Hybrid Fuzzy Metaheuristic Technique for Efficient VM Selection and Migration in Cloud Data Centers

C. Vijaya, P. Srinivasan*
Scope, Vit University, Vellore, India,
E-mail: c.vijaya2016@vitstudent.ac.in, srinivasan.suriya@vit.ac.in
*Corresponding author

Keywords: cloud computing, virtual machine migration, hybrid optimization, resource utilization and power consumption

Received: July 3, 2024

The rapid expansion of cloud computing has made maintaining Quality of Service (QoS) across dynamic workloads essential. Virtual machine (VM) migration is crucial for optimizing resource management; however, traditional migration techniques, which rely on static parameters, often lead to inefficiencies, such as increased energy consumption, higher migration costs, and suboptimal resource utilization. To address these challenges, a novel fuzzy-based hybrid optimization technique, FCSFFC, is proposed, integrating fuzzy logic with advanced optimization methods, Cuckoo Search and Firefly Colony Optimization. This technique introduces a dynamic threshold-based load prediction mechanism that adapts to real-time conditions, ensuring efficient VM placement and migration. The performance of this algorithm was rigorously evaluated using real workload data in a CloudSim simulation environment. Compared to state-of-the-art algorithms, the proposed approach demonstrated a 31.7% improvement in migration cost, achieving the lowest migration cost. Additionally, the proposed approach achieved the lowest energy consumption, using 2% less energy than other methods. In terms of load management and resource availability, the algorithm showed a significant reduction in the load parameter and the highest resource availability, minimizing unnecessary migrations. It also achieved the shortest computation time, completing tasks in 4.003 seconds compared to up to 9.2 seconds for traditional techniques. These results underscore the effectiveness of the proposed method in enhancing cloud service efficiency by optimizing energy consumption, reducing migration costs, and improving overall system performance.

Povzetek: Razvita je nova hibridno metoda FCSFFC za učinkovito migracijo virtualnih strojev (VM) v oblačnih podatkovnih centrih. Tehnika združuje mehko logiko in optimizacijska algoritma Cuckoo Search ter Firefly Colony Optimization. Rezultati kažejo 31,7 % nižje stroške migracije ter zmanjšanje porabe energije za 2 %, kar kaže potencial metode za izboljšanje učinkovitosti upravljanja virov.

1 Introduction

Cloud computing has emerged as a highly lucrative sector within the Information Technology industry, offering virtual services for data processing and storage to enterprises of all sizes. Businesses increasingly opt for cloud services over developing their own infrastructure due to the high costs involved. The primary advantage of cloud computing lies in its virtual provisioning, which enables resource access from any location at any time. This pay-per-use model has led to a surge in cloud adoption. However, as the benefits of mobility, adaptability, and compatibility drive more users to the cloud, the associated computing demands have risen significantly. To handle both expected and unexpected loads and meet computational requirements, cloud service providers have implemented hybrid clouds. These systems combine private cloud resources with additional public cloud resources, preventing the need for additional servers and ensuring efficient resource utilization. Despite these advantages, the increase in cloud usage escalates computational costs. Effective load management is crucial to maintaining service quality and meeting service level agreements (SLAs). This necessity highlights a significant gap in current approaches, which often rely on static thresholds for overload detection. Static methods fall short in dynamic cloud environments, where load conditions can fluctuate significantly. Existing methods for VM migration often rely on predefined thresholds and static policies to determine when and how to migrate resources. These approaches can lead to suboptimal decisions, increased costs, and delays in responding to dynamic workloads. Consequently, there is a pressing need for automated migration techniques that are accurate and adaptable to varying application contexts. This gap underscores the importance of developing more efficient migration strategies to enhance traffic management, reduce hardware maintenance, and lower energy consumption. Virtual machine (VM) migration is critical in commercial virtualization applications where resource requirements exceed available complementary resources. The challenge lies in managing these migrations without compromising service quality or breaching SLAs. In order to achieve better traffic management, reduced hardware

maintenance, energy management, and server consolidation, the majority of cloud services enable virtual machine migration. In conventional methods, the relocation procedure is typically governed by predefined rules and static thresholds, which may not effectively adapt to the dynamic nature of workloads. This causes bad decisions to be made, which raises costs and delays the migration process. Reducing costs and energy consumption is the main goal of the virtual machine migration; nevertheless, static thresholds and predefined rules can lead to suboptimal decisions and increased costs, highlighting the need for automated migration techniques that adapt to the dynamic nature of workloads. Dynamic VM migration is crucial for addressing these challenges, particularly in terms of energy and resource management. Effective VM migration strategies can help reduce energy consumption and optimize resource utilization, which are critical for improving overall data center efficiency. Current techniques often fall short in dynamically adapting to changing conditions, leading to inefficiencies and increased costs [1] [2]. Therefore, enhancing migration strategies to incorporate real-time data and adaptive thresholds is essential. There are two primary categories of migration: live and non-live [3]. Service providers generally prefer non-live migration to minimize user service interruptions. Several techniques have been proposed to assess VM utilization and relocate underutilized VMs to target systems. However, selecting the appropriate VM for migration remains essential to prevent performance degradation and negative impacts on QoS. The FCSFFC algorithm aims to address these challenges by developing a hybrid optimization algorithm that integrates the Fuzzy Cuckoo Search (FCS) algorithm with the Fuzzy Firefly Colony (FFC) algorithm. This hybrid approach aims to enhance VM migration efficiency, reduce energy consumption, and improve overall performance. Current optimization models for VM migration have limitations concerning QoS and SLA parameters. By combining these fuzzy-based algorithms, the proposed algorithm seeks to bridge these gaps and provide a more effective solution for managing VM migrations in cloud computing environments. Furthermore, the migration needs to be accurate and flexible to fit different application contexts. Similar to this, choosing the right virtual machine for the migration process is essential since the decision system needs to choose the right VM or else the target application's and the VM's performance would deteriorate and negatively impact QoS.

2 Related works

In cloud computing, virtualization is the most challenging research topic. This section presents an analysis and survey of related issues. In datacenters, virtual machines are mapped to the appropriate physical machines [5]. A one-dimensional Virtual Machine placement algorithm was studied by R. Panigrahy *et al.,* [6]. To operate the data center effectively while taking multiple objectives into account, the physical machines must provide the virtual machines with the proper support [7]. Numerous scholars have examined the approaches that employ meta-heuristic algorithms in cloud computing settings. However, the initial positioning of the VMs is the main focus of these algorithms. In order to give users a high-quality experience, virtual machine placement should prioritize power conservation and adhere to service level agreements [8]. To arrange virtual machines for effective power management and resource utilization, a variety of algorithms are available that offer the best possible placement.

The authors in [9] have presented a hybrid algorithm combining Chaotic Particle Swarm Optimization algorithm with adaptive mutation and energy aware algorithms used for better resource utilization. Metaheuristic evolutionary methods such as ACO-based approaches [19], GWO based methods [28], and VM migration using Ballooning [30] and Hill Climbing based approaches, Particle Swarm Optimization PSO) based approaches, Memetic approaches [10], and Genetic method-based approaches [34], Biogeography Based Optimization (BBO) approaches have been hybridized so far in Virtual Machine Placement. According to these papers, hybrid meta-heuristic approaches will be the most promising area of study for resolving the virtual machine placement problem. The use of meta-heuristic algorithms in cloud computing environments has been extensively researched. Effective server consolidation techniques are one way to enhance data centers. The proposed method lowers a data center's power consumption, which is currently the most difficult thing to do to keep data centers sustainable. A multi-objective grouping Genetic Algorithm [11] for reducing energy consumption and resource waste has been presented by C. Sonklin *et al.* A fitness function that weighs the two goals and makes a trade-off between the objectives has been identified.

An algorithm has been developed by Zhang *et al.* [12] to cluster the current generation's population and choose people from various groups with fewer cross-over operations. They have demonstrated that their algorithm outperforms the conventional genetic algorithm by using the runtime to ascertain the preference of virtual machines over physical machines as well as to generate the first solutions. X. Wang *et al.,* [13] define a mathematical model to lower make span, cost, and overall tardiness. A pre-selected dynamic resource allocation has been suggested. They have used a classifier to filter the subproblem solutions in the decision space. A chaotic multi-objective optimization algorithm was defined by S. Garepasha *et al.,* [14] for virtual placement in data centers. To accomplish server load balancing and cut down on resource waste, they have hybridized Sine Cosine (SCA) and Ant Lion Optimizer (ALO). In their work, P. Boominathan *et al.* [15] used a fuzzy hybrid bio-inspired technique to solve the server consolidation problem related to virtual machine placement. To select the next virtual machines (VMs) for the current server, fuzzy rules were created. The new ideal solution has been discovered using the Cuckoo search method. Thus, they have created an algorithm for server consolidation by merging ACS and the Firefly Colony Algorithm. Comparing both of them to

other algorithms that are similar, such as FFC, ACS, MMAS, and FFD, has shown which one produces the best results. M. Gagwero *e al.,* [16] examined how the MPC placement algorithm, which outperforms traditional heuristics in datacenters, minimizes power consumption, minimizes the effects of churn, and enforces security requirements. In their paper, Sharma *et al.,* [17] discovered the Gravitational search algorithm, a population-based meta- heuristic algorithm for solving non-linear problems based on the laws of motion and gravity. The number of physical machines used is taken into account when calculating fitness. The outcomes were compared with the Ant Colony
Optimization (ACO), FLC, and FFD algorithms. It works better than any other method, according to the results. In their work, M. Wang *et al.* [18] improve the Sine Cosine Algorithm (SCA) for optimization problems by adding a linear search path and a parameter that prevents the SCA from sinking into the local optimal solution. The focus of H. Xing *et al.* [19] is on reducing the amount of energy and network bandwidth used. They first choose the server with the lowest power consumption, and then they favor the server with the lowest network bandwidth resource consumption. and demonstrated that their algorithm (ETA-ACO) outperforms the algorithms that were compared.
N. Chalabi *et al.,* [20] suggest an improved marine predator algorithm based on Epsilon dominance and the Pareto archive for multi-objective optimization. Generally, the Marine Predator algorithm is used to solve single objective problems. But they have designed it to solve multi objective problems. Pareto dominance and Epsilon dominance concepts are used to arrive at the solution. The Whale Optimization Algorithm was enhanced by M.A. Basset *et al.,* [21] by altering the distance control factor and quickening the convergence. They believe that the most appropriate algorithms for optimization are evolutionary ones. A multimodal evolutionary algorithm was presented by Z.Ding *et al.*,[22]. They have embraced numerous subpopulations, and each one assesses its own autonomous evolution. Subpopulations are clustered hierarchically to avoid population discarding. Z. Xiang [23] enhanced the algorithm's optimization capability by combining the salp swarm algorithm and the sine-cosine algorithm for the shape matching process. A stochastic optimization problem that places a probabilistic limit on resource overflow is used to formulate the entropy of resource requirements by virtual machines (VMs). A stochastic VM placement algorithm takes this uncertainty into account. [24].
In order to find the best location for the virtual machines, a make span model, a cost and utilization mathematical model, and the efficiency of the Levy flight of cuckoos have been developed [25]. An Ant Colony system that uses an optimized chaotic Grey Wolf knowledge base to determine where to place virtual network functions and distribute paths based on knowledge of software-defined network controllers. The suggested algorithm has been shown to converge with fewer iterations in a shorter amount of computing time [26]. An artificial bee colony and chicken swarm optimization algorithm have been suggested in this paper for an efficient virtual machine

placement considering load, migration cost, and power consumption to prove it performs better than existing techniques. The Simulated Annealing Approach [28], which is combined with Grey Wolf Optimization, used in container-based virtualization, has been shown to perform better than other existing algorithms in terms of make span and load variation. This approach replaces traditional algorithms for virtualization. A hybridization of the Ant Colony Optimization and Sine Cosine algorithms has been performed for solving a multi-objective meta-heuristic problem in datacenters, considering minimization of resource wastage and power consumption. The experimental findings demonstrate that the ACOSCA algorithm [29] increased resource utilization by 16%, reduced power consumption by 24%, and improved execution time by 3%. A live virtual machine migration algorithm called live migration with efficient ballooning (LMEB) [30] has been proposed. Its main goal is to minimize the amount of data that must be moved from the source server to the destination server in order to lower the migration's overall energy consumption. The LMEB algorithm achieves this by optimizing the data transfer process, which helps reduce the associated downtime and operational disruptions. Additionally, its focus on minimizing data movement leads to lower overall energy costs and improved performance during the migration process. Using a VM migration technique, the authors of [31] proposed a resource management algorithm called "RU-VMM." The study took into account both successful and unsuccessful migrations when determining the resource utilization threshold. Three algorithms, namely Host Selection Migration Time (HSMT), VM Reallocation Migration Time (VMRMT), and VM Reallocation Bandwidth Usage (VMRBU) were proposed by the authors of [32] to minimize the overall migration time. A 25% decrease in overall migrations was also achieved. A 13% energy reduction was attained. By utilizing elastic scheduling, which the smart elastic scheduling algorithm (SESA) influences, the authors of [33] have created a virtual machine allocation and migration algorithm that is more energy-efficient. In our proposed algorithm, minimizing energy consumption, minimizing resource wastage and improving the migration cost are our main objectives. In Section 3, the proposed work, evolutionary multi-objective optimization objective functions, host overload detection, and the basics of the Cuckoo Search Algorithm (CS), the FFC algorithm, and the FCSFFC algorithm have been explained. This section provides a comprehensive overview of the methodologies employed, highlighting their significance in enhancing virtual machine migration efficiency. In Section 4, results were discussed. In Section 5, the conclusion of the paper is given. This section presents the suggested hybrid optimization methodology for choosing the optimal VMs for virtual machine migration.

Table 1: Summary table optimizing resource usage and power consumption

S. No	Optimization objectives	Optimization Method	Study	Limitations
1	Minimizing Resource Usage and power Consumption and maximizing Efficiency	β-Hill Climbing Algorithm	Hybrid Approach for Virtual machine Allocation in Cloud Computing [34]	Easily enters into a local optimum
2	Minimize Execution time and Migration cost	Enhanced Firefly algorithm PSO algorithm and Coyote Optimization Algorithm	A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems [35]	Migration time is high
3	Spacing and overall non-dominated vector generation	A Multi objective evolutionary algorithm	Energy-efficient Virtual Machine Placement in distributed cloud using NSGA-III algorithm [36]	Migration time is high
4	Minimize the number of Physical Machines and Power Consumption	Levy based Whale optimization algorithm	An improved Levy based Whale Optimization Algorithm for Bandwidth efficient virtual Machine Placement in Cloud Computing Environment [37]	Migration time is high
5	Minimizing Resource Usage Energy Consumption	simulated annealing-intelligent water drop cycle algorithm	Hybrid Metaheuristic Technique for Optimization of Virtual Machine Placement in Cloud [38]	Easily enters into a local optimum
6	Minimizing Resource Wastage Power Consumption	Resource Usage Factor Model using reward and Penalty Mechanism	An Energy-efficient Cuckoo Search Algorithm for Virtual Machine Placement in cloud computing data centers [39]	Limited to two dimensions
7	Minimizing Power Consumption and Network Latency and Maximization of Economic Revenue	Bat Algorithm with decomposition	Virtual Machine Placement using Multi-objective Bat Algorithm with decomposition in Distributed Cloud: MOBA/D for VMP [40]	Easily enters into a local optimum
8	Minimizing Resource Usage Power Consumption	Bio inspired FFC approach for server consolidation and VM Placement	A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Datacenters [41]	A few simplifying steps to accomplish the objective
9	Minimizing No. of Migrations and Improving QoS	Multilevel intrinsically controller-assisted modified ant colony optimization	Multilevel controller-assisted intrinsically modified Ant Colony Optimization heuristic-based load-balancing model for mega cloud infrastructures [42]	Migration time is high
10	Minimizing Power Consumption	Fuzzy Logic-Based Improved Cuckoo Search Algorithm	Novel Fuzzy Logic-Based Improved Cuckoo Search Algorithm [43]	Migration time is high
11	Minimizing Migration, Service Level Agreement Violations, and energy consumption	Multi-decision AHP (Analytic Hierarchy Process) method	Towards Virtual Machine Scheduling research based on multi-decision AHP method in the cloud computing platform [44]	Migration time is high
12	Minimizing Power consumption and Service Level Agreement Violation (SLA-V).	Cuckoo search Algorithm for VM Selection	An Optimal Cuckoo Search Algorithm for VM Selection for Energy Efficient Migration in Cloud Computing [45]	The migration takes too much time in live migration
13	Higher Computational Accuracy	Gravitational Acceleration and Cuckoo Search Algorithm	Virtual Machine Placement Optimization for Bigdata applications in Cloud computing [46]	Ignoring the extra energy required for moving virtual machines

Cloud computing and wireless sensor networks make extensive use of optimization models like Artificial Bee Colony optimization, Ant Colony Optimization, and other similar models [23]. Nevertheless, the features of optimization models in virtual machine migration have only been covered in a small number of publications. This work suggests a hybrid method for virtual machine migration that combines Cuckoo Search optimization with Firefly Colony approach. Combining these two algorithms with fuzzy rules incorporated into them, enhances the performance and allows it to successfully escape from local optima, which raises overall performance during the migration phase. Cuckoo search uses less parameters than other optimization algorithms, making it easier to set up than others. The superior computing efficiency sets FFC apart from other methods. When these algorithms are hybridized, complicated problems can have an improved optimal solution. The energy consumption, Migration cost, Resource availability and Computation time are studied in our research work.

A dynamic CPU threshold is employed in this algorithm to evaluate the overload detection of the host. The IQR method is used for determining the CPU threshold. If there are more user resource requests than available resources, the physical machine is deemed to be overloaded in the resource allocation process. In that instance, conventional resources must be made available to lower the physical machine load. To meet user requirements, virtual machines (VMs) are migrated to physical machines during this stage of the process. Energy use, resource use, computation costs, and migration costs are all taken into account during the migration process. Avoiding needless migrations improves resource utilization characteristics and cloud performance, as resource usage requirement prediction raises overall resource utilization. Four major objectives considered in this research are: Minimize energy consumption, resource wastage, migration cost and computation time.

A table summarizing the state-of-the-art algorithms with its optimization objectives for Virtual Machine placement, optimization methods chosen, and the name of the study is presented in Table 1 and the overview of the proposed optimization is presented in Fig. 1.

3 Proposed work

In the context of optimizing cloud data centers, accurately modeling energy consumption, resource wastage, and migration costs is crucial to achieving efficient resource utilization and minimizing operational costs. This section presents the mathematical models used to evaluate these factors, beginning with energy consumption, which is a significant concern in large-scale data centers. Following this, we address resource wastage, a critical measure of the efficiency of VM placement strategies, and finally, the migration cost, which plays a vital role in ensuring minimal disruptions and enhanced performance during VM relocations. These models collectively provide a comprehensive framework for assessing and improving the overall efficiency and sustainability of cloud data center operations.

a. Energy consumption modeling

Assume that there are 'n' Virtual Machines (VMs) on 'm' Physical Machines (PMs). It is given that under the capacity constraint, the capacity of any VM is not greater than the capacity of any PM. T_{pr} *and* T_{my} denote the maximum capacity of a single Physical Machine and R_{pr} *and* R_{my} reflects the CPU need of each virtual machine.

$$\boldsymbol{P_j}= \begin{cases} \left[\left(\boldsymbol{P_{busy}} - \boldsymbol{P_{idle}}\right)\boldsymbol{X}\ \boldsymbol{U_J^C}\right] + \boldsymbol{P_{idle}}\ , \boldsymbol{if}\ \boldsymbol{U_J^{pr}} > \boldsymbol{0} \\ \boldsymbol{0}\ ,\ \boldsymbol{Otherwise} \end{cases} \quad (1)$$

$\boldsymbol{P_j}$ is the power consumed by the server *j*, $\boldsymbol{P_{busy}}$ *and* $\boldsymbol{P_{idle}}$ are the power consumed in the busy state and the idle state of the server. $\boldsymbol{U_J^{pr}}$ is the normalized CPU usage which is calculated as remaining CPU capacity divided by total capacity. The cost of power consumption is given in Eqn. (1). By summing up the power consumption values over discrete time intervals, the total energy consumption $\boldsymbol{E_j}$, is approximated. The cost of energy consumption is given in Eqn. (2).

$$\boldsymbol{E_j} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} \boldsymbol{P_j(k)} \quad (2)$$

$\boldsymbol{P_j(k)}$ is the power consumption of server $\boldsymbol{j}$ at time $\boldsymbol{k}$.

b. Resource wastage modeling

There are many VM placement solutions which vary in the number of resources remaining on each server. Multidimensional resources should be completely utilized. Therefore, the cost of wasted resource is evaluated by the below equation:

$$\boldsymbol{w_j} = \frac{\left|\boldsymbol{L_j^{pr}} - \boldsymbol{L_j^{my}}\right| + \Delta}{\left|\boldsymbol{U_j^{pr}} - \boldsymbol{U_j^{my}}\right|} \quad (3)$$

w_j represents the resource wastage in j[th] server. $U_j^{pr} - U_j^{my}$ represents the CPU and memory usage utilized in a physical machine. It is the ratio between used resources to the total resources available. $L_j^{pr} - L_j^{my}$ represents remaining resources in terms of CPU and memory. Δ is a small positive integer to avoid the capacity of the physical machine coming down to zero and it is set to 0.0001. This addition is because of the uncertainties in the cloud environment. These models use Minimization of Migrations for hot spot selection and Modified Best Fit decreasing for virtual machine placement.

c. Migration cost modeling

The next objective is calculating migration cost triggered by a placement of $\boldsymbol{VM_i}$. Given a current placement of $\boldsymbol{n}$ VMs, the migration cost in terms of load aims to optimize resource utilization and performance while

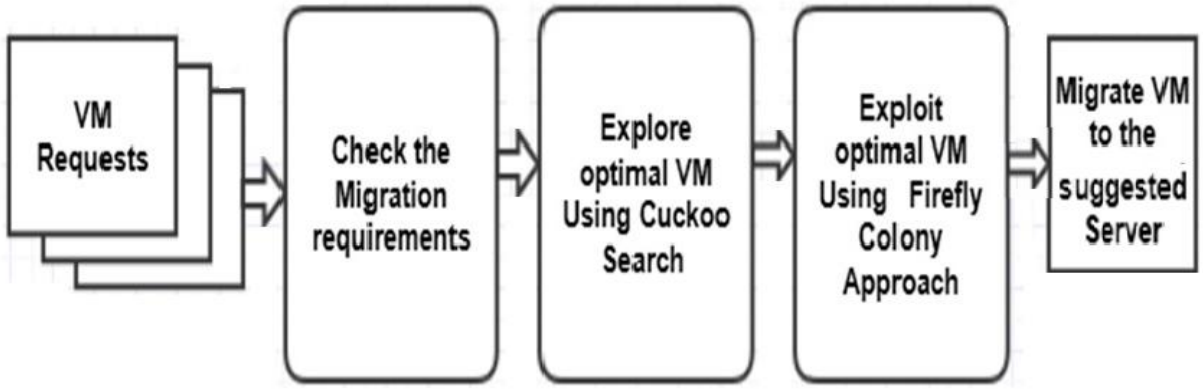

Figure 1: Overview of proposed optimization

minimizing operational disruptions and costs associated with VM movements in cloud environments at different states is calculated using the formula in Eqn. (4).

$$\mathbf{f(x)}=\sum_{\mathbf{i\in 1,n}} \mathbf{C(vm_i, x_i)} + \mathbf{C(vm_i, x_i')}, \quad (4)$$

Where $\mathbf{C(vm_i, x_i)}$ is the cost of VMs in source and $\mathbf{C(vm_i, x_i')}$ is the cost of VMs in destination.

3.1 Overloaded host prediction

A stochastic prediction model has been developed to identify overloaded hosts. Each host executes an algorithm periodically to assess current load conditions, aiding in the identification of overloaded nodes and ensuring SLA compliance. Unlike many existing methods that rely on fixed thresholds for overload detection, CPU utilization per host is estimated to detect overloads. Static thresholds are less effective in dynamic cloud environments, particularly in IaaS scenarios where load conditions fluctuate. Hence, dynamic CPU utilization is utilized for overload detection.

The dynamic thresholding scheme adjusts CPU utilization thresholds based on real-time variations. Larger deviations in CPU utilization suggest the need for lower upper thresholds. Increased deviations in usage patterns heighten the risk of nearly maximum CPU utilization, thereby increasing the chances of SLA breaches. To address these challenges, the hybrid model integrates inter-service relationships and variational data for dynamically adjusting thresholds. The hybrid model employs both Inter Quartile Range (IQR) and modified Local Regression Robust (LRR) techniques to estimate dynamic CPU thresholds [40]. The IQR method calculates the range between the first and third Quartiles of the CPU requirements of VM.:

$$IQR=Q3 - Q1 \quad (5)$$

Using the IQR value, the upper CPU utilization threshold per node is estimated as follows:

$$T_i=1 - s\cdot IQR \quad (6)$$

$\mathbf{T_i}$ is the upper threshold to determine the CPU utilization per node.

3.2 Evolutionary multi-objective optimization

Evolutionary multi-objective algorithms, in general, employ a population-based method to identify a close to optimal solution. Solutions that could be improved in more than one objective function are known as Pareto optimal

solutions. However, multiple objective functions ought to be optimized concurrently. There's little doubt that the performance in the remaining functions will suffer. The concept of dominance is used by the majority of algorithms in the selection process. One way to phrase a multi-objective minimization problem is as follows:

Minimize

$$\vec{f}(\overrightarrow{dv}) = [f1(dv_1 \dots dv_m), \dots . fn(dv_1 \dots dv_m)] \quad (7)$$

$$\overrightarrow{dv} = (dv_1 \dots dv_m) \in X$$
$$\overrightarrow{f} = (f_1 \dots f_n) \in Y$$

Where 'm' is set of decision variables and 'n' are the set of objectives. $\overrightarrow{dv}$ denotes the decision vector, $\overrightarrow{fv}$ denotes the objective vector, X is the variable space and Y is the objective area. The dominating points are those in which the decision vector $\overrightarrow{dv}$ has better objective than any other decision vector. Find every non-dominated, multi-objective solution set within a solution set. Take the first-choice variable first. For domination, compare the first variable with every other variable that is still present. When the first solution, dv_i, achieves its goals more successfully than the second, dv_j, it dominates the other. With the exception of the marked solution, all other solutions are non-dominated. Mark the dominating solution. Let's say we have a certain number (n) of physical machines that are currently running applications. Assume that each application is handled as a distinct

virtual machine (VM) that needs to be run. A multidimensional vector packing problem is mapping a virtual machine to a PM. The various dimensions are represented by different CPU and memory utilization rates. For illustration, consider two requests: one for a virtual machine (VM) with 20% CPU and 30% memory, and another for a VM with 35% CPU and 40% memory. Then, 55% of the server's usage will be computed as CPU and 70% as memory. For every dimension, the resources will be used up to 90% of the time. In order to prevent server performance degradation and potential VM migration, we must impose a 90% utilization boundary, which is less than 100%. Server Consolidation does not utilize the entire 100 percent resources of its server in order to avoid the performance degradation [48].

4 Preliminaries of fuzzy cuckoo search (FCS) and fuzzy firefly colony (FFC) optimization

In order to address local minima and convergence, we concentrated on using the adaptive genetic algorithm (AGA); however, it required a notably large computation time over extremely high dynamism and with multiple VM-host setup, indicating a higher probability of downtime. In the case of fog computing or IoT-based cloud (IoT-cloud, for short), where various devices use data randomly and even very dynamically, its severity may even be more frequent. Creating a metaheuristic model that incorporates a diverse and small but highly significant initial population is a workable solution that this research suggests. In this reference, the Cuckoo Search based meta heuristic model was taken into consideration with Fuzzy rules. In this model, the solution probability is estimated iteratively by multiple agents. Furthermore, the Fuzzy rules, Fuzzification, and Defuzzification concepts have all been optimized in the suggested Cuckoo Search model. These innovations make it possible for our suggested model to demonstrate quick and precise VM-host mapping, which aids in the achievement of SLA- sensitive VM migration scheduling.

4.1 Cuckoo search algorithm

Enabling a global controller to execute dynamic VM placement is a technique utilized by VM migration technologies [25]. The suggested work is expanded to include the best possible VM selection for starting the migration process with the Fuzzy Cuckoo Search algorithm. By taking into account the minimal parameters, the population-based technique resolves the optimization problem. The suggested algorithm takes into account the behavior of the birds, which lay one egg at a time. Nests are chosen at random for placement, and they are subsequently dumped. The following generation chose the best egg and determined which nest was the best. The probability of notifying an egg in a fixed nest is [0,1]. In such a case, the egg is either taken out of the host nest or the nest-hosting bird leaves the existing nest and onstructs a new one. The pertinent solution is found and added for the virtual machine location. The migration process is started by choosing the appropriate virtual machine (VM) while taking into account the probability factors. The host PM then either chooses the migrated VM or moves on to the next VM based on selection probability in order to determine the best optimal solution for VM Placement.

The ordinal measures of the CS architecture are illustrated in order to comprehend the suggested Fuzzy CS algorithm. The nest containing the egg signifies the correct answer, and the cuckoo lays one egg in direct proportion to the one solution. One egg leads to one solution, and each egg in the nest represents a workable solution. Every nest and egg location denotes a solution [4]; the first solution is chosen at random, and the procedure for updating the position is as follows:

$$o_i(j+1) = o_i(j) + \delta \otimes \varrho(\varphi) \quad (8)$$

where $o_i(j+1)$ represents the (j+1) th generation's nest position. $o_i(j)$ represents the nest position for the j^{th} generation. The random size vector is produced using the levy distribution ϱ(φ) and step size δ. Multiplication is performed using ⊗, and global search optimization is carried out. Levy Flight is a significant concept that is represented mathematically as:

$$\varrho(\varphi) \sim \upsilon = t^{-\psi} \quad (1<\varphi<3) \quad (9)$$

The subsequent steps are taken into consideration when creating a random walk process, and the levy walk is employed to generate the solutions. The solutions found by taking the Levy process into consideration speed up both the local and global searches. Randomization is the local solution, and it is sufficiently removed from the optimal and best current solution. This procedure makes sure that the problem of VM placement resolves the issue of locating the local optimal solution. Every simulation round, the levy flight is taken into account for an updated solution, and the optimal solution is found. Later, until the termination criteria are satisfied, the solutions that are kept or removed at random.

$$wt = wt_{max} - (wt_{max} - wt_{min}) X \frac{S_{current}}{S_{max}} \quad (10)$$

where wt_{max} and wt_{min} denote the maximum and minimum coefficients, respectively. The optimization technique's maximum number of simulation rounds is denoted by S_{max}, and the current simulation round is represented by $S_{current}$. New positions are found and the fitness function for the updated positions is calculated using the aforementioned equations. A better nest is chosen for the population by comparing the fitness values of the past and present, and the new nest position is indicated as follows:

$$G_m = [y_1(m), y_2(m), y_3(m), \dots, y_k(m)]^T \quad (11)$$

G_m is the current nest position.

$$G_{m+1}=[y_1(m+1),\ y_2(m+1),\ y_3(m+1),\dots,y_k(m+1)]^T \quad (12)$$

P_a, the probability factor, is then used to determine the nest status. The nests need to be replaced if the hosts identify them as such. To formulate this, a ℓ-dimensional vector is utilized as $\boldsymbol{R_l}= [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}, \dots, \boldsymbol{r_k}]$.If $\boldsymbol{R_{j>}P_d}$, the nest position is changed randomly. Next, using the probability factor $\boldsymbol{P_d}$, the nest status is determined. $\boldsymbol{G_{m+1}}$ is the new nest position. Changes must be made to the nests if the hosts identify them as such. This is formalized as ι-dimensional vector. The best nest position is found out by comparing the previous and present fitness of the nest and better nest is chosen for new generation.

$$N_{m+1} = [y_1(m+1), y_2(m+1), y_3(m+1), \dots, y_k(m+1)]^T \quad (13)$$

Where N_{m+1} is the next generation solution. The stopping condition is checked in the last phase. The solution is found if the stopping condition is met by the current solution; if not, move on to determining the initial population's fitness value.

Algorithm 1: Traditional Cuckoo Search Optimization algorithm

1. Calculate the Fitness Function for the initial position
2. Identify the Optimal nest position y_{Gb}^0
3. Update nest position based on the weight coefficient w
4. Calculate the fitness function for the new position
5. Compare the present and past fitness values and select new position N_{m-1}
6. If the termination condition is satisfied, then the optimal output position of the nest is achieved.
7. Terminate the process otherwise repeat 5.

The classic CS algorithm is a very simple algorithm in terms of parameters as shown in Algorithm 1. Some of the shortcomings of the traditional CS algorithm are that the cuckoos don't talk to each other about the solution; There is no use of global nest Levy flight guidance with a fixed step size. We suggest a Fuzzy Cuckoo Search based on fuzzy set-based population to remedy this shortcoming. The selection of either server or a VM for placement depends on a random number q_0.

The suggested migration model incorporates Firefly swarm optimization to improve the Cuckoo search optimization. This research study is distinctive in that it uses the FFC algorithm in the cuckoo search position. The ideal solution is updated with convergence using the positions of the fireflies after the initial search using levy flight. By avoiding local optima solutions through the use of random elimination in cuckoo search optimization, the hybrid strategy improves overall searching performance in discovering the best solutions. Figure 1, shows the process flow of the hybrid optimization model for choosing virtual machines during the migration process.

4.2 Firefly colony algorithm

Inspired by firefly behaviour, Yang created the firefly algorithm (FA), a swarm intelligence-based metaheuristic technique. Simple insects that live in groups are fireflies. The flashing of fireflies serves as a signalling mechanism to draw in other files. These patterns of flashing and firefly behaviour form the foundation of the firefly algorithm. The following are the features of the typical firefly algorithm [47]:

(1) Due to their unisex nature, fireflies are drawn to other fireflies regardless of their gender.

(2) Brighter fireflies draw the attention of the less brilliant ones. When the distance rises, a firefly's attraction and brightness diminish, and if there are no brighter fireflies in their path, they will begin to move aimlessly.

(3) A firefly's brightness is determined by the objective function.

The firefly colony optimization is a swarm intelligence-based metaheuristic approach which is based on ACO technique. Firefly Colony optimization is inspired by the flashing behaviour of fireflies. The solution is constructed using firefly state transition rule [39].

$$i = \begin{cases} argmax_{u\epsilon\Omega_k(j)}\left\{\beta_{uj} * e^{-\gamma RW_{uj}^m}\right\}, q \le q_{0,} \\ explore\ e \qquad otherwise \end{cases} \quad (14)$$

The heuristic information $\frac{1}{\gamma RW_{uj}^m}$ is termed as η_{ij}.

γ is the absorption coefficient of the light and it is initialized to 1. In equation 16, if q is less thanq_0, then a VM u with higher attractiveness is chosen from the set of eligible virtual machines. If q is greater than q_0, then the cumulative sum of attractiveness of all eligible VMs are obtained and then the VM having the higher attractiveness than a generated random number is chosen to be the next VM for placement. The cumulative sum of the attractiveness is obtained by:

$$\text{Attractiveness vector} = cumsum\left(\beta_{ij}^k\right), i \in \Omega_k(j)$$

$$\beta_{ij}^k(t) = \begin{cases} \beta_{ij} * \eta_{ij}, i\epsilon\Omega_k(j), \\ 0\ otherwise \end{cases} \quad (15)$$

$$\beta_{ij} = \begin{cases} \sum_{u\in\Omega_k(j)} \beta_{ui} \\ 1\ Otherwise \end{cases}, \quad \text{if} \quad \Omega_k(j) - \{i\} \neq 0 \quad (16)$$

The local update attractiveness is

$$\Delta\beta_{ij}(t) = \begin{cases} ff_{sc}(s^{Gb}), if\ VM\ is\ selected \\ 0\ otherwise \end{cases} \quad (17)$$

$$\beta_{ij}(t) = \beta_{ij}(t) = \alpha\left(rand - \frac{1}{2}\right)\beta_{ij}(t-1) + \beta_0 \quad (18)$$

where α is the attractiveness decay parameter, the initial value for β_0 is calculated using $\beta_0 = \frac{1}{nRW(S_0)}$, The global update of attractiveness is:

$$\beta_{ij}(t) = \alpha\left(rand - \frac{1}{2}\right)\beta_{ij}(t-1) + \Delta\beta_{ij}^{best}\ , \quad (19)$$

The next VM_i to be placed in the currently chosen server should be decided as described in [31]. This section explains about the conditions generated to determine which VM_i should be selected for the Chosen Physical Machine.

The process of converting a crisp input value into a fuzzy set is called "fuzzification." The terms "linguistic variable" and "membership function" are crucial in fuzzy logic.

A linguistic variable is a language structure that is further subdivided into several subfields. It is a part of the set and the range of true and false values. The linguistic variable stores the VM Placement's input and output. Rather than using numbers to represent the value, it uses words. Four sets of the linguistic variable of load—Very Low, Low, High, and Very High—are present based on the position of the virtual machine.

The Fuzzy rules to be followed to place a VM on a PM are given below. This hybrid technique uses the minimal and max-min operations in implication and composition should be assigned to an individual server j.

1. If β_{ij} is medium and η_{ij} is low then the efficacy e_{ij} of choosing VM_i is very very low.
2. If β_{ij} is medium and η_{ij} is low then the efficacy e_{ij} of choosing VM_i is very low.
3. If β_{ij} is high and η_{ij} is low then the efficacy e_{ij} of choosing VM_i is low.
4. If β_{ij} is low and η_{ij} is medium then the efficacy e_{ij} of choosing VM_i is low.
5. If β_{ij} is medium and η_{ij} is medium then the efficacy e_{ij} of choosing VM_i is medium.
6. Ifβ_{ij} is high and η_{ij} is medium then the efficacy e_{ij} of choosing VM_i is high.
7. If β_{ij} is low and η_{ij} is high then the efficacy e_{ij} of choosing VM_i is high.
8. If β_{ij} is medium and η_{ij} is high then the efficacy e_{ij} of choosing VM_i is very high.
9. If β_{ij} is high and η_{ij} is high then the efficacy e_{ij} of choosing VM_i is very high.

In a fuzzy inference system, the hybrid technique of using the minimal and max-min operations in implication and composition, respectively, allow for a more flexible and efficient way of reasoning with uncertain or imprecise information. The minimal operator is used in the implication step to determine the degree to which each rule is satisfied based on the membership value of the input variables. This allows for a more nuanced consideration of the rules and their applicability to the current situation. The max-min operator is then used in the composition step to combine the output membership values of the rules and determine the overall output value of the system. This method takes into account the contributions of all pertinent rules and ensures that no single rule has an undue influence on the output value. By combining these two operations in a hybrid technique, the fuzzy inference system can achieve a more accurate and robust performance in dealing with fuzzy input data and making decisions based on uncertain or incomplete information.

The maximum defuzzification method, similar to fuzzy

Table 2: Fuzzy rules for FCSFFC algorithm

Antecedent Part		Consequent Part
β_{ij} (Attractiveness)	η_{ij} (Heuristics)	e_{ij} $(Efficacy)$
Low	Low	$Very\ Very\ Low$
$Medium$	Low	$Very\ Low$
$High$	Low	Low
Low	$Medium$	Low
$Medium$	$Medium$	$Medium$
$High$	$Medium$	$High$
Low	$High$	$High$
$Medium$	$High$	$Very\ High$
$High$	$High$	$Very\ Very\ High$

strategy, is used on fuzzy probability to select the next virtual machine (i) for the current server (u) among u eligible virtual machines. The antecedent part of the fuzzy rule comprises heuristic information and attractiveness, while the subsequent part discusses the effectiveness of selecting the next virtual machine. For fuzzy implication, this method uses the minimum operation, and for composition, it uses the max-min operator. As the maximum efficacy for every virtual machine i, we finally arrive at $ek_{ij.}$ In this instance, we present two approaches, the fuzzy strategy and the fuzzy probable strategy in Eqn. (16) to carry out the exploitation and exploration process of choosing which virtual machine VM_i to install on the

$$I=\begin{cases} \text{Fuzzy strategy}, & q \le q_0\ , \\ \text{Fuzzy probable strategy}, & q > q_0 \end{cases} \quad (20)$$

Each strategy produces a discrete number that indicates which virtual machine should be installed on the current

server j next. The fuzzy rules for deciding which Virtual Machine to be placed next in the Physical Machine is given in Table 2. Equation (14) is applied to decide which VM_i is chosen for a server j. As we have seen, fuzzy strategy is chosen for exploitation and fuzzy probable strategy is chosen for exploration process.

4.3 Proposed hybrid optimization algorithm (FCSFFC)

The proposed hybrid optimization algorithm, Fuzzy Cuckoo Search with Fuzzy Firefly Colony (FCSFFC), integrates the exploration capabilities of the Cuckoo Search (CS) algorithm with the exploitation strengths of the Firefly Colony (FFC) algorithm, aiming to enhance virtual machine (VM) placement in cloud data centers. This section delves into the specific mechanisms and procedural steps of the hybrid algorithm, focusing on how the Cuckoo Search initiates the exploration phase, and how the Firefly algorithm refines the solutions during the exploitation phase.

Cuckoo search exploration

The Cuckoo Search algorithm models the movement of virtual machines using two primary perturbation functions: one for VM selection and another for server selection. The decision of whether to prioritize servers or VMs is based on a randomly generated number between 0 and 1. If the number is less than 0.5, the VM selection function is invoked; otherwise, the server selection function is chosen [4]. These functions are crucial as they determine the migration of VMs by evaluating resource utilization across servers. The movement of VMs within the cloud infrastructure is guided by sorting and optimizing the VMs based on resource requirements. The output of this selection process is a crisp number, which identifies the next VM to be placed on a server. The optimization process is iterative, ensuring that VMs are placed in a manner that balances resource utilization, reduces power consumption, and minimizes migration costs. The Cuckoo Search algorithm begins by initializing parameters such as the population size and the number of iterations. Using objective functions like power consumption, resource wastage, and migration cost, the algorithm evaluates the fitness of each solution. The optimal positions of individuals in the population are computed in each iteration, and solutions are updated based on the Levy flight mechanism—a random walk that aids in escaping local minima. If the fitness of a new solution is better than the old one, the old solution is discarded. This process continues until the exploration phase is complete. VM selection identifies VMs for migration based on resource needs and load balancing, while server selection finds optimal target servers with sufficient resources to minimize wastage and enhance efficiency. Together, they ensure effective VM placement, reducing power consumption and migration costs.

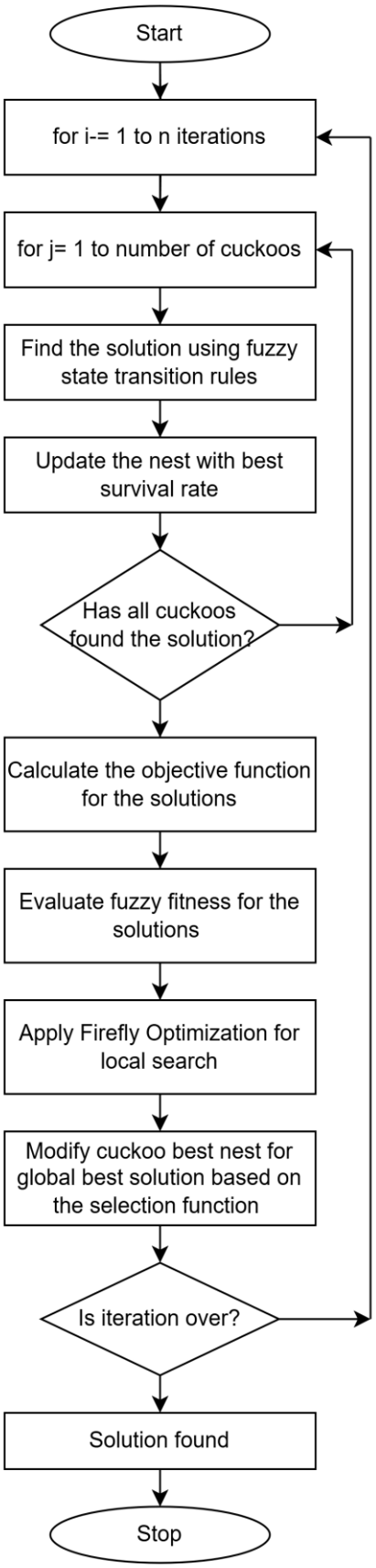

Figure 2: Flowchart for FCSFFC optimization VM placement

1. ***VM selection function:*** In a solution, a random selection of VMs is made from the current population, and they are removed out of the servers. Next, the involved servers are arranged according to the total resource usage in decreasing order. Subsequently, 132 the chosen virtual machines are likewise arranged in descending order of total resource requirements. Each virtual machine is now placed on the first server that has capacity for them. It is placed on a server chosen at random if none of the involved servers can host it. If one of the virtual machines is chosen from the perfectly placed server, it is deleted and another is chosen.

2. ***Server selection function:*** Once a set of b servers is chosen at random, all of its virtual machines are deleted, leaving the set of servers' empty. Put the virtual machines in decreasing order of overall resource requirements. Subsequently, each virtual machine is now placed on the first server that has capacity for the VMs. It is placed on a server chosen at random if none of the involved servers can host it.

Procedure for virtual machine placement using firefly algorithm for exploitation

The algorithm begins by initializing parameters such as the number of fireflies, attractiveness coefficient, and light absorption coefficient, along with generating initial random solutions. In the iterative process, fireflies move towards brighter (more attractive) fireflies, with their positions updated based on a function that considers brightness and distance. Optional local search can refine solutions further. The algorithm recalculates and updates the brightness of fireflies, ranks them accordingly, and continues until a stopping criterion is met, such as reaching the maximum number of iterations. This approach leverages the simplicity and flexibility of FA, offering global search capabilities and scalability across various problem sizes. Its adaptability allows integration with other optimization methods and it has been successfully applied in fields such as engineering design, scheduling, machine learning, and resource management, demonstrating its effectiveness in achieving optimal solutions and enhancing performance. The hybrid FCSFFC algorithm is designed to address multi-objective optimization problems in VM placement. By combining the exploration capabilities of Cuckoo Search with the exploitation strengths of Firefly optimization, the algorithm effectively balances the trade-offs between different objectives, such as minimizing power consumption and reducing migration costs. The fuzzy fitness function evaluates the quality of solutions based on their performance across multiple objectives. This approach ensures that the selected solutions are not only optimal in terms of individual objectives but also robust and adaptable to changing conditions in the cloud environment.

In FCSFFC algorithm, fuzzy logic enhances decision-making for VM placement and migration by processing imprecise and uncertain data. It begins with fuzzification, which translates exact numerical metrics related to VM and server conditions into fuzzy sets using qualitative terms like "Low," "Medium," and "High." This approach allows the algorithm to handle real-world uncertainties more effectively. Linguistic variables are employed to represent the load and attractiveness of VMs and servers, making complex data easier to interpret. For example, load conditions are categorized into "Very Low," "Low," "High," and "Very High" based on resource usage and server capacity. This categorization simplifies the processing of these metrics. Fuzzy rules are then applied to evaluate the suitability of placing a VM on a particular

Algorithm 2: Hybrid FCSFFC algorithm for VM placement

1. Initialize optimization parameters, population, weight coefficient
2. Generate random VM requests
3. Begin
4. For i= 1 to n
5. Determine the population's initial fitness
6. Calculate the global optimal individual position
7. Loop1: Construct cuckoo parameters
8. Calculate the position of individual levy flight search applying (9)
9. Evaluate ($\boldsymbol{fit_{new}}$ $\boldsymbol{and}$ $\boldsymbol{fit_{old}}$)
10. If ($\boldsymbol{fit_{new}} > \boldsymbol{fit_{old}}$)
    New generation= current position
    Else
    New generation = old position
    If $\boldsymbol{q_0}$ < 0.5, select a server else select VM for placement.
11. Initialize FFC parameters
12. Update the heuristic parameters of the individuals using (14).
13. For each Firefly k=1 to N
14. Repeat
15. For each $\boldsymbol{PM_j}$=1 to m do
16. Issue new server
17. Determine the eligible virtual machines using Fitness function
18. Find iterative best Global optimal and individual solutions using (18) and (19)
19. Update the fitness function of new and old nest.
20. Obtain optimal individual and optimal nest
21. Calculate the entire population
22. While termination criteria reached, terminate the current process
23. Else
    Goto l0
24. End

server. These rules take into account factors such as the VM's attractiveness and the server's heuristics. For instance, if a VM's attractiveness is medium and its heuristic is low, the rule might suggest a low efficacy for that VM's placement. These rules are designed to manage uncertainty by providing flexible and adaptive criteria for decision-making. The fuzzy inference system processes these rules and combines their outcomes using operators. The minimal operator is used to assess how well each rule is satisfied based on input values, while the max-min operator aggregates the results to make a decision. This system helps in making nuanced and well-informed decisions by considering all relevant factors and their interactions. Finally, defuzzification converts the fuzzy results back into precise actions, determining which VM should be placed on which server. This step ensures that the decisions are actionable and practical, contributing to better resource management, reduced migration costs, and overall improved system performance.

Procedure for virtual machine placement

The overall procedure for VM placement using FCSFFC is illustrated in Algorithm 2. The process begins with the initialization of PMs, VMs, and iterations, followed by the setup of Cuckoo Search parameters. Lines 1-4 initializes the number of PMs, VMs and number of iterations. Line 4 sets up the cuckoo search algorithm parameters. Using the objective functions specified for each nest solution, the algorithm evaluates the quality of the solutions based on their fitness values, guiding the search process to optimize the overall performance. Then the population's fitness, is determined which includes the objective functions of power consumption, resource wastage and energy consumption. solution is considered to be of a new generation. If not, the previous solution is kept in place for the following generation. The next statement computes the globally optimal individual positions for each iteration. Equation (7) is applied to the Levy flight walk in an attempt to find the solution. If the new solution's fitness is found to be lower than the old one, the old solution is discarded, and the new solution is accepted. For every iteration, the exploration phase has concluded, and now that the exploration phase is complete, the Firefly algorithm is executed to determine whether the solutions produced by the Cuckoo Search algorithm can be further refined for optimal performance. This procedure is referred to as exploitation. Repeat the above procedure until the stopping condition is satisfied. Equation (20) is applied to decide which $\boldsymbol{VM_i}$ is chosen for a server j. The algorithm then iteratively evaluates the fitness of each solution, updates the positions of individuals, and refines the solutions using the Firefly algorithm. The stopping condition, such as the maximum number of iterations or a convergence threshold, determines when the algorithm terminates. The final solution represents the optimal VM placement configuration, balancing the competing objectives of power efficiency, resource utilization, and migration cost.

4.1 Experimental set up

The simulation environment is configured with an Intel i3 processor and 8 GB of RAM, running on the Windows 10 operating system. Five datacenters were simulated with 10 physical machines and 50 virtual machines. The total number of tasks is varied from 25 to 75, and 150 Virtual Machines were used. Each VM was allocated 2GB of memory, and each host had 4GB. Host MIPS was set to 10,000, and VM MIPS to 1,500. VM Bandwidth was 100 Mbit/s, while Host Bandwidth was 1 Gbit/s. The load factor, which is based on VM resource utilization to complete user tasks, was calculated.

In FCSFFC algorithm, fuzzy logic enhances decision-making for VM placement and migration by processing imprecise and uncertain data. It begins with fuzzification, which translates exact numerical metrics related to VM and server conditions into fuzzy sets using qualitative terms like "Low," "Medium," and "High." This approach allows the algorithm to handle real-world uncertainties more effectively. Linguistic variables are employed to represent the load and attractiveness of VMs and servers, making complex data easier to interpret. For example, load conditions are categorized into "Very Low," "Low," "High," and "Very High" based on resource usage and server capacity. This categorization simplifies the processing of these metrics. Fuzzy rules are then applied to evaluate the suitability of placing a VM on a particular server. These rules take into account factors such as the VM's attractiveness and the server's heuristics. For instance, if a VM's attractiveness is medium and its heuristic is low, the rule might suggest a low efficacy for that VM's placement. These rules are designed to manage uncertainty by providing flexible and adaptive criteria for decision-making. The fuzzy inference system processes these rules and combines their outcomes using operators. The minimal operator is used to assess how well each rule is satisfied based on input values, while the max-min operator aggregates the results to make a decision. This system helps in making nuanced and well-informed decisions by considering all relevant factors and their interactions.

Finally, defuzzification converts the fuzzy results back into precise actions, determining which VM should be placed on which server. This step ensures that the decisions are actionable and practical, contributing to better resource management, reduced migration costs, and overall improved system performance.

4.2 Computational study

The load analysis of the algorithms is illustrated in Fig. 3. The study began with 25 tasks and progressively included 50 and 75 tasks, carefully evaluating the load parameter for each scenario. The results demonstrate that our FCSFFC algorithm consistently maintains a lower load compared to other methods when selecting and migrating VMs based on resource requirements. In terms of load, the Fuzzy Cuckoo Search Fuzzy Firefly Colony Optimization Algorithm (FCSFFC) consumes the least resources, with

a load value of 0.0025. This indicates that FCSFFC is the most efficient algorithm in terms of resource utilization for VM placement and migration among the algorithms compared.

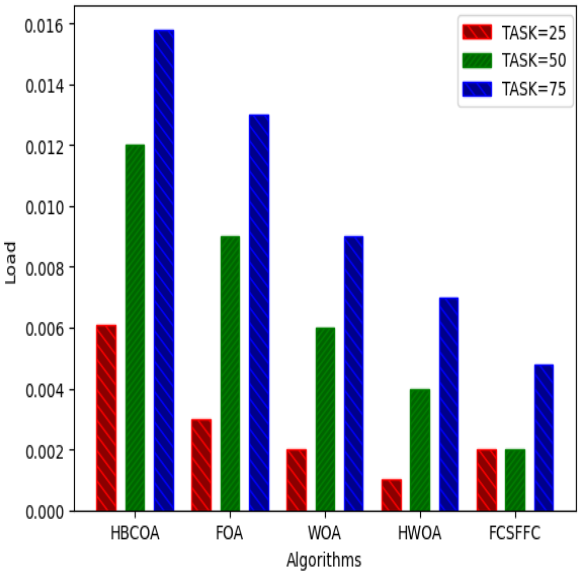

Figure 3: Load analysis

The lower load means it requires minimal computational resources, leading to better overall performance and reduced strain on the cloud infrastructure. This is largely due to the algorithm's efficient VM-server matching process, which minimizes the frequency migrations.

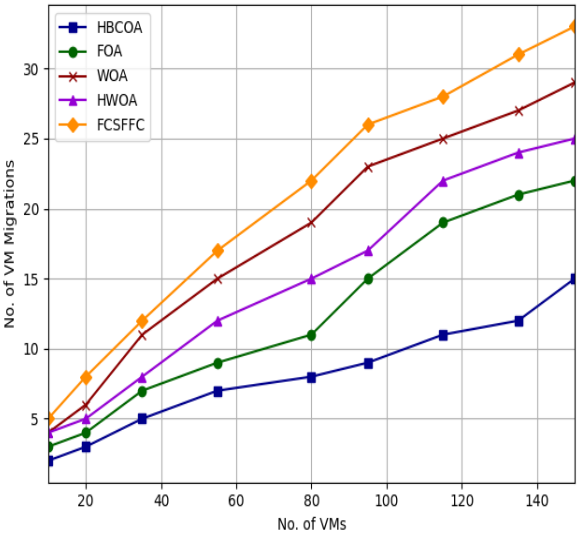

Figure 4: Migration analysis

As a result, the overall system stability is enhanced, reducing the operational overhead associated with VM migrations. Figure 4 presents the migration analysis, highlighting the superior performance of the FCSFFC algorithm in comparison to alternative algorithms. The reduced migration frequency improves resource utilization and also minimizes disruption to ongoing processes by performing only necessary migrations and avoiding unnecessary migration requests.

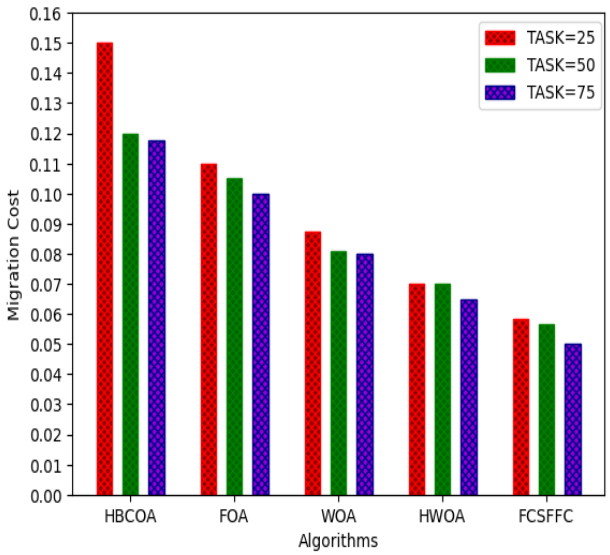

Figure 5: Migration cost analysis

Figure 4 depicts the analysis of migration costs for the proposed model compared to other models. The analysis considers both the total number of VMs and those specifically migrated to handle requested tasks. By selecting optimal VMs, the proposed method significantly reduces unnecessary migrations, demonstrating superior performance compared to alternative models. The number of virtual machines (VMs) moved for the requested tasks as well as the actual VMs needed for the task were noted. When compared to other algorithms, it is found that the FCSFFC algorithm achieves the least amount of migration. When comparing the migration cost of state-of-the-art techniques, optimal VM selection minimizes the number of migrations and makes the migration more effective by performing only necessary migrations and avoiding unnecessary migration requests. Figure 5 depicts the analysis of migration costs for the proposed model compared to other models. The analysis considers both the total number of VMs and those specifically migrated to handle requested tasks. By selecting optimal VMs, the proposed method significantly reduces unnecessary migrations, demonstrating superior performance compared to alternative models. Also, Figure 5, utilizes the ratio of completed migrations to the total number of migrations within the cloud environment to determine the migration cost impacting the performance of the hybrid optimization FCSFFC algorithm. Based on the migration approaches, the suggested model's migration cost is, in comparison, 70% less than HBCOA, 60% less. The results underscore its effectiveness in enhancing overall system performance and cost efficiency compared to existing models.to calculate the Migration Cost.

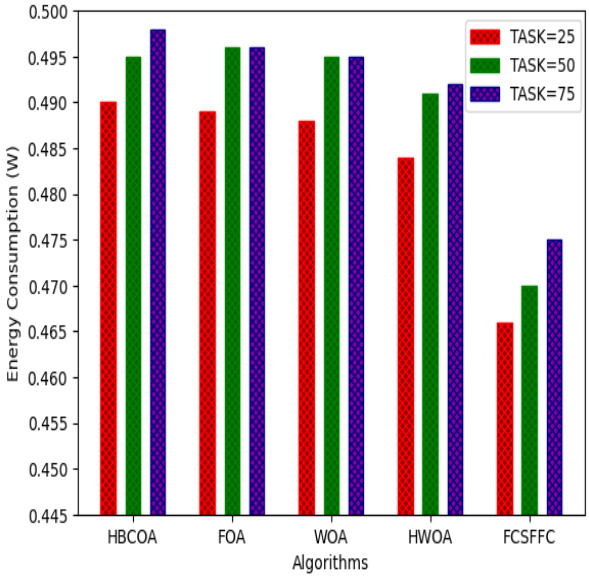

Figure 6: Energy consumption analysis

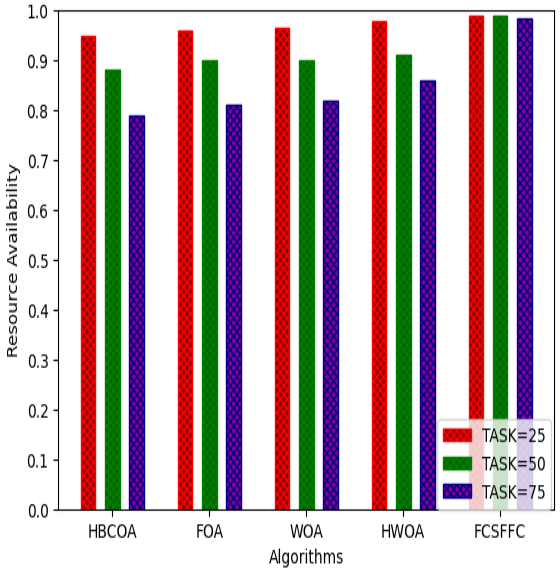

Figure 7: Resource availability analysis

For 75 tasks, the migration cost is 0.048; for HWOA, it is 0.067; for WOA, it is 0.083; for FOA, it is 0.10; and for HBCOA, it is 0.115. These numbers demonstrate how well the suggested than FOA, 50% less than WOA, and 20% less than HWOA. Figure 6 compares the energy consumption of the current methods with the proposed hybrid optimization approach. For 25 tasks, the proposed hybrid optimization achieves the lowest energy consumption at 0.465 W, followed by 0.47 W for 50 tasks, and 0.475 W for 75 tasks. In comparison, the HWOA method consumes 0.488 W. The average energy consumption for the proposed technique remains at 0.47 W, whereas other methods, including WOA, FOA, and HBOA, show an average energy consumption of 0.49 W, with minimal variation. Incorporating the Interquartile Range (IQR) into the Fuzzy Cuckoo Search Fuzzy Firefly Colony Optimization algorithm significantly improves its capability to handle dynamic loads in cloud data centers. By analyzing historical workload data, including CPU utilization, memory usage, and network bandwidth, IQR helps identify the distribution and variability of these metrics. This understanding allows for better trend detection and anomaly identification. Integrating IQR into the Fuzzy Cuckoo Search Fuzzy Firefly Colony Optimization framework enhances the fuzzy logic. Notably, the proposed model demonstrates a 2.2% reduction in energy consumption compared to HBCOA, FOA, and WOA, and a 1.8% reduction when compared to the HWOA algorithm. The Firefly algorithm, known for its superior local search capabilities, complements CSO by refining these solutions to find even more energy-efficient configurations. Firefly algorithm's attractiveness-based movement helps in focusing on the most promising areas of the search space, which leads to incremental improvements in energy consumption across different task loads. The synergy between CSO's exploration and the Firefly algorithm's exploitation ensures that the proposed model consistently outperforms other methods in both energy consumption and overall system performance. As a result, it effectively balances efficiency and resource utilization across varying workloads.

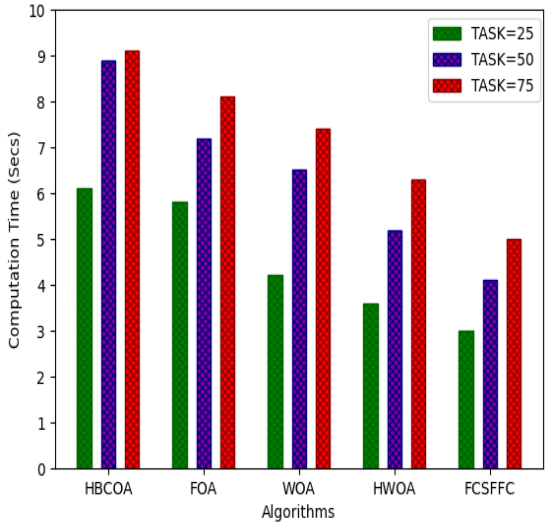

Figure 8: Computation time analysis

The impacts of the proposed optimization based VM migration and that of the existing VM migration algorithms are analyzed in terms of resource availability in Fig. 7. The proposed method's time for 75 tasks is 5 seconds; for HWOA, it takes 7.4 seconds. The table provided compares several algorithms in terms of load, energy consumption, migration cost, and execution time. When it comes to resource availability, the FCSFFC

Table 3: Overview of the energy consumption, migration cost, resource availability, and comp. time

SNo	Algorithm	Load	Energy Consumption	Migration Cost	Resource Availability	Computation Time in Sec.
1	Hybrid bee Colony optimization Algorithm (HBCOA)	0.010367	0.4850	0.1370	0.8522	8.252
2	Firefly Optimization Algorithm (FOA)	0.00732	0.4839	0.1155	0.8822	7.128
3	Whale Optimization Algorithm (WOA)	0.005473	0.4829	0.0942	0.9022	6.232
4	Hybrid Whale Optimization Algorithm (HWOA)	0.002793	0.4678	0.0688	0.9186	5.332
5	Fuzzy Cuckoo Search Fuzzy Firefly Colony Optimization Algorithm (FCSFFC)	0.0025	0.4593	0.0655	0.9814	4.003

algorithm demonstrates superior efficiency. This is reflected in its lower load value (0.0025), indicating that it consumes fewer resources compared to the other algorithms. The reduced load means that the FCSFFC algorithm is more effective at matching VMs with servers that have sufficient resources, which in turn reduces the strain on the physical machines. This efficient resource allocation minimizes the need for frequent migrations, leading to a lower migration cost (0.0467) and faster execution time (4.003 seconds). In contrast, the HBCOA and FOA algorithms show higher load values of 0.010367 and 0.00732, respectively, suggesting that they require more resources to achieve the same tasks. This higher resource demand can lead to increased migration costs and longer execution times. As a result, while these algorithms may still perform adequately, their resource management is less optimal compared to FCSFFC, leading to less efficient overall performance in cloud environments. The table highlights the importance of selecting algorithms that not only minimize energy consumption and migration costs but also efficiently manage and allocate available resources, ensuring better performance and scalability in practical applications.

Figure 8 shows the total computation time for both the suggested and current approaches. The proposed method's maximum computation time for 75 tasks is 5 seconds; for HWOA, it takes 7.4 seconds, For WOA, it takes 8.2 seconds, and HBCOA, it is computed for 9.2 seconds. Figure 7 illustrates a notable advantage of the proposed method in terms of computation time. The proposed algorithm achieves a maximum computation time of 5 seconds for 75 tasks, which is significantly faster than the Hybrid Whale Optimization Algorithm (HWOA), which requires 7.4 seconds, and the Whale Optimization Algorithm (WOA), which takes 8.2 seconds. The Hybrid Bee Colony Optimization Algorithm (HBCOA) has the longest computation time at 9.2 seconds. This reduced computation time of the proposed method not only improves operational efficiency but also enhances its suitability for real-time applications where rapid decision-making is crucial. The efficiency gains in computation time can lead to faster responses and better scalability, making the proposed algorithm a competitive choice for dynamic cloud environments with high task loads.

Table 3 provides an overview of the Energy Consumption, migration costs, Resource Availability, and Migration Cost for both the proposed and existing models' performance analyses. The average values are displayed according to the previously presented findings from various tasks. The outcomes unequivocally show that, the suggested hybrid optimization outperforms the current techniques in every element. It is clear from the results that the FCSFFC optimization algorithm achieves the lowest possible. The task's resource request, the VM selection, and the migration from one resource pool to the current resource pool all affect the computation time. The time required to move a set of tasks from one virtual machine to another is commonly referred to as computation time. It is found that the computation times for all methods are lowest for 25 tasks, computation time. Though the computation time is noted with the fewest elements, it may differ in a real-time setup. The average values are shown in accordance with the results from different tasks that were previously presented. The results clearly demonstrate that the proposed hybrid optimization works better than the existing methods in all aspects, owing to a unique analysis of migration requests the quality of service. FCSFFC achieves the lowest load parameter, indicating more efficient VM placement and a reduced need for frequent migrations. It also shows a substantial reduction in migration costs, which enhances overall system efficiency. The algorithm's energy consumption is the lowest among the compared methods, contributing to reduced operational costs and greater sustainability. Additionally, FCSFFC results in the highest resource availability, indicating effective resource utilization and minimized unnecessary migrations. Its computation time is the shortest, reflecting its efficiency in processing and decision-making. The FCSFFC algorithm integrates fuzzy logic with hybrid metaheuristic techniques, offering a novel approach that adapts to varying load conditions more effectively than traditional methods. Unlike static

threshold-based approaches, FCSFFC uses dynamic fuzzy rules and hybrid optimization to improve decision-making processes. This innovation includes a stochastic prediction model for overload detection, which estimates CPU utilization dynamically. This model overcomes the limitations of static thresholds, enhancing the accuracy of overload detection and SLA compliance. The dynamic thresholding scheme adjusts CPU utilization thresholds based on real-time variations, improving the handling of fluctuating load conditions. The use of Inter Quartile Range and modified Local Regression Robust techniques for adjusting thresholds further refines this approach, offering a robust mechanism for managing CPU utilization and preventing SLA breaches.

In comparison to existing studies, the proposed algorithm demonstrates significant improvements in key performance metrics, such as energy consumption, resource wastage, and migration cost, by refining fuzzy rules and membership functions based on the observed variability, which leads to more accurate load predictions. Moreover, applying IQR-based adjustments to the Cuckoo Search and Firefly Algorithm components ensures that optimization strategies are dynamically adapted to current conditions, improving resource allocation and minimizing migration costs. This dynamic approach to load prediction aids in balancing resources more effectively across physical machines, thus optimizing overall system performance. Continuous monitoring and comparative analysis further validate the effectiveness of IQR integration, demonstrating its contribution to enhanced load prediction accuracy and overall system efficiency.

In the Chaotic Particle Swarm Optimization (CPSO) algorithm with adaptive mutation [9] focuses on better utilization but also significantly reduces power. The hybrid meta-heuristic methods, such as the one combining Sine Cosine and Ant Lion Optimizer (SCA-ALO) for server load balancing [14], primarily address server load balancing and resource waste reduction. However, these methods often struggle with the trade-offs between multiple objectives. In contrast, our algorithm effectively balances these trade-offs, leading to a notable reduction in energy consumption and resource wastage Moreover, while algorithms like the Marine Predator Algorithm [20] and the enhanced Whale Optimization Algorithm (WOA) [21] have been adapted for multi-objective optimization, they do not specifically address the challenges of virtual machine migration in cloud data centers. Our approach, by integrating the strengths of Cuckoo Search and Firefly Colony Optimization, offers a more tailored solution for VM migration, resulting in better overall performance.

The effectiveness of the approach is further emphasized when compared to the ACOSCA algorithm [29], which achieved a 24% reduction in power consumption. The suggested algorithm’s average energy consumption was 2.2% lower, and the migration cost was notably 37% lower than that of the other compared algorithms. The proposed algorithm’s significant reduction in migration costs demonstrates its effectiveness and potential advantages in real-world scenarios. Our method not only achieves comparable or better power savings but also reduces the migration cost, making it a more efficient solution for real-time cloud data center operations. Additionally, the proposed algorithm significantly improves resource utilization by reducing unnecessary migrations, leading to lower operational overhead. This combined effect of enhanced energy efficiency and minimized migration cost underscores its suitability for optimizing large-scale cloud infrastructures.

The novelty of the solution lies in its hybrid optimization approach, which combines the strengths of Cuckoo Search and Firefly Colony Optimization with fuzzy logic, creating a robust and adaptive VM placement strategy. Unlike previous works that primarily focus on single or limited objectives, our algorithm addresses critical factors, including energy efficiency, resource utilization, and migration cost, in a balanced manner.

This makes the solution particularly significant for practical applications in cloud data centers, where the demand for energy-efficient and cost-effective operations is ever-growing. By reducing energy consumption and improving resource management, our approach contributes to the sustainability and scalability of cloud infrastructures, offering a practical and impactful solution for modern data centers. The combination of CSO's global search capabilities and FA's local optimization strengths create a powerful synergy. CSO prevents premature convergence by exploring a wide range of possibilities, while FA ensures that the solution is refined and optimized, resulting in a balanced and effective VM placement strategy. To further enhance the hybrid approach, fuzzy logic is integrated to manage the uncertainties and complexities inherent in cloud environments. Fuzzy rules are designed to evaluate and adjust the algorithm's parameters dynamically based on real-time conditions such as workload intensity, resource availability, and energy consumption. Fuzzy logic allows for more nuanced decision-making, enabling the algorithm to adapt its behaviour based on the specific needs of the cloud data center at any given moment.

In practical terms, it has been demonstrated that the migration cost associated with the Fuzzy Cuckoo Search Fuzzy Firefly Colony algorithm is 31.7% lower than that of other compared algorithms. This improvement underscores the effectiveness of FCSFFC in optimizing resource allocation and reducing the overhead typically encountered during virtual machine migrations in cloud computing environments. Its dynamic thresholding and prediction capabilities address the challenges of modern cloud environments, making it a valuable tool for optimizing cloud data center operations.

The scalability of the Fuzzy Cuckoo Search Fuzzy Firefly Colony Optimization Algorithm was assessed by varying the number of virtual machines and tasks in the simulation environment. The algorithm's performance metrics, including energy consumption and computation time, were evaluated as the data scale increased. Additionally, the algorithm's adaptability to varying workload intensities ensures that resources are allocated efficiently, minimizing downtime and improving user satisfaction. By incorporating fuzzy logic with metaheuristic optimization, the FCSFFC algorithm offers a robust solution for managing complex cloud

infrastructures effectively. The combination of CSO's broad exploration and Firefly colony approach focused exploitation leads to better VM placements that minimize energy consumption across the data center. By optimizing both the initial placement and the migration paths, the hybrid approach reduces the amount of data transferred and the associated costs during VM migration. The synergy between CSO and FA ensures that resources are utilized more efficiently, reducing wastage and enabling higher workload handling. The fuzzy logic component ensures that the algorithm adapts to changing conditions in the data center, maintaining optimal performance even as workloads and resource availability fluctuate. This adaptability is crucial for sustaining consistent efficiency and reliability in dynamic cloud environments, where demand can be unpredictable.

5 Scalability and computational complexity

The FCSFFC algorithm demonstrated strong scalability, effectively managing up to 1,000 VMs and 5,000 tasks without significant performance degradation. This indicates that the algorithm is well-suited for large-scale cloud environments where the number of VMs and tasks can be substantial. The performance metrics remained efficient as the workload increased, highlighting the algorithm's capacity to handle large and complex cloud computing scenarios. The computational complexity of the FCSFFC algorithm is influenced by its hybrid optimization approach, which integrates fuzzy logic with cuckoo search and firefly algorithms. The complexity can be expressed as O(N·M·K) where N represents the number of VMs, M denotes the number of tasks, and K stands for the number of iterations in the optimization process. The algorithm exhibits linear scalability with respect to the number of VMs and tasks, which ensures efficient performance even as the size of the problem grows.

6 Conclusion

A hybrid optimization approach for cloud computing virtual machine migration has been created. The combined technique of Fuzzy Cuckoo search and Fuzzy Firefly optimization algorithms is used to accomplish VM migration. To minimize needless migrations, the VMSs are moved to the host based on the best outcome produced by the suggested method. Improved performance in terms of energy consumption, resource availability, migration costs, and computing time is validated by simulation study of the suggested model. The suggested hybrid optimization algorithm's performance is compared with that of existing migration techniques, such as HBCOA, FOA, WOA, hybrid Whale and HWOA. The proposed algorithm utilizes less resources on average than the state-of-art techniques, and also lesser power consumption. Compared to state-of-the-art methods, the proposed algorithm uses less resources and consumes less power as discussed in Section 4. The proposed algorithm outperforms in computation time and also in reducing the migration cost. Future research on this topic may focus on introducing Flight random walk and Preferences random walk optimization [45] to speed up the search and enhance migration performance. Additionally, exploring the integration of lightweight heuristics, such as Quantum Particle Swarm Optimization, could further reduce computational overhead while maintaining solution quality. Another potential avenue for research is to dynamically adjust the algorithm's parameters based on varying cloud workloads and infrastructure conditions. Future research could explore the integration of advanced machine learning techniques with optimization strategies to enhance decision-making processes in resource allocation and energy management

References

[1] Duong-Ba, T., et al. (2018). A Dynamic Virtual Machine Placement and Migration Scheme for Data Centers. *IEEE Transactions on Services Computing*, vol. 14(2), pp. 329–341.

[2] Khan, M. A. (2021). An Efficient Energy-Aware Approach for Dynamic VM Consolidation on Cloud Platforms. *Cluster Computing*, vol. 24(4), pp. 3293–3310.

[3] Khan, M. S. A., and Santhosh, R. (2022). Hybrid optimization algorithm for VM migration in cloud computing. *Computers and Electrical Engineering*, Elsevier, pp. 102:108152. https://doi.org/10.1016/j.compeleceng.2022.108152

[4] Sait, S. M., Bala, A., and El Maleh, A. H. (2016). Cuckoo Search Based Resource Optimization of Datacenters. *Applied Intelligence*, Springer, vol. 44, no. 3, pp. 489-506.

[5] Chamas, N., Pires, F. L., and Baran, B. (2017). Two-Phase Virtual Machine Placement Algorithms for Cloud Computing: An Experimental Evaluation under Uncertainty. *Proceedings of the IEEE Conference (CLEI)*, IEEE, Cordoba, Argentina.

[6] Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. Heuristics for Vector Bin Packing, https://www.microsoft.com/en-us/research/wp-content/uploads/2011/01/VBPackingESA11. pdf.

[7] Shirvani, H., and Mirsaeid, H. (2021). Bi-objective Webservice Composition problem in Multi-cloud Environment: A Bi-objective Time Varying Particle Swarm Optimization Algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 33(2), pp. 179-202, Mar. 2021.

[8] Rukmini, S., and Shridevi, S. (2023). An Optimal Solution to Reduce Virtual Machine Migration SLA Using Host Power. *Measurement: Sensors*, vol.25, pp.1-5. https://doi.org/10.1016/j.measen.2022.100628

[9] Al-Mahruqi, A. A., Morison, G., Stewart, B. G., and Athinarayanan, V. (2021). Hybrid Heuristic

Algorithm for Better Energy Optimization and Resource Utilization in Cloud Computing. *Wireless Personal Communications*, 118, pp:43-73.

[10] Rashida, S. Y., Saba, M., Ebadzadeh, M. M., and Rahmani, A. M. (2020). A Memetic Grouping Genetic Algorithm for Cost Efficient VM Placement in Multi Cloud Environment, *Journal of Cluster Computing*, 23(2), 797-836, Jun. 2020.

[11] [3] Sonklin, C., and Sonlin, K. (2023). A Multi-Objective Grouping Genetic Algorithm for Server Consolidation in Cloud Data Centers. *Proceedings of JCSSE 2023*, IEEE, Phitsanulok, Thailand, pp. 1–16.

[12] Zhang, B., Wang, X., and Wang, H. (2021). Virtual Machine Placement Strategy Using Cluster-Based Genetic Algorithm. *Journal of Neurocomputing*, vol. 428, pp. 310-316.

[13] Wang, X., Lou, H., and Dong, Z. (2023). Decomposition Based Multi-Objective Evolutionary Algorithm for Virtual Machine and Task Joint Scheduling of Cloud Computing in Data Space. *Journal of Swarm and Evolutionary Computation*, vol. 77, pp. 1-17.

[14] Gharehpasha, S., and Masdari, M. (2021). A discrete chaotic multi-objective SCA-ALO optimization algorithm for an optimal virtual machine placement in cloud data center. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 9323-9339.

[15] Boominathan, P., Aramudan, M., and Saravanaguru, K. R. (2017). Fuzzy Bio-Inspired Hybrid Techniques for Server Consolidation and Virtual Machine Placement in Cloud Environment. *Cybernetics and Information Technologies*, 17(4), pp. 52-68.

[16] Gagwero, M. G., and Caviglione, L. (2019). Model Predictive Control for Energy Efficient, Quality-Aware Virtual Machine Placement. *IEEE Transactions on Automation Science and Engineering*, 16(1), pp. 420-432, Jan. 2019.

[17] Sharma, S., Kumar, S., Mohapatra, S., and Rani, R. (2020). Discrete Gravitational Search Algorithm for Virtual Machine Placement in Cloud Computing. *International Journal of Advanced Science and Technology*, 29(8s), pp. 1261-1267.

[18] Wang, M., and Lu, G. (2021). A Modified Sine Cosine Algorithm to Solve Optimization Problems. *Journal of IEEE Access*, pp. 27434-27450.

[19] Xing, H., Zhu, J., Qu, R., Dai, P., Luo, S., Muhammad, and Iqbal, A. (2022). An ACO for Energy-Efficient and Traffic-Aware Virtual Machine Placement in Cloud Computing. *Journal of Swarm and Evolutionary Computation*, vol. 68, pp. 1-18

[20] Chalabi, N. E., Attia, A., Bouziane, A., and Hasaballah, M. (2023). An Improved Marine Predator Algorithm Based on Epsilon Dominance and Pareto Archive for Multi-Objective Optimization. *Journal of Engineering Applications of Artificial Intelligence, 1*19(1), pp. 1-18

[21] Basset, M. A., Mohamed, R., and Mirjalili, S. (2021). A Novel Whale Optimization Algorithm Integrated with Nelder Mead Simplex for Multi-Objective Optimization Problems, *Journal of Knowledge-Based Systems*, vol. 212, pp. 1-28.

[22] Ding, Z., Cao, L., Chen, L., Sun, D., Yi, X., Zhang, Z., and Tao, Z. (2023). Large Scale Multimodal Multi-Objective Evolutionary Optimization Based on Hybrid Hierarchical Clustering. *Journal of Knowledge-Based Systems*, vol. 266, pp. 1-22

[23] Xiang, Z., Zhou, G., and Luo, Q. (2021). Golden Sine Cosine Salp Swarm Algorithm for Shape Matching Using Atomic Potential Function. *Journal of Expert Systems*, 39(15), pp. 1-24, Nov. 2021.

[24] Z. Xiang. G. Zhou and Luo, Q. (2020). "A New fusion of Salp Swarm with Sine Cosine for Non-Linear Function", Journal *of Engineering with Computers*, pp.185-212.

[25] Salami, H., Bala, A., & and Sait, S. (2021). An Energy Efficient Cuckoo Search Algorithm for Virtual Machine Placement in Data Centers. *The Journal of Supercomputing*, vol. 77, no. 11, pp. 1-28, Apr. 2021.

[26] Farshin, A., and Sharifian, S. (2019). A Modified Knowledge-based Ant Colony Algorithm for Virtual Machine Placement and Simultaneous Routing of NFV in Distributed Cloud Architecture. *The Journal of Supercomputing*, vol. 75, no. 8, pp. 5520–5550.

[27] Sharma, S. K., and Ghai, W. (2023). Artificial Bee Colony Optimized VM Migration and Allocation using Neural Network Architecture. *Journal of Advanced Technology and Engineering Exploration*, vol. 10, no. 102, pp. 590–607.

[28] Patra, M., Misra, S., Sahoo, B., and Turuk, A. (2022). GWO-Based Simulated Annealing Approach for Load Balancing in Cloud for Hosting Container as a Service. *Journal of Applied Sciences*, vol. 12, no. 21, pp. 1–22, Nov. 2022.

[29] Vijaya, C., and Srinivasan, P. (2024). Multi-objective Meta-heuristic Technique for Energy Efficient Virtual Machine Placement in Cloud Computing Data Centers. *Informatica*, vol. 48, no. 1, pp. 1–18. https://doi.org/10.31449/inf.v48i6.5263.

[30] Gupta, N., Gupta, K., Qahtani, A. M., Gupta, D., Shalharithi, F. S., Singh, A., and Goyal, N. (2022). Energy Aware Live VM Migration using Ballooning in Cloud Data Centers. *Electronics*, vol. 11, no. 23, pp. 1–16.

[31] Pande, S. K., Panda, S. K., Das, S., Sahoo, K. S., Luhach, A. K., Jhanjhi, N. Z., Alrobaea, R., & Sivanesan, S. (2023). A Resource Management

Algorithm for Virtual Machine Migration in Vehicular Cloud Computing. *Computers, Materials and Continua*, vol. 67, no. 2, pp. 2647–2663.

[32] Gupta, A., & Namasudra, S. (2022). A Novel Technique for Accelerating Live Migration in Cloud Computing. *Automated Software Engineering*, vol. 29, no. 1, May 2022. DOI:10.1007/s10515-022-00332-2.

[33] Kaur, A., Kumar, S., Gupta, D., Hamid, Y., Hamdi, M., Ksibi, A., Elmannai, H., & Saini, S. (2023). Algorithmic Approach to Virtual Machine Migration in Cloud Computing with Updated SESA Algorithm. *Sensors*, vol. 23, no. 13, pp. 1–18.

[34] Booba, B., Anitha, J. J., Mohan, C., & Jeyalakshmi, S. (2024). Hybrid Approach for Virtual Machine Allocation in Cloud Computing. *Sustainable Computing: Informatics and Systems*, vol.41, DOI: 10.1016/j.suscom.2023.100922.

[35] Aydilek, I. B. (2018). A Hybrid Firefly and Particle Swarm Optimization Algorithm for Computationally Expensive Numerical Problems. *Applied Soft Computing*, vol. 66, pp. 232–249.

[36] Thirugnanasambandam, K., Rajkumar, R., Alghamdi, A. S., Alshamrani, S. S., Maharajan, K., & Rashid, M. (2023). Energy Efficient Virtual Machine Placement in Distributed Cloud using NGSA III Algorithm, *Journal of Cloud Computing*, vol. 12(124).

[37] Basset, M. A., Abdle-Fatah, L., & Sangaiah, A. K. (2018). An Improved Levy Based Whale Optimization Algorithm for Bandwidth-Efficient Virtual Machine Placement in Cloud Computing Environment. *Cluster Computing*, vol. 22(1), Jan. 2018, pp. 8319–8334.

[38] Bhatt, C., & Singhal, S. (2023). Hybrid Metaheuristic Technique for Optimization of Virtual Machine Placement in Cloud. *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 23(3), Sept. 2023, pp. 353–364.

[39] Azizi, S., Zandsalimi, M., & Li, D. (2020). An Energy Efficient Algorithm for Virtual Machine Placement Optimization in Cloud Data Center, *Cluster Computing*, vol. 23(1), Mar. 2020, pp. 3421–3434.

[40] Gopu, A., & Venkataraman, N. (2021). Virtual Machine Placement using Multi-objective Bat Algorithm with Decomposition in Distributed Cloud: MOBA/D for VMP. *Applied Metaheuristic Computing*, vol. 12(4), Oct. 2021, pp. 62–77.

[41] Boominathan, P., & Aramudan, M. (2016). A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Datacenters. *Advances in Fuzzy Systems*, vol. 2016, Article ID: 6734161, pp. 1–15. https://doi.org/10.1155/2016/6734161.

[42] Govardhan, P., & Srinivasan, P. (2022). Multilevel Controller-Assisted Intrinsically Modified Ant Colony Optimization Heuristic-Based Load-Balancing Model for Mega Cloud Infrastructures. *International Journal of Communication Systems*, vol. 35(6), Jan. 2022. DOI:10.1002/dac.5091.

[43] Dhal, K. G., Das, A., & Galvez, J. (2021). A Novel Fuzzy Logic-Based Improved Cuckoo Search Algorithm. *International Journal of Applied Metaheuristic Computing*, vol. 13(1). https://orcid.org/0000-0002-6748-0569.

[44] Gu, H., Wang, J., Yu, J., Wang, D., Li, B., He, X., & Yin, X. (2023). Towards Virtual Machine Scheduling Research Based on Multi-Decision AHP Method in Cloud Computing Platform. *PeerJ Computer Science*, vol. 2023(1675), Nov. 2023, p. 22. DOI:10.7717/peerj-cs.1675.

[45] Mukhija, L., & Sachdeva, R. (2023). An Optimal Cuckoo Search Algorithm for VM Selection for Energy-Efficient Migration in Cloud Computing. *Eur. Chem. Bull.*, vol. 12(7), pp. 1596–1607.

[46] SeyyedSalehi, S. M., & Khansari, M. (2022). Virtual Machine Placement Optimization for Big Data Applications in Cloud Computing. *IEEE Access*, vol. 10(1), pp. 96112–96127. https://doi.org/10.1109/ACCESS.2022.3203057.

[47] Liu, P., & Zhang, S. (2021). A Novel Cuckoo Search Algorithm and its Application. *Applied Sciences*, vol. 11(1), pp. 1071–1081.

[48] Vijaya, C., & Srinivasan, P. (2024). Multi-Objective Meta-Heuristic Technique for Energy-Efficient Virtual Machine Placement in Cloud Data Centers. *Informatica*, vol. 48(6), pp. 1–18, Feb. 2024.