applied sciences

MDPI

Article

Secured VM Deployment in the Cloud: Benchmarking the Enhanced Simulation Model

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; stormy.umer@gmail.com
[2] School of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou 450001, China; zhangyuhong001@gmail.com
[3] Key Laboratory of Grain Information Processing and Control, Ministry of Education, Henan University of Technology, Zhengzhou 450001, China; lizhihui@haut.edu.cn
* Correspondence: gm1013315793@gmail.com

Abstract: Cloud computing has gained widespread recognition for facilitating myriad online services and applications. However, the current stages of commercial cloud computing employ a moderate design, wherein computational resources like storage and servers are housed in a few sizable worldwide data centers. System reliability, efficiency, and low latency are all goals of virtual machine (VM) placement. Load balancing has emerged as a crucial challenge for attaining energy efficiency in a fictitious grid computing architecture where a variety of users' workloads are distributed across several virtual machines. We propose a more effective optimization technique known as the twin fold moth flame algorithm. This algorithm considers multiple constraints, including computation time, stability, and placement cost. The proposed model's effectiveness will be evaluated based on relocation costs, reaction times, and stability assessments. The most significant gains of the presented work are 4.24%, 9.73%, 11.10%, 28.83%, 7.63%, and 10.62% for 20 count data of nodes for 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: virtualization; virtual machine deployment; cloud-based solutions; placement algorithm; load balancing

Citation: Nauman, U.; Zhang, Y.; Li, Z.; Zhen, T. Secured VM Deployment in the Cloud: Benchmarking the Enhanced Simulation Model. *Appl. Sci.* **2024**, *14*, 540. https://doi.org/10.3390/app14020540

Academic Editor: Peng Zhao

Received: 2 November 2023
Revised: 25 December 2023
Accepted: 5 January 2024
Published: 8 January 2024

1. Introduction

Cloud-based solution technologies have seen extraordinary development over the past few years. Cloud computing has become a popular model for providing support for a diverse set of web-based applications and businesses [1–3]. For instance, the medical industry leverages cloud computing solutions to securely store and retrieve data. It does this to improve patient care through the use of virtual healthcare applications by monitoring and identifying critical diseases such as cancer and immunological disorders [4,5].

The inefficient allocation of resources in cloud-based infrastructure poses a significant impediment. Cloud resource management stands as a forefront domain of study today. Because it does not rely on previous information, the robust module is superior to the stationary module in terms of both its adaptability and its effectiveness. In cloud computing, load balancing is seen as a multidimensional challenge of resource allocation that requires optimal utilization of resources without compromising the quality of service offered to customers [6–10]. This requires optimal utilization of resources to fulfill the demands of the problem at hand.

In the complex world of cloud computing, placing virtual machines (VMs) is a crucial task. It involves purposefully assigning virtual computers to physical servers on cloud platforms. The process of determining where virtual machines should be placed is made more difficult by the dynamic nature of workloads and shifting resources. Moreover, load

balancing turns into a crucial component that carefully distributes network traffic and computational workloads among several servers to maximize resource efficiency and avoid overloading any one node. The variety of hardware configurations, the unpredictable nature of workloads, and the critical requirement for reliability are a few of the issues that come with these operations [7]. It can be difficult to forecast and successfully manage resource requirements when dealing with dynamic workloads, which change in response to user demands and application requirements. Decisions about load distribution are made more difficult in heterogeneous cloud environments, which are frequently made up of various hardware and software configurations. Effective load balancing is essential for preserving resource equilibrium, preventing server under- or over utilization, and guaranteeing cost-effectiveness. Moreover, it is necessary for scalability because it enables horizontal scaling, which makes it simple to manage varying workloads. Achieving this equilibrium is essential for maximizing the use of resources, scalability, and fault tolerance, which in turn leads to improved user experience and energy efficiency. By addressing these issues in-depth, the suggested improvements to the simulation model for protected virtual machine deployment aim to establish a standard for attaining peak performance and economy in cloud-based applications.

Load-balancing algorithms typically optimize the selection of prospective target hosts throughout algorithm periods. Load balancing follows host selection. The immediate effect can maximize resource consumption, but it does not ensure task execution performance [7]. Heuristics methods, such as artificial bee colony (ABC) and ant colony optimization (ACO) [11] readily acknowledge this reality. The reduction of energy expenditure in VM migration requires promising technologies. Although such methods may yield rapid outcomes in terms of resource utilization, they fail to ensure the utmost efficiency in work completion pace. Algorithm-based approaches acknowledge and compensate for this problem to a great extent by employing well-established techniques like ABC and ACO. Despite this, the situation calls for certain solutions that are both practical and environmentally friendly to guarantee effective energy management during the placement of VM [12–16].

To optimize cloud computing load balancing, our proposed study presents the twin fold moth flame algorithm, which takes several constraints into account, and compares the suggested model to other algorithms. The twin fold moth flame (TFM) algorithm as a novel optimization technique for handling the complexity of virtual machine (VM) placement and load balancing in the ever-evolving cloud computing environment, where effective load balancing remains crucial. The TFM algorithm provides a fresh method of optimization and is modeled after the inventive characteristics of flames and moths. TFM uses a two-pronged strategy that combines the accuracy of confluence similar to flame management with the adventurous options reminiscent of moth flight patterns, all based on the notions of biological imitation. Because of this special conjunction, TFM is able to negotiate the complex field of virtual machine placement, taking stability, placement cost, and computation time into account. TFM intends to transform cloud computing effectiveness by adding aspects to load-balancing analysis. This work provides an in-depth evaluation that compares the suggested TFM algorithm with state-of-the-art benchmark methods, assessing its edge in terms of stability, placement cost, and computation time. Our work has made substantial improvements to the field, including the development of an effective load-balancing infrastructure, the TFM algorithm, and the thorough evaluation used to validate its benefits.

The contributions of our work can be summarized in the following:

- Algorithm development: The principal contribution of this research is the development of the twin fold moth flame (TFM) algorithm, a novel optimization technique designed to address the challenges related to load balancing and virtual machine (VM) placement in cloud computing environments. This algorithm incorporates special features that are influenced by both flames and moths. It uses a two-pronged approach that combines flame control precision with dynamic exploration similar

to moth flight patterns. This creative method is a significant development in cloud computing load-balancing techniques.

- A systematic method for load balancing: The TFM algorithm presents a comprehensive method of load balancing by taking into account several factors, such as computation time, placement cost, and stability. In contrast to traditional approaches, TFM integrates these crucial aspects to navigate the complex VM placement terrain, guaranteeing a more complete and efficient load-balancing solution for cloud computing settings.
- Biological ingenious optimization: By using biological imitation principles—more specifically, mimicking the traits of flames and moths—this research advances the area by developing an algorithm that successfully negotiates the challenges associated with virtual machine deployment. This novel methodology offers a new viewpoint on optimization techniques by taking inspiration from nature to meet cloud computing difficulties.
- Extensive comparative analysis: This research offers a comprehensive analysis that pits the suggested TFM algorithm against the most advanced benchmark techniques. This evaluation takes into account important parameters including calculation time, placement cost, and stability, providing a thorough examination that validates the algorithm's efficacy and highlights its advantage over current approaches. The thorough analysis performed in this study highlights the useful advantages of the TFM algorithm in optimizing cloud computing settings and advances our knowledge of load-balancing techniques.

This paper presents an advanced simulation approach to address the ongoing problem of optimizing the deployment of virtual machines (VMs) in the ever-changing cloud computing ecosystem. The enhanced methodology described in this paper attempts to address the complicated issues related to virtual machine deployment, redefining standards for efficiency and dependability in cloud-based environments. This study advances the area by proposing a novel simulation methodology that improves and protects virtual machine (VM) deployment in cloud-based systems. This study aims to set new benchmarks for reliability and efficiency in the rapidly evolving field of cloud computing.

The paper is organized as follows in the following sections. The details of cloud computing are covered in detail in Section 2, with particular attention to load balancing, energy efficiency, and virtual machine (VM) placement. Subsequently, Section 3 offers a thorough exposition of the design ideas and aims that dictate the suggested virtual machine placement technique, with a focus on load balancing, system dependability, and minimal latency. The paper presents a multifaceted issue formulation in Section 4 that includes parameters like placement cost, stability, and calculation time for evaluating virtual machines. The twin fold moth flame algorithm and its multifunctional optimization capabilities designed for effective load balancing are the main topics of Section 5's in-depth investigation of the suggested model. The focus of Section 6 is on the investigation of the twin fold moth flame's three stages. Following this, a comparative analysis is presented to assess the suggested model in relation to current methodologies. In contrast, Section 7 presents the findings and recommendations for further investigation.

2. Materials and Methods

2.1. Related Work

Optimizing dynamic fault-tolerant virtual machine placement in 2023 was a significant step toward improving the reliability of cloud data center architecture, as noted in [17]. The authors used an advanced technique for virtual network resuscitation that maximizes virtual machine selection for traffic routing analysis by using integer programming. As a result, the cloud data center virtual machines' failure recovery method was enhanced. A group selection-based virtual machine placement approach was introduced in 2023 as a key contribution, as reported in [18]. This strategy aimed to improve migration efficiency by taking into account many parameters, including voyage cost, communication

overhead, and virtual heat. This strategy was successful in lowering the cost of relocation and transmission. But in 2021, a VM travel approach called the enhanced bee colony approach [19] was developed to strengthen cloud security and thwart intrusions. Statistical security analysis was made possible by this method, which used the bee colony approach to simulate VM operation. Although these contributions show promise for strengthening security, increasing migration efficiency, and optimizing fault-tolerant virtual machine placement, there may be some disadvantages, such as computational overhead, difficulties implementing complex techniques in practice, and the requirement for thorough validation in real-world cloud environments.

According to [19], an entirely novel IP Sec routing technique for live VM transfers across infrastructure sites surfaced in 2021. The implementation of this technique created a complex router path that was intentionally hidden to prevent possible man-in-the-middle attacks. Interestingly, an onion routing strategy was used to improve virtual machine transfer privacy, demonstrating a strong security layer. A configurable and energy-efficient live virtual machine trip approach was introduced by [20] in 2022, with an emphasis on reducing idle machine power consumption to conserve energy. The framework provides a comprehensive solution by incorporating many modules, including resource monitoring, distribution, task allocation, analysis of optimization techniques, remote voyage agent, voyage scheduler, and energy regulator. Nonetheless, researchers used flexible clustering strategies in 2020 [21], describing the relocation and deployment of virtual machines through probabilistic incentive networks. One advanced method is dynamic evaluation that makes use of optimization incentive functions. Although these contributions show promise for improving energy economy, security, and privacy during live virtual machine transfers, there may be certain disadvantages due to implementation complexity, resource constraints, and the requirement for extensive validation in various cloud computing settings.

A security evaluation method based on an accessibility model was presented by [22] in 2022 for virtualized infrastructures with VM migration, namely for virtual machine manager (VMM) recovery. Establishing the best restoration strategy while taking availability, threat, and stability into account was the main goal. The paper emphasized the difficult balancing act between mobility and potential security risks inherent in VM migration for resurrection, highlighting vulnerabilities like as man-in-the-middle and denial of service (DOS) assaults. Meanwhile, in 2021, ref. [23] presented a multifunctional, encrypted virtual machine deployment approach that combined a non dominated sorting genetic algorithm with a whale optimization genetic algorithm. This novel approach minimizes inter-communication latency and facilitates energy-efficient resource allocation among virtual machines (VMs), with a focus on secure and rapid user application deployment. Some potential drawbacks may include the need for extensive testing in real-world scenarios, computational overhead associated with advanced algorithms, and the practical implementation challenges in various cloud computing environments, although these contributions demonstrate strengths in security evaluation, restoration planning, and encrypted virtual machine deployment.

2.2. Limitations and Establishing Grounds for a New Proposal

Addressing these restrictions and proposing novel solutions will increase dependability, scalability, real-time responsiveness, and security in dynamic fault tolerant VM placement and migration in cloud data centers.

2.2.1. Key Limitations

- Scalability issues: The scalability issues surrounding virtual machine deployment and migration in expansive cloud data centers are not covered in the current research. Real-world cloud infrastructures must be scalable, yet the boundaries of present scaling are rarely investigated.
- Restricted security scope: Although certain studies in the same field discuss security concerns such as denial of service (DoS) and man-in-the-middle attacks, their coverage

is still quite narrow. Emerging risks could arise from security flaws in the migration and installation of virtual machines.

- Inadequate research on suggested techniques: Suggested procedures are not frequently the subject of thorough examinations in related papers. This makes it difficult to conduct a thorough evaluation of solutions because there are insufficient experimental setups, comprehensive performance metrics, and comparisons with other cutting-edge techniques.

2.2.2. Establishing Grounds for a New Proposal

- Scalability: In large-scale cloud data centers, VM deployment and migration provide scalability difficulties that should be given top priority in the suggested solution. Sustained performance requires strategies to effectively manage expanding workloads and a growing number of virtual machines.
- Instantaneous adaptability: The new approach should concentrate on latency and reaction time because it acknowledges the significance of time-sensitive applications. Meeting the needs of real-time applications requires ensuring secure virtual machine installation and migration while minimizing the impact on application performance.
- Integrated stability procedure: The new plan must recognize and address a wide range of security risks and vulnerabilities to preserve the integrity, confidentiality, and availability of virtual machines and cloud infrastructure. It is important to incorporate a comprehensive security approach into the VM deployment and migration procedures to protect against new threats.

3. A Compact Architectural Description of the Proposed VM Placement Approach

The presented virtual machine placement technique presents a small-scale, yet all-inclusive architectural framework intended to maximize virtual machine (VM) distribution in cloud computing settings. The architecture's primary goal is to improve availability and performance by giving priority to effective load balancing. The system is carefully engineered to handle the resource consumption issue by dynamically reassigning overworked virtual machines (VMs) to underutilized counterparts, which creates equilibrium in the distribution of the total load. The architectural model leverages knowledge from almost 10 years of intensive research in the field to stress the smooth coordination of virtual machine placement in distributed systems. Sophisticated algorithms and mechanisms are contained in this small design, which also includes real-time monitoring and analysis to help with decision-making on virtual machine transfer. The confidentiality and integrity of the VM placement procedure are guaranteed by the smooth integration of security measures. The suggested architecture is a comprehensive and forward-thinking solution that embodies the changing requirements of cloud computing environments and positions itself as a strong way to deal with the difficulties associated with virtual machine deployment in a constantly changing and developing technological environment.

3.1. Proposed Framework: Task Constraints and Cloud Infrastructure Prototype

The proposed workflow for securing VM voyages on effective load balancing is explained in Figure 1.

Table 1 summarizes research on cloud computing VM migration. The table lists the pros and cons of each technique or algorithm. These details highlight the vast range of cloud VM migration improvements offered. VM migration solutions address effectiveness, dependability, cost savings, resource use, and performance. By summarizing each technique and highlighting its pros and downsides, the specifics aim to illuminate VM migration's considerations and trade-offs. Cloud computing researchers and professionals can utilize this knowledge to understand present tactics, investigate their applications, and develop new VM migration methods.

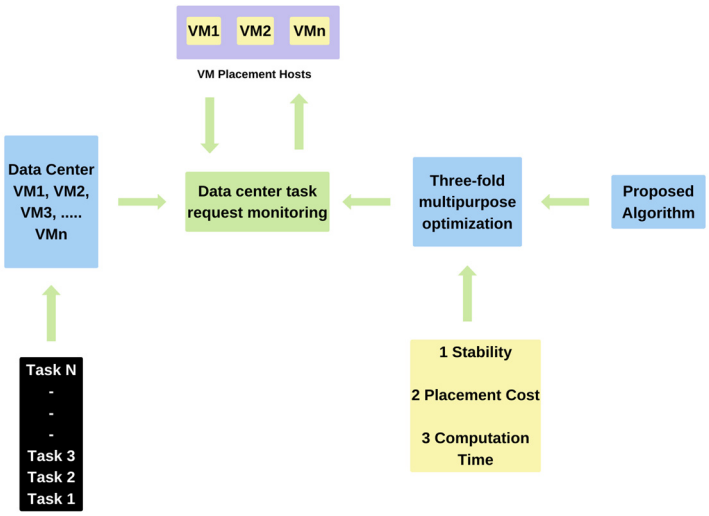

Figure 1. The workflow of the suggested VM placement for effective load balancing.

Table 1. Listing the foreground and issues of present VM voyage strategies.

Citation	Blueprint	Foreground	Issues
[19]	DFTM	VM migration improves cloud livability quickly and easily.	As VM consumption rises, so does resource waste.
[20]	VMMAGS	To increase system reliability and cut communication and migration expenses.	The migration was risky due to varying response time.
[21]	ABC-BA	Low energy consumption and failure rate reduce VM migration failures.	Network traffic overload necessitates the effective use of cloud resources.
[22]	ACO	Energy-efficient and fast migration.	To speed up migration, shrink the VM.
[23]	SPN	Low mean service time is associated with high throughput.	Low accessibility and low work service rate necessitates lowering power usage.
[24]	HMGOWM	Saving the CPU reduces the high access probability of virtual machine migration.	Avoid the prohibited overlap at all costs.
[25]	LVMM	Can increase the effectiveness of migrations and lower the cost of migrations.	Hefty and additional expenses.
[26]	SnapMig	By deleting unnecessary data blocks, migration times are reduced.	Hardware expenditures reduce IO and migration performance.
[27]	CSA	Conserve energy and resources.	Time draining.
[28]	KH	Boost data hub's energy efficiency.	Convergence may take time.
[29]	WOGA	Power, resource, and communication costs.	No VM mapping to server clusters.
[26]	ILWOGA	Optimum bandwidth reduces the number of active PMs.	Dynamic relocation and multipurpose VM allocation must be considered.

3.2. Resource Pooling

Each server in a cloud platform resource pool is denoted by $S_{i,}$ where $i = 1, 2, \ldots, n$. Under the initial state k, $S_i = VM_{i1}, \ldots, VM_{ik}$, and the server acknowledges and transmits

virtual machines at each endpoint. The i-th virtual machine is VM_i. Servers (S_i) are equipped with disk storage, computing resources, and networking. Dynamic host selection for VM allocation in cloud environments is difficult. Every kind of management has a specific QoS requirement, expressed as T_i,i = 1, 2, 3, . . ., n in the server's request for services. It should be noted that administration requests may have varying arrival, ripening tenure, and other conditions. The cloud platform user assigns T_i,i = 1, 2, 3,. . ., n to VMs. Task scheduling is determined by VM function execution and VM resource availability. Every task in the interval T_i,i = 1, 2, 3,. . ., n has three attributes: T_i = {CTU, Stab, CoP}. The three critical parameters for effective load balancing throughout VM placement in this context are computation time utilization (CTU), stability (Stab), and placement cost (CoP). This optimization is addressed using a proposed technique.

4. Multipurpose Problem Description and Evaluation for Quantification of Virtual Machine

The preliminary goal of this synopsis is to minimize the three-fold multipurpose problem, which is described statistically in Equation (1).

$$obj = min\{CTU, \text{Stab}, CoP\} \tag{1}$$

4.1. Computation Time Utilization

CTU, the total amount of time a computational task takes to execute a set of activities, is commonly referred to as CTU utilization [23]. Virtual machines are also associated with the resources of the actual server when a client requests a cloud-based commodity. These virtual machines are configured to accommodate the host's computational needs so that the steps involved should be immediately achievable. It is calculated considering the objectives specified in Equation (2).

$$UCTU(tasks) = \sum CountCTU\left(T_q+, \ldots + T_n\right) \tag{2}$$

4.2. Stability

For efficient and safe load balancing, this VM placement must also be theft-proof. The stability of VM (X(i)) is calculated by comparing the PM and the workload. The probability of risks involved (P_{risk}) model is used to conduct this analysis. The syntax can be followed using Algorithm 1.

Algorithm 1: Risk Propensity Based on Stability

```
for   i = 1:length(X)
if    X(i) <= 0
   P_risk = 0;
else if   X(i) > 0 & X(i) <= 1
   P_risk = 1 − exp((−1/2) × X(i));
else if   X(i) > 1 & X(i) <= 2
   P_risk = 1 − exp((−3/2) × X(i));
else
   P_risk = 1;
end
   P_risk(i) = P_risk;
end
Stability = mean(P_risk);
```

4.3. Placement Cost (CoP)

Another crucial indicator for evaluating task handling is the CoP. The CoP can be performed by continually implementing relocations, which are associated with incentive

potential. It calculates the total expense of the virtual machine changing tasks over time. A mathematical definition of CoP is given in Equation (3).

$$CoP = \sum_{Ti=1}^{n} CoM(T_{i;i} = 1, 2, \ldots, n) \quad (3)$$

5. Virtual Machine Placement Optimization using Solution Embeddings

5.1. Solution Embeddings

The process of solution embedding is depicted in Figure 2. The allocation of blocks for comprehensive tasks, denoted as "$B_{in}; i = 1, \ldots, N$", is twice the multiple of the chromosomal length. Each unit comprises two components: the physical machine responsible for task completion and the virtual machine identification assigned by the PM to receive the task (VM^{id}). The size of the block may vary for each individual job.

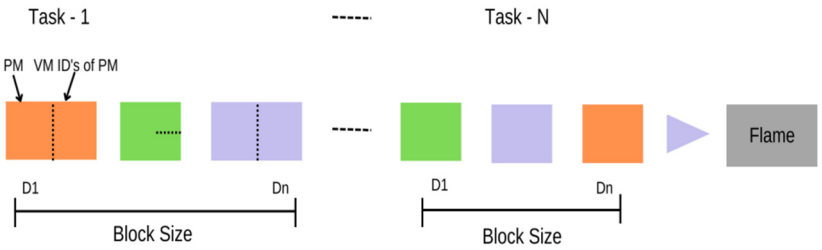

Figure 2. Solution embeddings.

As an illustration, let us consider a scenario in which the count of blocks amounts to 44, resulting in a chromosome length of 88. Consequently, the initial 44 chromosomes are occupied by the physical machine, whereas the remaining ones are allocated for the virtual machine.

5.2. The Proposed Model

The moth is lavish and resembles a group of butterflies [17]. Because the moon is distant, transversal orientation usually aligns the moth and moves linearly. The methods employed in the proposed TFM are shown below:

Stage 1: The general population of the moth $Moth_m$ and the $Flame_w$ are led off. Here, $Moth_m$ is the m^{th} moth of w^{th} flames. The general census of flames is $Count_{flames}$, and q signifies the conventional iteration census. Moreover, random values $ran1$ and $ran2$ are also initialized.

Fitness function determines the function for all moths, according to Equation (1).

Stage 2: When $q <= Count_{flames}$

(a) If $ran1 < 0.5$

The conventional moth flame optimization technique is used to modify both the moth's position and flame. The following section elaborates on the actions undertaken:

Equation (4) presents a mathematical formulation of the procedure for transverse alignment for the positioning revamp of the moth with regard to flame. Whereas Equation (5) symbolizes the mathematical representation of the logarithmic spiraling locks of the searching agent.

$$Moth_m = S(Moth_m, Flame_w) \quad (4)$$

$$S(Moth_m, Flame_w) = R_m e^{bt}.cos(2_\pi t) = Flame_w \quad (5)$$

For the w^{th} with the renown of the m^{th} moth, R_m refers to the shape of the logarithmic spirals, which is quantitatively presented in Equation (6). Furthermore, the number of flames over the length of t iterations can be calculated using Equation (7).

$$R_m = |Flame_w - Moth_m| \quad (6)$$

$$Count_{flames} = round(N - q \times N - 1/T) \quad (7)$$

Thus, *Fn* and *In* stand for the total number of flames and repetitions, correspondingly.

(b) If $ran > 0.5$

Using the flame distance and moth Equations (8) and (9), calculate the distance.

$$dist = neatps(Flame) - mothps \quad (8)$$

$$mothps = dist + neatps(Flame) \quad (9)$$

Stage 3: Moreover, the moment $q > Count_{flames}$

(a) If $ran < 0.5$

The conventional MFO method, which is theoretically presented in Equations (7)–(10), updates the position of the moth and the flame.

(b) In case $ran\ 2 > 0.5$

The proposed approach changes the location of the iteration of the algorithm (moth and flame).

Use Equations (10) and (11) to calculate the separation between the flame *dis* and the *moth*, correspondingly.

$$dist = neatps(Flame) - sort(pop) \quad (10)$$

$$mothps = dist + neatps(Flame) \quad (11)$$

Algorithm 2 displays the pseudocode of the suggested model.

Algorithm 2: Pseudocode of Generalized Architecture

The initialization population pop of the $Flame_w$ and the $Moth_m$ is calculated.
Initialize $Count_{flames}$, q, $ran\ 1$, $ran\ 2$.
Evaluate fitness input according to Equation (1).
for ($q_{max} > q$)
    If the termination requirement is not satisfied:
    if 1 ($q <= Count_{flames}$)
    if 2 ($ran\ 1 < 0.5$)
Equation (4) implements transverse alignment for flame placement updating with respect to moth.
Equation (5) is used to modify the search agent's logarithmic spiraling location.
Equation (7) can be used to compute the number of flames.
    else 2
Using Equations (8) and (9) to determine the distance between the flame and the moth:
    end if 2
     else if 1 ($q > Count_{flames}$)
    if 3 ($ran\ 2 < 0.5$)
Equation (4) implements the transverse alignment method to update the moth's flame position.
Equation (5) is used to modify the search agent's logarithmic spiraling location.
Equation (7) can be used to compute the number of flames.
    else 3
Using Equations (10) and (11) to determine the distance between the flame and the moth:
    end if 3
  end if 1
end
Terminate

5.3. Design of Databases

The physical machine's data for the virtual machine placement load-balancing database comes from https://www.kaggle.com/datasets/discdiver/clouds/code (accessed on 6 November 2022). The database creates virtual machine data. The variables in Table 2 are

in the PM's database. VM databases are built using actual machine datasets. The PM's attributes are also included. The following is a discussion of the database design process for all constraints and limitations:

Table 2. Database restrictions.

In	Iterations	Data
1	Census of PM	Inaugural Findings
2	Tasks involved	Articulated Findings
3	Fence tenure	Articulated Findings
4	Cost of PM	Articulated Findings
5	Stability	Articulated Findings
6	Count of computations	Articulated Findings

Let T_i;i (1, 2, 3 ,.., n) represent the PM tasks to be completed. The block sizes for tasks 1–4 are 5, 10, and 12, respectively. The task is given to the PM, which breaks it down and schedules each work in a different virtual machine. Each of the 13 PMs in use here is believed to have 10 virtual machines. There are 130 virtual machines in total to complete the duties.

5.4. Time and Space Complexities of the Proposed Model

Time complexity: The algorithm's time complexity is determined by the quantity (q) of iterations, as well as the size of the moth and flame populations. The moths and flames are subject to calculations and modifications at each level. The complexity of the equations used to calculate distances and update positions will determine the precise time complexity. The twin fold moth flame algorithm's overall time complexity can be roughly calculated as $O(q)$, where q is the total number of iterations and assumes that each stage's calculations can be thought of as constant time operations.

Space complexity: The memory needed to hold the population of moths and flames, as well as any other variables used for computations and updates, determines how much space the program takes up. The size of the issue, including the quantity of moths and flames, will also affect the challenge of space. The space complexity can be approximated as $O(N)$, where N is the total number of moths and flames in the population, and it is assumed that the storage needs for each moth and flame are constant.

6. Observations

6.1. Simulation Methodology

A virtualized placement strategy that was adapted for effective load handling was subjected to extensive testing and MATLAB analysis. An extensive evaluation procedure was applied to the suggested model, including important parameters such as placement cost, stability analysis, and CTU use. The effectiveness of the methodology was evaluated using careful comparisons with prior research findings, offering a strong foundation for evaluating its performance in the field of virtualized placement. MATLAB analysis and testing not only confirmed the validity of the suggested technique but also provided important insights into how well it might optimize load handling, bolstering its potential as a cutting-edge solution in the dynamic field of virtualized settings.

Modifications to virtual machines (VMs) and blocks are part of the assessment. Block characteristics are defined as follows: [20, 25, 30, 35, 40] is the fluctuation range that represents the variability in block characteristics. Meanwhile, for physical machines (PMs), the range of virtual machines (VMs) is [10–50], indicating the variation in virtual machine characteristics linked to each PM. Assignments 1–4 include the replacement of blocks from the available block assortment, implying a dynamic reorganization of components inside the system. This evaluation covers a wide range of adjustments, highlighting the flexibility and diversity inherent in the blocks and virtual machines (VMs), establishing the groundwork for an environment that is both responsive and flexible.

According to the examined database:

- Type 1 are [4 4 6 7] 20 total blocks = [T4 T1 T3 T2]
- Type 2 includes [5 6 6 8] 25 total blocks = [T4 T3 T1 T2]
- Type 3 includes [6 7 7 10] 30 total blocks = [T1 T3 T4 T2]
- Type 4 includes [7 8 9 11] 35 total blocks = [T1 T3 T4 T2]
- Type 5 includes [5 10 10 15] 40 total blocks = [T1 T2 T3 T4]

6.2. Assessment of CTU Usage

Central processing unit (CPU) utilization, sometimes known as CTU utilization, is the term used to describe the total amount of processing time needed to run a program in cloud computing. Applications that run in the cloud usually require virtual machines (VMs) located in server zones. Once assigned, these virtual machines (VMs) can accomplish a wide range of tasks that correspond with the host's processing needs. Minimizing CTU utilization is necessary for efficient resource use in cloud environments because it ensures that computing resources are used to their fullest potential. Therefore, reducing CTU consumption is essential to improving overall resource efficiency and making the cloud infrastructure capable of supporting a wide variety of workloads and operations.

In Figure 3, a graphical representation of the CTU usage between the proposed work and the previous studies is exhibited. As shown in Table 3, for 10 VMs: TF-MFA perpetrates a CTU utilization of 886.2, which is much lower than ILWOA (5692.2), WOGA (4712.1), KH (1280.8), CSA (1253.9), ACO (1323.2), and ABC + BA (935.61). This suggests that TF-MFA surpasses the other algorithms in the context of resource utilization for this VM count. For 20 VMs: TF-MFA demonstrates a CTU utilization of 1228.8, which is on descending analogized to ILWOA (5071.9), WOGA (3290.9), KH (1790), CSA (1202.9), ACO (1535.9), and ABC + BA (1679.2). TF-MFA exemplifies better resource utilization efficiency for this VM count, as well. For 30 VMs: TF-MFA attains the lowest CTU utilization value of 963.14 compared with ILWOA (2212.3), WOGA (3215.1), KH (2665.1), CSA (1253.9), ACO (1094.3), and ABC + BA (1597.1). This reveals that TF-MFA provides the most efficient utilization of computational resources for 30 VMs. For 40 VMs and 50 VMs: TF-MFA invariably harbors lower CTU utilization values compared with ILWOA, WOGA, KH, CSA, ACO, and ABC + BA, although the differences vary for each algorithm.

Table 3. Portrays CTU allocation for various VMs in proposed and existing models.

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC + BA
10	886.2	5692.2	4712.1	1280.8	1253.9	1323.2	935.61
20	1228.8	5071.9	3290.9	1790	1202.9	1535.9	1679.2
30	963.14	2212.3	3215.1	2665.1	1253.9	1094.3	1597.1
40	1135	3121.6	1858.5	1675.3	1501.7	1614.7	1598.9
50	1389.7	5818.5	2418.1	2246.8	1139.7	1311.8	985.04

The comprehensive analysis clearly shows that the twin fold moth flame algorithm (TF-MFA) model performs better than other algorithms in terms of central processing unit (CTU) use, exhibiting greater efficiency at different counts of virtual machines. This validation highlights how well the TF-MFA model works to optimize resource allocation and improve overall cloud computing environment virtual machine placement efficiency. The findings support the model's ability to achieve more efficient use of computing resources and highlight its promise as a cutting-edge and practical approach to handling the difficulties associated with placing virtual machines in a variety of dynamic computing settings.

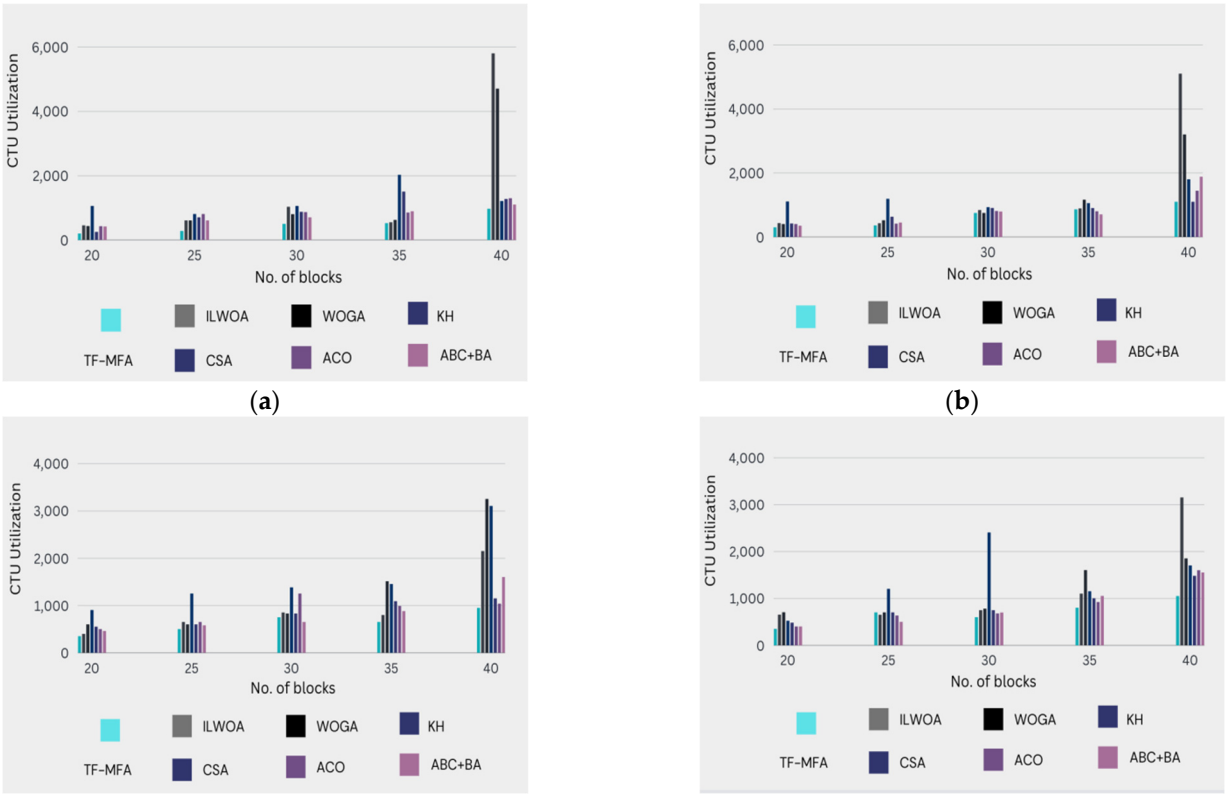

Figure 3. Compares CTU usage for proposed and current work when a PM distributes (**a**) VM = 10, (**b**) VM = 20, (**c**) VM = 30, (**d**) VM = 40 virtual machines for load balancing. Each subfigure represents a virtual machine distribution scenario with a full quantitative evaluation to improve adaptability. Conversely, the figure shows the present work's CTU consumption to benchmark the suggested technique. The research covers the same virtual machine distribution spectrum, providing an immediate and insightful comparability. CTU utilization, reaction times, and system throughput are examined to understand CTU use patterns under different scenarios. These subfigures help stakeholders optimize the use of resources and reliability by evaluating energy utilization, and efficiency.

6.3. Assessment of Stability

The assessment presented in Figure 4, which is measured in Table 4, and the corresponding stability factor values provide a thorough comparison of the advanced technique (TF-MFA) with alternative models, particularly concerning stability influence. This analysis is broken down into a detailed analysis of important stability metrics that show how the TF-MFA model stacks up against other methods. In the context of the proposed evaluation, this thorough assessment is essential for evaluating the TF-MFA model's stability performance and establishing its effectiveness relative to other models already in use. As such, it provides insightful information in the field of stability optimization.

Table 4. Stability for various VMs assigned by a PM for the current and suggested models.

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC + BA
10	0.22457	0.28106	0.50062	0.12212	0.11804	0.068857	0.088531
20	0.27208	0.18888	0.18888	0.13604	0.078694	0.082776	0.092612
30	0.2045	0.15561	0.60420	0.16100	0.068853	0.10240	0.16550
40	0.25816	0.21022	0.36543	0.17131	0.092612	0.19222	0.1332
50	0.32367	0.3159	0.58498	0.12869	0.1082	0.2229	0.088531

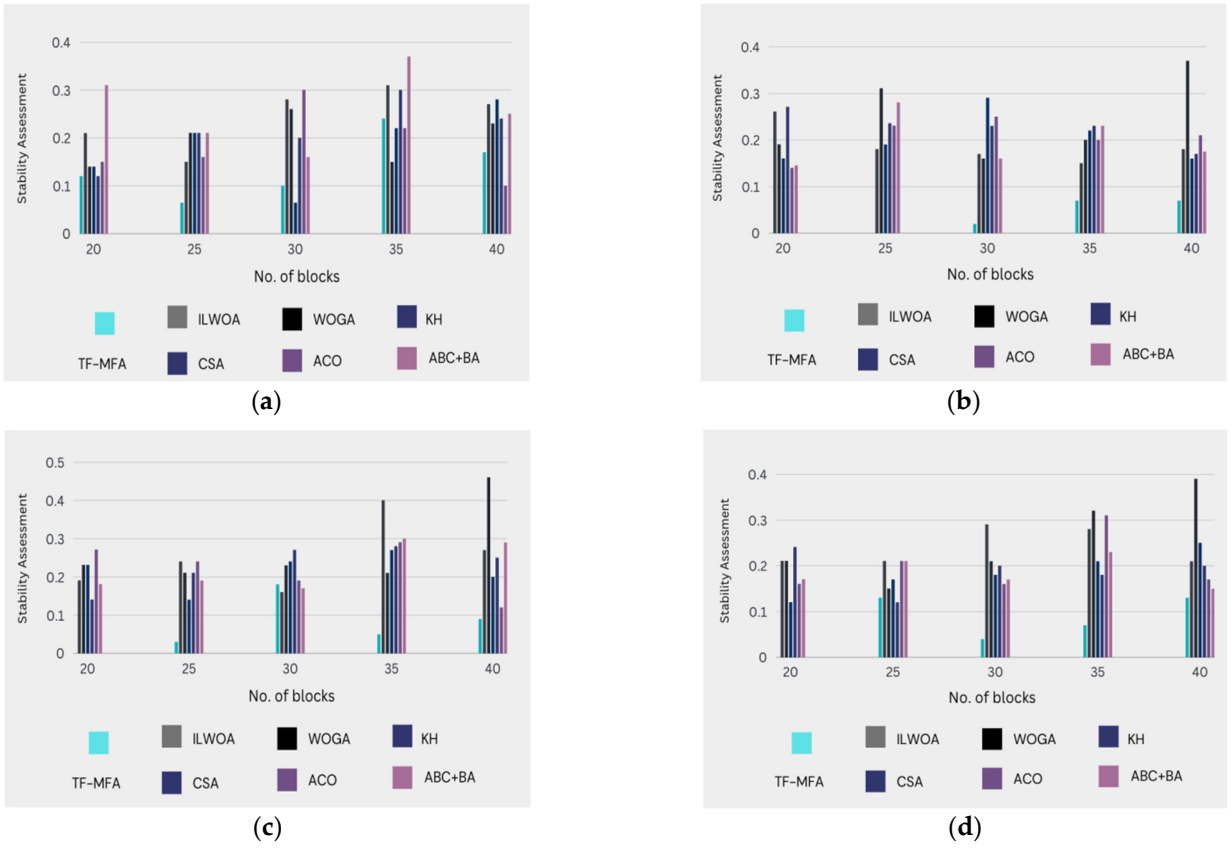

Figure 4. Portrays the stability of the proposed methodology over ongoing projects when a physical machine distributes (**a**) VM = 10, (**b**) VM = 20, (**c**) VM = 30, (**d**) VM = 40 virtual machines for load balancing. The approach's resilience and coherence across diverse virtual machine deployments has been depicted. A physical computer deploys virtual machines for load balancing. This examination evaluates the stability of the system. The recommended technique for active projects is tested for reliability and efficacy under varied workloads. Each subfigure shows the architecture's reaction to different computing levels of demand by representing a virtual machine distribution case. This figure assesses key stability indications and performance indicators, such as system reactivity and overall equilibrium, to provide useful information into the methodology's reliability and aid ongoing endeavors seeking an equilibrium between productivity and flexibility.

- Graphical analysis: In Figure 4, it can be observed that the proposed technique (TF-MFA) guarantees immaculate stability in load balancing while posing fewer risks compared to classic models. The graphical representation emphasizes the downsized risk and enhanced stability provided by TF-MFA.
- Probability comparison: The suggested methodology illustrates a significantly lower probability of high-risk circumstances compared to prior models. Particularly at 35 counts of clusters in the 30th virtual machine, TF-MFA is 82.32%, 81.71%, 81.45%, 81.13%, 75.18%, and 86.79% superior in terms of reduced probability when analogized to ILWOA, WOGA, KH, CSA, ACO, and ABC + BA, respectively.
- Stability factor: The stability factor quantifies the stability achieved by each model. The presented method perpetrates a stability factor of 0.2045 when a physical machine (PM) is assigned 30 virtual machines (VM) for a given role. In comparison, the stability factor values for ILWOA, WOGA, KH, CSA, ACO, and ABC + BA are 0.15561, 0.60420, 0.16100, 0.068853, 0.10240, and 0.16550, respectively, TF-MFA portrays a higher stability factor, signifying improved stability performance.

The overall results of the comparative analysis highlight how much more stable and successful the suggested method, the twin fold moth flame algorithm (TF-MFA), is than

other models. Specifically, the TF-MFA model delivers a higher stability factor, reduces the probability of high-risk scenarios, and guarantees outstanding load balancing stability. All these facts lend credence to the claim that the TF-MFA model is a better way to guarantee consistent and fair resource distribution in cloud computing settings. The findings further our knowledge of stability optimization techniques and establish the TF-MFA model as a reliable and practical method for handling the challenges associated with resource allocation in dynamic cloud computing settings.

6.4. Assessment on Placement Cost

The information shown in Figure 5 and Table 5 clarifies the assessment of the twin fold moth flame algorithm (TF-MFA), which is the methodology provided, in addition to other models, regarding placement cost. This breakdown includes a thorough examination of important placement cost parameters, providing information on how the TF-MFA model performs in comparison to other strategies. The thorough evaluation helps determine how cost-effective the TF-MFA model is when compared to other models that are already in use, and it offers insightful information about the financial aspects of virtual machine deployment strategies in cloud computing settings.

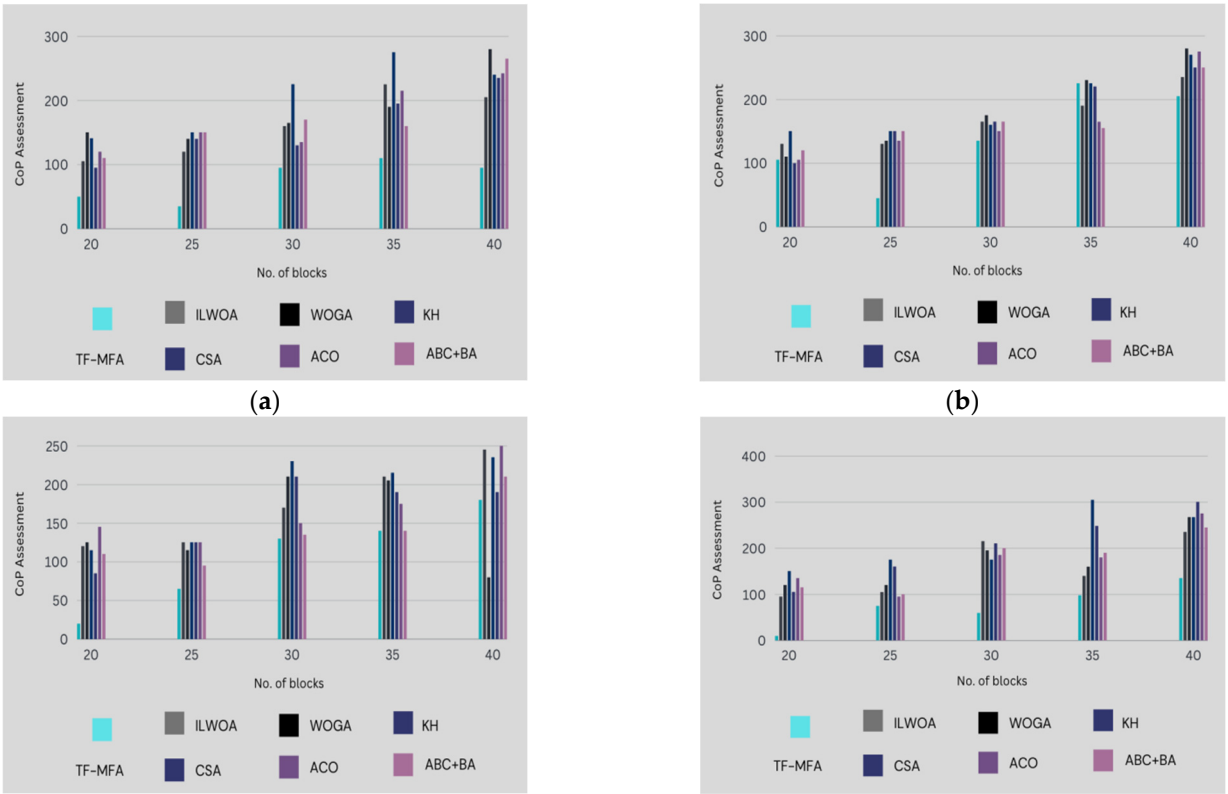

Figure 5. Portrays the assessment of placement costs for new and current work while allocating load-balancing virtual machines of (**a**) VM = 10, (**b**) VM = 20, (**c**) VM = 30, and (**d**) VM = 40. The subfigures shows the cost dynamics and financial implications of placing these virtual machines in new project creation and operation. Each subfigure represents a VM distribution scenario, showing how placement costs change with load balancing size. This figure helps decision-makers reconcile the distribution of resources and costs for new and current workloads by examining cost trends, possible savings, and effectiveness.

Table 5. Placement costs for different VMs to a PM for the current and potential models.

No. of VM	TF-MFA	ILWOA	WOGA	KH	CSA	ACO	ABC + BA
10	88.80	287.15	234.93	291.75	252.03	259.1	295.65
20	190.82	268.47	256.54	268.06	254	247.12	272.99
30	221.24	237.11	69.62	248.36	243.15	247.69	227.09
40	153.47	281.76	244.26	292.56	294.68	276.49	231.2
50	147.57	281.47	74.481	257.07	301.89	208.99	217.64

- Graphical analysis: Figure 5 displays the placement cost of the presented methodology in comparison to other techniques. It illustrates that the presented work consistently earns the lowest placement cost across different variations in VM and block size, thereby fulfilling the theme of minimizing placement cost.
- Percentage comparison: In the case of 40 censuses of blocks and VM = 10, the presented work outperforms prior methods by 64.81%, 61.22%, 59.57%, 60.42%, 65.45%, and 52.5%. This signifies a noteworthy reduction in placement cost compared to alternative procedures.
- Total placement overhead: Table 5 displays a synopsis of the placement overhead for different VM allocations. The presented work reveals a minimal total placement cost, attaining a placement cost of 88.80 when assigning 10 VMs to 1 PM. In comparison, classic approaches show higher placement costs.
- Specific comparisons: The presented method also outperforms constant references like ABC + BA, ACO, CSA, KH, WOGA, and ILWOA when allocating 40 VMs to 1 PM. The presented work achieves reductions in placement cost of 33.62%, 44.49%, 47.92%, 47.54%, 37.17%, and 45.53% compared with these reference models, respectively.

Overall, the comparison analysis's results show that the twin fold moth flame algorithm, or TF-MFA, consistently has the lowest placement cost among the approaches and dramatically lowers installation expenses. This result highlights how economical the TF-MFA technique is for maximizing resource placement and allocation expenses in the context of cloud computing. The results of this study enhance our comprehension of the financial viability of virtual machine placement techniques and establish TF-MFA as a notable and efficient method for attaining optimal resource allocation in dynamic cloud settings while minimizing installation costs.

6.5. Probabilistic Assessment: Proposed Model vs. Traditionally Used Models

Each algorithm undergoes execution 10 times to obtain the figures of the overall integrity that need to be decreased to guarantee an equitable comparison. The quantitative consequence for different counts of VMs allocated to one PM for performing the task scheduling function is displayed in Table 6. On observing the mean value for 10 counts of VMs, the presented work has the best lowest value of 1786.4, whereas the existing models have higher mean values of ABC + BA = 1996.3, ACO = 2042.5, CSA = 1985.3, KH = 2754.1, WOGA = 2110.5, and ILWOA = 2110.7. When 20 counts of VM are allocated to 1 PM, the best value of the presented model is 3.23%, 8.72%, 10.09%, 27.82%, 6.62%, and 9.61% better than existing ABC + BA, ACO, CSA, KH, WOGA, and ILWOA methods. Furthermore, in the median case scenario, the proposed TF-MFA model is 0.84%, 1.82%, 8.49%, 26.97%, 9.61%, and 7.42% better than existing ABC + BA, ACO, CSA, KH, WOGA, and ILWOA methods for 40 counts of VM. Thus, it is evident from the table that the presented work is much more sufficient for load balancing during VM migration. Additionally, for 40 counts of VM, the offered TF-MFA approach outperforms the current ABC + BA, ACO, CSA, KH, WOGA, and ILWOA approaches by 0.84%, 1.82%, 8.49%, 26.97%, 9.61%, and 7.42% in the median case circumstance. Overall, Table 6 makes it clear that the amount of work that has been given is far more adequate for managing load throughout virtual machine relocation.

Table 6. Probabilistic assessment of a PM's dynamic VM allocation for load balancing purposes.

VM at 10 Counts							
Behaviors	**TF-MFA**	**IILWOA**	**WOGA**	**KH**	**CSA**	**ACO**	**ABC + BA**
Best	1786.4	2110.7	2110.5	2754.1	1985.3	2042.5	1996.3
Worst	3421.4	3607.4	3807	4983.4	4313.8	3685	3734
Mean	2550.7	2950.6	2935.1	3563.7	3069.4	2908.6	2846.4
Median	2497.5	3042.2	2911.4	3259.1	2989.4	2953.4	2827.6
STD	715.48	732.15	746.18	1030.7	989.64	698.48	748.99
VM at 20 counts							
Behaviors	TF-MFA	IILWOA	WOGA	KH	CSA	ACO	ABC + BA
Best	1982.4	2193.3	2123.1	2746.8	2205.1	2171.9	2048.6
Worst	3639.5	3657.7	4012.3	3798.8	3733.5	3612.8	3468.4
Mean	2750.5	2893.7	3020.4	3268.1	2969	2835.7	2763
Median	2690.1	2862.4	2973.2	3263.4	2968.6	2779.1	2767.6
STD	734.3	655.5	785.28	431.17	650.1	624.92	611.02
VM at 30 counts							
Behaviors	TF-MFA	IILWOA	WOGA	KH	CSA	ACO	ABC + BA
Best	1685.1	2003.4	2168.6	2538.5	2092.3	2087.6	1946.5
Worst	3208.2	3625.8	4235.3	4101.8	3841.8	3648.9	3469.5
Mean	2539.3	2898.2	3054	3396	2979.7	2992	2697.6
Median	2632	2981.9	2906	3467.8	2992.3	3115.7	2687.2
STD	694.09	713.6	901.66	695.84	770.09	757.58	648.34
VM at 40 counts							
Behaviors	TF-MFA	IILWOA	WOGA	KH	CSA	ACO	ABC + BA
Best	1763.9	2200.1	2355.4	2166.4	2046.8	1985.7	1913.5
Worst	3450.1	3890.3	4448.4	4691.1	3833.1	3607.8	3600.9
Mean	2661.8	2989.8	3203.8	3574.6	2954.5	2781.9	2748.5
Median	2716.7	2934.5	3005.6	3720.5	2968.9	2767	2739.8
STD	700.33	760.01	942.95	1104.3	755.04	701.04	733.28

7. Conclusions and Future Directions

The development of a virtual machine placement strategy designed for utilization over wide area network (WAN) links involves the crucial aspect of offline VM selection. The primary objective of offline VM selection is to carefully choose one or more potential virtual machines for placement, ultimately mitigating resource demands on the hosts under consideration. This task is achieved through the implementation of the twin fold moth flame algorithm (TF-MFA) model, which is built upon three key goals: optimizing computing time unit (CTU) utilization, ensuring system stability, and minimizing placement costs. By addressing these goals, the TF-MFA model offers an effective and comprehensive solution for the strategic placement of virtual machines, particularly tailored for scenarios involving WAN links in cloud computing environments. The presented model's effectiveness was examined by calculating placement costs, computation times, and stability assessment. The most significant result of the presented work was 3.23%, 8.72%, 10.09%, 27.82%, 6.62%, and 9.61% for 20 count data of nodes for 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, as ambiguously portrayed in Figures 3–5.

The swift progress of computer science and technology is significantly influencing many aspects of our everyday lives and work settings. Looking ahead, we believe that combining state-of-the-art applied sciences—most notably, artificial intelligence and big data [30]—will be essential. This evolution is consistent with the ongoing creation of strong and affordable security controls, especially for large-scale networks and software-defined networks (SDN) in particular [31]. Furthermore, we anticipate creating end-to-end internet of medical things (IoMT) infrastructures using a mixed integer linear programming (MILP) mathematical paradigm [32]. These expected advancements represent the changing field of computer science and hold the potential to fundamentally alter how we engage with technology and tackle challenging problems across a range of industries, including healthcare and network security.

Author Contributions: All authors contributed to this research as follows: U.N. conceptualized this study and led the methodology design; U.N. and Y.Z. collected and analyzed the data; U.N. and Y.Z. contributed to data analysis and played a key role in writing the initial draft of the manuscript; U.N., Y.Z. and Z.L. provided essential support in data visualization and interpretation; T.Z. and Y.Z. secured funding for this research project. All authors have read and agreed to the published version of the manuscript.

Funding: 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 (No: 2022YFD2100202).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The information that underpins this study's conclusions is freely accessible at https://www.kaggle.com/datasets/discdiver/clouds/code, accessed on 6 November 2022.

Conflicts of Interest: The authors claim no possible conflicts of interest.

References

1. Annadanam, C.S.; Chapram, S.; Ramesh, T. Intermediate node selection for Scatter Gather VM migration in cloud data center. *Eng. Sci. Technol. Int. J. Commun.* **2020**, *23*, 989–997. [CrossRef]
2. Patel, Y.S.; Nagdev, M.; Choubey, A.; Das, S.K. On demand clock synchronization for live VM migration in distributed cloud data centers. *J. Parallel Distrib. Comput.* **2020**, *138*, 15–31. [CrossRef]
3. Pattanaik, B.C.; Sahoo, B.K.; Pati, B.; Laha, S.R. Default Fault Tolerance Management Algorithm for VM Migration in Cloud Data Centers. *Int. J. Intell. Syst. Appl. Eng. (IJISAE)* **2023**, *11*, 85–96.
4. Andrei, I.G.; Alexander, B.; Abbas, E.A.; Roman, V.P. Complex Anterior chest wall reconstruction after ongolic resections: A narrative review. *Shanghai Chest* **2021**, *5*, 41.
5. Parisi, G.F.; Mòllica, F.; Giallongo, A.; Papale, M.; Manti, S.; Leonardi, S. Cystic fibrosis transmembrane conductance regulator (CFTR): Beyond cystic fibrosis. *Egypt. J. Med. Hum. Genet.* **2022**, *23*, 94. [CrossRef]
6. Ahmad, S. Load-balacing in Cloud Computing Empowered with Dynamic Divisible Load Schedulling Method. *Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol.* **2021**, *5*, 44–53.
7. Lasemi, M.A.; Alizadeh, S.; Assili, M.; Yang, Z.; Baboli, P.T.; Arabkoohsar, A.; Raeiszadeh, A.; Brand, M.; Lehnhoff, S. Energy cost optimization of globally distributed Internet Data Centers by copula-based multidimensional correlation modeling. *Energy Robot* **2023**, *9*, 631–644. [CrossRef]
8. Tang, K.; Wei, X.F.; Jiang, Y.H.; Chen, Z.W.; Yang, L. Adaptive Ant Colony Optimization for Solving Large-Scale Traveling Salesman Problem. *Mathematics* **2023**, *11*, 4349. [CrossRef]
9. Alharbi, H.A.; Yosuf, B.A.; Aldossary, M.; Almutairi, J. Energy and Latency Optimization in Edge-Fog-Cloud Computing for the Internet of Medical Things. *Comput. Syst. Sci. Eng.* **2023**, *47*, 1299–1319. [CrossRef]
10. Javadpour, A.; Wang, G.; Rezaei, S. Resource Management in Peer to Peer Cloud Network for IoT. *Wirel. Pers. Commun.* **2020**, *115*, 2471–2488. [CrossRef]
11. Golightly, L.; Chang, V.; Liu, B.S.C. Adoption of cloud computing as innovation in the organization. *Int. J. Eng. Bus. Manag.* **2022**, *14*. [CrossRef]
12. Nadimi-Shahraki, M.H.; Zamani, H.; Fatahi, A.; Mirjalili, S. MFO-SFR: An Enhanced Moth Flame Optimization Algorithm Using Effective Stagnation Finding and Replacing Strategy. *Mathematics* **2023**, *11*, 862. [CrossRef]

13. Nirmala Sreedharan, N.P.; Ganesan, B.; Raveendran, R.; Sarala, P.; Dennis, B.; Boothalingam, R.R. Grey wolf optimization based feature selection and classification for facial emotion recognition. *IET Bio Metr.* **2018**, *7*, 490–499. [CrossRef]
14. Sharma, S.R.; Alshathri, S.; Singh, B.; Kaur, M.; Mostafa, R.R.; El-Shafai, W. Hybrid Multilevel Thresholding Image Segmentation Approach for Brain MRI. *Diagnostics* **2023**, *13*, 925. [CrossRef] [PubMed]
15. Putnins, M.; Androulakis, L.P. Self-selection of evolutionary stages: Adaptive *versus* non-adaptive forces. *Heliyon* **2021**, *7*, e06997. [CrossRef] [PubMed]
16. Javadpour, A.; Wang, G. cTMvSDN: Improving resource management using combination of Markov-process TMDA in software-defined networking. *J. Supercomput.* **2021**, *78*, 3477–3499. [CrossRef]
17. Riabko, A.V.; Vakaliuk, T.A.; Zaika, O.V.; Kukharchuk, R.P.; Kontsedailo, V.V. Cluster fault tolerance model with migration of virtual machines. In Proceedings of the 3rd Edge Computing Workshop, Zhytomyr, Ukraine, 7 April 2023.
18. Kaur, A.; Kumar, S.; Gupta, D.; Hamid, Y.; Hamdi, M.; Ksibi, A.; Elmannai, H.; Saini, S. Algorithmic Approach to Virtual Machine Migration in Cloud Computing with Updated SESA Algorithm. *Sensors* **2023**, *23*, 6117. [CrossRef] [PubMed]
19. Talwani, S.; Singla, J. Enhanced Bee Colony Approach for reducing the energy consumption during VM migration in cloud computing enviroment. In *IOP Conference Series: Materials Science Engineering, Sanya, Hainan, 12–14 November 2021*; IOP Publishing: Bristol, UK, 2021.
20. Kaur, H.; Anand, A. Review and analysis of secure energy optimization approaches for virtual machine migration in cloud computing. *Meas. Sens.* **2022**, *24*, 100504. [CrossRef]
21. Sutar, S.G.; Mali, P.J.; More, A.Y. Resource utilization enhancement through live VM migration in cloud using ant colony optimization algorithm. *Int. J. Speech Technol.* **2020**, *23*, 79–85. [CrossRef]
22. Choudhary, R.; Perinpanaygam, S. Applications of Virtual Machine Using Multi-Objective Optimization Scheduling Algorithm for Improving CPU Utilization and Energy Efficiency in Cloud Computing. *Energies* **2022**, *15*, 9164. [CrossRef]
23. Surya, K.; Rajam, V.M.A. Prediction of resource contention in cloud using second order Markov model. *Periodicals* **2021**, *103*, 2339–2360. [CrossRef]
24. Saxena, D.; Gupta, I.; Kumar, J.; Singh, A.K.; Wen, X. A secure and multi-objective virtual machine placement framework for cloud data center. *IEEE Syst. J.* **2022**, *16*, 3163–3174. [CrossRef]
25. Torquato, M.; Maciel, P.; Vieira, M. Analysis of VM migration scheduling as moving target defense against insider attacks. In Proceedings of the 36th Annual ACM Synopsium on Applied Computing: SAC'21, Virtual, Republic of Korea, 22–26 March 2021; pp. 194–202.
26. Abdel-Basset, M.; Abdel-Fatah, L.; Sangaiah, A.K. An improved Lévy based whale optimization algorithm for bandwidth efficient virtual machine placement in cloud computing environment. *Clust. Comput.* **2019**, *22*, 8319–8334. [CrossRef]
27. Jiang, C.; Yang, L.; Shi, R. An energy-aware virtual machine migration strategy based on three-way decison. *Energy Robots* **2021**, *7*, 8597–8607.
28. Netaji, V.K.; Bhole, G.P. Optimal container resource allocation using Hybrid SAMFO algorithm in cloud architecture. *Multimed. Res.* **2020**, *3*, 11–20.
29. Poluru, R.K.; Kumar, L.R. Enhancement of ATC by optimizing TCSC configuration using Adaptive Moth Flame Optimization Algorithm. *J. Comput. Mech. Power Syst. Control* **2019**, *2*, 1–9.
30. Zhang, Y.; Nauman, U. Artificial Unintelligence: Anti-intelligence of Intelligent Algorithms. In *International Conference on Intelligence Science*; Springer: Beijing, China, 2018; pp. 333–339.
31. Javadpour, A.; Ja'fari, F.; Taleb, T.; Shojafar, M.; Yang, B. SCEMA: An SDN-Oriented cost effective edge-based MTD approach. *IEEE Trans. Inf. Forensics Secur.* **2022**, *18*, 667–682. [CrossRef]
32. Javadpour, A. Improving Resource Management in Network Virtualization by Utilizing a Software-based Network. *Wirel. Pers. Commun.* **2019**, *106*, 505–519. [CrossRef]