SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I3P118

Volume 12 Issue 3, 179-189, March 2025

Original Article

Two-Stage Optimal Virtual Machine Load Balancing Algorithm for Cloud Computing

E. Suganthi[1], F. Kurus Malai Selvi[2]

[1,2] *Department of Computer Science, Government College for Women (A), Kumbakonam ,*
(Affiliated to Bharathidasan University, Trichy), Tamilnadu, India.

[1]*Corresponding Author : esuganthics@gmail.com*

Received: 11 January 2025 Revised: 12 February 2025 Accepted: 16 March 2025 Published: 29 March 2025

Abstract - *Cloud Computing (CC) is allocating resources flexibly to deliver services to end users via the Internet. To implement CC, it is necessary to tackle various obstacles, including resource finding, security, scheduling, and Load Balancing (LB). LB is the most difficult of these research problems. LB aims to allocate workloads to optimize resource usage and boost performance. This research paper proposes an efficient LB model for CC using a two-stage optimal meta-heuristic algorithm called TSOVM_LB. In the first stage, the Virtual Machine (VM) is chosen based on the Minimum Utilization and Migration (MUM) time. In the second stage, a multi-objective optimization algorithm, Modified Fish Swarm Optimization (MFSO), is used for VM allocation. This model allows the VM to the Physical Machine (PM). The proposed method was assessed using CloudSim, incorporating massive VMs and workload traces from the PlanetLab platform. The outcomes showed that the proposed technique attained much higher levels of energy efficiency, SLA compliance, and fewer VM migrations related to other modern techniques. The results presented here provide evidence of the efficacy of the proposed technique in optimizing the allocation of VMs in a cloud environment.*

Keywords - *Load Balancing, Optimization, Fish Swarm, Cloud Computing, Virtual Machine*.

1. Introduction

CC describes the provision of resources via the Internet. The resources encompass computers, storage, databases, and networking [1]. Implementing a cloud environment has encountered numerous challenges. These include resource discovery, scheduling, security, and privacy. LB is a critical topic among these challenges. Distributed LB is the process of dispersing the workload among numerous machines. LB is distributing the workload across multiple computer platforms [2]. LB aims to optimize the efficiency, resource usage, and performance of VMs. Workload balancing is a crucial aspect of CC architecture as it effectively distributes computing resources. Each VM in the cloud has a diverse processing speed, memory, and capacity [3]. LB matches workloads with VMs to prevent overload, while dynamic web computing can cause request overload in CC. LB is the most complex and important area of research in CC, as it comprises allocating workloads among VMs in data centres. LB is significant for optimal resource consumption and service quality in heterogeneous CC environments [4]. Load balancers are vital in allocating resources equitably and efficiently to workloads, ensuring customer satisfaction while minimizing costs. However, existing LB methods face various challenges that require immediate attention. This has prompted researchers to develop improved LB policies to address these difficulties [5].

In CC, workload balancing within the architecture is a critical factor in resource allocation. To optimize resource use, the cloud system utilizes a range of LB methods [6]. However, high computational costs, energy usage, additional burdens, limited scalability, and time limits plague most conventional LB methods. The use of meta-heuristics-based approaches [7] for LB has gained significant popularity in recent times due to their superiority in handling discontinuous problems through intensification (exploitation), diversification (exploration), flexibility, multimodal optimization, efficient randomization, and so on [8].

Different kinds of meta-heuristics-based approaches like nature-inspired (cuckoo search, flower pollination) [9], bio-simulated (grey wolf), evolutionary (genetic), and swarm (PSO, Ant colony) based approaches are used in a CC for LB [10]. The efficient management of resources in CC environments remains a significant challenge, specifically as cloud infrastructures continue to grow in scale and complexity. Optimizing resource distribution across VMs is crucial for maximizing overall performance and minimizing response times [11]. As cloud systems handle diverse workloads with varying requirements, it is significant to ensure that resources are allotted dynamically and effectively to prevent overloading or underutilization. This motivates the need for advanced strategies that can intelligently balance the computational load across distributed systems, improving both efficiency and scalability [12].

This research study presents an effectual LB technique for CC. The approach utilizes a two-stage optimum meta-heuristic method known as TSOVM_LB. A threshold-based technique determines the current consumption of PM. The PMs are categorized into three states: typically loaded, underutilized, and overwhelmed. The first stage involves selecting a VM based on the MUM time. The process's second stage uses a multi-objective optimization technique called Modified Fish Swarm Optimization (MFSO) for VM allocation. This method effectively assigns the VM to the PM. The primary research contribution is outlined as follows:

- A two-stage optimal VM allocation is proposed to achieve effective LB.
- A method based on thresholds is employed to determine the current utilization of PMs. The PMs are divided into three states: normally loaded, underloaded, and overloaded.
- Formulate a VM state-based algorithm to identify suitable VMs for host migration using VM MUM time.
- A VM placement technique utilizing MFSO is suggested to distribute migrated VMs evenly and achieve optimal service performance.
- The suggested technique's effectiveness is evaluated using CloudSim and Planet Lab workload. The investigative outcome highlights that the proposed methodology minimizes EC and SLA violations.

2. Related Works

Sayadnavard et al. [13] provide a Discrete-Time Markov Chain (DTMC) method, which utilizes the reliability technique of PMs. The e-dominance-based Multi-Objective Artificial Bee Colony (e-MOABC) approach effectively meets SLA and QoS requirements. Dubey et al. [14] expanded upon the intelligent water drop method. The technology reduces energy usage in the cloud data centre and improves overall system performance. The Water Drop VM Allocation (WDVMA) approach distinguishes between low-and high-utilization hosts. It then migrates VMs to increase server utilization. Radi et al. [15] present a Modified Genetic-based VM Consolidation (MGVMC) technique. This approach employs a Genetic Algorithm (GA) to transfer VMs to suitable PMs to limit the occurrence of over- and under-utilized PMs to the greatest extent possible. It explicitly highlights workloads that require a significant amount of CPU processing power.

A cloud environment that accurately simulates real-world conditions is necessary to evaluate the method. Kanagaraj et al. [16] suggest using Uniform Distribution Elephant Herding Optimization (UDEHO) to maximize resources by spotting hosts that are too busy or not busy enough. The UDEHO technique accurately forecasts future resource utilization. To detect under-loaded hosts, it is recommended to use a power-saving value based on power usage and migration numbers. Thakur et al. [17] propose a VM consolidation strategy for CC using the Cuckoo Search Algorithm. Optimizing energy conservation without affecting system performance or cloud service quality is unattainable. Most existing methods for VM consolidation rely on load and threshold concepts.

Alsadie et al. [18] propose the Modified Feeding Birds Algorithm (ModAFBA) methodology. Madhusudhan et al. [19] suggest a VM placement strategy for cloud data centres based on the Harris Hawk Optimization model. Durairaj et al. [20] introduce a meta-heuristic optimization technique called the Multi-Objective Mayfly VMP (MOM-VMP), which utilizes a vast CDC (Computation and Data Centre) with diverse and multi-dimensional resources. An integrated approach is used through a multi-objective dynamic VMP technique. Pandey et al. [21] introduce the Energy-Efficient Particle Swarm Optimization algorithm (EEVMPSO) model, which aims to optimize LB while minimizing energy consumption. The Particle Swarm Optimization (PSO) model is utilized for energy-aware VM migration to achieve dynamic VM placement. Medara et al. [22] present a dynamic VMC model. The Modified Water Wave Optimization (MWWO) technique is utilized. Studies have shown that a host under excessive load uses more energy in a given period than a host operating at normal capacity. Lu, Zhou, and Zou [23] propose a two-stage optimization strategy: a Greedy Algorithm (GA) for Coarse-Grained LB across VMs and a GA for fine-grained resource allocation within each VM. Gabhane, Pathak, and Thakare [24] introduce EAGLE modified approach for optimal VM placement in CC. Li et al. [25] present a novel Multi-Objective Flower Pollination Algorithm (MOFPA/D) method, integrating a discrete FPA.

Menaka and Kumar [26] propose a strategy-based mixed support and LB for task scheduling in cloud computing. The Time-Conscious Scheduling with Supportive PSO (SPSO-TCS) technique to reduce make-span time and achieve LB. Ma et al. [27] introduce a two-stage-VNS approach for effectual discrete variable search and Sigmoid activation support, ensuring global optimality. Qora, implemented on Kubernetes, automates resource provisioning in a serverless system. Kaur et al. [28] propose an Enhanced K-means Clustering (EKCLB) approach for task and VM allocation at the fog layer in smart cities by clustering tasks and VMs based on priority, burst time, and capacity. Bano et al. [29] present the Levelized Multi-workflow Heterogeneous Earliest Finish time (LMHEFT) methodology. It features task prioritization utilizing level attributes and upward rank, followed by task allocation to the best-suited VM for minimizing completion time. Li et al. [30] introduce the Lyapunov and Multi-agent Deep Deterministic Policy Gradient (LAMETO) approach, a distributed two-stage task offloading architecture based on Lyapunov and MADDPG. It optimizes offloading delays and

RSU energy consumption in VEC subsystems. Zhang [31] proposes an automated container arrangement and resource optimization algorithm that predicts and adjusts resources based on real-time and historical data, ensuring effective utilization and optimal application performance. Zhu et al. [32] propose SA2CTS, a containerized task scheduling framework based on reinforcement learning and cross-modal contrastive learning. It utilizes a two-stage pipeline: pretraining on image-text pairs for extracting scheduling features, followed by fine-tuning with multisource cluster feedback for task-oriented, semantic-aware scheduling.

The limitations of the existing studies comprise a lack of comprehensive approaches that incorporate both energy efficiency and real-time optimization for VM placement and task scheduling across multiple cloud environments. Many existing methods concentrate on either resource utilization or LB, neglecting the interdependency of energy consumption and system performance. Moreover, the scalability of these algorithms in large-scale cloud or fog computing environments remains a challenge, with few studies addressing dynamic task allocation in real-world, heterogeneous settings. Further research is required to integrate these strategies into more adaptive and energy-efficient frameworks while ensuring improved handling of variable workloads and resources in large-scale cloud infrastructures.

3. Proposed TSOVM_LB

This section describes the proposed VM LB in CC. This methodology comprises two stages: MUM-based VM selection and MFSO-based VM allocation. Figure 1 illustrates the working flow of the TSOVM_LB method.

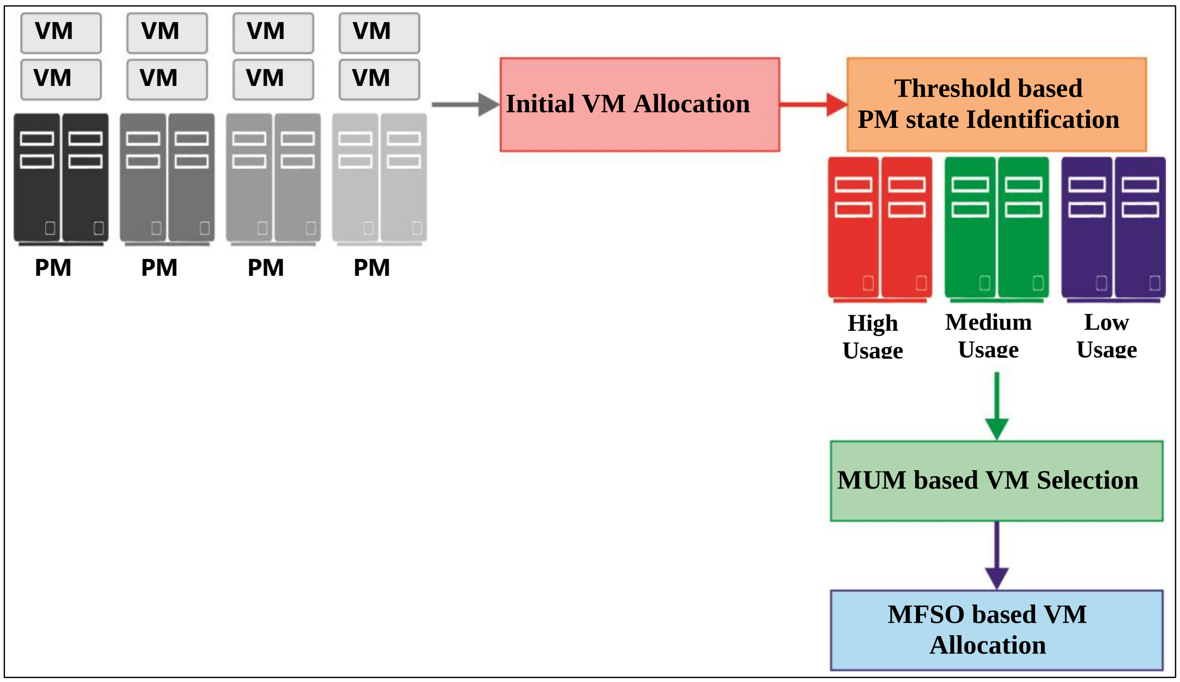

Fig. 1 TSOVM_LB workflow

3.1. PM State Identification

Let's assume that the CDC consists of m PMs, denoted as PM = $\{pm_1, pm_2, pm_3,..., pm_m\}$, and n VMs, denoted as VM = $\{vm_1, vm_2, vm_3,..., vm_n\}$. Every machine possesses various resources, including CPU, memory, and bandwidth. The system's Energy Consumption (EC) metric directly influences a data centre's PM load status. A widely used host will have a negative effect on response times and service quality, whereas a less utilized host will consume additional energy. Hence, the state detection of PMs is a crucial determinant for allocation. This research employs a threshold-based technique [33] to determine the status of the PM. There are three states of power management: high utilization, medium usage, and low usage.

The two threshold values, T_{up} and T_{low}, are utilized to determine the current PM state. Farahnakian et al. [34] established threshold values of 0.5 and 1.0; Li et al. [35] placed them at 0.1 and 0.9; and Liu et al. [36] determined lower and higher threshold values of 0.3 and 0.8, respectively. A higher T_{up} will lead to persistent overloading and significant SLA violations, whereas a lower T_{up} will lead to inefficient resource utilization. Many VMs will relocate if the T_{low} exceeds a certain level. Setting the T_{low} value too low could lead to the insufficient shutdown of minimally used hosts. This study calculates the T_{low} and T_{up} values in real time, considering the current level of resource usage. A detailed explanation of the algorithm is found in [33].

3.2. MUM-Based VM Selection

To prevent any adverse effects on the migration procedure and to promptly restore the host to its original load status, it is essential to efficiently choose the VM to migrate when the server host becomes overloaded. In cases where hosts are overloaded, it is necessary to move certain VMs until the host's utilization level falls below the high threshold. Every VM on the host needs to migrate if it is underloaded. The host enters sleep mode when all VMs migrate. Adjusting the lower threshold reduces excessive consolidation, minimizing migrations and SLAVs. This paper suggests MUM-based VM selection. The CPU and RAM utilization of the VM is computed as,

$$U_{vm_j}^{CPU} = \frac{Allocated\ MIPS\ of\ vm_j}{vm_j^{MIPS}} \tag{1}$$

$$U_{vm_j}^{RAM} = \frac{Allocated\ memory\ of\ vm_j}{vm_j^{RAM}} \tag{2}$$

$U_{vm_j}^{CPU}$ and $U_{vm_j}^{RAM}$ represent the CPU and RAM utilization of j^th VM (vm_j). vm_j^{MIPS} and vm_j^{RAM} indicate the MIPS of memory of vm_j.

Algorithm-1: MUM-Based VM Selection
Input: High usage host list (PM_{hu}) **Output:** Selected VM list (VM_{sel}) Step01: For each pm_i in PM_{hu}, do Step02: vmList = get list of VM in pm_i Step03 minUM =MaxValue Step04: For every vm in vmList do Step05: Compute CPU utilization of vm ($U_{vm_j}^{CPU}$) using Equation (1) Step06: Compute Memory utilization of vm ($U_{vm_j}^{RAM}$) using Equation (2) Step07: Compute migration time $T_{vm_j}^{mig}$ using Equation (3) Step08: cUM = $T_{vm_j}^{mig} \times \left(1 - U_{vm_j}^{CPU}\right) \times \left(1 - U_{vm_j}^{RAM}\right)$ Step09: If cUM < minUM then Step10: minUM = cUM Step11: Add vm to VM_{sel} Step12: EndIf Step13: Find the pm_i current state Step14: If pm_i is high usage, then Step15: break; Step16: EndIf Step17: EndFor Step18: EndFor Step19: Return VM_{sel}

The VM migration time is computed as,

$$T_{vm_j}^{mig} = \frac{vm_j^{RAM}}{pm_i^{Band}} \tag{3}$$

Where $T_{vm_j}^{mig}$ represents the duration needed for the migration of vm_j. pm_i^{Band} indicates the bandwidth of i^th PM (pm_i). The VMs are selected based on the MUM time. Algorithm 1 explains the proposed VM selection algorithm.

Algorithm 1 takes the high-consumption hosts as input and produces the chosen VMs that must be migrated as output. The first step involves obtaining the list of VMs on the host with high consumption, as indicated in line 2. Subsequently, the utilization of each VM and the time required for migration are computed using Equations (1), (2), and (3), as described in lines 4-7. The VMs with the lowest consumption and shortest migration time are added to the selected VMs (lines 8–12). Following each stage of the VM selection process, assessing whether host overload persists after VM migration is essential. If the host is experiencing excessive load, the process of selecting a VM continues; otherwise, the process of VM selection stops (lines 13-16).

3.3. MFSO-Based VM Allocation

Selecting target hosts is identical to the VM initialization placement, requiring a mapping between the VM and the appropriate host that meets resource needs while optimizing energy, LB, and resource usage. Finding suitable destinations for the moved VMs is the next crucial task after identifying overloaded and underloaded servers and selecting certain VMs for migration. This section presents a VM placement technique that relies on multi-objective MFSO.

FSO is a metaheuristic algorithm for solving optimization problems. The algorithm utilizes the actions of fish swarms, including preying, swarming, and following (chasing). Figure 2 shows the vision concept of artificial fish [37]. Let X_i be the current position of an artificial fish, X_v its view at a given moment, and Visual its scope. X_a and X_b are fish within X_i's visual range. Step is the fish's maximum step, and δ is the congestion factor. The food concentration is directly related to the fitness function f(X). The behaviour patterns exhibited by fish swarms are shown below:

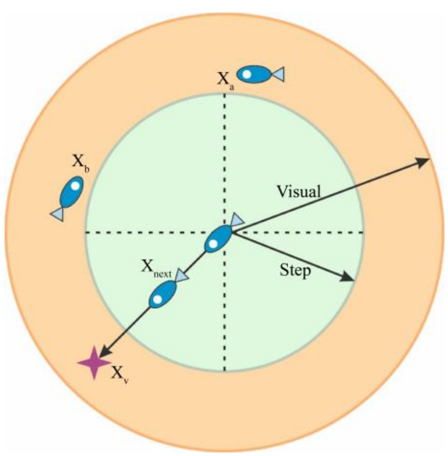

Fig. 2 Vision concept of artificial fish [37]

Swarming behaviour is triggered when the value of $f(X_c)$ is greater than that of $f(X_i)$, where X_c represents the central point within the visual range of point X_i. Let X_c be denoted as X_v. The fish at X_i will get closer to the location at X_c by taking a step.

Chasing behavior occurs when the objective function value at X_{max}, the best point in the Visual, exceeds that at X_i, and X_i's Visual is not congested. In this case, the chasing behaviour is executed. Let X_{max} be denoted as X_v. The fish at X_i will move closer to point X_{max}.

Preying behaviour occurs in two situations: (1) when $f(X_c) < f(X_i)$, $f(X_{max}) < f(X_i)$, and the Visual is not crowded, and (2) when the Visual is crowded.

This model randomly selects a point X_j within the visual range of point X_i. The technique executes the preying behaviour if the objective function value at X_j exceeds the value at X_i. The fish at X_i then moves towards X_j, taking X_j as its new position. If the objective function value at X_j is not greater than Xi's, the fish at X_i moves randomly within its visual range. Each iteration marks the best-obtained solution as a "board." After a set of iterations, the search ends, and the "board" solution is final. For artificial-preying fish, the position update is as follows:

$$X_{next} = X_i + rand \times \frac{step \times \left(X_j - X_i\right)}{norm\left(X_j - X_i\right)} \tag{4}$$

X_{next} is the next fish position; X_i and X_j are the current and better positions; rand is an arbitrary value between -1 and 1, and norm $(X_j - X_i)$ is the distance between the positions.

The position update for artificial swarming fish and artificial chasing fish is expressed in Equations (5) and (6).

$$X_{next} = X_i + rand \times \frac{step \times (X_c - X_i)}{norm(X_c - X_i)} \tag{5}$$

$$X_{next} = X_i + rand \times \frac{step \times (X_{max} - X_i)}{norm(X_{max} - X_i)} \tag{6}$$

In this work, the FSO method is used for VM allocation. When hosts are underloaded, assign all VMs that require relocation to new hosts. Initially, the VMs chosen from the hosts experiencing excessive demand are transferred, followed by the migration of the VMs from the hosts experiencing insufficient load. If VMs are evenly dispersed across hosts, the likelihood of SLAVs at a host decreases. This work suggests a balanced placement approach for VMs to optimize LB and resource utilization in a data center following VM consolidation. The technique is based on the MFSO. The following is the objective function of the VM placement approach:

$$F = \min(EC * SLAV + VM_{MC}) \tag{7}$$

The main objective of the placement approach is to mitigate the objective function value. Algorithm 2 shows the algorithm of the proposed MFSO.

Algorithm-2: MFSO Based VM allocation

Input: PM = {pm_1,pm_2,pm_3,…,pm_m}, VM = {vm_1,vm_2,vm_3,…,vm_n}, List of PM_{hu}, PM_{mu}, PM_{lu}
Output: Allocated VMs
Step01: Initialize algorithm parameters (Visual, Step, Crowd Factor, MaxIter)
Step02: Initialize the random population
Step03: While the stop condition is not attained, do
Step04: For each p ϵ pop, do
Step05: F1 = Compute F(p) using Equation (7)
Step06: Apply Prey behavior
Step07: Update p based on prey behavior (p_pb) using Equation (8)
Step08: F2 = Compute F(p_pb) using Equation (7)
Step09: Apply Swarm behaviour
Step10: Update p based on swarm behavior(p_sb) using Equation (9)
Step11: F3 = Compute F(p_sb) using Equation (7)
Step12: Apply Chase's behaviour
Step13: Update p based on chase behavior(p_cb) using Equation (10)
Step14: F4 = Compute F(p_cb) using Equation (7)
Step15: Find the minimum (F1, F2, F3, F4)
Step16: Update the board
Step17: EndFor
Step18: EndWhile
Step19: Return Optimal Solution

In Algorithm 2, the parameters Visual (1.5), Step (0.3), Crowd Factor (0.61), and MaxIter (50) are initialized and randomly generated based on the hosts and VM numbers (Steps 1 and 2). In the subsequent steps, the objective function for the initial population (Step 5) is computed, and the behaviours of prey, swarm, and chase are applied. The objective function for the updated population (Steps 6-14) is calculated, and the minimum objective is found. Finally, the board is updated (Step 16).

In this algorithm, for artificial preying fish, the position update is expressed as follows:

$$X_{next} = X_i + (rand - 0.5) \times step \times \left(X_j - X_i\right) \times Dist(X_j, X_i) \tag{8}$$

For artificial swarming fish, the position update is expressed as follows:

$$X_{next} = X_i + (rand - 0.5) \times step \times (X_c - X_i) \times Dist(X_c, X_i) \times \omega \tag{9}$$

For artificial chasing fish, the position update is expressed as follows:

$$X_{next} = X_i + (rand - 0.5) \times step \times (X_{max} - X_i) \times Dist(X_{max}, X_i) \times \omega \quad (10)$$

Where $Dist(X_j, X_i)$ represents the distance between positions, and ω is the weight factor.

4. Experimental Results

This section gives an elaborate description of the experimental design. It presents experimental results evaluating a two-stage optimal LB strategy for enhancing energy efficiency and reducing SLA violations. The suggested method was simulated using CloudSim 4.0, a popular CC platform offering virtualization technology, virtual cloud modelling, and simulation capabilities. CloudSim represents several components of a cloud data centre, including hosts, VMs, brokers, and power models, as simulated entities. Both the hosts and VMs have their computing capabilities.

The study utilized a cloud data centre of 800 servers with varying specifications. Table 1 shows the physical and VM configuration.

The model's performance was evaluated using workload data from the PlanetLab project [38], a global computer cluster collecting CPU utilization data from VMs across over 500 locations. From March 3 to April 20, 2011, data includes 288 CPU utilization records per VM, taken every 5 minutes, covering various VM counts and resource utilization metrics like average CPU consumption and Standard Deviation (SD). Table 2 demonstrates the workload dataset characteristics.

Table 1. PM and VM configuration

	Type	MIPS	Core	RAM (MB)	Bandwidth (Gbps)
PM	HP ProLiant ML110 G4 - Xeon 3040	1860	2	4096	1
	HP ProLiant ML110 G5 – Xeon 3075	2660	2	4096	1
VM	High	2500	1	870	100
	Extra Large	2000	1	1740	100
	Small	1000	1	1740	100
	Micro	500	1	613	100

Table 2. PlanetLab workload characteristics [39]

Workload	Date	# VM	Mean (%)	SD (%)
1	03-03-2011	1052	12.31	17.09
2	06-03-2011	898	11.44	16.83
3	09-03-2011	1061	10.70	15.57
4	22-03-2011	1516	9.26	12.78
5	25-03-2011	1078	10.56	14.14
6	03-04-2011	1463	12.39	16.55
7	09-04-2011	1358	11.12	15.09
8	11-04-2011	1233	11.56	15.07
9	12-04-2011	1054	11.54	15.15
10	20-04-2011	1033	10.43	15.21

This study uses the following measures to analyze the model's performance: EC, number of migrations required to finish the workload, SLAV, and SLA Time per Active Host (SLATAH). EC refers to the aggregate number of energy processing equipment utilized to accomplish a specific task. Minimizing EC is desirable, as it is the primary consideration for developing effectual allocation strategies. Power consumption varies with CPU usage, and different host types have different power demands at the same utilization level. VM migration uses network bandwidth, affecting performance and increasing SLAVs with excessive migrations. Minimizing migrations is crucial for service quality. SLATAH shows the percentage of time hosts' CPU is at 100%, indicating possible VM capacity issues. The SLATAH is defined as,

$$SLATAH = \frac{1}{M} \sum_{i=1}^{M} \frac{To_i}{Tr_i}$$

M represents the overall PMs, To_i is the overload duration, and Tr_i is the host's running time. PDM refers to the decline in the performance of VMs caused by migration. It is computed as,

$$PDM = \frac{1}{N}\sum_{j=1}^{N}\frac{Cd_j}{Cr_j}$$

Where Cr_j is the entire resource the VM has requested, Cdj is the expected performance value deterioration caused by migrations, and N is the overall VMs.

SLAVs, linked to SLATAH and PDM, signal reduced service quality due to host overload and VM migration. The SLAV is computed as,

$$SLAV = SLATAH \times PDM$$

ESV, based on EC and SLAVs, measures the overall performance of the VM consolidation strategy. It is defined as,

$$ESV = EC \times SLAV$$

An increase in either of these metrics will raise the ESV value because it results from both. A lower ESV score signifies a higher trade-off between EC and the SLAV. Table 3 presents the performance metrics under different workloads.

The proposed TSOVM_LB is compared with MOABC-VMC [13], VMS-EDMVM [40], GM-DPSO [41], and ADT-CAU-IEABF [39]. Table 4 illustrates the average results of diverse approaches for different metrics. Compared to existing approaches, the proposed method reduces EC, VM migration, SLAV, and ESV.

Table 3. PlanetLab workload performance

Workload	Number of VMs	EC	Number of Migrated VMs	SLAV	ESV
03-03-2011	1052	46.68	889	0.0002	0.00093
06-03-2011	898	42.42	792	0.00025	0.00106
09-03-2011	1061	46.02	875	0.0002	0.00092
22-03-2011	1516	56.24	1099	0.00012	0.00067
25-03-2011	1078	47.20	886	0.00014	0.00066
03-04-2011	1463	52.72	1043	0.00014	0.00074
09-04-2011	1358	51.45	1014	0.00013	0.00067
11-04-2011	1233	52.24	941	0.00011	0.00057
12-04-2011	1054	46.08	857	0.00017	0.00078
20-04-2011	1033	45.03	870	0.00019	0.00086

Table 4. Average results comparison

Approach	EC	Number of Migrated VMs	SLAV	ESV
MOABC-VMC	105.24	6717	0.08635	9.0875
VMS-EDMVM	104.45	2202	0.00091	0.09504
GM-DPSO	110.2	2303	0.0019	0.20938
ADT-CAU-IEABF	129.96	2841	0.00048	0.06238
TSOVM_LB	48.608	926	0.000165	0.000786

Figure 3 portrays the comparison of EC. The average EC of the proposed approach is 48.608. The proposed method reduces EC by 53.81%, 53.46%, 55.89%, and 55.89% for MOABC-VMC, VMS-EDMVM, GM-DPSO and ADT-CAU-IEABF, respectively. The suggested approach efficiently decreases the active host number and appropriately distributes resources among hosts, minimizing the frequency of host switching. Hence, the approach possesses certain benefits in diminishing energy use.

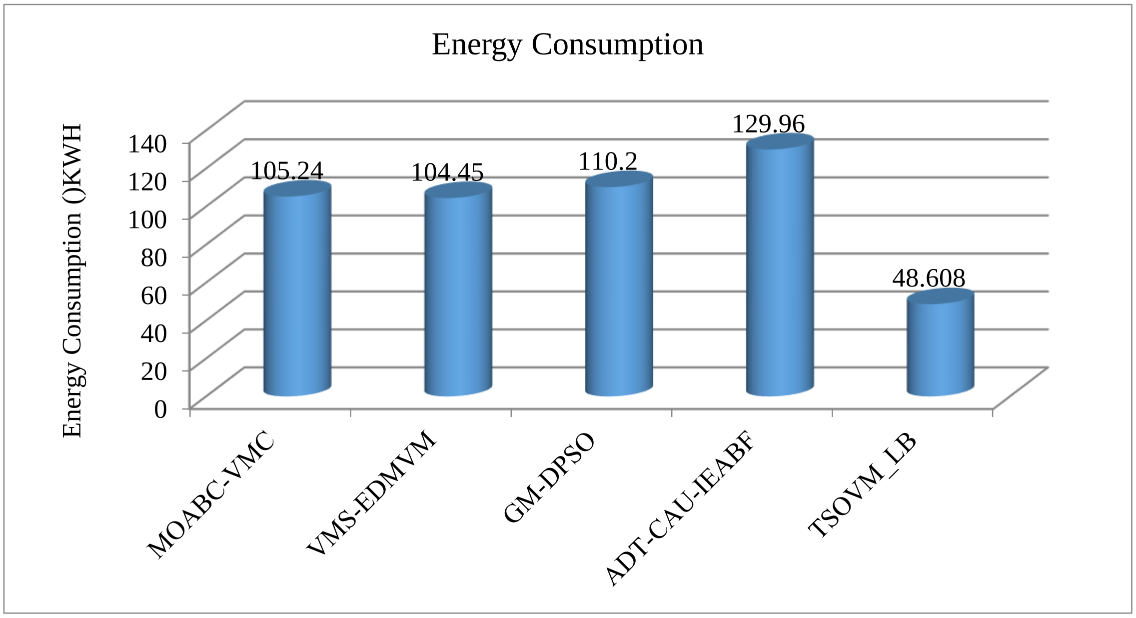

Fig. 3 EC comparison

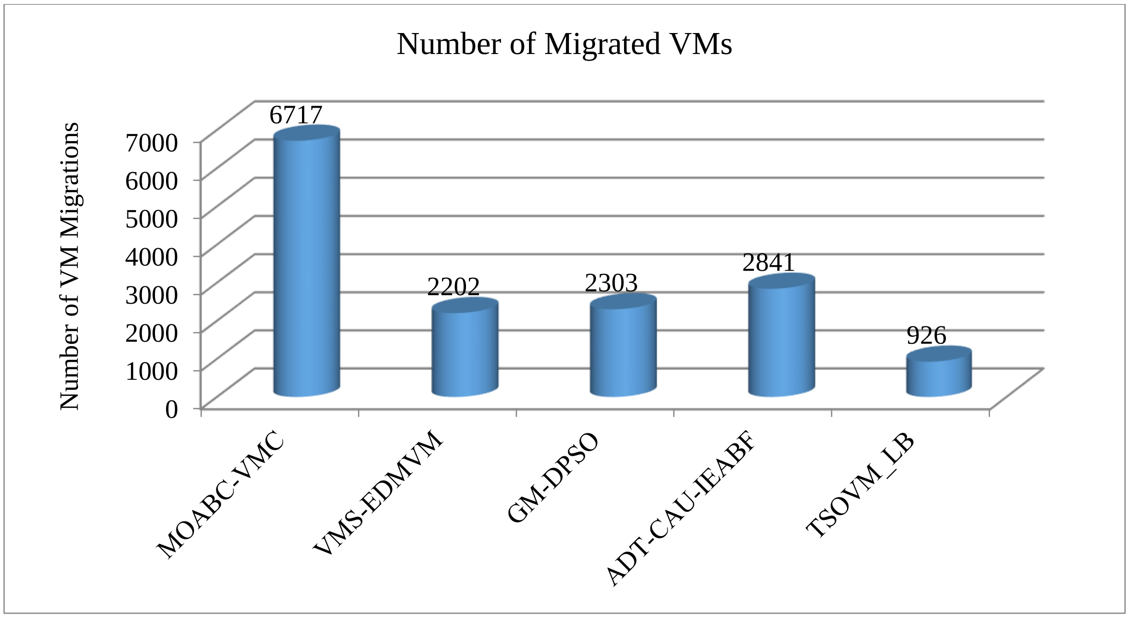

Fig. 4 VM migration comparison

Figure 4 compares the migrated VM numbers. The average number of VM migrations is 926. This method reduces VM migration by 86.21%, 57.95%, 59.79%, and 67.41% for MOABC-VMC, VMS-EDMVM, GM-DPSO, and ADT-CAU-IEABF, respectively. The LB-based placement technique minimizes the likelihood of a host becoming overwhelmed and reduces the frequency of successive migrations.

Figure 5 compares SLA violations. The proposed approach's average SLAV is 0.000165. The approach reduces SLA violations by 99.81%, 81.87%, 91.32%, and 65.63% for MOABC-VMC, VMS-EDMVM, GM-DPSO, and ADT-CAU-IEABF, respectively. It can effectively prevent excessive consolidation and decrease the likelihood of resource shortages.

Figure 6 shows the comparison of ESV. The proposed approach's average ESV is 0.000786. The proposed method reduces ESV by 99.99%, 99.17%, 99.62%, and 98.74% for MOABC-VMC, VMS-EDMVM, GM-DPSO, and ADT-CAU-IEABF, respectively.

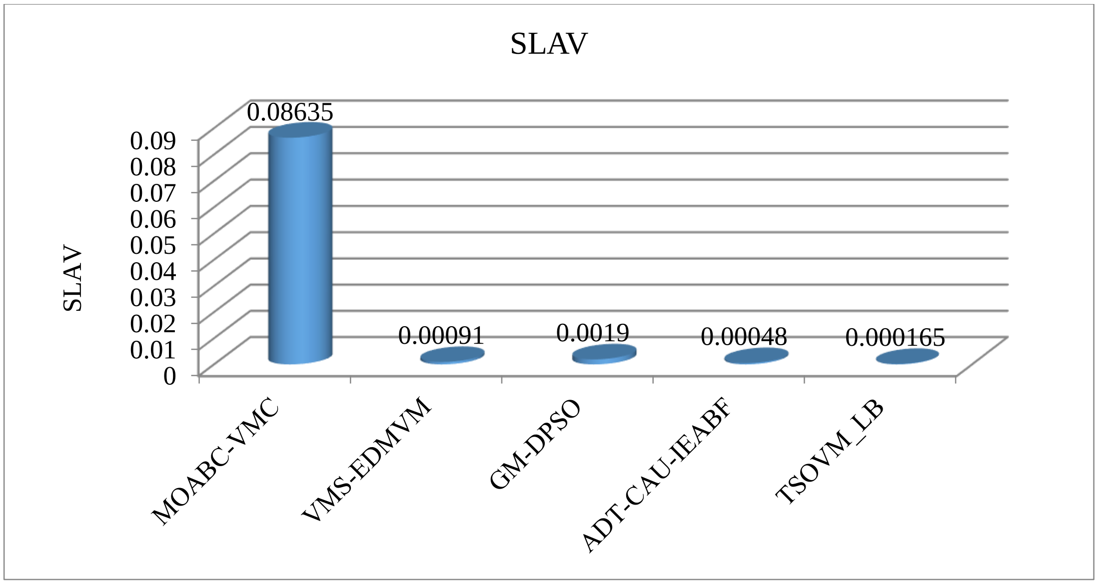

Fig. 5 SLAV comparison

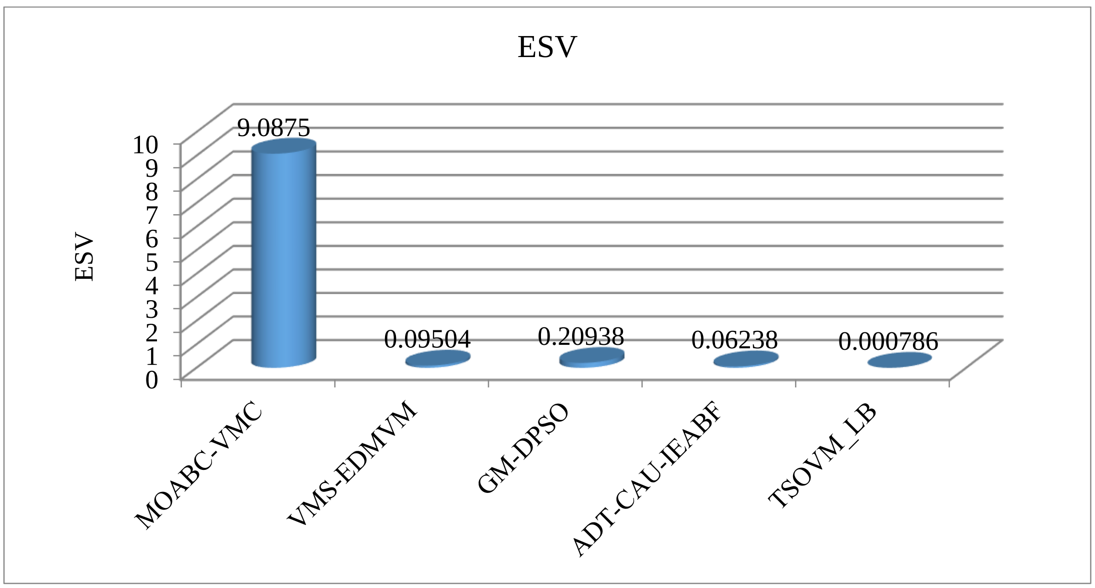

Fig. 6 ESV comparison

The ESV metric reflects the total energy consumed and the level of service quality provided. Compared to other tactics, the proposed strategy outperforms them in terms of the ESV result, regardless of the load conditions. Therefore, it may efficiently ensure the QoS in a datacenter while minimizing energy usage. The proposed technique enhances the efficiency of CC and ensures equitable allocation of resources to each computer unit, strengthening the system's scalability.

5. Conclusion and Future Enhancement

LB is crucial in CC to optimize resource use and improve load distribution. Various tactics and procedures are proposed to tackle issues connected to LB. This research study presents a very effective LB model for CC. The approach utilizes a two-stage optimum meta-heuristic method named TSOVM_LB. During the initial phase, the VM selection is determined by considering the least consumption and migration time. The second stage involves utilizing an MFSO method to allocate VMs in a multi-objective optimization environment. The proposed approach reduced VM migration energy usage and host numbers, lowering the system's overall EC-the experiments conducted on an extensive scale utilized real-world data obtained from execution traces of PlanetLab VMs. The outputs indicated that the suggested technique outperforms existing strategies in optimizing energy usage, VM migration frequency, and SLA violations, providing significant improvements.

References

[1] Bhagyalakshmi Magotra, Deepti Malhotra, and Amit K. Dogra, “Adaptive Computational Solutions to Energy Efficiency in Cloud Computing Environment Using VM Consolidation,” *Archives of Computational Methods in Engineering*, vol. 30, pp. 1789-1818, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mana Saleh Al Reshan et al., “A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing,” *IEEE Access*, vol. 11, pp. 11390-11404, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Arunkumar Gopu, and Neela Narayanan Venkataraman, “Virtual Machine Placement using Multi-objective Bat Algorithm with Decomposition in Distributed Cloud: MOBA/D for VMP,” *International Journal of Applied Metaheuristic Computing*, vol. 12, no. 4, pp. 62-77, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Soumen Swarnakar, Souvik Bhattacharya, and Chandan Banerjee, “A Bio-inspired and Heuristic-based Hybrid Algorithm for Effective Performance with Load Balancing in Cloud Environment,” *International Journal of Cloud Applications and Computing*, vol. 11, no. 4, pp. 59-79, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Harvinder Singh et al., “Metaheuristics for Scheduling of Heterogeneous Tasks in Cloud Computing Environments: Analysis, Performance Evaluation, and Future Directions,” *Simulation Modelling Practice and Theory*, vol. 111, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dalia Abdulkareem Shafiq, N.Z. Jhanjhi, and Azween Abdullah, “Load Balancing Techniques in Cloud Computing Environment: A Review,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3910-3933, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jincheng Zhou et al., “Comparative Analysis of Metaheuristic Load Balancing Algorithms for Efficient Load Balancing in Cloud Computing,” *Journal of Cloud Computing*, vol. 12, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Muhammad Junaid et al., “Modeling an Optimized Approach for Load Balancing in Cloud,” *IEEE Access*, vol. 8, pp. 173208-173226, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Kethavath Prem Kumar et al., “An Efficient Load Balancing Technique Based on Cuckoo Search and Firefly Algorithm in Cloud,” *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 3, pp. 422-432, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Bhavesh N. Gohil, and Dhiren R. Patel, “Load Balancing in Cloud Using Improved Gray Wolf Optimizer,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 11, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Insha Naz et al., “A Genetic Algorithm-Based Virtual Machine Allocation Policy for Load Balancing Using Actual Asymmetric Workload Traces,” *Symmetry*, vol. 15, no. 5, pp. 1-22, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yogita Yashveer Raghav, and Vaibhav Vyas, “ACBSO: A Hybrid Solution for Load Balancing Using Ant Colony and Bird Swarm Optimization Algorithms,” *International Journal of Information Technology*, vol. 15, pp. 2847-2857, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Monireh H. Sayadnavard, Abolfazl Toroghi Haghighat, and Amir Masoud Rahmani, “A Multi-Objective Approach for Energy-Efficient and Reliable Dynamic VM Consolidation in Cloud Data Centers,” *Engineering Science and Technology, an International Journal*, vol. 26, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Kalka Dubey, and S.C. Sharma, “An Extended Intelligent Water Drop Approach for Efficient VM Allocation in Secure Cloud Computing Framework,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3948-3958, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Mohammed Radi, Ali A. Alwan, and Yonis Gulzar, “Genetic-Based Virtual Machines Consolidation Strategy with Efficient Energy Consumption in Cloud Environment,” *IEEE Access*, vol. 11, pp. 48022-48032, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] G. Kanagaraj, and G. Subashini, “Uniform Distribution Elephant Herding Optimization (UDEHO) Based Virtual Machine Consolidation for Energy-Efficient Cloud Data Centres,” *Automatika: Časopis za Automatiku, Mjerenje, Elektroniku, Računarstvo i Komunikacije*, vol. 64, no. 3, pp. 529-539, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Puja Thakur, Jagapreet Sidhu, and Kushal Kanwar, “Dynamic Virtual Machine Consolidation in the Cloud: A Cuckoo Search Approach,” *Procedia Computer Science*, vol. 230, pp. 769-779, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Deafallah Alsadie, and Musleh Alsulami, “Efficient Resource Management in Cloud Environments: A Modified Feeding Birds Algorithm for VM Consolidation,” *Mathematics*, vol. 12, no. 12, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] H.S. Madhusudhan et al., “A Harris Hawk Optimisation System for Energy and Resource Efficient Virtual Machine Placement in Cloud Data Centers,” *Plos one*, vol. 18, no. 8, pp. 1-27, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Selvam Durairaj, and Rajeswari Sridhar, “MOM-VMP: Multi-Objective Mayfly Optimization Algorithm for VM Placement Supported by Principal Component Analysis (PCA) in Cloud Data Center,” *Cluster Computing*, vol. 27, pp. 1733-1751, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Abhishek Kumar Pandey, and Sarvpal Singh, "An Energy Efficient Particle Swarm Optimization Based VM Allocation for Cloud Data Centre: EEVMPSO," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 10, no. 5, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Rsmbabu Medara, and Ravi Shankar Singh, "Dynamic Virtual Machine Consolidation in a Cloud Data Center Using Modified Water Wave Optimization," *Wireless Personal Communications*, vol. 130, pp. 1005-1023, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Chaoze Lu, Jianchao Zhou, and Qifeng Zou, "An Optimized Approach for Container Deployment Driven by a Two-Stage Load Balancing Mechanism," *PloS One*, vol. 20, no. 1, pp. 1-32, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jyotsna P. Gabhane, Sunil Pathak, and Nita Thakare, "An Improved Multi-Objective Eagle Algorithm for Virtual Machine Placement in Cloud Environment," *Microsystem Technologies*, vol. 30, pp. 489-501, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Zhihua Li et al., "Resource-Efficient and Quality-Aware Virtual Machine Consolidation Method," *Journal of Grid Computing*, vol. 23, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[26] M. Menaka, and K.S. Senthil Kumar, "Supportive Particle Swarm Optimization with Time-Conscious Scheduling (SPSO-TCS) Algorithm in Cloud Computing for Optimized Load Balancing," *International Journal of Cognitive Computing in Engineering*, vol. 5, pp. 192-198, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Ruifeng Ma et al., "Qora: Neural-Enhanced Interference-Aware Resource Provisioning for Serverless Computing," *IEEE Transactions on Automation Science and Engineering*, pp. 1-16, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[28] Harpreet Kaur et al., "Enhanced K-Means Clustering of Tasks and Virtual Machines for Load Balancing in Fog Environment," *13th International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, pp. 565-570, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[29] Farheen Bano et al., "A Levelized Multiple Workflow Heterogeneous Earliest Finish Time Allocation Model for Infrastructure as a Service (IaaS) Cloud Environment," *Algorithms*, vol. 18, no. 2, pp. 1-31, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[30] Xuehan Li et al., "Two-Stage Offloading for an Enhancing Distributed Vehicular Edge Computing and Networks: Model and Algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 17744-17761, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Feng Zhang, "Design of Automated Container Layout and Resource Optimization Algorithm Based on Cloud Computing Technology," *International Conference on Mechatronics and Intelligent Control*, Wuhan, China, vol. 13447, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[32] Lilu Zhu et al., "Two-Stage Learning Approach for Semantic-Aware Task Scheduling in Container-Based Clouds," *IEEE Transactions on Cloud Computing*, vol. 13, no. 1, pp. 148-165, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[33] Peisong Li et al., "Reinforcement Learning Based Edge-End Collaboration for Multi-Task Scheduling in 6G Enabled Intelligent Autonomous Transport Systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1-14, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[34] Fahimeh Farahnakian et al., "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187-198, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[35] Lianpeng Li et al., "SLA-Aware and Energy-Efficient VM Consolidation in Cloud Data Centers Using Robust Linear Regression Prediction Model," *IEEE Access*, vol. 7, pp. 9490-9500, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[36] Fagui Liu et al., "A Virtual Machine Consolidation Algorithm Based on Ant Colony System and Extreme Learning Machine for Cloud Data Center," *IEEE Access*, vol. 8, pp. 53-67, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[37] Zhenrui Peng et al., "Modification of Fish Swarm Algorithm based on Levy Flight and Firefly Behavior," *Computational Intelligence and Neuroscience*, vol. 2018, no. 1, pp. 1-13, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[38] Zoltan Adam Mann, and Mate Szabó, "Which is the Best Algorithm for Virtual Machine Placement Optimization?," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[39] Zhoujun Ma et al., "Virtual Machine Migration Techniques for Optimizing Energy Consumption in Cloud Data Centers," *IEEE Access*, vol. 11, pp. 86739-86753, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[40] S. Supreeth, and Kirankumari Patil, "VM Scheduling for Efficient Dynamically Migrated Virtual Machines (VMS-EDMVM) in Cloud Computing Environment," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 16, no. 6, pp. 1892-1912, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[41] Yifan Shao et al., "A Dynamic Virtual Machine Resource Consolidation Strategy Based on a Gray Model and Improved Discrete Particle Swarm Optimization," *IEEE Access*, vol. 8, pp. 228639-228654, 2020. [CrossRef] [Google Scholar] [Publisher Link]