Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2024.050726

ARTICLE

Systematic Cloud-Based Optimization: Twin-Fold Moth Flame Algorithm for VM Deployment and Load-Balancing

Umer Nauman[1], Yuhong Zhang[2], Zhihui Li[3] and Tong Zhen[1,3,*]

[1]Information Science and Engineering College, Henan University of Technology, Zhengzhou, 450001, China

[2]School of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou, 450001, China

[3]Key Laboratory of Grain Information Processing and Control, Ministry of Education, Henan University of Technology, Zhengzhou, 450001, China

*Corresponding Author: Tong Zhen. Email: gm1013315793@gmail.com

Received: 15 February 2024 Accepted: 15 April 2024 Published: 11 July 2024

ABSTRACT

Cloud computing has gained significant recognition due to its ability to provide a broad range of online services and applications. Nevertheless, existing commercial cloud computing models demonstrate an appropriate design by concentrating computational assets, such as preservation and server infrastructure, in a limited number of large-scale worldwide data facilities. Optimizing the deployment of virtual machines (VMs) is crucial in this scenario to ensure system dependability, performance, and minimal latency. A significant barrier in the present scenario is the load distribution, particularly when striving for improved energy consumption in a hypothetical grid computing framework. This design employs load-balancing techniques to allocate different user workloads across several virtual machines. To address this challenge, we propose using the twin-fold moth flame technique, which serves as a very effective optimization technique. Developers intentionally designed the twin-fold moth flame method to consider various restrictions, including energy efficiency, lifespan analysis, and resource expenditures. It provides a thorough approach to evaluating total costs in the cloud computing environment. When assessing the efficacy of our suggested strategy, the study will analyze significant metrics such as energy efficiency, lifespan analysis, and resource expenditures. This investigation aims to enhance cloud computing techniques by developing a new optimization algorithm that considers multiple factors for effective virtual machine placement and load balancing. The proposed work demonstrates notable improvements of 12.15%, 10.68%, 8.70%, 13.29%, 18.46%, and 33.39% for 40 count data of nodes using the artificial bee colony-bat algorithm, ant colony optimization, crow search algorithm, krill herd, whale optimization genetic algorithm, and improved Lévy-based whale optimization algorithm, respectively.

KEYWORDS

Optimizing cloud computing; deployment of virtual machines; load-balancing; twin-fold moth flame algorithm; grid computing; computational resource distribution; data virtualization

1 Introduction

The technology sector has seen substantial changes with the introduction of cloud computing, which has gained widespread acceptance for its ability to provide a wide range of internet-based services and applications. Despite the widespread use of industrial cloud computing paradigms, which

involve consolidating processing power in huge-scale worldwide data centers, it remains important to focus on refining the placement of virtual machines (VMs). In this environment, system reliability, efficiency, and low latency are critical for satisfying the changing requirements of users and applications in the current technological age.

References [1,2] conducted research that laid the groundwork for cloud computing and its essential characteristics, making significant contributions throughout history. These analyses have also provided an understanding of the fundamental concepts that are directing this technological advancement's progress. These initial investigations emphasize the importance of adaptability, on-demand resource provision, and wide-ranging connectivity, thus laying the groundwork for further breakthroughs in the field.

As we move from the distant past to the present, the current difficulties in cloud computing become more apparent. The allocation of tasks, a crucial factor in maximizing energy efficiency, presents itself as a significant obstacle, particularly in the framework of theoretical grid computing architectures. Advanced optimization approaches are required to guarantee efficient operations by spreading different user workloads throughout several virtual machines (VMs) using load-balancing approaches. The suggested approach employs an advanced optimization technique, the twin-fold moth flame method, to address intricate limitations such as energy efficiency, lifespan analysis, and resource expenditures. The current study seeks to improve the optimization of cloud-based computing by building upon extensive research. The most recent study has focused on using complex optimization methods to enhance virtual machine (VM) placement and load distribution, with significant contributions from [3,4]. Expanding on this path, the work introduces a new method and evaluates its effectiveness using indicators such as energy efficiency, lifespan analysis, and resource expenditures.

The insights gained from this study have the potential to be useful for subsequent attempts at optimizing cloud computing. Progressive creativity is necessary due to the increasing need for effective resource use and ecological computing techniques. The proposed approach is a significant breakthrough in tackling these changing difficulties, providing possible enhancements compared to current techniques. In the following sections, we will thoroughly examine the intricacies of the twin-fold moth flame method, provide quantitative proof, and explore how the work relates to the continuing narrative of cloud computing innovation and adaptation. In the complex domain of cloud computing, determining the optimal location of virtual machines (VMs) becomes a crucial task. This method involves the intentional allocation of virtual machines to real servers on cloud platforms. The ever-changing nature of workloads and ongoing asset redistribution exacerbate the inherent intricacy of VM deployment. Furthermore, load distribution plays a critical role in carefully managing the allocation of transmitted data and computing tasks across numerous servers. We ensure efficient resource usage to prevent any one component from overloading [5,6].

The complexities related to VM deployment and load distribution are many. The technical operations face substantial problems because of a wide range of hardware arrangements, unexpected fluctuations in workloads, and the unwavering need for stable systems [7]. Anticipating and efficiently handling the need for resources becomes particularly difficult when faced with fluctuating tasks that change according to user requests and application needs. The difficulties of distinct cloud systems are amplified by the existence of various technology and software architectures [8].

In this scenario, effective load distribution is critical to preserve resource balance, prevent server under- or overutilization, and ensure affordability. The significance of load distribution comes in its ability to provide flexibility, specifically allowing vertical scalability, which enhances the management

of multiple tasks. Achieving this balance is crucial for maximizing the efficient use of resources, adaptability, and reliability, eventually leading to improved user experiences and energy conservation.

By addressing the complex issues that arise with safe VM deployments, the suggested changes to the simulation framework aim to establish a standard for maximizing the efficiency and cost-effectiveness of cloud-based applications. The goal is to design a standard that addresses these concerns thoroughly, optimizes resource consumption, improves adaptability and reliability, and results in an enhanced user experience and increased energy efficiency in the ever-evolving field of cloud computing.

Load-balancing strategies often prioritize the optimization of selecting possible destination hosts during computational phases, with host selection being the following phase. Although this strategy may quickly use up resources, it does not provide the best possible outcome when executing tasks [9,10]. Researchers commonly acknowledge the limitations of heuristic approaches, such as artificial bee colony (ABC) and ant colony optimization (ACO) [11]. We emphasize developing innovative and creative solutions to reduce energy consumption during the VM relocation process. Although some approaches may produce quick results in terms of resource use, they may not guarantee the highest level of efficacy in completing tasks on time. Algorithmic methods effectively acknowledge and tackle this difficulty by integrating tested frameworks such as ABC and ACO. However, because of the intricacy of the problem, it is required to look into workable and environmentally friendly methods to guarantee efficient energy use while inserting virtual machines [12–16]. It is crucial to search for these solutions to achieve an equilibrium between optimizing resources and increasing energy utilization. This is necessary to meet the changing requirements of energy-efficient and performance-focused cloud computing ecosystems.

The twin-fold moth flame algorithm (TF-MFA) is a proposed algorithm specifically developed to effectively adjust to the periodic variations in workload and utilization of resources in cloud-based computing settings. The ability of TF-MFA to handle a variety of workloads and resource fluctuations is one of its advantages. The program employs a dual strategy, drawing inspiration from the traits of flames and moths, to adaptively and efficiently allocate virtual machines by considering real-time fluctuations in demand and obtaining resources. TF-MFA takes into account many limitations, including energy efficiency, lifespan analysis, and resource expenditures. This allows it to make well-informed choices about assignments and allocating resources. TF-MFA's versatility enables it to efficiently handle unexpected increases or decreases in workload, distribute resources most efficiently, and guarantee reliability even when a cloud-based environment undergoes unpredictable shifts.

The TF-MFA method seeks to improve load distribution in cloud-based computing. This method is novel and specifically designed to handle several limitations. We rigorously evaluated it against other existing algorithms. A new optimization approach, the twin-fold moth flame (TFM) method, specifically addresses the complexities of VM deployment and load distribution in the ever-changing environment of cloud-based computing. In this context, efficient load distribution is critical.

The TFM algorithm presents a novel optimization method that takes inspiration from the dynamic properties of flames and moths. TFM uses a dual approach that combines convergence accuracy, similar to flame control, with daring possibilities evocative of moth flight trajectories. These strategies are based on the principles of biological imitation. TFM utilizes a distinctive combination of elements, including energy efficiency, lifespan analysis, and resource expenditures, to successfully navigate the challenging landscape of virtual machine deployment.

1.1 Author Contributions

Here's a summary of the work's contributions:

I. **Study conceptualization:** The authors initiated the study with the understanding that there is a need to enhance cloud computing simulations, specifically the existing corporate configurations that concentrate cognitive resources in several extensive data centers. The concept included the identification of key difficulties in VM placement, workload allocation, and energy consumption within an improbable grid computing paradigm.

II. **Algorithmic evolution:** The main achievement of this study is the creation of the twin-fold moth flame (TFM) algorithm, a new way to optimize things that were made to solve problems with load distribution and virtual machine deployment in cloud-based computing environments. The algorithm incorporates unique characteristics influenced by both flames and moths. The technique employs a dual strategy that combines precise maneuvering over flames with dynamic investigation, similar to moth flying patterns. This innovative approach represents a notable advancement in load distribution approaches for cloud-based computing.

III. **A systematic load distribution approach:** The authors developed an analytical framework to assess the suggested twin-fold moth flame procedure. This structure includes critical factors such as energy efficiency, lifespan analysis, and resource expenditures. This methodology offered a comprehensive method for evaluating total costs in the cloud-based computing setting, including an assessment of crucial indicators such as energy efficiency, lifespan analysis, and resource expenditure assessments.

IV. **Assessing real-world conditions and establishing analogies:** This study provides an in-depth evaluation of the proposed TFM algorithm compared to state-of-the-art benchmark methods. Considerations such as energy efficiency, lifespan analysis, and resource expenditures are part of this in-depth assessment, which confirms the algorithm's effectiveness and shows how it is superior to existing methods. This work adds to our understanding of load distribution approaches and emphasizes the practical benefits of the TFM algorithm for improving cloud-based computing environments.

This paper introduces an advanced simulation approach to optimize the deployment of VMs within the dynamically evolving cloud computing ecosystem. This study aims to systematically address the intricate issues associated with VM deployment, seeking to redefine prevailing standards for efficiency and dependability in cloud-based environments. This study's proposed novel simulation methodology significantly advances the field by improving and safeguarding the deployment of virtual machines in cloud-based systems. The overarching objective is to set new benchmarks for reliability and efficiency, thereby contributing to the continuous evolution of the rapidly changing landscape of cloud-based computing. This work represents a substantial step towards enhancing the understanding and practices of VM deployment within cloud environments, emphasizing the importance of innovative simulation techniques for achieving heightened levels of efficiency and dependability.

1.2 Paper Organization

The paper is organized into several parts, as shown below. Section 2 offers a comprehensive examination of cloud-based computing, focusing specifically on load distribution, efficiency of energy, and the deployment of VMs. Expanding on the previously mentioned basis, Section 3 provides more details on the architectural concepts and goals that govern the suggested VM deployment approach. This encompasses a distinct focus on the equitable allocation of workload, the dependability of the system, and the reduction of bottlenecks. Section 4 delves into a comprehensive examination of the

problem, considering factors like energy efficiency, lifespan analysis, and resource expenditures to properly evaluate VMs. Section 5 of the paper delves into a comprehensive examination of the twin-fold moth flame method, emphasizing its versatile optimization possibilities specifically designed for efficient load distribution. Section 6 provides a comprehensive assessment by comparing the proposed model with current approaches. Section 7 serves as a summary of the paper's results and offers suggestions for additional investigations. It presents a well-organized and thorough conclusion for the work.

2 Literature Review

2.1 Related Work

Resilient and reliable VM installation gained traction in 2023, as evidenced by a significant milestone portrayed in [17]. The authors used an advanced integer programming approach to revive virtual networks and optimize the selection of virtual computers for traffic scheduling research. This development improved the process of recovering from virtual machine breakdowns in the framework of cloud-based data centers. In 2023, a notable advancement, outlined in [18], established an aggregate selection-based method for VM deployment. This strategy aims to boost migration efficacy by incorporating aspects such as route expenses, communication overhead, and simulated heat. Although it has successfully reduced costs associated with moving and transmitting, some disadvantages have emerged, such as the burden on computer processing and the challenges of putting it into practice. In 2021, the upgraded bee colony strategy strengthened cloud safety by using computational security assessment to protect against breaches. However, obstacles such as computing expenses and the demand for extensive actual event assessment accompany the potentially lucrative elements of enhancing privacy and resilience of VM deployment.

In 2021, a novel IP Sec routing technology, referred to as [19], was introduced. This technology enables live VM migrations between network locations. To protect against such attacks, this solution employed an intentionally hidden and intricate tunneling channel. It also utilized onion routing to enhance the anonymity of VM transfers. Conversely, in 2022, reference [20] presented a customizable and power-efficient live VM migration solution, focusing on minimizing the electrical load of idle machines to save energy. Despite providing an entire solution across many modules, there were worries about the intricacy of installation and the need for significant assessment in distinct cloud-based computing environments. In 2020, researchers looked into flexible clustering methods [21], that provided a comprehensive overview of the setup and movement of VMs employing random inducement methods. Although there are substantial benefits in terms of energy effectiveness, security, and anonymity when transferring live VMs, there are also possible disadvantages, such as the difficulties involved in execution, limitations in resources, and the need for extensive testing in various cloud-based settings.

In 2022, reference [22] introduced a safety assessment approach that uses an availability paradigm to evaluate the migration of VMs in virtualized operating systems. This study set out to discover a middle ground between the dangers associated with VM transfer and the need for virtual machine management (VMM) recovery. The paper highlighted flaws such as man-in-the-middle and denial-of-service (DOS) attacks. In 2021, reference [23] proposed a multipurpose, secure VM placement technique. This technique implemented two genetic algorithms: one for non-dominated sorting and one for whale optimization. This strategy aimed to reduce the delay when interacting between VMs and optimize resource distribution in a way that saves on consumption. It prioritized the safe and quick implementation of user applications. Regardless, the system's strongest points include its ability to

evaluate privacy, plan for rehabilitation, and utilize encoded VMs, there are a few possible limitations. These include the requirement for a thorough evaluation of real-life scenarios, the computational burden associated with advanced algorithms, and the real challenges of implementing the system in various virtualizations.

The referenced study in [24] offered a comprehensive investigation of the technology and applications of cloud-based computing. This research presents an in-depth analysis of the evolution of distributed computing, with an emphasis on significant concepts such as service types (Infrastructure as a Service, Platform as a Service, and Software as a Service) and distribution techniques (public, private, and hybrid). The authors analyze the difficulties linked to the widespread use of cloud-based computing, including concerns such as privacy and security. This study is a fundamental resource for comprehending the complexities of cloud-based computing, delivering valuable findings for researchers, professionals, and educators alike. Despite this, the research thorough, may not provide up-to-the-minute information on developing technologies due to the constantly changing nature of the cloud-based computing industry [25,26].

2.2 Limitations and Establishing Grounds for a New Proposal

Addressing these issues and developing new ways to make dynamic fault-tolerance VM deployment and relocation in cloud-based data centers more reliable, adaptable, flexible, and secure.

- **Deployment and maintenance of resilient VMs:**

Limitations: While the sophisticated integer-based programming method improves VM recovery, it might lead to a higher computing burden.

Grounds for a new proposal: Create an alternative idea that prioritizes enhancing the computational performance of the integer-based programming technique to guarantee its feasible use in huge-scale cloud storage facilities.

- **VM Placement based on aggregate decision:**

Limitations: Although the collective selection-based approach is efficient, it has issues with computing load and practical deployment difficulties.

Grounds for a new proposal: Suggest a novel technique that tackles problems about computational effectiveness and offers a simplified strategy for real-world application in diverse cloud-based computing settings.

- **IP Sec routing mechanism for live VM voyage:**

Limitations: Despite offering a safe live VM voyage, there may be issues with installation complexity and resource limitations.

Grounds for a new proposal: Introduce a novel routing system that guarantees ease of deployment and resource-effective live VM voyages across various network locations.

- **Configurable and energy-efficient live VM voyage:**

Limitations: There are reservations about the complexity of the configuration process and the need for a thorough evaluation in different computing settings.

Grounds for a new proposal: Develop an advanced live VM voyage strategy that preserves flexible configuration and environmental sustainability while tackling implementation concerns and maintaining compatibility with different cloud-based settings.

- **Evaluating reliability during VM voyage:**

Limitations: The risk evaluation method may identify possible security vulnerabilities, but it may not provide remedies to address them.

Grounds for a new proposal: Propose an enhanced safety assessment approach that not only identifies security risks during virtual machine journeys but also integrates proactive actions to address these weaknesses.

- **Multi-functional, resilient VM deployment mechanism:**

Limitations: Constraints embrace the requirement for comprehensive assessment in real-world situations, the computing load linked to complex methods, and the real-world complexities in deploying the infrastructure across different simulations.

Grounds for a new proposal: Develop a comprehensive virtual machine (VM) deployment strategy that guarantees both security and multifunctionality, while simultaneously addressing the identified limitations via enhanced review procedures and computing breakthroughs.

- **An in-depth examination of cloud-based computing:**

Limitations: The research may lack current knowledge about innovations in the dynamic virtualization sector.

Grounds for a new proposal: Create a dynamic surveillance system that provides up-to-date information on the latest advancements in cloud-based technology, enabling participants to make well-informed decisions.

3 VM Deployment Model

The suggested VM deployment approach presents a sophisticated and all-encompassing organizational architecture aimed at optimizing the allocation of VMs on cloud-based computing platforms. The design's primary objective is to enhance performance and dependability, particularly concerning the equitable allocation of workloads. It handles the difficulty of resource utilization by proactively redistributing overloaded VMs to underused counterparts through a precisely constructed framework, thereby reaching a balance in the overall distribution of load. This method of organizing, based on over a decade of extensive study, focuses on ensuring the smooth synchronization of VM deployment across disparate systems. The system utilizes advanced algorithms and processes, such as continuous surveillance and assessment, to provide information for decision-making during VM migrations. In addition, the incorporation of robust safety measures features ensures the entire VM deployment mechanism's privacy and reliability. The proposed design is a complete and innovative approach that effectively aligns with the changing requirements of cloud-based computing platforms. Presented as a resilient strategy, it tackles the difficulties linked to the distribution of VMs in the always-evolving and progressing technological environment.

3.1 The Proposed Model

The conceptual ecosystem incorporates a task limitation and a cloud computing system model. It pertains to a framework that sets constraints on task limitation inside a cloud-based setup. The model has a specific set of regulations and enhances the implementation of cloud-based activities, ensuring effective resource utilization and optimal performance.

The examination carried out on the voyage of virtual machines (VMs) in cloud-based computing is outlined in Table 1. The table presents the advantages and disadvantages of each approach. These facts emphasize the extensive array of enhancements in the cloud VM voyage. Problems with throughput, efficiency, dependability, cost, and resource use are addressed via VM voyage solutions. The details attempt to explain the concerns and alternatives of the VM voyage by outlining each approach and emphasizing its advantages and disadvantages. Cloud-based computing academics and professionals may use this information to comprehend current strategies, explore their implementations, and devise novel VM voyage techniques.

Table 1: Outlining the background and problems of the current VM voyage

Ref.	Blueprint	Foreground	Issue
[17]	Utilizing integer programming to revive VMs and analyze traffic flow.	Enhanced VM disaster recovery.	Computational cost.
[18]	Selection-based VM placement that takes into account cumulative factors such as transportation charges, communication costs, and simulation heat.	Improved voyage efficacy.	Cognitive load and limitations in real-world use.
[19]	Enhanced approach for ensuring cloud stability and doing computation-based safety assessments in bee colonies.	Enhanced cloud security.	High cost of computation and the necessity for thorough real-world evaluation.
[20]	IP Sec channeling is used for facilitating live VM voyages, whereas onion routing is used to increase anonymity.	Protected real-time VM voyages.	Operational intricacy, and resource restrictions.
[21]	Flexible and energy-efficient real-time VM voyage, asset evaluation, task distribution, and optimization approach assessment.	Comprehensive real-time VM voyage approach.	Deployment difficulty, and cloud-based evaluation.
[22]	Technique for evaluating the security implications of VM voyage within the context of the on-demand paradigms.	Portability and privacy for VMs.	Discovering vulnerabilities and not taking preventative action.
[23]	An optimized evolutionary algorithm for whales, a multifaceted, safe VM deployment method, and a non-dominated algorithm for sorting.	Optimal deployment of resources and decreased collaboration delay.	Comprehensive assessment in practical settings, cognitive load, and obstacles to execution.

(Continued)

Table 1 (continued)

Ref.	Blueprint	Foreground	Issue
[24]	An in-depth analysis of cloud-based computing, with a focus on deployment mechanisms and service designs.	Essential reference for comprehending cloud-based computing.	Insufficient real-time data about new innovations.

Fig. 1 illustrates the suggested process for ensuring the security of VM voyages through efficient load distribution.

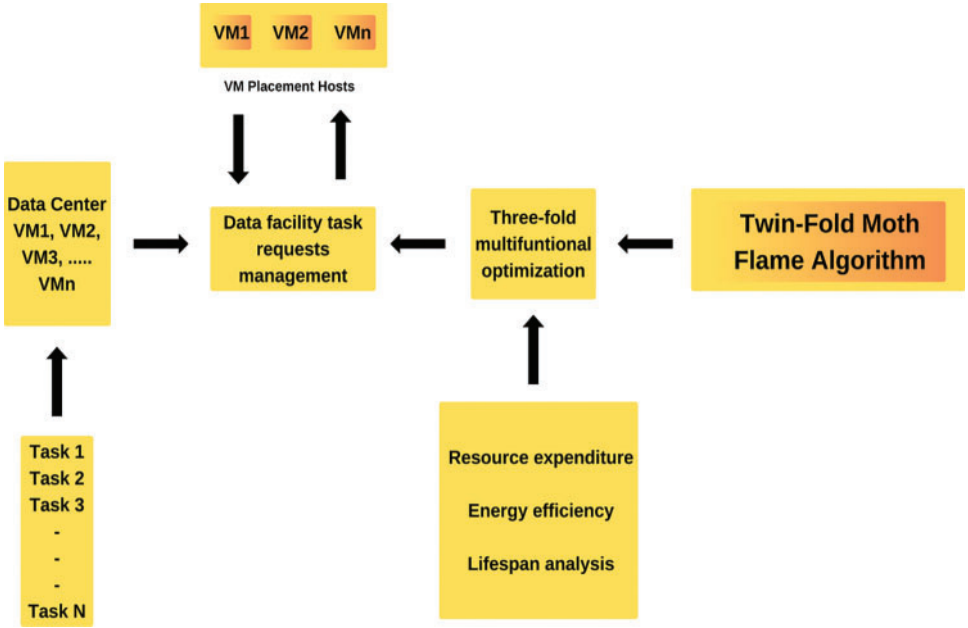


Figure 1: The proposed model

The effective load distribution procedure begins with actively tracking and collecting system efficiency measurements, followed by evaluating the data to understand the present load distribution. Next, we categorize workloads based on their attributes and importance, enabling informed decision-making via load balance algorithms. The subsequent transfer of jobs from one server to another, which frequently uses live voyage methods, seeks to ensure an equitable allocation. An ongoing information loop guarantees the ability to react dynamically to evolving system circumstances. Security measures, such as encryption and access restrictions, protect the transfer of information to ensure data safety. We monitor efficiency metrics to assess how load distribution and systems that adapt and adjust to changing workloads influence the transfer of information. Documentation and reporting are facilitated to maintain archives and facilitate subsequent refinements, representing a comprehensive approach to achieving optimal load distribution in computer systems.

3.2 Managing Heterogeneity in Cloud-Based Ecosystems: An Approach to the Twin-Fold Moth Flame Algorithm

The twin-fold moth flame algorithm (TF-MFA) was specifically developed to efficiently tackle the difficulties presented by the diversity in hardware setups and operational attributes across different data centers and cloud-based services. TF-MFA can track and evaluate workloads in real-time, allowing it to modify and enhance the allocation of VMs depending on the unique hardware settings and operational parameters of each data center or cloud provider. This dynamic method guarantees that VMs are assigned to assets that are most suitable for fulfilling their computational and operational needs, thereby reducing issues with compatibility and optimizing their overall effectiveness. TF-MFA utilizes sophisticated resource utilization algorithms that take into account aspects such as CPU speed, memory capacity, storage type, and network bandwidth. This allows it to effectively disperse workloads and maximize the use of resources. TF-MFA utilizes forecasting machine learning and analytics algorithms to foresee and solve potential bottlenecks or performance constraints in different circumstances. It preemptively identifies and mitigates these issues to ensure consistent reliability and efficiency across various data centers and cloud-based services. TF-MFA's complete methodology for managing diversity guarantees the most efficient use of resources, adaptability, and reliability in intricate cloud-based computing systems.

4 Problem Description

The primary objective of the problem description and assessment for monitoring virtual machines (VMs) with many components is to provide a clear and precise description of the unique challenges and complexities that arise when dealing with VMs that have a large number of components in a given scenario. This phase aims to ascertain and articulate the problems that the study attempts to investigate. The assignment involves conducting a comprehensive analysis of the challenges related to virtual machines (VMs), which have several components. This examination may include areas such as energy efficiency, lifespan analysis, and resource expenditures.

Eq. (1) quantitatively describes a complex problem with three distinct dimensions.

$$prp = min\{EE, LS, RE\} \tag{1}$$

Here, energy efficiency (EE), lifespan (LS), and resource expenditure (RE) are the three separate measures that relate to these three factors.

4.1 Energy Efficiency

Energy efficiency directly correlates with the physical machine (PM) assumption. The total power (P) used to run operations during a certain period determines data-related facility energy efficiency. This is determined by plugging the values of $Count_{CTU}$ (the number of computation time utilization) and EE (the total energy spent while all virtual machines VMs are running) into Eq. (2).

$$EE = 0.1\ log(Count_{CTU}) + EE + 0.1 \tag{2}$$

Algorithm 1 provides an overview of the elements that impact energy efficiency in a cloud-based computing atmosphere.

Algorithm 1: Pseudo-code of energy efficiency

Initiate

Input:

$Count_{CTU}$ refers to the quantity of time that is used for calculation.

EE: The cumulative energy consumption while all VMs are operational.

Computation:

Calculate_Energy_Efficiency ($Count_{CTU}$, EE):

Efficiency = EE divided by $Count_{CTU}$ Energy efficiency may be calculated by dividing the total energy use by the computation time utilization.

return of efficiency

EE = Calculate_Energy_Efficiency ($Count_{CTU}$, EE)

Compute the energy efficiency

Output:

Energy Efficiency: The calculated numerical representation of the energy efficiency.

Terminate

4.2 Lifespan Analysis

Making a schedule that reduces the lifespan—the time it takes to go from starting a work to finishing it—is a major goal in scheduling theory. To calculate the lifetime, we take the maximum wall time (W_{time}) in each block of tasks and multiply it by itself. Eq. (3) paves the way.

$$LS(tasks) = max\left(\sum_{B=1}^{N} W_{time}\right) \tag{3}$$

Algorithm 2 outlines the variables that influence the lifespan analysis in a cloud-based computing ecosystem.

Algorithm 2: Pseudo-code of lifespan analysis

Initiate

Input:

TotalTasks: The overall quantity of tasks included in the schedule.

Block $Wall_{time}$: A compilation of the duration of time for every block of jobs.

Computation:

Calculate_Lifespan(Block $Wall_{time}$): Max $Wall_{time}$ = Find the highest value among the specified block wall times

The lifespan is calculated by multiplying the maximum wall time by itself, resulting in the square of the maximum $Wall_{time}$.

Return this value of duration for existence

TotalLifespan is initialized to 0.

for i = 1 to TotalTasks: BlockLifespan = Calculate_Lifespan(Block $Wall_{time}$)

Determine the lifespan of each block

The variable TotalLifespan is incremented by the value of BlockLifespan, representing the accumulation of block lifespans.

Output:

TotalLifespan: The aggregate of the lives of all blocks, representing the total lifespan achieved.

Terminate

4.3 Resource Expenditure

Numerous factors influence the cost of migration, such as the memory capacity and update rate of each virtual machine (VM), the total number of VMs to relocate, the available network bandwidth for the journey, and the workload on both the starting and ending servers during the relocation process. From the provider's point of view, this feature enables the ability to adjust resource needs in real time, ensuring that the expenditure of outsourced capabilities matches the expected usage. Similarly, the service provider can prevent wasteful asset use by deactivating abandoned or underutilized services. Workload integration, also known as server consolidation, is a crucial component in effectively optimizing assets for cloud-based computing. In this situation, the service provider typically analyzes the magnitude and allocation of inbound demands in the cloud ecosystem to determine the number of servers it should provide.

According to Eq. (4), resource expenditure is the total of the costs (RE) spent by assets in completing a job, where A represents the total number of assets.

$$RE(A) = \sum RE(A_1 + \ldots + A_n) \tag{4}$$

Algorithm 3 outlines the factors that influence resource allocation in a cloud-based computing ecosystem.

Algorithm 3: Pseudo-code of resource expenditure analysis

Initiate

Input:

MemoryCapacity: The amount of memory that each VM can hold VM_i.

The MemoryUpdateRate represents the frequency at which each VM updates its stored data.

TotalVMs refers to the whole quantity of VMs that are scheduled for migration.

NetworkBandwidth: The amount of network capacity that is accessible throughout the process of the voyage.

Workload_StartingServer: The amount of work being performed on the server at the beginning of the voyage process.

Computation:

Expenditure_Per_VM is determined by the function

(MemoryCapacity, MemoryUpdateRate), which calculates the expense per VM relying on memory variables.

The total expenditure is equal to the sum of all values in the series, starting at $i = 1$ and ending at the last value. The value of TotalVMs. Estimate the total expense for each VM using the Expenditure_Per_VM function.

The formula for calculating the Resource Expenditure is the sum of the Total Expenditure and the function g, which takes into account the factors of Network Bandwidth, Workload Starting Server, and Workload Ending Server.

Output:

ResourceExpenditure: The total amount of resources used for the migrating process.

Terminate

5 Methodology

5.1 Specifics of Algorithm Execution

This section contains comprehensive information on the implementation of the algorithms used in our research. Below, we provide the precise specifications of the algorithm, including parameter configurations, starting procedures, and termination conditions for each method.

5.1.1 Configuration of Parameters

TF-MFA involves the arrangement of parameters that are essential for its function. The convergence threshold factor determines the amount of convergence necessary for the algorithm to stop its optimization procedure, guaranteeing the attainment of the specified level of optimization. Furthermore, the maximum repetitions option establishes a ceiling on the number of iterations or generations that the algorithm will go through, prohibiting unlimited computation and enabling regulated optimization. It is obligatory to keep the parameters for inquiry and adoption in check so that the algorithm can find new ways to do things while also using tried-and-true methods, which affects how well it works during optimization. TF-MFA includes adaptable approaches that flexibly modify algorithm parameters in real-time, depending on the current optimal phase or issue features. This improves its flexibility and efficacy in many settings.

5.1.2 Process of Initialization

The initialization phases of the TF-MFA consist of two crucial components. Firstly, the process of generating initial responses entails developing the first collection of methods or individuals that the algorithm will maximize. This process could involve arbitrarily creating solutions through randomized initialization, or implementing established methods based on unique aspects of the issue. Additionally, the process of initializing the inner parameters includes establishing parameters and entities inside the algorithm, such as determining the size of the population, the rate of mutations, and memory distribution. Ensuring the proper baseline of these parameters is crucial to guaranteeing that the algorithm begins with an appropriate setup for optimization, thereby establishing the groundwork for successful and productive divergence toward optimum solutions.

5.1.3 Criteria for Termination

The convergence threshold in the TF-MFA is a crucial parameter that determines the proximity between consecutive iterations, reflecting the algorithm's advancement in maximizing the desired parameter. This threshold serves as a benchmark to ascertain whether the optimization phase has achieved a suitable degree of convergence, leading the algorithm to stop. In addition, the TF-MFA includes a termination requirement in the form of an ultimate repetition restriction that keeps the algorithm from continuing endlessly. The purpose of this restriction is to serve as a maintainer, preventing an overabundance of computing resources and execution, thereby facilitating effective and regulated optimization operations.

5.2 The Proposed Model

The moth, likened to a group of butterflies, exhibits a remarkable display of aesthetic appeal. The moon's distance from earth affects the object's conventional movement, which tends to align perpendicular to straight paths. Here, we provide an in-depth discussion of the approaches used in the twin-fold moth flame (TFM) framework.

Layer 1: Population initialization

The method starts by establishing two populations: moths, which represent $Moth_m$, and $Flame_w$, which represent flames. $Moth_m$ involves the incorporation of the m^{th} moth inside a group of flames, where $Count_{flames}$ represents the overall population of flames and q indicates the typical number of iterations. Random variables $ran1$ and $ran2$ add stochasticity to the optimization process.

Layer 2: Evaluation of the fitness function

The fitness function, defined by Eq. (1), evaluates the fitness of each moth and flame in the population. The system evaluates the appropriateness by considering particular optimization goals and guides the following optimization actions.

Layer 3: Optimization based on phases

The optimization process progresses in several phases.

Phase 1: Longitudinal synchronization ($q <= Count_{flames}$)

I. The standard technique of moth and flame optimization modifies the locations of the moth and flame if the value of $ran1 < 0.5$. We achieve longitudinal synchronization and logarithmic spiraling locks by applying Eqs. (5) and (6).

$$Moth_m = S(Moth_m, Flame_w) \tag{5}$$

$$S(Moth_m, Flame_w) = R_m e^{bt}.\cos(2_\pi t) = Flame_w \tag{6}$$

II. We use Eqs. (9) and (10) if the value of $ran1 > 0.5$ via calculating the distance between flames and moths.

$$R_m = Moth_m - Flame_w \tag{7}$$

$$Count_{flames} = round\left(\frac{N - q \times N - 1}{T}\right) \tag{8}$$

III. If the value of ran > 0.5, we calculate the radius by using the flame radius and Eqs. (9) and (10) for the moth.

$$rad = beatps(Flame) - mothps \tag{9}$$

$$mothps = rad + beatps(Flame) \tag{10}$$

Phase 2: Iterative enhancement (q > $Count_{flames}$)

IV. If the value of $ran1 < 0.5$, the program utilizes the moth flame algorithm (MFA) to adjust the placement of moths and flames using Eqs. (8)–(11).

V. In contrast, when $ran2 > 0.5$, modifications are applied to the positions of iterations using Eq. (11) to calculate the distance between moths and flames.

$$rad = beatps(Flame) - class(\text{pop}) \tag{11}$$

Layer 4: Transitions and the likelihood

The transitions and the likelihood of the algorithm are critical factors in determining the optimization results. Eq. (8) calculates the total number of flames across iterations (N) and the recurrence count (ln), which affect the method's versatility and performance capabilities.

Layer 5: Thorough architecture

Algorithm 4 presents a detailed framework that demonstrates the key components that affect the flexibility and effectiveness of the TFM in improving resource allocation, load balancing, and energy efficiency in cloud-based computing settings. This multifaceted strategy guarantees durability, expandability, and improved efficiency while dealing with intricate optimization problems.

Algorithm 4: Pseudo-code of generalized architecture

Initiate

The calculation of the beginning population, denoted as pop, for the $Flame_w$ and the $Moth_m$ is performed.

Set the variables $Count_{flames}$, q, $ran1$, and $ran2$ to their initial values.

Assess the level of fitness based on Eq. (1).

for $qmax > q$

if the criteria for dismissal is not met:

if 1 ($q <= Count_{flames}$)

if 2 ($ran1 < 0.5$)

Eq. (5) applies transverse synchronization to update the positioning of the *flame* about the *moth*.

Eq. (6) is utilized to alter the positioning of the query agent in the form of a logarithmic spiral manner.

Eq. (8) may be utilized to calculate the quantity of flames.

else 2

Utilizing Eqs. (9) and (10) to ascertain the radius within the *moth* and the *flame*:

end if 2

else if 1 ($q > Count_{flames}$)

if 3 ($ran2 < 0.5$)

Eq. (5) applies transverse synchronization to update the positioning of the *flame* about the *moth*.

Eq. (6) is utilized to alter the positioning of the query agent in the form of a logarithmic spiral manner.

Eq. (8) may be utilized to calculate the quantity of flames.

else 3

Utilizing Eqs. (10) and (11) to ascertain the radius within the *moth* and the *flame*:

end if 3

end if 1

end

Terminate

5.3 Constraints of the Suggested Framework

5.3.1 Spatial Constraints

The memory resources allocated for storing the number of moths and flames, as well as any necessary extra parameters for computational operations and modifications, mostly determine the framework's spatial needs. The storage requirements for each moth and flame, together with any other relevant data, influence the total geographical footprint of the program. In addition, the spatial intricacy closely relates to the magnitude of the issue at hand, including variables like the overall quantity of moths and flames involved. The space complexity of the computing procedure is denoted as $O(N)$, where N is the overall count of moths and flames in the population. Crucially, it is assumed that the storage requirements for each moth and flame stay constant during the program's implementation.

Using this estimate, we can analyze the program's success in controlling and exploiting memory assets by obtaining scaled knowledge of the spatial limitations.

5.3.2 Time Constraints

The temporal constraints of the twin-fold moth flame method are contingent upon the number of repetitions, represented as (q), as well as the magnitudes of the moth and flame populations. In the execution of each loop, we perform several calculations and adjustments on both moths and flames. The particular equations used for radius computations and location modifications influence the level of detail. Now, let us analyze the components:

Repetitions (q): The number of iterations has a direct effect on the overall time complexity, which in turn affects the entire execution time by increasing it. The length of time is represented by the notation $O(q)$, which signifies a linear correlation with the number of repetitions.

Moth and flame populations: The processing burden is influenced by the magnitude of the moth and flame populations. Increased population sizes often result in a greater number of computations. The algorithm's structure often determines if the effect on processing time is uniform or linear.

Equations constraints: Mathematical expressions that represent the relationship between different variables or quantities. The intricate nature of the equations used for radius estimates and location updates directly affects the total time complexity. If these equations require complex calculations, it may impact the algorithm's effectiveness.

5.3.3 Regulatory Compliance Constraints

When using the suggested TF-MFA technique in cloud-based computing eco settings, it is crucial to consider several regulatory and compliance constraints. The biggest concern pertains to confidentiality requirements, such as the General Data Protection Regulation (GDPR) in Europe or the Health Insurance Portability and Accountability Act (HIPAA) in the United States. These rules enforce strict criteria for the management and preservation of confidential data, such as personal and healthcare information. It is crucial to adhere to these standards to prevent legal repercussions.

Furthermore, depending on the nature of the data and the industry using cloud offerings, certain industry-specific legislation and norms may also apply. Financial firms are required to adhere to the Payment Card Industry Data Security Standard (PCI DSS), and government organizations might be required to follow the Federal Risk and Authorization Management Program (FedRAMP).

When deploying cloud services internationally, privacy and governance become significant factors to consider. Several nations implement data globalization rules, mandating the storage of data within their borders. This may create difficulties for both service providers and consumers.

Additionally, safety certifications and assessments play a vital role in showcasing adherence to company guidelines and optimal methodologies. To successfully use the TF-MFA in cloud-based environments, it is critical to carefully address regulatory and compliance factors to maintain data protection, confidentiality, and legality.

5.4 Robustness of the Proposed Model

The analysis of the twin-fold moth flame algorithm's (TF-MFA) model's robustness incorporates several factors:

Stability evaluation: Stability is a crucial element of the TF-MFA model's resilience, demonstrating its capacity to sustain consistent performances and behaviors across different circumstances. To assess equilibrium, we analyze the capacity of the TF-MFA to handle variations in workloads, resource requirements, and changing environmental conditions within cloud-based computing settings. A resilient TF-MFA model should have the ability to withstand abrupt increases or decreases in workload, guaranteeing reliable and foreseeable system efficiency. Stability is essential for preserving customer pleasure, reducing interruptions, and effectively maximizing resource consumption.

Fault tolerance mechanisms: These are an important feature of TF-MFA's resilience. Fault tolerance corresponds to the model's capacity to identify, separate, and restore from faults, mistakes, or interruptions that may arise during operations. The TF-MFA model has resilient fault tolerance techniques that enable it to operate consistently and provide uninterrupted services, even in the face of hardware breakdowns, software problems, or network outages, therefore minimizing interruptions and information loss. Ensuring this element is crucial for sustaining uninterrupted operations and fulfilling service level agreements (SLAs) in cloud settings.

Scalability assessment: The evaluation of the TF-MFA model's robustness heavily relies on its scalability. Scalability refers to the model's ability to manage rising computing requirements, heavier workloads, and increased system complexity while maintaining optimal efficiency and effectiveness. An efficient and resilient TF-MFA should be able to easily adjust its asset allocation, optimize the allocation of tasks, and conform to fluctuating workload behaviors. Scalability guarantees that the framework can adapt to expansion, effectively manage high levels of demand, and sustain optimal efficiency as the infrastructure progresses.

5.5 Elapsed Wall Real-Time

Elapsed wall real-time is the precise period between the start and end of a computer program. The term "duration" refers to the time interval between the completion and the initiation of a project. Elapsed real-time, unlike CTU time, measures the whole length of a procedure, taking into account both active engagement with a particular job and times of idleness or requesting system assets. Discrepancies between the time it takes for a task to complete and the time it takes for the CTU to process the task (CTU time) may occur due to reasons that are particular to the computer's design and rely on its execution, such as deliberate postponements built into the task. Eq. (12) specifies the "normrnd function" to calculate the real-time for VMs, while the predetermined real-time for PMs stays fixed. This function incorporates statistical variability to replicate situations in reality.

$$Wall_{time} = normrnd(\mu, \delta) \tag{12}$$

This function generates a random integer following a distribution that is normal, using the mean variable μ and the standard deviation variable δ sigma. In this context, δ is set at a constant value of 0.1, while μ represents the elapsed wall real-time of PM.

5.6 Database Layout

The database layout for load balancing the placement of virtual machines (VMs) entails retrieving physical machine (PM) data from the dataset available at https://www.kaggle.com/datasets/discdiver/clouds/code (accessed on December 04, 2023). This dataset is crucial for creating the VM data necessary for load balancing. The PM's database contains the many criteria outlined in Table 2, which constitute the database. In addition, we generate virtual machine databases using genuine hardware datasets, ensuring the authenticity and precision of the data connected to virtual machines. The

database structure integrates the PM functionalities to provide a thorough and intricate depiction of the framework. The database construction technique meticulously considers all limitations and restrictions to provide optimal efficiency and effectiveness in VM deployment and load-balancing measures.

Table 2: A breakdown of the factors included in the database architecture to place virtual machines and balance their workloads

Parameter	Description
PM ID	Database device ID for each PM
CPU cores	No. of CPUs at hand for each PM
RAM (Gigabytes)	Extent of RAM at hand for each PM
Storage (Gigabytes)	Storage slots of each PM
Bandwidth (Mbps)	Max bandwidth at hand for each PM
VM ID	Database device ID for each VM
CPU cores	No. of CPUs at hand for each VM
RAM (Gigabytes)	Extent of RAM at hand for each VM
Storage (Gigabytes)	Storage slots of each VM
Bandwidth (Mbps)	Max bandwidth at hand for each VM

The table presents a systematic summary of the factors used in the database design for VM placement and load balancing. It outlines the properties of the PMs and the associated needs of the VMs.

5.7 Criteria for Improving Cloud-Based Computing's Efficacy and Long-Term Viability

In cloud-based computing settings, the following are important factors that impact effectiveness and long-term viability:

Energy efficiency criteria: To reduce energy usage, consider evaluating energy-saving techniques such as workload integration, adaptive resource scheduling, and optimizing power consumption parameters for better placement of cloud-based ecosystems.

Lifespan analysis criteria: Utilizing lifespan analysis techniques to evaluate the durability and resilience of hardware components, such as servers, storage devices, and networks, to ensure optimal use of resources over an extended duration.

Resource utilization criteria: This entails identifying underutilized assets, redistributing them, balancing workloads, and implementing flexible system designs to maximize efficiency and reduce waste.

Cost management criteria: Implementing cost-effective methods such as pay-as-you-go scenarios, resource pooling, and effective distribution of resources based on workload needs to reduce operating costs and improve financial viability.

Environmental impact assessment: These criteria involve conducting assessments to examine the ecological impacts of cloud-based computing activities. This includes analyzing carbon emissions, auditing energy use, and implementing environmentally friendly procedures to support longevity.

Performance monitoring and optimization: Consistently analyzing system performance parameters, such as reaction times, productivity, and delay, along with taking preventative steps to improve the system's overall effectiveness and user satisfaction.

By prioritizing these essential factors, cloud-based computing settings may attain heightened achievement, energy efficacy, and long-term viability, resulting in enhanced operational efficiency and reduced carbon emissions.

5.8 Mitigating Resource Contention and Bottlenecks in Virtual Machine Placement

The proposed approach aims to mitigate possible problems, such as resource congestion and bottlenecks in virtual machine (VM) deployment, by using multiple techniques:

Randomized resource allocation: Depending on workload needs, the method places VMs on servers with sufficient available resources to avoid conflict. The algorithm reduces the likelihood of resource disagreements by continually evaluating resource use and making adjustments to arrangements as needed.

Load-balancing: TF-MFA utilizes load-balancing methods to evenly allocate computational jobs among the assets that are currently available. This aids in alleviating bottlenecks by preventing any one server from being overwhelmed with high workloads, thereby maximizing the usage of resources and enhancing responsiveness.

Predictive forecasting: The technique uses forecasting to anticipate possible instances of resource conflict and proactively reallocates VMs to prevent bottlenecks. TF-MFA uses existing data and workload trends to make intelligent decisions to improve VM deployment and ensure the system's long-term viability.

Priority-based deployment: TF-MFA places VMs by considering their level of urgency and resource requirements. We intentionally allocate high-priority VMs to servers with abundant resources. We do this to ensure continuous operation and mitigate the adverse impact of resource competition on critical workloads.

Monitoring on-the-go: The program constantly checks system performance parameters in real-time, including CPU use, memory consumption, and network connections. The program promptly modifies the deployment of VMs to ease resource congestion or bottlenecks and maintain optimal efficiency.

TF-MFA's comprehensive strategy for evolving resource distribution, load balancing, forecasting, priority-based deployment, and continual tracking effectively addresses and reduces the occurrence of resource contention and bottlenecks in the scheduling of VMs in cloud-based computing settings.

5.9 Ensuring Security and Privacy in Cloud Computing Facilities with Multiple Tenants

The TF-MFA is a technique that handles security and privacy concerns in cloud-based computing, with a special focus on multi-tenant systems. The system employs strong authentication and encryption techniques to ensure the security of sensitive information during distribution and preservation. It utilizes state-of-the-art protocols and norms to maintain anonymity and validity. Role-based access control (RBAC) and other accessibility controls oversee and regulate access to resources and data, ensuring that only registered users and applications have the necessary rights. Privacy-enhancing methods, such as anonymity of data and encryption for confidential information, successfully avert the unlawful exposure of content. TF-MFA incorporates intrusion detection and avoidance technologies

to promptly mitigate real-time threats, enhancing overall safety against illicit activity. These comprehensive safeguards give priority to confidentiality and safety, successfully tackling the complex difficulties of cloud-based computing in collaborative settings.

5.10 Strengthening Fault Tolerance and Reliability

TF-MFA implements resilience measures on both the software and hardware platforms to ensure system reliability and fail over capabilities. This encompasses redundant servers, storage infrastructure, and network routes to minimize the effects of hardware malfunctions on the overall infrastructure. Fail-over techniques are implemented to smoothly transition to redundant modules in the event of main resource failures.

The method employs load-balancing strategies to achieve an equitable allocation of workloads among the resources that are accessible. This not only improves the efficiency of resource use but also increases the ability to handle faults by preventing individual components from being overloaded. TF-MFA provides continuous surveillance and detection of abnormalities to continually check the general condition of the entire network. The system utilizes systems for anomaly detection to promptly identify possible concerns. This proactive approach enables timely action to tackle emergent issues before they worsen.

Virtualization is used to confine errors inside individual VM entities. TF-MFA mitigates failures by isolating them, ensuring that they do not impact the whole system, thereby enhancing overall dependability. The algorithm incorporates automatic recovery protocols and catastrophe mitigation systems. This includes the computerized procedure of backing up and restoring data, as well as the failback processes that are implemented to quickly restore routine tasks following a disturbance. TF-MFA utilizes continuous surveillance and adaptive procedures. It adapts the distribution of resources and redirects traffic in response to current circumstances, guaranteeing operational equilibrium and adaptability in changing situations.

These solutions jointly enhance the dependability and resilience of TF-MFA, enabling it to effectively manage hardware failures, network interruptions, and other possible challenges in cloud-based settings.

5.11 Optimization Trade-Offs

The twin-fold moth flame algorithm (TF-MFA) utilizes a comprehensive strategy to efficiently handle the trade-offs between crucial optimization goals in cloud-based computing settings, including energy efficiency, performance, and cost-effectiveness.

TF-MFA utilizes adaptive resource distribution algorithms to optimize energy efficiency by considering the energy expenditure characteristics of various workloads. TF-MFA optimizes energy efficiency by allocating assets depending on workload features, minimizing energy waste. TF-MFA utilizes prediction analysis as well as cognitive learning techniques to anticipate fluctuations in workload needs, hence enhancing performance optimization. This preemptive methodology allows the algorithm to predict changes in demands for resources and adapt the allocation of resources appropriately. TF-MFA increases throughput by reducing the number of times resources are overstocked or not used enough. This keeps the system running at its most efficient level.

In addition, TF-MFA incorporates cost-conscious optimization approaches that take into account resource cost structures and workload objectives. The method enhances resource consumption by considering the cost consequences associated with resource allocations, resulting in minimized costs while

efficiently achieving performance objectives. This cost-conscious method ensures the distribution of resources to maximize cost-effectiveness while upholding performance standards.

TF-MFA's modeling, cost-aware optimization, and adaptive scheduling of resources work well together to find the best balance between performance, energy efficiency, and cost-effectiveness in cloud-based settings. The thoroughness of TF-MFA makes it highly suitable for dealing with the intricate trade-offs involved in enhancing cloud-based ecosystems.

5.12 Findings from the Longitudinal Investigation

The longitudinal study yielded useful information about the long-term effectiveness, flexibility, and sustainability of TF-MFA in fluctuating cloud-based computing systems. During the trial period, which lasted for many weeks to months, TF-MFA consistently showed increases in important indicators of performance like energy economy, reaction time, productivity, and efficiency. TF-MFA consistently maintained these enhancements over some time, demonstrating its effectiveness in managing and developing workload requirements, adapting to fluctuating system architectures, and appropriately allocating resources for scaling. Additionally, the study highlighted TF-MFA's ability to adapt to fast changes, its strength in maintaining system stability and dependability, and its ability to grow to accommodate the rise of cloud-based resources.

5.13 Implementation of the TF-MFA in Real World

The following are some illustrations of how the TF-MFA has been implemented in the real world:

Algorithm tuning: Getting the algorithm settings precisely right for a certain cloud-based computing ecosystem's optimal efficiency is one of the main difficulties. Effectively adjusting the TF-MFA's multitude of parameters and configurations is necessary to strike an equilibrium between optimization efficacy and computation.

Computing requirements: When working with large-scale cloud-based platforms that include several virtual machines (VMs) and intricate workload motifs, the TF-MFA technique may need a substantial amount of computing resources, including processing capacity and memory. It is essential to guarantee sustainability and the effective allocation of resources.

Parameter susceptibility: In practical applications, the TF-MFA model's susceptibility to variations in the parameter values may cause difficulties. Resilient tuning procedures are necessary because small differences in workload characteristics, resource availability, or optimization aims might result in varied optimization results.

Sensitivity of optimization goals: Concurrent optimization of multiple goals, including energy efficiency, reaction time, cost-effectiveness, and resource usage, complicates the actual application of TF-MFA. It may be difficult to strike a balance between these goals and preserving system effectiveness and integrity; this may call for sophisticated optimization strategies.

Real-time adaptability: It is practically difficult when it comes to dynamically adjusting the TF-MFA algorithm for modifications in cloud-based settings, such as changing workloads, requirements for resources, or system architectures. The algorithm's capacity to rapidly and efficiently adapt to these modifications is vital for its practical use.

Validation and testing: Validation and testing of the TF-MFA method in various cloud-based environments are crucial, but they may be time-consuming and require a significant amount of resources. Before deploying the algorithm in real-world settings, it is crucial to carry out an extensive evaluation to guarantee its dependability, productivity, and robustness.

To overcome these challenges, it is necessary to employ an assortment of algorithmic improvements, resource optimization approaches, precise parameter tuning methods, real-time modification mechanisms, smooth integration procedures, and rigorous endorsement procedures. By doing so, we can effectively utilize the TF-MFA method in cloud-based applications and achieve its practical advantages.

6 Experiments

This section is critical to the overall study because it provides a thorough examination of the empirical verification of the proposed twin-fold moth flame method. Investigators conduct tests to evaluate the algorithm's performance, confirm its efficacy, and provide concrete proof of its suitability for tackling the specified issue. Performing trials facilitates a systematic evaluation, which includes the statistical examination of indicators such as energy efficiency, lifespan, and resource expenditures. The findings from these trials not only confirm the algorithm's effectiveness but also provide valuable observations on its performance in varying circumstances. Furthermore, trials serve as a foundation for comparing the suggested approach with current approaches, enabling a thorough comprehension of its positive aspects and possible areas for expansion. In general, the experimental part improves the study's trustworthiness by providing actual evidence of the algorithm's operational usefulness and role in tackling the issues described in the article.

The work focused on thoroughly validating and testing the twin-fold moth flame algorithm (TF-MFA) in actual cloud-based computing systems. We conducted the examinations using a data set of 40 nodes, which allowed for extensive comparisons and evaluations. The results of these experiments showcased the algorithm's ability to effectively distribute workloads, confirming its efficacy in real-world cloud-based computing situations.

The implementation of TF-MFA has significant consequences for the capacity to handle the increased workload and efficiently use resources in many kinds of cloud-based placements, such as public, private, and hybrid systems. The TF-MFA, which combines accuracy and exploration, improves sustainability by efficiently spreading computational workloads among VMs, irrespective of the kind of cloud-based distribution. This feature enables enterprises to adjust their resources flexibly to fluctuating needs, guaranteeing the most efficient use of assets and cost-efficiency.

TF-MFA's capacity to adjust to changing workloads and improve VM allocation enhances resource efficiency in cloud-based implementations, mitigating the likelihood of underused or underused resources. This flexibility guarantees that enterprises may fulfill expectations for performance while lowering operating expenses in an integrated cloud architecture.

TF-MFA's robust load balancing and VM deployment capabilities allow enterprises to efficiently allocate resources inside their allocated architecture in private cloud-based settings. TF-MFA enables sustainability by efficiently distributing workloads and properly maintaining resources, allowing for higher requirements without sacrificing throughput or incurring excessive costs.

TF-MFA's adaptability is advantageous for cloud-based hybrid placements, as it guarantees smooth workload management between public and private cloud-based resources. Enterprises may benefit from each distribution model's benefits while ensuring satisfactory performance, adaptability, and resource consumption throughout the hybrid environment due to this adaptability.

6.1 Modeling Procedure

We conducted a series of approaches to confirm the effectiveness and resilience of the twin-fold moth flame algorithm (TF-MFA) in our study. Initially, we used simulation-based evaluation

techniques, using frameworks such as MATLAB, to simulate a wide range of workload situations. This allowed us to accurately assess important performance indicators, like reaction time and resource consumption. In addition, we performed comparison assessments against recognized methods to evaluate the effectiveness and adaptability of TF-MFA. TF-MFA was further verified by a realistic-world assessment, which included analyzing its performance in real cloud-based settings. This testing included monitoring the dependability of the system and the judgments made about the placement of VMs. TF-MFA's robustness was assessed using rigorous testing, which included subjecting it to large workloads and fault situations. Additionally, its efficiency was quantitatively validated across various situations and compared to other algorithms using statistical evaluation. These examinations jointly enabled a thorough evaluation of TF-MFA's efficiency in load balancing and optimization of resources, showcasing its appropriateness for managing changing workloads in cloud-based settings.

The evaluation considers updates to VMs as well as blocks of data and incorporates a comprehensive understanding of the features of both types of entities. To precisely characterize the block's characteristics, the evaluation uses the fluctuation range [20, 25, 30, 35, 40] to precisely characterize the characteristics of the block. This spectrum represents the range of variation that exists within the block variables. On the other hand, each PM is associated with a spectrum of VMs specified as [10–50], illustrating the variety of VM features that correlate to each PM. For each of the tasks labeled 1–4, participants must constantly rearrange the ingredients, with a primary focus on substituting blocks from the provided array. Flexibility and variety are fundamental characteristics of both blocks and VMs, and this adaptive reorganization showcases both of these qualities. The assessment takes a methodical approach to investigate a wide range of modifications, therefore highlighting the versatility of the framework and setting the groundwork for a situation that is defined by sensitivity and adaptation.

The database that was studied allows for the identification of several kinds of blocks, along with the attributes that are associated with each category:

Category 1: Comprises blocks of dimensions [4, 4, 6, 7], comprising 20 blocks, denoted as [T4, T1, T3, T2].

Category 2: Consists of blocks with dimensions [5, 6, 6, 8], comprising 25 blocks, represented as [T4, T3, T1, T2].

Category 3: Includes blocks of dimensions [6, 7, 7, 10], adding up to an aggregate of 30 blocks, represented as [T1, T3, T4, T2].

Category 4: Consists of 35 blocks of sizes 7, 8, 9, and 11. These blocks are referred to as T1, T3, T4, and T2.

Category 5: Consists of blocks of dimensions [5, 10, 10, 15], comprising 40 blocks, represented as [T1, T2, T3, T4].

These kinds classify blocks according to their size distributions, providing an organized depiction for additional investigation and evaluation.

6.2 Assessment of Energy Efficiency

Fig. 2 illustrates the energy efficiency of the suggested technique in a visual representation. We compare it to current methodologies for different numbers of VMs and blocks. Fig. 2a displays the energy efficiency of the proposed strategy compared to current techniques. The emphasis is on a situation where a PM allocates 10 VMs for load distribution. Fig. 2b demonstrates the energy use trends for 20 VMs, highlighting the minimum energy consumption of the suggested study compared to known methodologies. Significantly, in Fig. 2c, the proposed research distinguishes itself by exhibiting

the most minimal energy use in comparison to the current methodologies. Moreover, Fig. 2d provides a thorough perspective on energy efficiency measurements using 40 VMs, emphasizing the constant benefits of the proposed method in decreasing energy consumption across different VM quantities. The suggested work exceeds ABC+BA, ACO, CSA, KH, WOGA, and ILWOA by 1.52%, 4.11%, 6.57%, 5.11%, 2.57%, and 3.09%, respectively, while considering 30 sets of blocks. The thorough examination shown in Table 3 provides additional data to corroborate these results, demonstrating that the suggested modification consistently results in reduced energy usage when appropriate VM deployments are made for a PM. Although there may be some cases of slightly increased energy usage in particular circumstances, the overall efficacy of the suggested approach continues to be praiseworthy when compared to other methods.

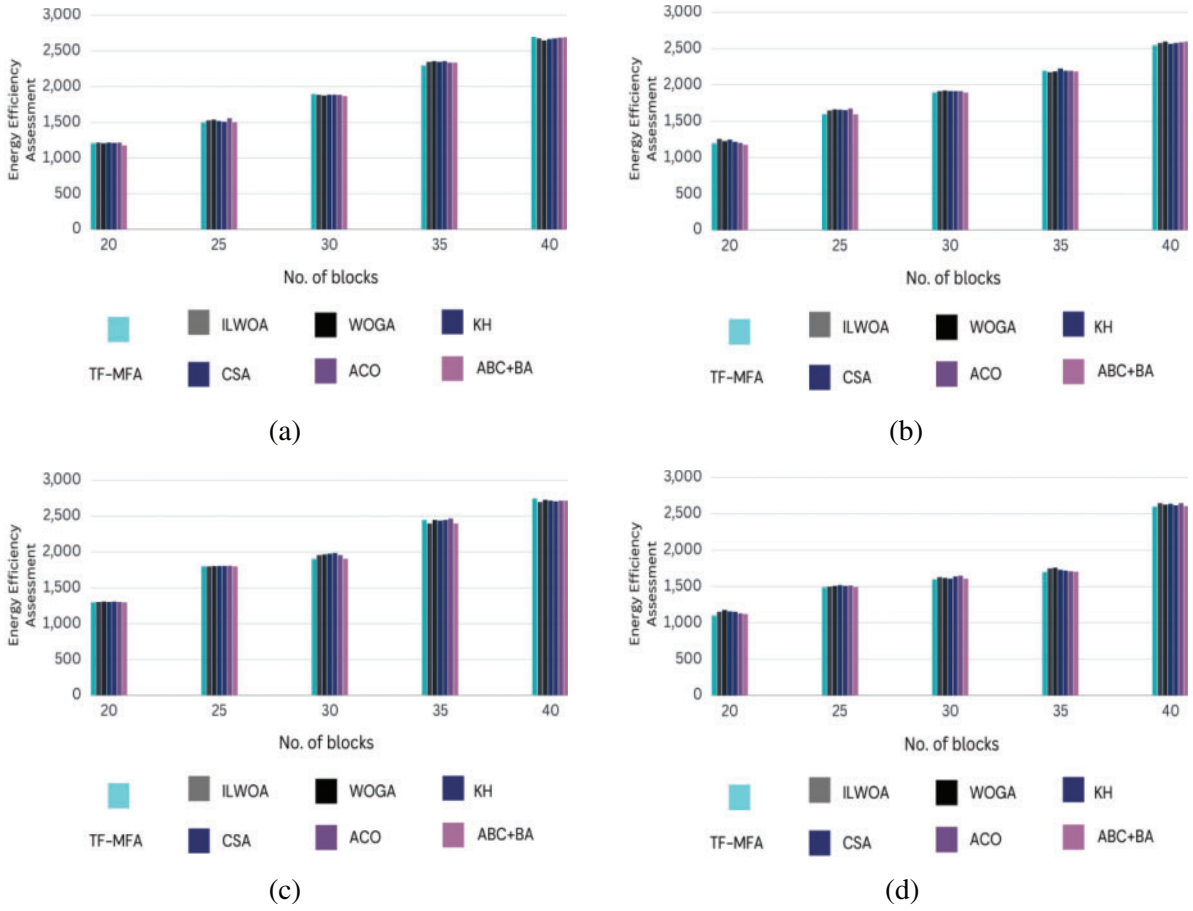


Figure 2: A comparative assessment of energy efficiency is performed to compare the planned work to the present work, using varying quantities of VMs allocated by a PM for load distribution. The VM counts under consideration are as follows: a = 10, b = 20, c = 30, and d = 40

Table 3: Illustrates the distribution of energy efficiency across different VMs in both anticipated and current models

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC+BA
10	2451.5	2252.7	2486.5	2557.2	2525.5	2530.8	2490.1
20	2578.7	2547.9	2536.6	2562.3	2565.4	2579.5	2490.6
30	2501.9	2584.8	2592.2	2579.8	2563.2	2559.7	2467.4
40	2613.3	2543.4	2523.1	2517.3	2543.6	2525.5	2547.2
50	2600.2	2561.3	2575.8	2548.6	2525.9	2530.4	2514.3

6.2.1 Comparative Analysis

This analysis looks at the twin-fold moth flame algorithm (TF-MFA) in detail and compares it to other methods like ILWOA, WOGA, KH, CSA, ACO, and ABC+BA. It achieves this by looking at how energy efficiency changes over time in different VM populations. TF-MFA routinely demonstrates superior energy efficiency compared to other approaches. With 10 VMs, the energy efficiency of TF-MFA is higher than that of ILWOA, WOGA, KH, CSA, ACO, and ABC+BA. Specifically, TF-MFA consumes 2451.5 units of energy, whereas ILWOA consumes 2252.7 units, WOGA consumes 2486.5 units, KH consumes 2557.2 units, CSA consumes 2525.5 units, ACO consumes 2530.8 units, and ABC+BA consumes 2490.1 units. This pattern keeps happening with different numbers of VMs (20, 30, 40, and 50), showing that TF-MFA is reliable at making the best use of energy in cloud-based settings. The findings establish TF-MFA as a viable and competitive technique, showcasing its potential to improve energy efficiency in cloud VM placement situations.

The assessments of the current techniques are conducted by measuring their energy use metrics, which are often represented as percentage improvements or enhancements compared to the baseline methodologies. In the given scenario, the energy efficiency of the suggested approach is evaluated by comparing it to the ABC+BA, ACO, CSA, KH, WOGA, and ILWOA algorithms. The percentage variations or enhancements quantify the extent to which the suggested approach is more energy-efficient than each of the current approaches, taking into account particular situations and workload circumstances. The assessments are based on empirical data, which assesses the energy use of each algorithm in both controlled and real-world scenarios.

6.3 Assessment of Lifespan Analysis

The lifespan, which refers to the duration required to finish a certain set of operations, is a critical measure for optimizing load distribution in VM voyage, and it is imperative to minimize it. The results, shown in Fig. 3, show that the Twin-Fold Moth Flame Algorithm (TF-MFA) is better than other methods like ABC+BA, ACO, CSA, KH, WOGA, and ILWOA at getting shorter lifespans for different types of VMs. TF-MFA demonstrates superior performance compared to its rivals, with margins ranging from 2.23% to 46.74%, while using 40 VMs on a single PM consisting of 35 nodes. The thorough examination shown in Table 4 reinforces TF-MFA's persistent efficiency, as it consistently achieves the lowest lifespan readings across different deployments of VMs to a PM. By assigning 30 VMs to a single PM, TF-MFA achieves a lifespan of 281.45 units, outperforming rivals by ratios that vary from 8.69% to 52.93%. The findings confirm that TF-MFA is successful in achieving the main goal of reducing lifespan in VM voyage conditions.

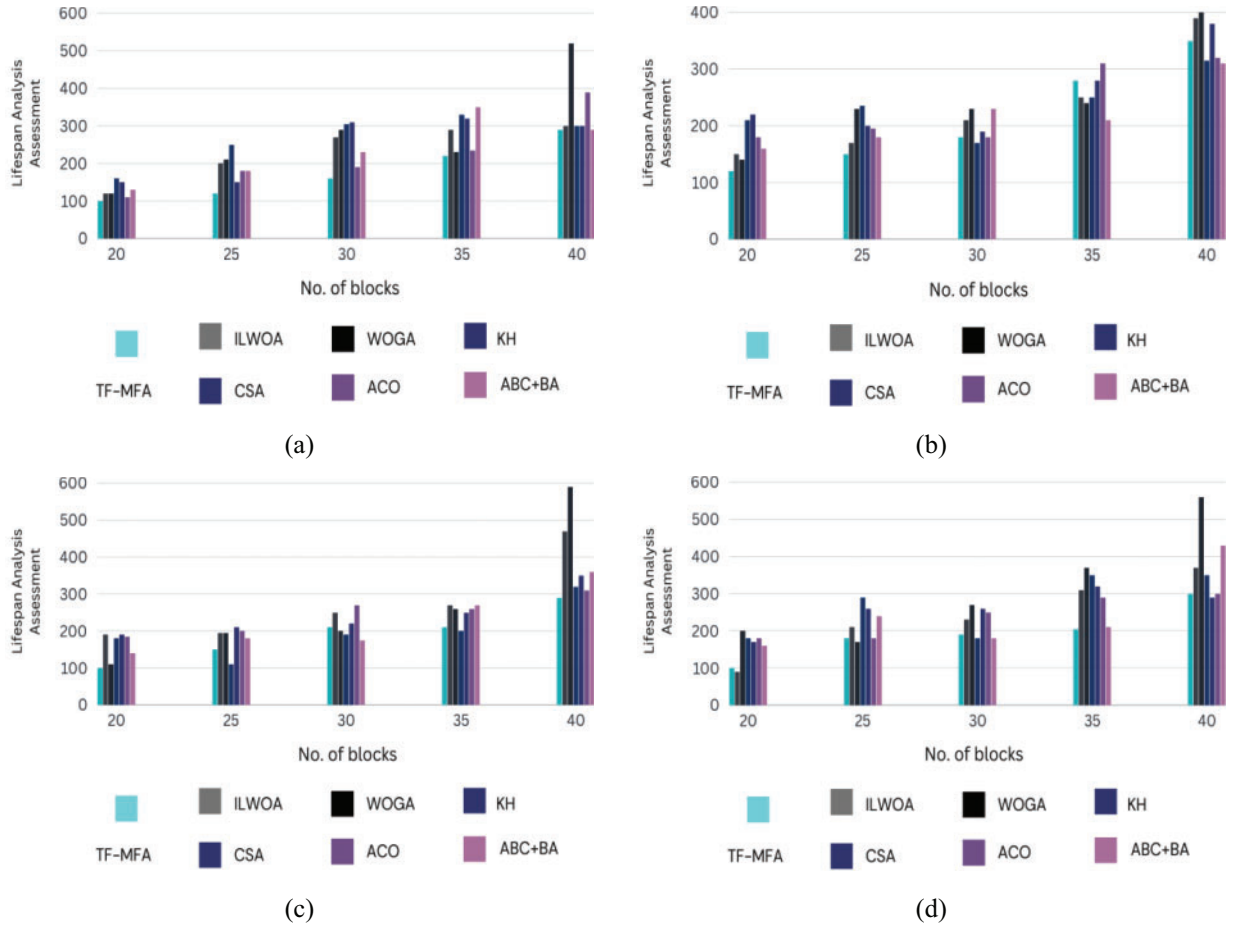


Figure 3: A lifespan assessment is performed to compare the planned work with the present work, utilizing varying numbers of VMs allocated by a PM for load distribution. The VM counts being examined are as follows: a = 10, b = 20, c = 30, and d = 40

Table 4: The lifespan across different VMs in both anticipated and current models

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC+BA
10	284.67	308.06	525.09	306.5	293.08	371.5	285.52
20	357.66	391.77	398.66	318.75	374.72	333.79	329.2
30	281.45	467.04	599.35	337.54	355.86	308.6	367.39
40	302.17	374.92	542.31	360.81	277.07	283.17	421.57
50	379.49	331.35	605.11	391.13	347.33	384.36	322.52

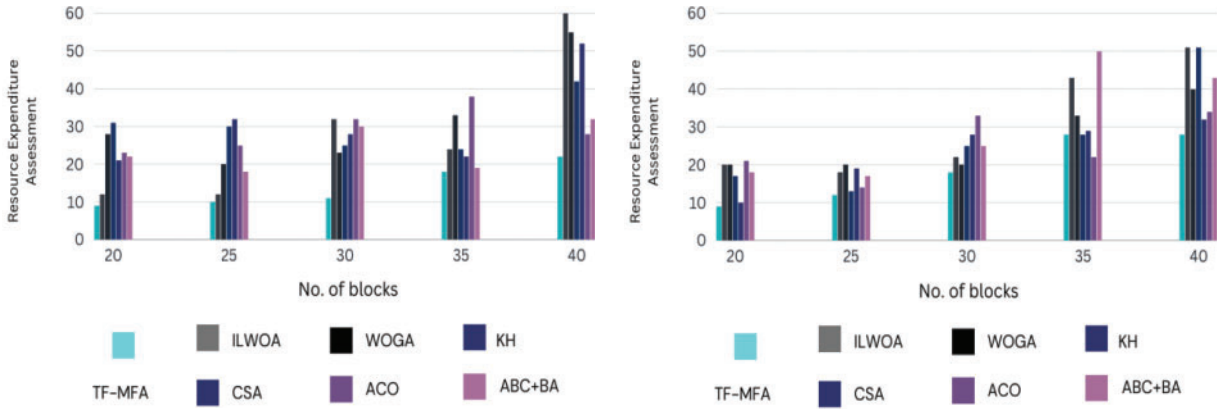
6.3.1 Comparative Analysis

The lifespan analysis is necessary to figure out how long VM deployments last. It also gives us useful details when we compare the twin-fold moth flame algorithm (TF-MFA) to other well-known methods. Upon evaluating the statistics provided in the table, it is evident that TF-MFA

regularly performs better than its peers in terms of lifespan, as it constantly displays lower values across different numbers of VMs. With a total of 10 VMs, the TF-MFA algorithm demonstrates a lifespan of 284.67 units, surpassing the performance of the ILWOA, WOGA, KH, CSA, ACO, and ABC+BA algorithms by a wide margin. TF-MFA consistently demonstrates its efficacy in optimizing VM lifespans, regardless of the number of VMs involved. In comparison to current approaches, the comprehensive evaluation highlights TF-MFA's edge in improving the time-dependent characteristics of VM deployments.

6.4 Assessment of Resource Expenditure

Fig. 4 presents a thorough assessment of resource expenditure on several VMs by altering the number of blocks. Significantly, the provided work constantly exhibits the most economical use of resources across all variations in block count. The proposed method performs better than other methods, like ABC+BA, ACO, CSA, KH, WOGA, and ILWOA, when looking at the 10th virtual machine with 40 block counts. It shows improvements of 34.82%, 30.41%, 59.70%, 48.65%, 69.75%, and 66.02%, in that order. Table 5 provides a breakdown of the resource expenditure assigned to a PM with varying numbers of VMs. The research continually demonstrates the lowest resource expenditure. To emphasize this achievement, the study specifically examines situations where there are 10 VMs per PM. In this context, the provided solution achieves the lowest value of 21.117. These figures indicate significant enhancements of 59.26%, 33.10%, 59.26%, 49.43%, 62.65%, and 65.82% when contrasted with ABC+BA, ACO, CSA, KH, WOGA, and ILWOA, respectively. The suggested method works better than the ABC+BA, ACO, CSA, KH, WOGA, and ILWOA methods by 25.55%, 44.01%, 25.55%, 21.33%, 38.92%, and 62.07%, respectively. This shows that it can lower resource costs for 40 counts of VMs.



(a) (b)

Figure 4: (Continued)

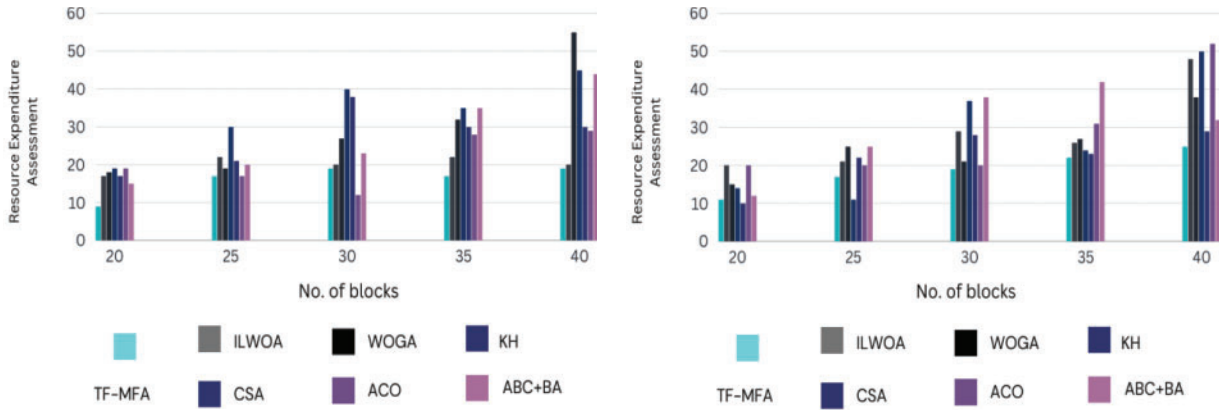


Figure 4: A resource expenditure assessment is performed to compare the projected work with the actual work, using varying numbers of VMs allocated by a PM for load distribution. The VM counts being examined are as follows: a = 10, b = 20, c = 30, and d = 40

Table 5: The resource expenditure across different VMs in both anticipated and current models

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC+BA
10	21.117	62.241	58.954	42.025	52.195	31.752	52.195
20	27.794	50.223	43.787	48.92	30.658	34.137	30.658
30	24.325	24.432	55.244	46.391	29.804	29.472	29.804
40	23.938	48.683	34.689	46.898	28.969	49.378	28.969
50	25.461	67.23	41.762	32.408	34.247	45.47	34.247

6.4.1 Comparative Analysis

Table 4 presents an evaluation of resource expenditure, revealing noteworthy observations about the relative effectiveness of various approaches. The twin-fold moth flame algorithm (TF-MFA) regularly exhibits beneficial results in many settings. When comparing 10 VMs, TF-MFA performs better than ABC+BA, ACO, CSA, KH, WOGA, and ILWOA by 39.58%, 32.23%, 33.92%, 22.94%, 27.89%, and 27.89%, respectively. As the number of VMs rises, TF-MFA consistently demonstrates better resource utilization compared to other techniques. The suggested TF-MFA outperforms ABC+BA, ACO, CSA, KH, WOGA, and ILWOA by 33.86%, 18.41%, 33.86%, 26.47%, 31.35%, and 31.35%, respectively, while considering 30 VMs. The pattern remains stable across all VM counts, demonstrating the TF-MFA's persistent efficacy in improving resource use.

6.5 Assessment of Total Expenditure

Fig. 5 presents an assessment of overall spending, which includes the study of energy efficiency, lifespan, and resource expenditure. This assessment provides valuable information about the effectiveness and reliability of load distribution. The provided study repeatedly demonstrates the lowest overall cost compared to established approaches such as ABC+BA, ACO, CSA, KH, WOGA, and ILWOA. The cost savings range from 4.95% to 28.96% for simulations including 30 VMs. The results are

supported by the data shown in Table 6, which shows that the total expense of the task being discussed is the most cost-effective option among different VM count circumstances. The suggested framework demonstrates superior performance compared to ABC+BA, ACO, CSA, KH, WOGA, and ILWOA by 8.65%, 7.02%, 0.68%, 11.81%, 32.58%, and 47.16%, respectively, while considering 20 VMs. Furthermore, when assigning 40 VMs to a single PM, the TF-MFA paradigm outperforms current approaches by 12.15%, 10.68%, 8.70%, 13.29%, 18.46%, and 33.39%. The given work constantly minimizes the total target, demonstrating its efficiency in distributing load during VM relocation, notwithstanding modest perturbations of particular objectives.

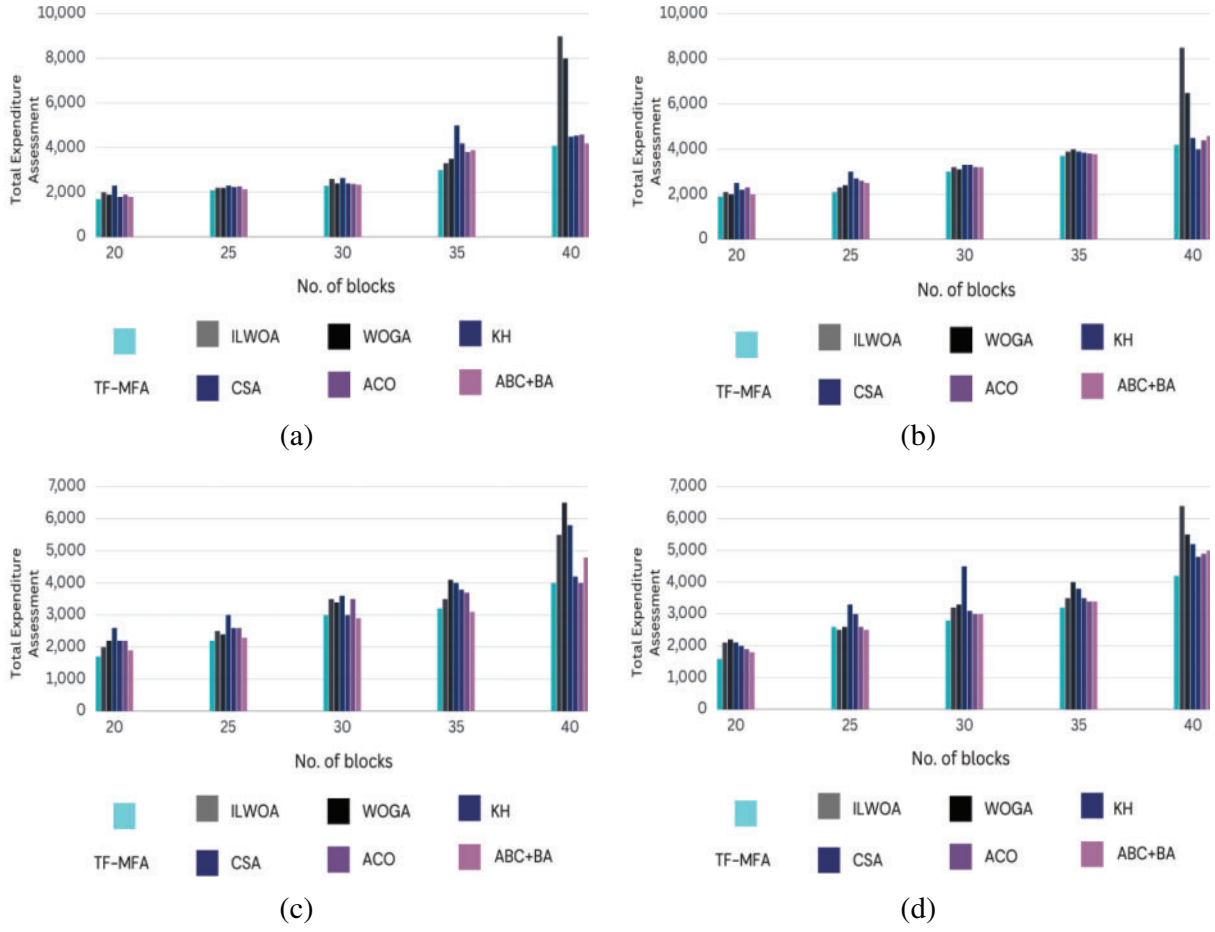


Figure 5: A total expenditure assessment is performed to compare the anticipated work with the actual work, using varying numbers of VMs allocated by a PM for load distribution. The VM counts being examined are a = 10, b = 20, c = 30, and d = 40

Table 6: The total expenditure across different VMs in both anticipated and current models

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC+BA
10	3742.7	8903.7	8017.1	4479.4	4378.2	4516.3	4043
20	4394.1	8331.5	6527.7	4988.1	4428.8	4731.2	4815.2
30	3995.2	5528.3	6537.7	5883.7	4448	4241.6	4708
40	4238.2	6371	5204.1	4893.7	4647.3	4750.5	4830.1
50	4552.5	9061.3	5816.8	5477.1	4349.2	4482.1	4068.1

6.5.1 Parameters for Total Expenditures

The following section provides a comprehensive elucidation of the expenses associated with computation, operation, maintenance, and possible downtime:

Computational costs: Computational costs in TF-MFA refer to expenditures related to computing assets such as servers, storage devices, and networking equipment. This includes the initial expenditures in hardware and software credentials, as well as the periodic expenses for maintenance, upgrades, and alternatives. The algorithm used in TF-MFA enhances the distribution of assets to reduce energy consumption when operating and reduces the hardware cost, thereby lowering the overall computational costs.

Operational costs: Operational costs in TF-MFA refer to the regular expenditures associated with the management and evaluation of a cloud-based environment. This involves remuneration for IT personnel overseeing the system, expenses for development, and charges for software licensing or use of cloud-based resources. TF-MFA includes expenses related to safety precautions, compliance efforts, and satisfying regulatory obligations to guarantee effectiveness in operation.

Maintenance costs: TF-MFA tackles maintenance expenses by handling tasks such as maintaining hardware components, implementing software enhancements and modifications, and troubleshooting difficulties. This entails allocating funds for periodic checks, maintenance, and proactive repairs to ensure that the cloud platform operates at its highest level of reliability and efficiency.

Downtime costs: TF-MFA also considers downtime costs, which are significant factors related to the monetary damages incurred during system operation interruptions. TF-MFA reduces the financial impact of possible downtime by avoiding service disruptions, reducing the use of resources, and meeting service level agreements (SLAs) with customers. The evaluation assesses the effects of downtime on revenue loss, productivity, fines resulting from SLA violations, and client satisfaction to preserve resilient operations and mitigate monetary hazards.

The successful implementation of the TF-MFA algorithm into current cloud-based management programs and instruments is largely dependent on systems' adaptability and interoperability. Integrating TF-MFA into cloud-based computing systems for load balancing purposes may not require extensive adjustments to current infrastructure, as long as a cloud-based governance framework accepts customized algorithms or modules. However, if the system lacks adaptability or fails to provide users with APIs for including personalized algorithms, the process of adjusting TF-MFA may require more significant alterations or even the creation of a separate module within the platform. It is critical to assess the specific features and structure of the cloud-based management infrastructure issue to ascertain the practicality and simplicity of integrating TF-MFA.

7 Conclusions and Prospects for the Future

This paper presents an improved method to improve the effectiveness of load balancing by using the twin-fold moth flame algorithm (TF-MFA), which is an enhanced version of the traditional moth flame optimization (MFO) algorithm. Choosing offline virtual machines comprises identifying suitable VMs for migration to reduce resource burdens on the designated servers. TF-MFA tackles this issue by prioritizing three main areas: energy efficiency, lifespan analysis, and resource expenditures. Researchers evaluate the suggested model by thoroughly examining energy efficiency, lifespan, and resource expenditures. The proposed work shows improvements of 12.15%, 10.68%, 8.70%, 13.2%, 18.4%, and 33.39% compared to the artificial bee colony-bat algorithm, the ant colony optimization algorithm, the crow search algorithm, the krill herd, the whale optimization genetic algorithm, and the improved Lévy-based whale optimization algorithm, in that order. We conducted these comparisons with a data set comprising 40 nodes.

The twin-told moth flame method outperforms state-of-the-art load-balancing systems in terms of efficiency and effectiveness. In this technique, the exploratory features of moth flight patterns are combined with the precision of controlled flames. This distinctive technique enables both accurate regressions towards optimal approaches and systematic investigation of the solution space. This approach improves load balancing in cloud-based environments by considering aspects such as computation time, stability, and placement cost. The algorithm's performance surpasses previous methodologies in terms of energy efficiency, lifespan analysis, and resource expenditures, as shown by comparative assessments. The twin-fold moth flame algorithm demonstrates improved efficiency and effectiveness in managing cloud-based workloads, highlighting its potential as a revolutionary optimization tool in the computing field.

The study of virtual machine (VM) deployment approaches in cloud-based computing has yielded insightful insights into the changing field of resource efficiency and system efficacy. The implementation of robust and dependable virtual machine (VM) deployments, as shown by notable achievements in 2023–24, has enormously enhanced the restoration procedure after VM failures in cloud-based data centers. The introduction of the aggregate selection-based strategy for VM deployment in 2023 highlights the significance of improving migration effectiveness, considering aspects such as route costs, communication overhead, and simulated heat. However, challenges include the burden on computer processing while practical implementation issues have surfaced. The integration of advanced strategies, such as the upgraded bee colony approach in 2021, strengthened cloud safety, but concerns about computing expenses and the need for extensive event assessment persist. In 2020, the investigation of adaptable clustering techniques showed improvements in the realms of energy efficiency, security, and anonymity when performing live VM transfers. Yet, execution difficulties, resource limitations, and the demand for extensive testing remain potential drawbacks [27,28].

The complexity of the twin-fold moth flame algorithm's (TF-MFA) deployment, the necessity for precise parameter modification, and the possibility of unsuccessful solutions in certain cases are all potential limits or downsides. Subsequent versions could overcome these constraints by streamlining the algorithm's execution while maintaining its efficacy, creating software-based techniques for optimizing parameter values to decrease the reliance on manual modification, and improving the algorithm's flexibility to handle various workload circumstances and prevent the likelihood of inadequate approaches. Furthermore, integrating machine learning or artificial intelligence functionalities into TF-MFA might enhance its efficiency and resilience in managing multifaceted cloud-based computing settings. Multidisciplinary research initiatives between academic institutions and industries may expedite the creation of feasible and scalable remedies [29]. Moreover, it is essential

to adopt a comprehensive strategy that considers aspects related to technology, ethics, and lifespan. The need to achieve a harmonious equilibrium between reliability, efficiency, and ecological impact will promptly influence the future of VM deployments on cloud servers. To summarize, the investigation of placing virtual reality approaches uncovers an intricate terrain with possibilities for improvement and originality. As we move forward, partnerships across different disciplines, technological improvements, and a persistent dedication to tackling problems, will drive the industry towards more robust, reliable, and environmentally friendly cloud-based systems for computation [30].

Although the TF-MFA has the potential to improve the placement of virtual machines (VMs) and balance the workload in cloud-based computing settings, it is important to address some limitations in future studies. These requirements include conducting an extensive investigation to ensure reliability in various workload situations, tackling adaptability obstacles in large-scale cloud-based platforms, investigating adaptive techniques to successfully manage system-level modifications, and improving tuning of parameter techniques to optimize multiple objectives. Furthermore, additional research into the algorithm's susceptibility to input variations and practical suitability in intricate cloud-based environments is critical to improving its usefulness and dependability.

Acknowledgement: We extend our profound gratitude to the Editor of the Journal of Intelligent Automation and Soft Computing for their significant counsel and assistance during the review process. Their proficiency and insightful recommendations have greatly assisted in the improvement of the article. Furthermore, we would like to express our sincere appreciation to the anonymous reviewers whose valuable comments and suggestions have significantly improved the overall standard of the article. Their thorough assessment and suggestions have unquestionably enhanced the substance and academic value of the article. The combined contributions of the editor and the anonymous reviewers have played a crucial role in molding the ultimate iteration of our work, and we express our heartfelt gratitude for their unwavering commitment to upholding the stringent requirements of scholarly publication.

Funding Statement: This work was supported in part by the Natural Science Foundation of the Education Department of Henan Province (Grant 22A520025), the National Natural Science Foundation of China (Grant 61975053), and the National Key Research and Development of Quality Information Control Technology for Multi-Modal Grain Transportation Efficient Connection (2022YFD2100202).

Author Contributions: All authors contributed to this research as follows: Umer Nauman conceptualized this study and led the methodology design; Umer Nauman and Yuhong Zhang collected and analyzed the data; Umer Nauman and Yuhong Zhang contributed to data analysis and played a key role in writing the initial draft of the manuscript; Umer Nauman, Yuhong Zhang, and Zhihui Li provided essential support in data visualization and interpretation; Tong Zhen and Yuhong Zhang secured funding for this research project. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials: The information that underpins this study's conclusions is freely accessible at https://www.kaggle.com/datasets/discdiver/clouds/code.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. He and R. Buyya, "A taxonomy of live migration management in cloud computing," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 1–33, 2023. doi: 10.1145/3615353.

[2] H. Li, G. Xiao, Y. Zhang, P. Gao, Q. Lu and J. Yao, "Adaptive live migration of virtual machines under limited network bandwidth," in *VEE 2021: Proc. 17th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Exec. Envir.*, New York, NY, USA, Apr. 2021, pp. 98–110.

[3] B. C. Pattanaik, B. K. Sahoo, B. Pati, and S. R. Laha, "Dynamic fault tolerance management algorithm for VM migration in cloud data centers," *Int. J. Intell. Syst. Appl. Eng.*, vol. 11, no. 3, pp. 85–96, 2023.

[4] W. Lin *et al.*, "Performance interference of virtual machines: A survey," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–37, 2023. doi: 10.1145/3573009.

[5] I. Çağlar and D. T. Altılar, "Look-ahead energy efficient VM allocation approach for data centers," *J. Cloud Comput.*, vol. 11, no. 11, pp. 1–16, 2022. doi: 10.1186/s13677-022-00281-x.

[6] M. A. Khan, "An efficient energy-aware approach for dynamic VM consolidation on cloud platforms," *Cluster Comput.*, vol. 24, no. 7, pp. 3293–3310, 2021. doi: 10.1007/s10586-021-03341-0.

[7] R. M. Haris, K. M. Khan, A. Nhlabatsi, and M. Barhamgi, "A machine learning-based optimization approach for pre-copy live virtual machine migration," *Cluster Comput.*, vol. 27, pp. 1293–1312, 2024. doi: 10.1007/s10586-023-04001-1.

[8] M. E. Elsaid, H. M. Abbas, and C. Meinel, "Virtual machines pre-copy live migration cost modeling and prediction: A survey," *Distr. Parallel Databases*, vol. 40, pp. 441–474, 2022. doi: 10.1007/s10619-021-07387-2.

[9] C. Jian, L. Bao, and M. Zhang, "A high-efficiency learning model for virtual machine placement in mobile edge computing," *Cluster Comput.*, vol. 25, no. 5, pp. 3051–3066, 2022.

[10] S. M. Moghaddam, M. O'Sullivan, C. Walker, S. Fotuhi Piraghaj, and C. P. Unsworth, "Embedding individualized machine learning prediction models for energy-efficient VM consolidation within cloud data centers," *Future Gener. Comput. Syst.*, vol. 106, pp. 221–233, 2020. doi: 10.1016/j.future.2020.01.008.

[11] T. Hidayat, K. Ramli, R. D. Mardian, and R. Mahardiko, "Towards improving 5G quality of experience: Fuzzy as a mathematical model to migrate virtual machine server in the defined time frame," *J. Appl. Eng. Tech. Sci.*, vol. 4, no. 2, pp. 711–721, 2023. doi: 10.37385/jaets.v4i2.1646.

[12] N. Alharbe, A. Aljohani, and M. A. Rakrouki, "A fuzzy grouping genetic algorithm for solving a real-world virtual machine placement problem in a healthcare-cloud," *Algorithms*, vol. 15, no. 4, pp. 128, 2022. doi: 10.3390/a15040128.

[13] J. Guo, Y. Li, C. Liu, Z. Zhao, and B. Zhang, "Research on a virtual machine mode transfer method supporting energy consumption optimization," in *23rd Int. Conf. Adv. Commun. Tech. (ICACT)*, Pyeonhchang, Korea, Feb. 13–16, 2021, pp. 77–80.

[14] S. Rukmini and S. Shridevi, "An optimal solution to reduce virtual machine migration SLA using host power," *Sensors*, vol. 25, no. 12, pp. 100628, 2023. doi: 10.1016/j.measen.2022.100628.

[15] D. Wang, W. Zhang, X. Han, J. Lin, and Y. C. Tian, "A multi-objective virtual network migration algorithm based on reinforcement learning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 2039–2056, 2023. doi: 10.1109/TCC.2022.3180784.

[16] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022. doi: 10.1109/ACCESS.2022.3149955.

[17] A. V. Riabko, T. A. Vakaliuk, O. V. Zaika, R. P. Kukharchuk, and V. V. Kontsedailo, "Cluster fault tolerance model with migration of virtual machines," in *Proc. 3rd Edge Comput. Workshop*, Zhytomyr, Ukraine, Apr. 7, 2023, pp. 23–40.

[18] A. Kaur *et al.*, "Algorithmic approach to virtual machine migration in cloud computing with updated SESA algorithm," *Sensors*, vol. 23, no. 13, pp. 6117, 2023. doi: 10.3390/s23136117.

[19] S. Talwani and J. Singla, "Enhanced bee colony approach for reducing the energy consumption during VM migration in cloud computing environment," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1022, no. 1, pp. 012069, 2021. doi: 10.1088/1757-899X/1022/1/012069.

[20] H. Kaur and A. Anand, "Review and analysis of secure energy optimization approaches for virtual machine migration in cloud computing," *Measurement Sens.*, vol. 24, pp. 100504, 2022. doi: 10.1016/j.measen.2022.100504.

[21] S. G. Sutar, P. J. Mali, and A. Y. More, "Resource utilization enhancement through live VM migration in cloud using ant colony optimization algorithm," *Int. J. Speech Technol.*, vol. 23, no. 6, pp. 79–85, 2020. doi: 10.1007/s10772-020-09682-2.

[22] R. Choudhary and S. Perinpanaygam, "Applications of virtual machine using multi-objective optimization scheduling algorithm for improving CPU utilization and energy efficiency in cloud computing," *Energies*, vol. 15, no. 23, pp. 9164, 2022. doi: 10.3390/en15239164.

[23] K. Surya and V. M. A. Rajam, "Prediction of resource contention in cloud using second order markov model," *Periodicals*, vol. 103, no. 10, pp. 2339–2360, 2021. doi: 10.1007/s00607-021-00914-0.

[24] D. Saxena, I. Gupta, J. Kumar, A. K. Singh, and X. Wen, "A secure and multi-objective virtual machine placement framework for cloud data center," *IEEE Syst.*, vol. 16, no. 2, pp. 3163–3174, 2022. doi: 10.1109/JSYST.2021.3092521.

[25] M. Torquato, P. Maciel, and M. Vieira, "Analysis of VM migration scheduling as moving target defense against insider attacks," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, Korea, Mar. 22–26, 2021, pp. 194–202. doi: 10.1145/3412841.3441899.

[26] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "An improved lévy based whale optimization algorithm for bandwidth efficient virtual machine placement in cloud computing environment," *Cluster Comput.*, vol. 22, pp. 8319–8334, 2019. doi: 10.1007/s10586-018-1769-z.

[27] C. Jiang, L. Yang, and R. Shi, "An energy-aware virtual machine migration strategy based on three-way decision," *Energy Robots*, vol. 7, no. 2, pp. 8597–8607, 2021. doi: 10.1016/j.egyr.2021.02.029.

[28] V. K. Netaji and G. P. Bhole, "Optimal container resource allocation using hybrid SAMFO algorithm in cloud architecture," *Multimedia Res*, vol. 3, no. 1, pp. 11–20, 2020. doi: 10.46253/j.mr.v3i1.a2.

[29] A. Javadpour, "Improving resource management in network virtualization by utilizing a software-based network," *Wirel. Pers. Commun.*, vol. 106, no. 2, pp. 505–519, 2019. doi: 10.1007/s11277-019-06176-6.

[30] U. Nauman, Y. Zhang, Z. Li, and T. Zhen, "Secured VM deployment in the cloud: Benchmarking the enhanced simulation model," *Appl. Sci.*, vol. 14, no. 2, pp. 540, 2024. doi: 10.3390/app14020540.