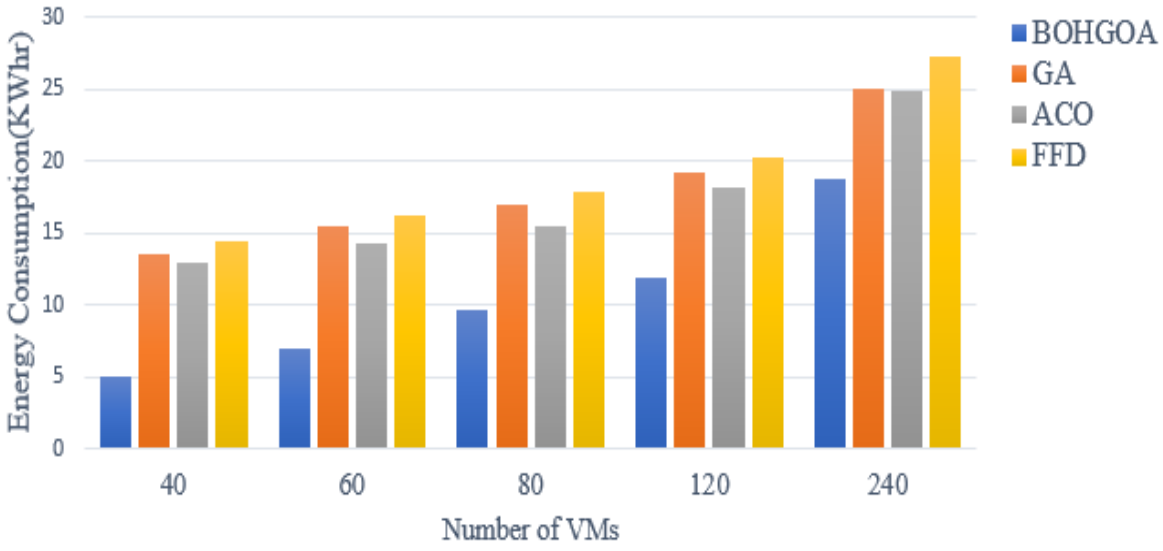

Figure 5.3. Energy consumption of PM of 240 VMs Placement

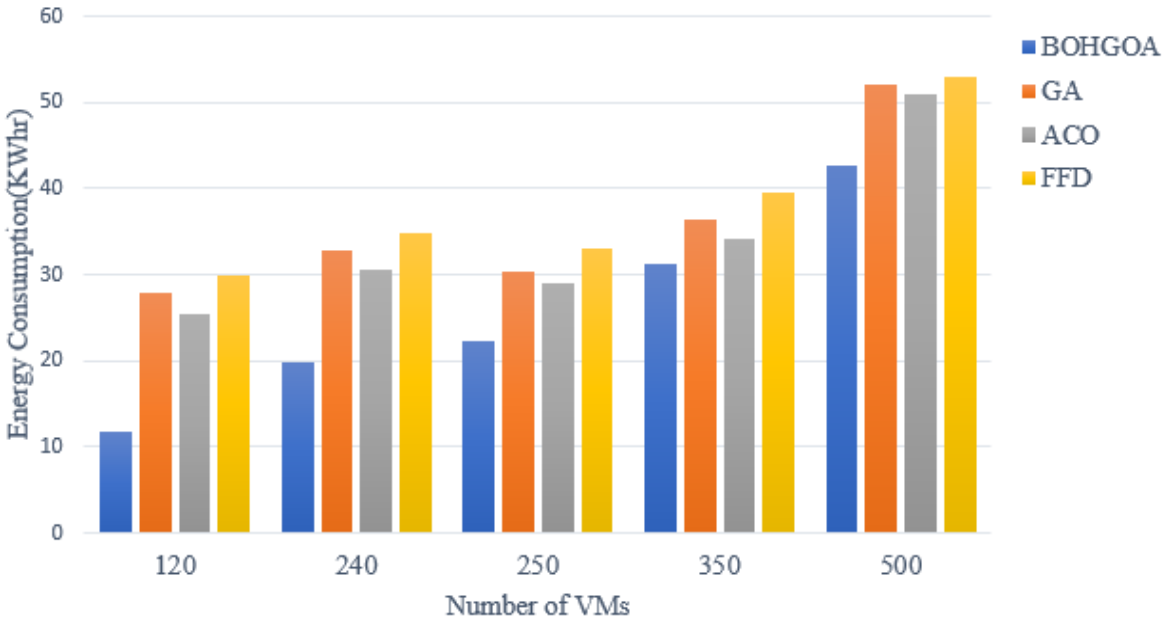

Figure 5.4. Energy consumption of PM of 500 VMs Placement.

Here, when compared to GA, ACO, and FFD algorithms with BOHGOA. BOHGOA achieved an overall reduction in energy usage of physical machines for the placement of 240 VMs by 37.71%, 34.01%, and 40.14%, respectively as depicted in Figure 5.3 above.

This means that BOHGOA consistently outperforms these algorithms, resulting in notable energy savings. Similarly, as seen in Figure 5.4 above, for the placement of 500 VMs, BOHGOA achieved a reduction in energy usage of physical machines by 27.50 %, 22.28%, and 30.52% when compared to GA, ACO, and FFD algorithms respectively. Therefore, these findings emphasize the significant advantages of BOHGOA in optimizing energy usage and improving overall efficiency in virtual machine placement. This energy-saving feature is critical for lowering operational costs and decreasing the environmental impact of data centers.

In general, this research work answered four research questions, below is the result concerning each research question discussed:

Q1. How to Formulate both bandwidth resources and energy consumption for efficient VMP?

In this study, we have formulated both bandwidth resources and energy consumption models for efficient VMP in cloud computing in subsection 4.2.1. Bandwidth resource consumption formulation has been formulated as a summation of the number of Switches between hosts and traffic demand between virtual machines that are placed on the host. Finally, we have formulated both of them simultaneously considering resource constraints in data centers. This formulation is to minimize both objectives simultaneously and to achieve bandwidth resource-energy consumption trade-off.

Q2. How to integrate GA with ACO for Efficient VMP in cloud computing?

This study, integrated GA with ACO to get BOHGOA in subsection 4.3.2. First, GA generates a diverse set of solutions, and ACO refines this solution and ensures that the algorithm does not get into local optima. An ACO takes the best fittest solution selected by GA. This ACO makes the solution to be converged quickly. Second, the best Solution at ACO that has the highest pheromone path is selected for crossover and mutation probability. finally, the best solution has been selected as the optimal solution considering the resource constraints.

Q3. How to place Virtual Machine considering bandwidth resource and energy consumption simultaneously using a Bi-objective Hybrid Genetic optimization Algorithm (BOHGOA)?

This thesis has deployed VMs considering both bandwidth resources and energy consumption by utilizing BOHGOA in subsection 4.3.3. In this study, VM is allocated to physical machines at a minimum bandwidth resource and energy. BOHGOA generates a Pareto set of a solution and a non-dominated solution selection approach has been employed to select the non-dominated solution in the Pareto set.

To answer **Q4**, a Proper evaluation and comparison of the proposed approach have been done against the three algorithms namely: GA, ACO, and FFD. The simulation was done by the cloudSim toolkit. The BOHGOA outperforms these algorithms, based on the two metrics known as bandwidth resource and energy consumption minimization in cloud computing. Therefore, BOHGOA achieved a better bandwidth resource-energy trade-off.

5.2**. Complexity analysis of BOHGOA**

To simplify the analysis of BOHGOA further we used the term time complexity. Time complexity is an abstract way to represent the running time of algorithm in terms of rate of growth and Big-O notations only. It is an approximate estimation of how much time an algorithm will take for large value of input size(Arora & Barak, 2006).

Here we discussed the time complexity of our proposed algorithm. The proposed BOHGOA approach is the execution of two meta-heuristics algorithms: GA and ACO. Therefore, the total time complexity can be determined by computing the complexity of the two methods given. So, the complexity of each meta-heuristic is determined by adding up the complexity of all the procedures.

Table 5.1. Time complexity

Procedure	Complexity
Population-size	O (Population-size* m * n^2)
Idle servers	O (Population-size* m * n)
Roulette wheel	O (population-size * Crossover rate)
Crossover rate	O (population-size * crossover rate)
Mutation rate	O (population-size *mutation rate)

Complexity for GA: O (maximum iteration * population-size * Crossover rate * m2 * n^2).

Table 5.2. Time complexity

Procedure	Complexity
Solution	O (number of ants* m *n^2)
Update	O (number of ants *m*n)
Idle servers	O (number of ants *m*n)

The Overall complexity for ACO: O (maximum iteration * number of ants * m^2 * n^2).

In this regard, having n VMs, m PMs, we considered the parameters defined in section 3.5, in Table 3.5. As shown in Tables 5.1 and 5.2 above, we have set the number of solutions namely, the population size, and number of ants in both meta-heuristics during the initializing phase as fixed numbers.

The Overall Complexity of BOHGOA.

O (maximum iteration * population-size * Crossover rate * m2 * n^2) + O (maximum iteration * number of ants * m^2 * n^2) =O [(Population-size* Crossover rate + number of ants) *(maximum iteration * m^2 * n^2)] = O (maximum iteration * m^2 * n^2). Therefore, the overall complexity of the BOHGOA algorithm can be simplified to O (maximum iteration * m^2 * n^2) This means that the complexity is mainly dependent on the maximum number of iterations and the sizes of the server set (m) and the VM set (n). These factors have a more significant impact on the overall complexity compared to the population size and number of ants.

CHAPTER SIX

CONCLUSION AND FUTURE WORKS

6.1. Conclusion

This chapter summarizes the research work that is presented in this thesis on bi-objective optimization of bandwidth resources and energy consumption for efficient virtual machine placement in cloud computing and highlights the key contributions.

Cloud computing, a utility-based computing trend, has undergone a revolution due to the working conditions in the IT industry. The need for cloud data centers is growing daily as a result of this upward trend, which uses up the majority of the world's computer power and results in significant energy and bandwidth consumption. VMP is a complex problem in cloud computing that involves allocating virtual machines to physical machines efficiently and optimally. This problem is classified as NP-hard, which means that finding an exact solution within a reasonable time is computationally challenging. Therefore, to tackle the complexity of VMP, evolutionary algorithms are required to be employed.

This thesis has focused on the bi-objective optimization of bandwidth resources and energy consumption for efficient virtual machine placement in cloud computing. After analyzing the shortcomings of existing virtual machine placement methods, we have formulated both bandwidth resource and energy consumption model for efficient virtual machine placement. Bi-objective optimization algorithm called BOHGOA which combines genetic algorithms with ant colony optimization (ACO) has been proposed. Then our bi-objective optimization method is proposed to optimally place virtual machines on hosts in minimum bandwidth resource and energy consumption.

In this case, there is no single optimal solution but rather a set of solutions that offer trade-offs between bandwidth resource utilization and energy consumption. Regarding a bi-objective optimization algorithm, our proposed algorithm efficiently explored and identified a diverse range of solutions along the Pareto front, representing trade-offs between bandwidth resource use and energy consumption. It was evaluated as properly placing virtual machines, considering both bandwidth resources and energy consumption.

The experimental results explain the superiority of BOHGOA compared to GA, ACO, and FFD in terms of both bandwidth resource, and physical machine energy consumption. BOHGOA offers a substantial improvement, with reductions of Bandwidth resource

consumption by 54.14%, 32.11%, and 57.47% for 240 VM placement and 38.12%, 22.76%, and 47.05% for 500 VM placement when compared with GA, ACO, and FFD, respectively. Additionally, the proposed algorithm reduced physical machine energy consumption by 37.70%, 34.01%, and 40.14% for 240 VM placement and 27.50%, 22.28%, and 30.52% for 500 VM placement when compared with GA, ACO, and FFD algorithms respectively.

These findings emphasize the effectiveness of BOHGOA in optimizing the allocation of virtual machines and achieving a better trade-off between bandwidth resource utilization and energy efficiency in cloud computing environments. From this, the proposed algorithm achieved superior performance in optimizing these objectives, indicating its effectiveness in finding more efficient and balanced solutions for VMP in cloud computing environments.

6.2.Contribution

This study contributes to the field of cloud computing by introducing bi-objective optimization techniques that focus on optimizing bandwidth resource and energy consumption in efficient virtual machine placement.

Formulation for both bandwidth and energy have been formulated for efficient virtual machine placement. The proposed bi-objective hybrid genetic optimization algorithm combines genetic algorithms with ACO, enabling simultaneous optimization and enhanced performance. This work provides valuable insights and decision support for cloud computing providers seeking to achieve optimal bandwidth resource utilization, and energy efficiency in cloud computing. To select the most optimal solutions, we utilized a concept called the Pareto set. The Pareto set consists of solutions that are not dominated by any other solution, meaning they are the best possible solutions. By using the Pareto set, BOHGOA was able to choose non-dominated solutions that balance bandwidth resources and energy consumption.

This capability is crucial in cloud computing, where minimizing bandwidth resource usage and energy consumption are critical factors for cost-effectiveness and sustainability. By optimizing virtual machine placement based on bandwidth resource utilization and energy consumption, Cloud providers can achieve better bandwidth resource utilization, and reduce energy consumption.

Therefore, this study has provided valuable insights and decision support for cloud computing providers aiming to achieve minimum bandwidth resource and energy consumption in their data centers.

6.3. Future Works

Even though this thesis presented its contribution towards bi-objective optimization of bandwidth resources and energy consumption for efficient virtual machine placement, there are still numerous open research challenges that were not addressed in this thesis and need to be addressed in the future:

- In future work, it will be possible to extend the dimensions of our problem by taking into account security. Considering security during VM placement is crucial in today's computing environments, where the protection of sensitive data and prevention of unauthorized access are paramount.

- The most essential advantage of cloud computing is that it provides the QoS services specified in the SLA agreement signed between the user and the provider of the services. Since these customers' requirements may change over time, it is critical to developing new algorithms that account for time fluctuations in SLA while utilizing a minimum number of physical servers, i.e., without raising the cost of the data center's infrastructure.
- Additionally, it is possible to consider faulty tolerance by leveraging the proposed algorithm in this thesis or developing new algorithm. By considering fault tolerance during VM placement, it is better to identify optimal placement strategies that minimize the impact of such faults, ensuring the reliability of cloud data centers.

Special Acknowledgment

This research project is funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/ 828/23

Adama, Ethiopia

REFERENCE

Abohamama, A. S., & Hamouda, E. (2020). A hybrid energy–Aware virtual machine placement algorithm for cloud environments. *Expert Systems with Applications*, *150*, 113306. https://doi.org/10.1016/j.eswa.2020.113306

Ajith Singh, N., & Hemalatha, M. (2014). Basip a Virtual Machine Placement Technique to reduce energy consumption in cloud data centre. *Journal of Theoretical and Applied Information Technology*, *59*(2), 426–435.

Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, *38*(4), 63–74. https://doi.org/10.1145/1402946.1402967

Alharbi, H. A., El-Gorashi, T. E. H., & Elmirghani, J. M. H. (2019). Energy efficient virtual machine services placement in cloud-fog architecture. *International Conference on Transparent Optical Networks*, *2019-July*, 1–6. https://doi.org/10.1109/ICTON.2019.8840258

Ammar, A. M., Luo, J., Tang, Z., & Wajdy, O. (2019). Intra-Balance Virtual Machine Placement for Effective Reduction in Energy Consumption and SLA Violation. *IEEE Access*, *7*, 72387–72402. https://doi.org/10.1109/ACCESS.2019.2920010

Arivuselvi, A., Amudha, T., & Babu, P. D. (2021). An Effective Ant Colony Optimization Methodology For Virtual Machine Placement In Cloud Data Centre. *Turkish Journal of Computer and Mathematics Education*, *12*(10), 3460–3467.

Arnold, T., He, J., Jiang, W., Calder, M., Cunha, I., Giotsas, V., & Katz-Bassett, E. (2020). Cloud Provider Connectivity in the Flat Internet. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 230–246. https://doi.org/10.1145/3419394.3423613

Bai, Y. (2011). Introduction to NetBeans IDE. *Practical Database Programming with Java*, 155–315. https://doi.org/10.1002/9781118104651.CH5

Balaji, K., Kiran, · P Sai, & Sunil Kumar, · M. (2022). Power aware virtual machine placement in IaaS cloud using discrete firefly algorithm. *Applied Nanoscience*, *1*, 3. https://doi.org/10.1007/s13204-021-02337-x

Buyya, R., & Murshed, M. (2002). GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience*, *14*(13–15), 1175–1220. https://doi.org/10.1002/CPE.710

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

Duan, P., & AI, Y. (2016). Research on an Improved Ant Colony Optimization Algorithm and its Application. *International Journal of Hybrid Information Technology*, *9*(4), 223–234. https://doi.org/10.14257/ijhit.2016.9.4.20

Erzurum Cicek, Z. I., & Kamisli Ozturk, Z. (2020). A Comparative Study of Scalarization Techniques on the Multi-Objective Single Machine-Scheduling Problem Under Sequence-Dependent Setup Time, Release Date and Due Date Constraints. *Gazi University Journal of Science*, *33*(2), 429–444. https://doi.org/10.35378/GUJS.581780

Faheem, M., Akram, U., Khan, I., Naqeeb, S., Shahzad, A., & Ullah, A. (2017). Cloud Computing Environment and Security Challenges: A Review. *International Journal of Advanced Computer Science and Applications*, *8*(10). https://doi.org/10.14569/ijacsa.2017.081025

Fatima, A., Javaid, N., Butt, A. A., Sultana, T., Hussain, W., Bilal, M., Hashmi, M. A. ur R., Akbar, M., & Ilahi, M. (2019). An Enhanced Multi-Objective Gray Wolf Optimization for Virtual Machine Placement in Cloud Data Centers. *Electronics 2019, Vol. 8, Page 218*, *8*(2), 218. https://doi.org/10.3390/ELECTRONICS8020218

Fatima, A., Javaid, N., Sultana, T., Hussain, W., Bilal, M., Shabbir, S., Asim, Y., Akbar, M., & Ilahi, M. (2018). Virtual Machine Placement via Bin Packing in Cloud Data Centers. *Electronics 2018, Vol. 7, Page 389*, *7*(12), 389. https://doi.org/10.3390/ELECTRONICS7120389

Feizollahibarough, S., & Ashtiani, M. (2020). A security-aware virtual machine placement in the cloud using hesitant fuzzy decision-making processes. *Journal of Supercomputing*, *77*(6), 5606–5636. https://doi.org/10.1007/S11227-020-03496-4/FIGURES/8

Feleke, F. (2018). *School of Electrical & Computer Energy Efficient Virtual Machine Placement Algorithms in OpenStack Neat Addis Ababa Institute of Technology Engineering Energy-aware Virtual Machine Placement Algorithms in OpenStack Neat By*.

Ganesan, S., & Ganesan, S. (2021). A multi-objective secure optimal vm placement in energy-efficient server of cloud computing. *Intelligent Automation and Soft Computing*, *30*(2), 387–401. https://doi.org/10.32604/iasc.2021.019024

Gao, L., & Rouskas, G. N. (2018). A spectral clustering approach to network-aware virtual request partitioning. *Computer Networks*, *139*, 70–80. https://doi.org/10.1016/J.COMNET.2018.04.005

Gupta, R. K., & Pateriya, R. K. (2014). Survey on Virtual Machine Placement Techniques in Cloud Computing Environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, *4*(4). https://doi.org/10.5121/ijccsa.2014.4401

Hariharan, B., Siva, R., Kaliraj, S., & Prakash, P. N. S. (2021). ABSO: an energy-efficient multi-objective VM consolidation using adaptive beetle swarm optimization on cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, *0123456789*. https://doi.org/10.1007/s12652-021-03429-w

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Ullah Khan, S. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, 98–115. https://doi.org/10.1016/J.IS.2014.07.006

Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., & Epema, D. H. J. (2008). The Grid Workloads Archive. *Future Generation Computer Systems*, *24*(7), 672–686. https://doi.org/10.1016/j.future.2008.02.003

Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whalley, I., & Snible, E. (2011). Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement. *Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011*, 72–79. https://doi.org/10.1109/SCC.2011.28

Jiang, J. W., Lan, T., Ha, S., Chen, M., & Chiang, M. (2012). Joint VM placement and routing for data center traffic engineering. *Undefined*, 2876–2880. https://doi.org/10.1109/INFCOM.2012.6195719

Karmakar, K., Das, R. K., & Khatua, S. (2022). An ACO-based multi-objective optimization for cooperating VM placement in cloud data center. *Journal of Supercomputing*, *78*(3), 3093–3121. https://doi.org/10.1007/S11227-021-03978-Z/FIGURES/5

LeeJeongkeun, TurnerYoshio, LeeMyungjin, PopaLucian, BanerjeeSujata, KangJoon-Myung, & SharmaPuneet. (2014). Application-driven bandwidth guarantees in datacenters. *ACM SIGCOMM Computer Communication Review*, *44*(4), 467–478. https://doi.org/10.1145/2740070.2626326

Li, P., & Cao, J. (2022). A Virtual Machine Consolidation Algorithm Based on Dynamic Load Mean and Multi-Objective Optimization in Cloud Computing. *Sensors 2022, Vol. 22, Page 9154*, *22*(23), 9154. https://doi.org/10.3390/S22239154

Liu, X., Cheng, B., & Wang, S. (2020). Availability-Aware and Energy-Efficient Virtual Cluster Allocation Based on Multi-Objective Optimization in Cloud Datacenters. *IEEE Transactions on Network and Service Management*, *17*(2), 972–985. https://doi.org/10.1109/TNSM.2020.2975580

Masdari, M., Nabavi, S. S., & Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, *66*(October 2017), 106–127. https://doi.org/10.1016/j.jnca.2016.01.011

Mellette, W. M., Snoeren, A. C., & Porter, G. (2016). P-FatTree: A multi-channel datacenter network topology. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 78–84. https://doi.org/10.1145/3005745.3005746

Mosa, A., & Paton, N. W. (2016). Optimizing virtual machine placement for energy and SLA in clouds using utility functions. *Journal of Cloud Computing*, *5*(1), 1–17. https://doi.org/10.1186/S13677-016-0067-7/TABLES/3

Mueen Uddin. (2012). Green Information Technology (IT) framework for energy efficient data centers using virtualization. *International Journal of the Physical Sciences*, *7*(13). https://doi.org/10.5897/IJPS11.1732

Niranjan, U. V., & Pillai, K. K. G. (2019). Energy Management of Cloud Data Center Using Neural Networks. *Proceedings - 7th IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2018*, 85–89. https://doi.org/10.1109/CCEM.2018.00022

Noormohammadpour, M., & Raghavendra, C. S. (2018). Datacenter Traffic Control: Understanding Techniques and Tradeoffs. *IEEE Communications Surveys and Tutorials*, *20*(2), 1492–1525. https://doi.org/10.1109/COMST.2017.2782753

Rawas, S., & Zekri, A. (2018). Location-Aware Energy-Efficient Workload Allocation in Geo Distributed Cloud Environment. *Journal of Computer Science*, *14*(3), 334–350. https://doi.org/10.3844/JCSSP.2018.334.350

Regaieg, R., Koubàa, M., Ales, Z., Aguili, T., & Aguili, · Taoufik. (2021). Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers. *Computing*, *103*(6), 1255–1279.

Rekha Yashwantrao, B. (2017). *Load Balancing and its Algorithms in Cloud Computing: A Survey*.

Riahi, M., & Krichen, S. (2018). A multi-objective decision support framework for virtual machine placement in cloud data centers: a real case study. *Journal of Supercomputing*, *74*(7), 2984–3015. https://doi.org/10.1007/s11227-018-2348-z

Saxena, D., & Singh, A. K. (2021). A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center. *Neurocomputing*, *426*(xxxx), 248–264. https://doi.org/10.1016/j.neucom.2020.08.076

Sethi, S. (2012). Efficient load Balancing in Cloud Computing using Fuzzy Logic. *IOSR Journal of Engineering*, *02*(07), 65–71. https://doi.org/10.9790/3021-02716571

Shah, M. (2014). *Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm*. *November*.

Sharma, M., & Sharma, P. (2012). Performance Evaluation of Adaptive Virtual Machine Load Balancing Algorithm. *IJACSA) International Journal of Advanced Computer Science and Applications*, *3*(2).

Shaw, S. B., & Singh, A. K. (2015). A survey on scheduling and load balancing techniques in cloud computing environment. *Proceedings - 5th IEEE International Conference on Computer and Communication Technology, ICCCT 2014*, 87–95. https://doi.org/10.1109/ICCCT.2014.7001474

Sloss, A. N., & Gustafson, S. (2019). *2019 Evolutionary Algorithms Review*.

Su, S., Su, Y., Shao, F., & Guo, H. (2016). A Power-Aware Virtual Machine Mapper Using Firefly Optimization. *Proceedings - 2015 3rd International Conference on Advanced Cloud and Big Data, CBD 2015*, 96–103. https://doi.org/10.1109/CBD.2015.25

Takouna, I., Rojas-Cessa, R., Sachs, K., & Meinel, C. (2013). Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers. *Proceedings - 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC 2013*, 251–255. https://doi.org/10.1109/UCC.2013.50

Tang, M., & Pan, S. (2015). A Hybrid Genetic Algorithm for the Energy-Efficient Virtual Machine Placement Problem in Data Centers. *Neural Processing Letters*, *41*(2), 211–221. https://doi.org/10.1007/s11063-014-9339-8

Victer Paul, P., Moganarangan, N., Kumar, S. S., Raju, R., Vengattaraman, T., & Dhavachelvan, P. (2015). Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems. *Applied Soft Computing*, *32*, 383–402. https://doi.org/10.1016/J.ASOC.2015.03.038

Wang, H. (2022). Research on the Application of Genetic Algorithm in Physical Education. *Journal of Mathematics*, *2022*. https://doi.org/10.1155/2022/8477945

Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010). CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 446–452. https://doi.org/10.1109/AINA.2010.32

Xu, J., & Fortes, J. (2011). A multi-objective approach to virtual machine management in datacenters. *Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC 2011 and Co-Located Workshops*, 225–234. https://doi.org/10.1145/1998582.1998636

Xu, M., Tian, W., & Buyya, R. (2010). A Survey on Load Balancing Algorithms for Virtual Machines Placement in Cloud Computing. *Pract. Exper*, *00*, 1–22. https://doi.org/10.1002/cpe

Yang, T., Pen, H., Li, W., & Zomaya, A. Y. (2017). An energy-efficient virtual machine placement and route scheduling scheme in data center networks. *Future Generation Computer Systems*, *77*, 1–11. https://doi.org/10.1016/J.FUTURE.2017.05.047

Yi, Q., & Singh, S. (2014). Minimizing Energy Consumption of FatTree Data Center Networks. *ACM SIGMETRICS Performance Evaluation Review*, *42*(3), 67–72. https://doi.org/10.1145/2695533.2695558

Yu, L., Shen, H., Cai, Z., Liu, L., & Pu, C. (2018). Towards Bandwidth Guarantee for Virtual Clusters under Demand Uncertainty in Multi-Tenant Clouds. *IEEE Transactions on Parallel and Distributed Systems*, *29*(2), 450–465. https://doi.org/10.1109/TPDS.2017.2754366

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. https://doi.org/10.1007/s13174-010-0007-6

Zhang, W., Chen, X., & Jiang, J. (2021). A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems. *Tsinghua Science and Technology*, *26*(1), 95–111. https://doi.org/10.26599/TST.2019.9010044

Zhao, D. M., Zhou, J. T., & Li, K. (2019). An energy-aware algorithm for virtual machine placement in cloud computing. *IEEE Access*, *7*, 55659–55668. https://doi.org/10.1109/ACCESS.2019.2913175

APPENDICES

Appendix I: Snapshots of Some Results.

Placement of 500vm on 80 Host

```
86100.10: [Host #76] energy is 125364.31 W*sec

86100.10: [Host #77] utilization at 85800.10 was 7.26%, now is 7.35%
86100.10: [Host #77] energy is 13654.56 W*sec

86100.10: [Host #78] utilization at 85800.10 was 1.53%, now is 1.66%
86100.10: [Host #78] energy is 124562.79 W*sec

86100.10: [Host #79] utilization at 85800.10 was 10.53%, now is 10.31%
86100.10: [Host #79] energy is 14047.97 W*sec

86100.10: Data center's energy is 4535032.73 W*sec


Over-utilized hosts:

Reallocation of VMs from the over-utilized hosts:
isPareto: true

Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 1 cloudlets
Simulation completed.

Number of hosts: 80
Number of VMs: 500
Total simulation time: 86400.00 sec
Energy consumption: 42.51 kWhr & Bandwidth Resource consumption: 7200.00 Mb/s
Number of host shutdowns: 8

BUILD SUCCESSFUL (total time: 1 minute 10 seconds)
```

Placement of #360 VMs on 60 Host

Appendix II: Sample Source Code Listing.

```
public class BoHGoA extends PowerVmAllocationPolicyMigrationAbstract {
   private static Random rnd;
    private List<BoHGoAInd> pop, pareto;
    private List<? extends Host> hostList;
    private List<? extends Vm> vmList;
    private int populationSize;
private int maxIteration;
private double mutationRate;
private double crossoverRate;
   private  double[][] pheromoneMatrix;
private static final double numberOfAnts = 100;
   private static final double MAX_PHEROMONE = 1.0;
public BoHGoA(List<? extends Host> hostList, PowerVmSelectionPolicy vmSelectionPolicy) {
```

```
        super(hostList, vmSelectionPolicy);
         this.vmList = vmList;
    }
   @Override
  public List<Map<String, Object>> optimizeAllocation(List<? extends Vm> vmList) {
        ExecutionTimeMeasurer.start("optimizeAllocationTotal");
        ExecutionTimeMeasurer.start("optimizeAllocationHostSelection");
        List<Map<String, Object>> migrationMap = new LinkedList<>();
         List<Vm> sortedVmList = sortVmListBasedOnTrafficDemand(vmList);
getExecutionTimeHistoryHostSelection().add(ExecutionTimeMeasurer.end("optimizeAllocationHo
stSelection"));
        printOverUtilizedHosts(getOverUtilizedHosts());
        saveAllocation();
        initBoHGoA(50, 100, 0.1, 0.8);
         // Add transformation of genetic algorithm solution into initial pheromone
        initializePheromone();
        while (true) {
           try { BoHGoAInd selectedInd = pareto.get(rnd.nextInt(pareto.size()));
              migrationMap.add(selectedInd.getMap());
              break;
           } catch (Exception e) {
           }}
        ExecutionTimeMeasurer.start("optimizeAllocationVmSelection");
        List<? extends Vm> vmsToMigrate = getVmsToMigrateFromHosts(getOverUtilizedHosts());
getExecutionTimeHistoryVmSelection().add(ExecutionTimeMeasurer.end("optimizeAllocationVmS
election"));
        Log.printLine("Reallocation of VMs from the over-utilized hosts:");
        ExecutionTimeMeasurer.start("optimizeAllocationVmReallocation");
        migrationMap = getNewVmPlacement(vmsToMigrate, new
HashSet<>(getOverUtilizedHosts()));
getExecutionTimeHistoryVmReallocation().add(ExecutionTimeMeasurer.end("optimizeAllocationV
mReallocation"));
getExecutionTimeHistoryTotal().add(ExecutionTimeMeasurer.end("optimizeAllocationTotal"));
         // Calculate fitness values
        calculateFitnessValues(pareto);
        displayNonDominatedSolutions(pareto);
```

```
    BoHGoAInd optimalSolution = findOptimalSolution(pareto);
        return migrationMap; }
  private List<Vm> sortVmListBasedOnTrafficDemand(List<? extends Vm> vmList) {
    // Create a map to store the traffic demand between VM pairs
    Map<Pair<Vm, Vm>, Double> trafficDemandMap = new HashMap<>();
    // Calculate the traffic demand between VM pairs
    for (int i = 0; i < vmList.size() - 1; i++) {
        for (int j = i + 1; j < vmList.size(); j++) {
            Vm vm1 = vmList.get(i);
            Vm vm2 = vmList.get(j);
            double trafficDemand = calculateTrafficDemand(vm1, vm2);
            trafficDemandMap.put(new Pair<>(vm1, vm2), trafficDemand);   }}
    // Sort the VMs based on traffic demand in descending order
    List<Vm> sortedVmList = new ArrayList<>(vmList);
    sortedVmList.sort((vm1, vm2) -> {
        Double trafficDemand1 = trafficDemandMap.getOrDefault(new Pair<>(vm1, vm2), 0.0);
        Double trafficDemand2 = trafficDemandMap.getOrDefault(new Pair<>(vm2, vm1), 0.0);
        return Double.compare(trafficDemand2, trafficDemand1);
    }); return sortedVmList; }
private double calculateTrafficDemand(Vm vm1, Vm vm2) {
    double trafficDemand = 0.0;
    // Check if vm1 or vm2 is null
    if (vm1 == null || vm2 == null) {
        System.err.println("Error: Invalid VMs provided.");
        return trafficDemand;
    } // Check if vm1 and vm2 are on the same host
    if (vm1.getHost() == null || vm2.getHost() == null || vm1.getHost().getId() ==
vm2.getHost().getId()) {
        return trafficDemand; }
    Host host1 = vm1.getHost();
    Host host2 = vm2.getHost();
    if (host1 == null || host2 == null) {
        System.err.println("Error: Invalid hosts assigned to VMs.");
        return trafficDemand;
```

```
    } // Calculate the traffic demand between the hosts
    double traffic = Math.min(host1.getBw(), host2.getBw());
    trafficDemand = traffic;
    return trafficDemand; }
private void displayOptimalSolution(BoHGoAInd ind) {
    //Log.printLine("Optimal Solution:");
    Log.printLine(ind.toString()); }
private BoHGoAInd findOptimalSolution(List<BoHGoAInd> inds) {
    BoHGoAInd optimalSolution = null;
    double minFitnessValue = Double.MAX_VALUE;
    for (BoHGoAInd ind : inds) {
        if (ind.getFitnessValue() < minFitnessValue) {
            optimalSolution = ind;
            minFitnessValue = ind.getFitnessValue(); }}
    if (optimalSolution != null) {
        Log.printLine("Optimal Solution:");
        Log.printLine(optimalSolution.toString());}
    return optimalSolution; }
 public void displayNonDominatedSolutions(List<BoHGoAInd> pareto) {
        int count = 1;
        for (BoHGoAInd ind : pareto) {
            if (!isDominated(ind, pareto)) {
                count++;  }}}
    private boolean isDominated(BoHGoAInd ind, List<BoHGoAInd> pareto) {
        for (BoHGoAInd otherInd : pareto) {
            if (otherInd.dominates(ind)) {
                return true;   }}
        return false; }
    private void calculateFitnessValues(List<BoHGoAInd> inds) {
    for (BoHGoAInd ind : inds) {
        ind.calculateFitnessValue(); }}
   @Override
  protected boolean isHostOverUtilized(PowerHost host) {
    double utilizationThreshold = 0.8; // Adjust this threshold as needed
```

```
    double utilization = host.getUtilizationOfCpu();
    if (utilization > utilizationThreshold) {
        // Host is over-utilized
        migrateVirtualMachines(host); // Perform the migration action
        return true; }
    return false;}
private void migrateVirtualMachines(PowerHost sourceHost) {
    List<PowerVm> vmsToMigrate = new ArrayList<>(sourceHost.getVmList());
    for (PowerVm vm : vmsToMigrate) {
        PowerHost targetHost = findTargetHostForMigration(vm);
        if (targetHost != null) {
            migrateVm(vm, sourceHost, targetHost); }}}
private PowerHost findTargetHostForMigration(PowerVm vm) {
    List<PowerHost> hostList = getHostList();
    for (PowerHost host : hostList) {
        if (!host.equals(vm.getHost()) &&
!host.getVmScheduler().getAllocatedMipsForVm(vm).isEmpty()) {
            // Return the first host with available resources for migration
            return host; }}
    return null; }
private void migrateVm(PowerVm vm, PowerHost sourceHost, PowerHost targetHost) {
    if (targetHost != null) {
        if (targetHost.vmCreate(vm)) {
            sourceHost.vmDestroy(vm);
            vm.setHost(targetHost);
            System.out.println("VM #" + vm.getId() + " migrated from Host #" + sourceHost.getId() + "
to Host #" + targetHost.getId());
        } else {
            System.out.println("Migration of VM #" + vm.getId() + " from Host #" + sourceHost.getId()
+ " to Host #" + targetHost.getId() + " failed");
        }
    } else {
        System.out.println("No suitable target host found for migrating VM #" + vm.getId());  }}
    private void initBoHGoA(int populationSize, int maxIteration, double mutationRate, double
crossoverRate) {
```

```
    this.populationSize = populationSize;
   this.maxIteration = maxIteration;
   this.mutationRate = mutationRate;
   this.crossoverRate = crossoverRate;
    rnd = new Random();
   pop = new ArrayList<>();
   pareto = new ArrayList<>();
   // Initialize the population with random individuals
   for (int i = 0; i < populationSize; i++) {
    try {
     addToPopulation(new BoHGoAInd(this));
    } catch (Exception e) {
     System.out.println(e.getMessage()); }}    for (int i = 0; i < maxIteration; i++) {
     mutation();
     crossover();  }}
   // Transformation of genetic algorithm-generated solution to pheromone trial.
private void initializePheromone() {
List<Vm> vmList = new ArrayList<>(); // Initialize vmList with appropriate values
List<Host> hostList = new ArrayList<>(); // Initialize hostList with appropriate values
List<BoHGoAInd> pareto = new ArrayList<>(); // Initialize pareto with appropriate values
   // Create a matrix to store the pheromone values
   double[][] pheromoneMatrix = new double[vmList.size()][hostList.size()];
   // Calculate the total fitness of the Pareto set
   double totalFitness = 0.0;
   for (BoHGoAInd ind : pareto) {
    totalFitness += ind.getFitnessValue();
   }   for (BoHGoAInd ind : pareto) {
    double fitnessRatio = ind.getFitnessValue() / totalFitness;
    double pheromoneValue = fitnessRatio * MAX_PHEROMONE; // Adjust MAX_PHEROMONE
according to your needs
    // Update the pheromone values for each VM-host pair in the individual
    for (int i = 0; i < vmList.size(); i++) {
      int hostIndex = ind.getHostIndex(i);
     pheromoneMatrix[i][hostIndex] += pheromoneValue;
```

```
        }}   this.pheromoneMatrix = pheromoneMatrix;
}
private void updatePheromone(BoHGoAInd ind, double pheromoneValue) {
    // Update the pheromone values for each VM-host pair in the individual
    for (int i = 0; i < vmList.size(); i++) {
         int hostIndex = ind.getHostIndex(i);
        pheromoneMatrix[i][hostIndex] += pheromoneValue;
    }} private void crossover() {    BoHGoAInd p1, p2;
    p1 = pop.get(rnd.nextInt(pop.size()));
    do {
        p2 = pop.get(rnd.nextInt(pop.size()));
    } while (p1.equals(p2));
    try {
        addToPopulation(new BoHGoAInd(p1, p2));
    } catch (Exception e) {    }
    // Crossover takes place here
    for (int i = 0; i < pop.size(); i++) {
        if (rnd.nextDouble() < crossoverRate) {
            p1 = pop.get(rnd.nextInt(pop.size()));
            do {
                p2 = pop.get(rnd.nextInt(pop.size()));
            } while (p1.equals(p2));   try {
                addToPopulation(new BoHGoAInd(p1, p2));
            } catch (Exception e) {            } } }}
public void runACO() {
    int numAnts = 100;
    int maxIterations = 50;
    for (int iteration = 0; iteration < maxIterations; iteration++) {
        for (int ant = 0; ant < numAnts; ant++) {
            // Perform ant-specific operations
            crossover();
            updatePheromone(individual, pheromoneValue);  }
    private void mutation() {
    for (BoHGoAInd ind : pop) {
```

```
        if (!ind.isPareto && rnd.nextDouble() < mutationRate) {
            try {
                ind.mutation();
            } catch (Exception e) {            }}}}
    private void removeIndividual(BoHGoAInd ind) {
        pop.remove(ind);
        pareto.remove(ind);
    }  private boolean addToPareto(BoHGoAInd ind) {
        List<BoHGoAInd> dominatedInds = new ArrayList<>();
        for (BoHGoAInd target : pareto) {
            boolean dominates = ind.dominates(target);
            if(dominates) {
dominatedInds.add(target);
} else {
return false; }}  for (BoHGoAInd gaInd : dominatedInds) {
        removeIndividual(gaInd); }
    if (pareto.size() < 20) {
        pareto.add(ind);
        ind.setPareto(true);
        return true;
    } return false; }
private boolean addToPopulation(BoHGoAInd ind) {
    if (pop.size() < 50) {
        pop.add(ind);
        addToPareto(ind);
        return true;}
    List<BoHGoAInd> dominatedInds = new ArrayList<>();
    for (BoHGoAInd gaInd : pop) {
        boolean dominates = ind.dominates(gaInd);
        if (dominates) {
            dominatedInds.add(gaInd); }  }
    if (!dominatedInds.isEmpty()) {
        BoHGoAInd random = dominatedInds.get(rnd.nextInt(dominatedInds.size()));
        removeIndividual(random);
```

```
        pop.add(ind);
        addToPareto(ind);
        return true;}
    return false; }
   private int calculateNumberOfSwitches() {
    List<Host> hosts = new ArrayList<>();
    Set<Integer> switchIds = new HashSet<>();
    for (int i = 0; i < getHostList().size(); i++) {
        hosts.add((Host) getHostList().get(i));
    }
    for (Host host : hosts) {
        switchIds.add(host.getSwitchId());
    }
    int numOfSwitches = switchIds.size();
    if (numOfSwitches == 1) {
        return 1; // Same edge switch
    } else if (numOfSwitches <= 4) {
        return 3; // Different edge switches in same pod
    } else {
        return 5;    }}
  public double calculateBandwidthConsumption(List<BoHGoAInd> inds) {
    int totalBw = 0;
    int maxLinkCapacity = 100; // Maximum link capacity in MB/s
    double upperThreshold = 0.3; // Upper threshold for LINK CAPACITY utilization
    List<Host> hosts = new ArrayList<>();
    for (int i = 0; i < getHostList().size(); i++) {
        hosts.add((Host) getHostList().get(i));   }
    for (Host host : hosts) {
        int hostBwThreshold = (int) (host.getBw() * upperThreshold); // Calculate host's bandwidth
threshold
        int hostTraffic = 0;
        for (Vm vm : host.getVmList()) {
            hostTraffic += (int) vm.getBw();}
        int hostBw = Math.min(hostTraffic, Math.min(hostBwThreshold, maxLinkCapacity));
```

```
        totalBw += hostBw;  }
    return totalBw; }
public double calculateEnergyConsumption(List<BoHGoAInd> pareto, List<? extends Host> hostList) {
    double energyConsumption = 0.0;
    for (Host host : hostList) {
        if (host instanceof PowerHost) {
            PowerHost powerHost = (PowerHost) host;
            double utilization = powerHost.getUtilizationOfCpu();
            double maxPower = powerHost.getPower();
            double energy = maxPower * utilization;
            energyConsumption += energy/10;}}
    return energyConsumption;
}
public List<BoHGoAInd> getPareto() {
    List<BoHGoAInd> pareto = new ArrayList<>();
    if (pop != null) { // Check if pop is not null
        for (BoHGoAInd ind : pop) {
            if (ind.isPareto()) {
                pareto.add(ind);
            }}}
    return pareto; }
public class BoHGoAInd {
   private boolean isPareto;
    private double bandwidthConsumption = 0.0;
    private double energyConsumption = 0.0;
   private HashMap<Integer, Integer> vmToHostMap;
    // Define other properties and methods for the GAInd class according to your requirements
    public BoHGoAInd(BoHGoA policy) {
    }
public void setPareto(boolean isPareto) {
        this.isPareto = isPareto;
    }
    public BoHGoAInd(BoHGoAInd parent1, BoHGoAInd parent2) {
```

```
        Random random = new Random();
        boolean isPareto;
        // Perform crossover for the 'isPareto' property
        if (random.nextBoolean()) {
            isPareto = parent1.isPareto;
        } else {
            isPareto = parent2.isPareto;
        }       this.isPareto = isPareto;
    }
    public boolean getPareto() {
        return isPareto;
    }
public boolean isPareto() {
        return isPareto;
    }
   public int getHostIndex(int vmIndex) {
        return vmToHostMap.get(vmIndex);
    }
    public Map<String, Object> getMap() {
        Map<String, Object> map = new HashMap<>();
        map.put("isPareto", isPareto);
        return map;  }
public double getMaxLinkBw() {    return 100;
}
    private boolean dominates(BoHGoAInd other) {
    if (isPareto && !other.isPareto) {
        return true; // This BoHGoAInd dominates the other
    } else if (!isPareto && other.isPareto) {
        return false; // This BoHGoAInd is dominated by the other
    } else {                return bandwidthConsumption <= other.bandwidthConsumption &&
energyConsumption <= other.energyConsumption; // This BoHGoAInd dominates the other
    }}
    public void mutation() {
        double mutationProbability = 0.1; // Adjust this value as needed
```

```
        if (Math.random() < mutationProbability) {
            isPareto = !isPareto; // Flip the value of isPareto
        }}
    private double fitnessValue;
     public double getFitnessValue() {
        return fitnessValue;
     }
     public void calculateFitnessValue() {
        fitnessValue = bandwidthConsumption + energyConsumption;
     }    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("isPareto: ").append(isPareto).append("\n");
        return sb.toString();
     } }
public static void main(String[] args) {
        boolean enableOutput = true;
        boolean outputToFile = false;
        String inputFolder = "/Users/Ketema Deresa/Documents/workload/WL";
        String outputFolder = "";
        String workload = "workload_500"; // workload
        String vmAllocationPolicy = "BoHGoA";
        String vmSelectionPolicy = "mmt";
        String parameter = "1.2";
        Path inputFolderPath = Paths.get(inputFolder).toAbsolutePath();
        String inputFolderPathStr = inputFolderPath.toString();
        System.out.println(workload + ": Running " + vmAllocationPolicy + "-" + vmSelectionPolicy);
        System.out.println("Running BOHGOA");
         new MyRunner( enableOutput, outputToFile, inputFolderPathStr, outputFolder,
                workload,
                vmAllocationPolicy,
                vmSelectionPolicy,
                parameter);
} }
}
```