PSO-JSO: A Hybrid Metaheuristic for Load Balancing in Cloud Computing

Na Li
Department of Information Technology, ZhengZhou Vocational College of Finance and Taxation, Zhengzhou, Henan, 450048, China
E-mail: xinxizhengzhou@126.com

Keywords: cloud computing, load balancing, resource utilization, optimization

Received: March 22, 2025

Cloud computing platforms face growing challenges in efficiently allocating resources and balancing loads due to the dynamic and heterogeneous nature of workloads. This work introduces PSOJSO, an innovative hybrid optimization algorithm that fuses Particle Swarm Optimization (PSO) and Jellyfish Search Optimization (JSO). It presents a dynamic time-control mechanism and adaptive coefficients to balance global exploration and exploitation. Experiments were simulated under a time-sharing scheduling policy using CloudSim 3.0.2 for 2 data centers, eight virtual machines, and 10–100 cloudlets. PSOJSO is compared against five baseline algorithms: ACO, ABC, BA, CSA, and PSO alone. PSOJSO achieved a reduction of up to 25.3% in makespan, a 19.8% reduction in energy consumption, and a 17.5% enhancement in resource utilization, proving its validity for dynamic cloud environments.

Povzetek: Hibridni algoritem PSOJSO (PSO-JSO) je namenjen uravnoteženju obremenitve v računalništvu v oblaku. Z dinamičnim časovnim nadzorom zmanjša makespan, porabo energije in izboljša izkoriščenost virov, kar potrjuje njegovo učinkovitost v dinamičnih oblačnih okoljih.

1 Introduction

Cloud computing offers on-demand computational resources over the Internet [1]. It also introduced scalability and a pay-as-you-use model, allowing for its use across various industries [2]. Large dynamic workloads and configurable Virtual Machine (VM) setups are significant challenges for resource and workload management [3]. This can lead to inefficient resource utilization, bottlenecks, energy waste, and increased operating costs [4]. Hence, creative techniques are necessary to optimize the system's performance and energy efficiency. Correlation research into creative grid energy management indicates adaptive control is essential under uncertain conditions [5]. Moreover, machine learning techniques have been promising for investigating and optimizing dynamic business economics-like systems [6].

Optimal performance in cloud environments is achieved by proper load balancing. Even workload distribution through load balancing avoids the underutilization and overloading of VMs, optimizes response times, reduces energy consumption, and improves resource utilization [7]. An efficient load balancing strategy is essential while maintaining Quality of Service (QoS) and adhering to Service-Level Agreements (SLAs) in dynamic and complex cloud systems [8].

Recently, metaheuristics have emerged as one of the most effective methodologies for solving NP-hard problems, such as load balancing. Nature-inspired meta-heuristics provide flexible and robust frameworks for exploring complex solution spaces [9, 10]. Algorithms inspired by nature, such as Particle Swarm Optimization (PSO) and Jellyfish Search Optimization (JSO), have already demonstrated considerable promise in minimizing makespan and enhancing resource efficiency [11-13]. However, combining global exploration with local exploitation remains an open challenge.

The hybridization of PSO with JSO is intuitively inspired by their complementary strengths. While PSO has been highly efficient in exploring the global solution space, JSO possesses a strong capability of local exploitation to refine the solutions. Such a combination provides an unprecedented way to balance exploration and exploitation. It mitigates some of the drawbacks of standalone implementations, particularly in terms of enhanced performance in complex load-balancing scenarios.

This paper presents the PSOJSO algorithm, combining PSO and JSO for load-balancing in cloud computing. The novelty of this work lies in three main contributions. First, we propose a dynamic hybrid framework that alternates between PSO and JSO phases based on a nonlinear time control function, rather than combining the two heuristics statically. This allows the algorithm to leverage PSO's global exploration in early iterations and JSO's local exploitation in later stages. Second, we incorporate nonlinear, time-varying inertia weights and adaptive cognitive/social coefficients to enhance convergence and prevent premature stagnation. Third, we apply this hybrid PSOJSO formulation to a cloud computing context with multiple conflicting objectives, including makespan, energy efficiency, and resource utilization. To the best of our knowledge, this is the first implementation of such a hybrid algorithm within this specific performance-driven context.

2 Related work

Annie Poornima Princess and Radhamani [14] presented an efficient cloud workload balancing problem with a hybrid metaheuristics optimization approach, using Pigeon-inspired Optimization (PO) and Harries Hawks Optimization (HHO) algorithms. Kruekaew and Kimpan [15] suggested a Multi-objective Artificial Bee Colony Algorithm with Q-learning (MOABCQ) for task scheduling and resource utilization in cloud computing. Thakur and Goraya [16] suggested a hybrid optimization approach using Phasor PSO and Dragonfly algorithms, PPSO-DA, for load balancing and resource management. Ramya and Ayothi [17] developed HDWOA-LBM, a hybrid method combining dingo and whale optimization algorithms to maximize resource utilization and reliability while minimizing makespan.

Narwal [18] presented a credit-based scheduling method integrating Walrus and Lyrebird optimization algorithms called HO-CB-RALB-SA. Gabhane, et al. [19] proposed a hybrid algorithm combining Ant Colony Optimization (ACO) and Tabu Search (TS), called ACOTS, for scalable load distribution. Singhal, et al. [20] suggested the Rock Hyrax (RH)-based algorithm, focusing on dynamic load balancing and energy efficiency using QoS parameters. Karuppan and Bhalaji [21] proposed the African Vultures Algorithm (AVA) that balances server workloads while reducing makespan and energy usage.

In various reviewed studies, as represented in Table 1, several hybrid algorithms are proposed to perform load balancing with significant improvement in performance indicators for makespan, resource utilization, and energy efficiency. Nevertheless, most methods concentrate on specific aspects, such as throughput optimization or resource utilization, and very few provide a holistic balance between exploration and exploitation.

While prior hybrid metaheuristics, such as HDWOA-LBM, MOABCQ, and ACOTS, have achieved improvements in specific metrics, they often fall short in addressing all critical QoS parameters holistically, particularly under dynamic and heterogeneous workloads. Furthermore, many studies lack adaptability mechanisms or focus solely on either exploration or exploitation. Our proposed PSOJSO algorithm addresses these gaps through a combination of techniques: (1) a time-adaptive switching mechanism to balance global and local search phases, (2) nonlinear, time-varying inertia and acceleration coefficients to dynamically modulate the search behavior, and (3) chaotic logistic initialization to improve population diversity and prevent premature convergence. These components, integrated within a single framework, provide a novel and scalable solution for robust load balancing in cloud computing environments.

3 Cloud load balancing

This section provides a two-part explanation of load balancing. The first part presents an overview of cloud load balancing. The primary metrics related to load balancing are discussed in the second part.

3.1 Problem statement

Balancing the load will be effective in the cloud computing environment, leading to better resource utilization and sustained high system performance. This is generally concerned with distributing the arriving tasks among the available VMs to avoid underutilization or overloading of resources. Several benefits can be ensured through load balancing, including minimized energy consumption, shorter response times, and excellent system reliability. However, designing effective load balancing strategies has become very challenging due to the changing and heterogeneous structure of cloud workloads.

Load balancing is also concerned with route selection, minimizing bottlenecks, and preventing overflows. Without a well-thought-out strategy, poor cloud environments are often plagued with performance bottlenecks and even SLA violations. The dynamic assignment of tasks to heterogeneous VMs in cloud data centers presents formidable challenges toward workload imbalance avoidance; inefficient workload allocation will lead to server underperformance and increased operational costs.

Network topology greatly influences how resources are allocated and balanced. The unbalanced system, shown in Figure 1(a), illustrates the outcome of an ineffective load-balancing strategy, where some PMs are overloaded and others underutilized. In contrast, a balanced system, Figure 1 (b), illustrates the benefit of equal workload distribution to ensure all resources are utilized efficiently.

Therefore, the challenge lies in applying the load-balancing strategy dynamically in cloud environments to accommodate variations in workloads and heterogeneity, while aiming to maximize resource utilization, minimize delays, and reduce energy consumption. The paper explores a hybrid algorithm to optimally balance these loads and achieve high system efficiency.

3.2 Problem formulation

Load balancing in cloud computing facilitates even distribution of tasks among available VMs while meeting specific performance targets, including better response time, resource utilization, and energy usage. The formulation of this problem involves defining tasks, VMs, dependencies, and performance metrics that guide the optimization process.

Table 1: An overview of relevant works

Study	Algorithm	Main contribution	Tool/testbed	Metrics	Drawbacks
[14]	HHO+PO	Optimized load balancing and resource utilization	JAVA	Makespan, cost, throughput, and latency	Limited focus on energy consumption and dynamic workload handling
[15]	MOABCQ	Improved task scheduling using reinforcement learning	CloudSim	Makespan, throughput, cost, and imbalance degree	Does not consider energy efficiency or real-time adaptability
[16]	PPSO-DA	Balanced resource allocation across physical machines	CloudSim	Resource usage and load imbalance	High computational complexity and limited energy optimization
[17]	HDWOA-LBM	Enhanced resource utilization and reliability	CloudSim	Throughput, reliability, and makespan	Limited scalability and lack of energy consumption focus
[18]	HO-CB-RALB-SA	Credit-based scheduling for equitable task distribution	CloudSim	Makespan, cost, execution time, and resource usage	Complex implementation and limited metrics evaluation
[19]	ACOTS	Faster file delivery and better multi-resource load balancing	CloudSim	Throughput and task completion time	A narrow focus on throughput, lacking other critical metrics
[20]	RH	Addressed local maxima and improved energy efficiency	CloudSim	Makespan and energy usage	Limited consideration of workload distribution
[21]	AVA	Load balancing with minimized makespan and energy usage	CloudSim	Resource utilization and makespan	Overhead in evaluating selection factors
[22]	DRABC-LB	Dynamic task scheduling based on resource awareness		Resource utilization, response time, makespan, and throughput	Limited adaptability to highly dynamic cloud environments

The tasks in the system are denoted as $T = \{T_1, T_2, \dots, T_n\}$, where each T_i (for $1 \le i \le n$) represents a non-preemptive task with a specific set of instructions. These tasks are executed across a set of VMs $VM = \{VM_1, VM_2, \dots, VM_m\}$, where VM_j (for $1 \le j \le m$) refers to the j^{th} VM in the system. The VMs may have heterogeneous configurations, reflecting variations in computational power, memory, and storage capacities.

The dependencies between tasks are represented using a Directed Acyclic Graph (DAG), $G = (T, E)$, where T denotes the set of tasks and E signifies the edges between them. An edge $e(T_i, T_j)$ indicates that T_j cannot start execution until T_i is completed. These dependencies ensure the logical execution of tasks and impact the scheduling process. To evaluate the efficiency of a load-balancing mechanism, multiple performance metrics are used, each addressing a critical aspect of system performance:

Energy consumption (E): Energy consumption is modeled based on the power usage of Complementary Metal Oxide–Semiconductor (CMOS)-based microprocessors. The dynamic power is given by:

$$P_e = acv^2 f \tag{1}$$

Where ac is the dynamic power parameter, v is the supply voltage, and f is the processor frequency. The total energy consumption for all tasks is calculated using Eq. 2.

$$E = \sum_{i=1}^{n} TCT_i . acv_i^2 f_i \tag{2}$$

Where TCT_i stands for the completion time of i^{th} task, v_i refers to the voltage supplied to the processor on which i^{th} task is executed, and n denotes the number of VMs.

Imbalance degree (I): Eq. 3 quantifies the imbalance in task distribution among VMs.

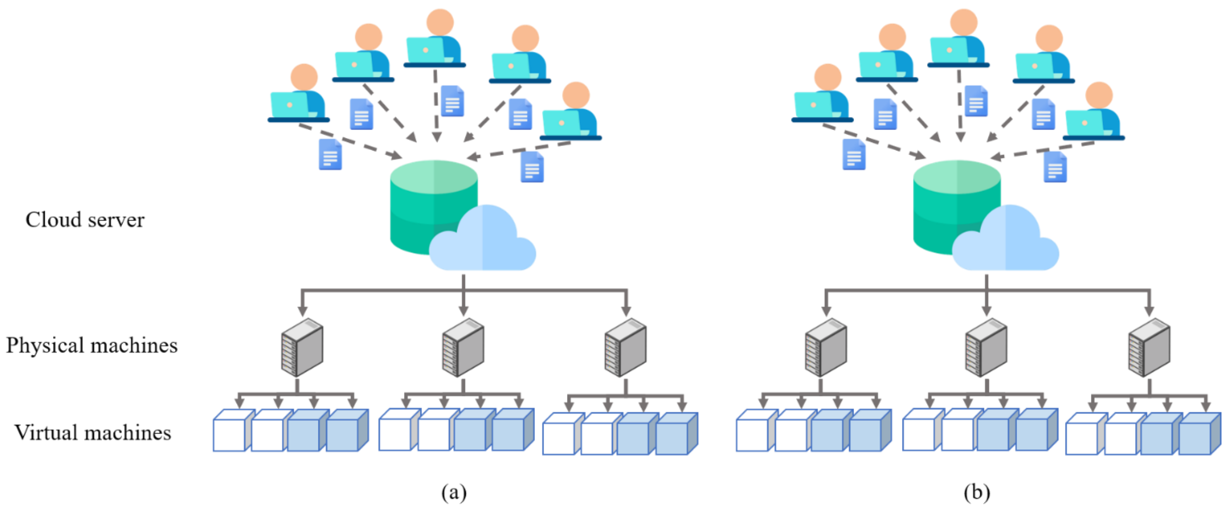

Figure 1: Network topology

$$D = \frac{T_{max} - T_{min}}{T_{avg}} \quad (3)$$

T_{max} and T_{min} are the maximum and minimum completion times across VMs. T_{avg} is the average task length calculated by Eq. 4, where m signifies the VM count.

$$T_{avg} = \frac{Total\ task\ lenght}{Total\ VM\ capcacity\ \times m} \quad (4)$$

Resource Utilization (*RU*): Resource utilization measures the percentage of resources used across all VMs and is given by:

$$RU_{VM} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} TCT_{ij}}{MS \times m} \times 100 \quad (5)$$

Higher resource utilization indicates more efficient use of VM resources.

Response Time (RT): The total response time for all tasks in the system is calculated as:

$$RT = \sum_{i=1}^{n}\sum_{j=1}^{m} TCT_{ij} \quad (6)$$

This metric evaluates the cumulative time spent on task execution across all VMs.

Makespan (MS): Makespan is the maximum completion time across all VMs, representing the time required to finish all tasks.

$$MS = max\left\{TCT_{VM_j} | j = 1,2,\dots,m\right\} \quad (7)$$

Task Completion Time (TCT): The time taken to execute a task T_i on the jth VM is calculated by E. 8.

$$TCT_{ij} = FT(T_i) - ST(T_i) \quad (8)$$

Where $ST(T_i)$ and $FT(T_i)$ are the start and finish times of Ti, respectively. For a specific VM VM_j, the total completion time of all its tasks is calculated as follows:

$$TCT_{VM_j} = \sum_{i=1}^{N_{VM_j}} TCT_{ij} \quad (9)$$

The optimization objective is to diminish the makespan, energy usage, and degree of imbalance while improving resource utilization and minimizing response time. This multi-objective problem is addressed using a fitness function that aggregates all metrics into a single-objective optimization framework as follows:

$$F = minimize\ \left(\beta_1 \frac{1}{RU} + \beta_2 MS + \beta_3 E\right) \quad (10)$$

Where β_1, β_2, and β_3 are weights assigned to each metric, reflecting their relative importance. The weights are normalized to ensure their sum equals 1. The following principles are applied to simplify the multi-target problem to a single-target one.

- Scalarization validity: It ensures linear trade-offs among objectives, making the single-target function properly represent the multi-target problem.
- Objective alignment: Maximization metrics, like resource utilization, are converted into minimization objectives to unify the fitness function.
- Normalization: Metrics are scaled to a standard range for meaningful aggregation.
- Weight assignment: Equal weights ($\beta_1 = \beta_2 = \beta_3 = 1/3$) are assigned initially, ensuring no metric is prioritized.

To guide the design and evaluation of the proposed PSOJSO algorithm, the following research questions (RQs) are formulated:

- RQ1: Can a hybrid PSO-JSO algorithm with dynamic time-control outperform existing metaheuristic methods in reducing makespan under variable cloud workloads?
- RQ2: Does PSOJSO lead to improved energy efficiency and resource utilization compared to standalone and hybrid baselines?
- RQ3: How does the use of nonlinear time-varying coefficients and chaotic initialization impact task scheduling accuracy and load balancing quality?
- RQ4: Is the PSOJSO framework scalable and adaptable to dynamic cloud environments with heterogeneous VM configurations and fluctuating task demands?

Although the scalarization method used in Eq. 10 simplifies multi-objective optimization by combining objectives into a single fitness function, it inherently assumes linear trade-offs between competing goals such as energy consumption and response time. This simplification is suitable for the current study's scope, which emphasizes algorithmic efficiency and comparative evaluation. Nonetheless, we acknowledge that Pareto-based multi-objective optimization could provide a more expressive treatment of conflicting objectives. Integrating a Pareto-dominance approach into PSOJSO is identified as a promising direction for future enhancement of this work.

4 System model

This part discusses the suggested load balancing scheme and the strategy for resolving the problem.

4.1 Proposed model

The cloud load balancing system in this approach utilizes a framework with multiple entities based on CloudSim. A multi-component task scheduling and load distribution mechanism optimizes energy consumption. The entire system processes tasks submitted by users through a centralized control mechanism, comprising several sub-entities, such as a Cloud Information Service (CIS), brokers, and VM managers.

The proposed model integrates multilevel components for cloud load balancing, ensuring efficiency in task scheduling, dynamic redistribution of loads, and energy-aware management. The proposed solution will be implemented using the CloudSim framework, allowing user-submitted tasks to be effectively allocated to the best resources.

The broker treats the tasks submitted by any user. The broker uses a CIS that maintains a continuously updated cloud data center database. The CIS records resource availability, configuration, and performance characteristics. Once the broker receives tasks, it queries CIS to find appropriate data centers. After selecting data centers, it finally allocates the tasks to them. The CIS plays a vital role in this process, ensuring that resource utilization is optimized while directing tasks to data centers with available capacity.

The arriving tasks are handled hierarchically within each data center, as shown by the multiple layers in Figure 2. The data center controller is responsible for executing the task. The scheduler executes tasks on a specific virtual machine according to predefined algorithms, such as Round Robin. After this, the load balancer monitors the dynamic task distribution and detects situations of under- or overloading. It finally controls the VM manager to handle the operational states, whether active or asleep, of the VMs according to the workload.

The load balancer plays an essential role in the system, distributing the workload evenly among the VMs. If the workload of a VM exceeds the upper threshold, the load balancer will remove tasks from it and redistribute them to underloaded VMs for optimal task allocation. If the workload of any VM exceeds the upper threshold, the load balancer removes tasks from it and reallocates them to underloaded VMs for optimal task allocation.

The hierarchical model ensures dynamic adaptability and scalability, effectively managing workloads and operational states of VMs while maintaining low energy consumption and optimizing resource utilization. This proposed model balances task execution at data centers effectively, providing a strong and energy-efficient solution for modern cloud environments.

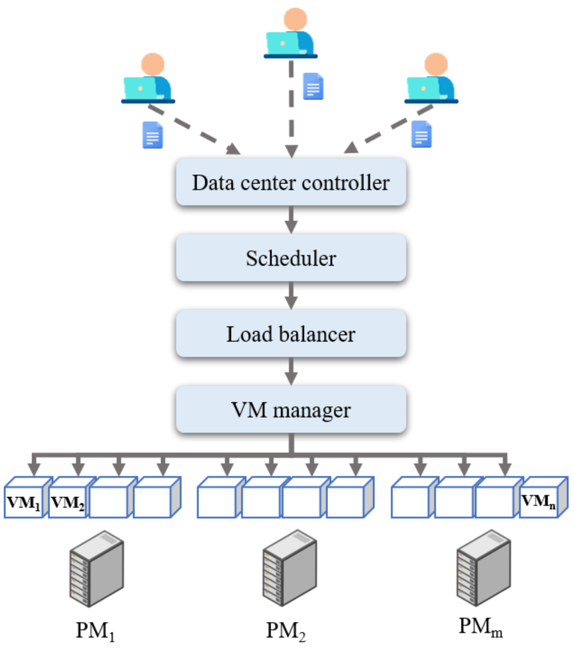

Figure 2: System model

4.2 Proposed method

The PSOJSO approach aims to combine the strengths of PSO and JSO by addressing their respective shortcomings. For example, PSO features global exploration capability and performs well in the early stages of the exploration process. In contrast, the JSO relies heavily on local exploitation to refine any solution. The PSOJSO approach balances the two algorithms, enabling the exploration and exploitation of load balancing in cloud computing. Novel elements are nonlinear time-varying weights, adaptive coefficients, and a time control mechanism that dynamically switches between PSO and JSO phases to ensure robust and efficient optimization. Detailed explanations of the components and mathematical formulations are given below.

PSO draws inspiration from the swarming tendency of birds and fishes. It was proposed as an iterative optimization method. In PSO, candidate solutions have been represented as a population of particles in an algorithm. Every particle updates its position in the search space concerning its best-known previous position and the global best position of the swarm. The update for the velocity of every particle should be made using Eq. 11.

$$V_i^{t+1} = \omega . V_i^t + c_1 . r_1 . (P_{best}^t - X_i^t) + c_2 . r_2 . (G_{best}^t - X_i^t) \tag{11}$$

Where w refers to the inertia weight controlling the balance between exploration and exploitation, c_1 and c_2 are cognitive and social coefficients, representing the particle's own experience and the swarm's collective knowledge, and r_1 and r_2 are random values in the range [0, 1].

The positions of particles are updated using Eq. 12.

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{12}$$

JSO models the dynamics of jellyfish within an ocean, including motions related to ocean currents or local swimming motions. The position of each jellyfish is updated based on ocean currents or movements within the swarm [23]. Initial positions of the individual jellyfish can be generated using a chaotic logistic map as follows:

$$X_i = LB + (UB - LB) . L_i \tag{13}$$

Where L_i is a logistic map value, and UB and LB are the upper and lower bounds of the search space, respectively.

The position update based on the ocean current is defined as follows:

$$X_i^{t+1} = X_i^t + r_1 . (X^* - 3 . r_2 . X_i^t) \tag{14}$$

Where X^* refers to the current global best position.

Movements within the swarm include passive motion and active motion, calculated as follows:

$$X_i^{t+1} = X_i^t + \gamma . r_1 . (UB - LB) \quad (15)$$

$$X_i^{t+1} = X_i^t + \omega . r_1 . \overrightarrow{Step} \quad (16)$$

Where γ is the motion coefficient and $\overrightarrow{Step}$ is determined by the direction of movement, depending on the quality of solutions.

PSOJSO alternates between PSO and JSO phases using a dynamic time control function $c(t)$, determining whether the algorithm follows the PSO or JSO phases.

$$c(t) = \left(1 - \frac{t}{T}\right) . (2r - 1) \quad (17)$$

Where t refers to the current iteration, and T refers to the total number of iterations. If $c(t) \geq 0.5$, the algorithm executes the PSO phase, favoring exploration. Otherwise, it switches to the JSO phase for exploitation.

To enhance exploration in the early stages and exploitation in the later stages, nonlinear time-varying inertia weights (w), cognitive coefficients (c_1), and social coefficients (c_2) are introduced:

$$\omega = \omega_{min} + (\omega_{max} - \omega_{min}) \left(1 - \frac{t}{T}\right)^{\beta} \quad (18)$$

$$c_1 = c_{min} + (c_{max} - c_{min}) sin\left(\frac{\pi}{2} . \left(1 - \frac{t}{T}\right)\right) \quad (19)$$

$$c_2 = c_{min} + (c_{max} - c_{min}) cos\left(\frac{\pi}{2} . \left(1 - \frac{t}{T}\right)\right) \quad (20)$$

Where β controls the rate of inertia weight decay, and w_{min}, w_{max}, c_{min}, and c_{max} define the lower and upper bounds of the weights.

In implementation, the control function $c(t)$ governs dynamic switching between phases. When $c(t) \geq 0.5$, the PSO phase is executed; when $c(t) < 0.5$, the algorithm enters the JSO phase. Thus, the first half of the iterations generally emphasize exploration, and the second half focuses on exploitation. The parameters were set as follows: $\omega_{max} = 0.9$, $\omega_{min} = 0.4$, $c_1 = c_2 = 2$, and β=1.5. These values were selected based on recommendations from prior hybrid metaheuristic literature and validated through empirical testing. A limited sensitivity analysis, varying each parameter within ±20%, demonstrated that the selected configuration maintained consistent performance and yielded superior results in terms of makespan and energy consumption.

PSOJSO uses the global search capability of PSO during the early stages and local exploitation by JSO during the later stages. Switching between PSO and JSO maintains a balance between exploration and exploitation, thus preventing early convergence. Such hybridization allows PSOJSO to adapt dynamically to the search space and optimizes cloud computing load balancing.

The algorithm initializes the position of all individuals in the population by using a chaotic logistic map to introduce randomness, thereby improving diversity and preventing convergence. Meanwhile, it sets the velocity of each particle or jellyfish to zero. The execution of the PSO or JSO phases is decided based on the time control mechanism. *c(t)* denotes a time control function indicating whether the PSO or JSO phases should be executed.

The computational complexity of the PSOJSO algorithm can be estimated as *O(N × D × T)*, where *N* is the number of particles, *D* is the dimensionality of the problem, and *T* is the number of iterations. This is consistent with the standard complexity of PSO and JSO, although PSOJSO introduces a slight overhead due to its dynamic switching mechanism and time-varying coefficient updates. However, this additional cost is marginal relative to the performance gains.

5 Results

The proposed load-balancing algorithm draws inspiration from nature-inspired optimization techniques. It involves identifying overloaded servers, selecting appropriate servers for task migration, and optimizing the fitness function to balance workloads across cloud environments.

The proposed algorithm was implemented in CloudSim version 3.0.2 on Windows 10. CloudSim is a well-known simulation tool that models and simulates cloud computing environments. This work utilized the tool to create virtual machines, data centers, and cloudlets, thereby realistically modeling a cloud computing scenario. The experimental environment has four data centers and workload resources with increasing task counts. Table 2 presents the details of the experimental environment configuration. The time-sharing policy was used for resource allocation in this architecture by the VMs across the four data centers.

While the overall system model is built upon CloudSim's default architecture, key components have been extended to support dynamic and energy-aware load balancing. The default round-robin policy was replaced with a custom task allocation policy governed by the PSOJSO fitness function. Additionally, the broker and load balancer were modified to dynamically monitor VM utilization and trigger task migration when workload thresholds are breached. These enhancements allow the system to respond adaptively to fluctuating workloads and to optimize key QoS metrics in real time.

Table 2: Simulation variables

Entities	Variables	Values
Data center	Count	2
VM	CPU	4
	Policy	Time-sharing
	MIPS	1000
Host	Operating system	Windows
	RAM	2 GB
	VMs	8
	Bandwidth	512
	Storage	40 GB
Cloudlets	RAM	4 GB
	Hosts	4
	Length	500-10000
User	Cloudlets	10-100
PSOJSO	Iterations	100
	Population size	50
Simulation	Random seed	42
	Runs per scenario	30

The parameters used in the PSOJSO algorithm were selected through empirical calibration and informed by prior literature on PSO and JSO hybrids. For the PSO phase, the inertia weight ω was varied nonlinearly from 0.9 to 0.4 using a decay exponent $\beta = 0.1$, while cognitive (c_1) and social (c_2) coefficients varied within [0.5, 2.5] using complementary sinusoidal schedules to encourage early exploration and late exploitation. These bounds were chosen based on commonly accepted best practices in swarm optimization. The motion coefficient γ for JSO was fixed at 0.1, ensuring a controlled range of swarm movement. The time control function $c(t)$ plays a critical role by governing the phase transition; its design promotes exploration during early iterations and gradual transition to exploitation.

To analyze its effectiveness, the proposed hybrid PSO-JSO algorithm was considered for various key QoS parameters, including makespan, energy efficiency, and response time. Makespan measures the total time to complete a set of tasks and reflects the scheduler's efficiency in allocating resources under dynamic workloads. It directly correlates with throughput and infrastructure utilization. Energy consumption is crucial due to the environmental and operational cost implications of large-scale data centers, were inefficient load distribution results in unnecessary power usage. Response time represents the delay experienced by users from task submission to completion, making it vital for meeting SLA requirements and user satisfaction.

Furthermore, the proposed method was tested against several prevalent optimization algorithms, including the Cuckoo Search Algorithm (CSA), Bat Algorithm (BA), ABC, PSO, and ACO. Results related to energy consumption, depicted in Figures 3 and 4, demonstrate that PSOJSO can reduce power consumption by dynamically adjusting the number of active VMs in response to workload demands. In this regard, resources are utilized efficiently during high workload periods, while idle servers are transitioned into a low-power state to minimize unnecessary energy consumption. This methodology not only enhances energy efficiency but also reduces operational costs.

Figures 5 and 6 depict that PSOJSO achieves lower and more consistent response times compared to the baseline methods, as observed across varying VM counts. In the proposed algorithm, the scheduler will be continuously updated in real-time by the broker about server availability to ensure that tasks are scheduled optimally. This reduces waiting time in server queues and maximizes the accuracy of estimated response times, thereby increasing user satisfaction and enabling the handling of more critical tasks.

Figures 7 and 8 illustrate the relationship between the number of cloudlets and the makespan. PSOJSO maintains a relatively stable makespan with increasing tasks, whereas methods such as ACO and PSO degrade their effectiveness with increased workloads. This is because the algorithm optimizes resource utilization and balances the workload effectively, thereby preventing excessive delays in task execution.

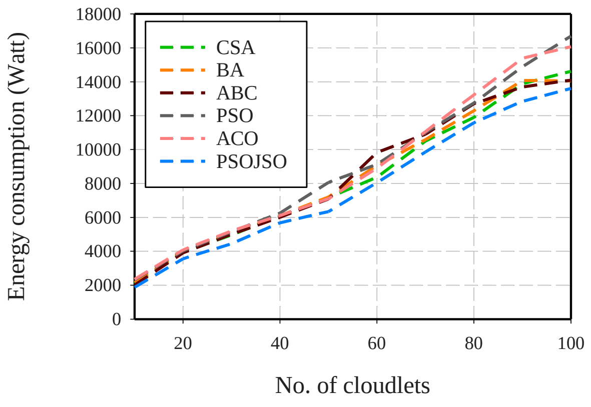

Figure 3: Energy consumption results for 10 VMs

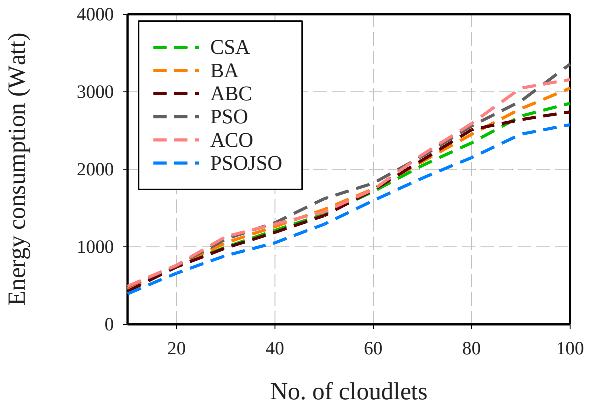

Figure 4: Energy consumption results for 50 VMs

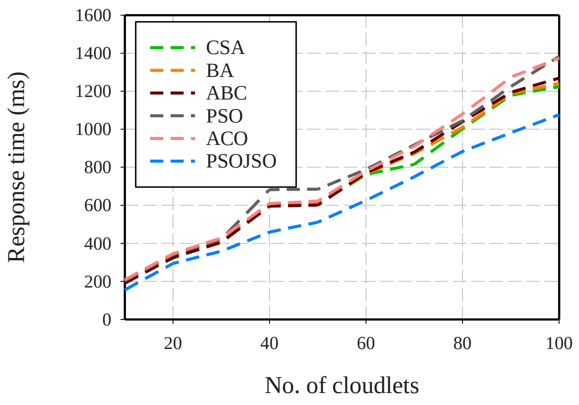

Figure 5: Response time results for 10 VMs

PSOJSO aims to enhance system performance by dynamically scaling the number of VMs up or down according to workload demand. This makes the model adaptable, ensuring sufficient resources to cope with high demands, thereby avoiding bottlenecks and slowdowns. During low times, it deactivates unnecessary VMs, minimizing energy wastage. These measures contribute to

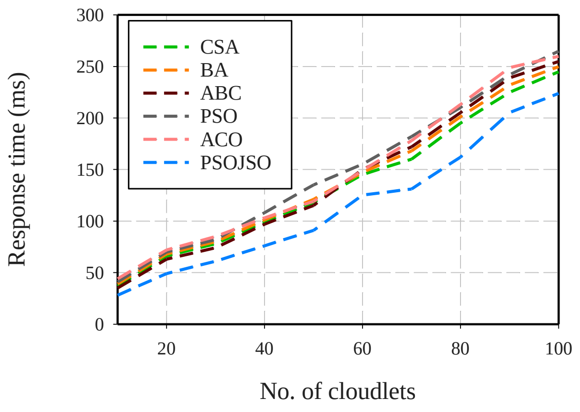

Figure 6: Response time results for 50 VMs

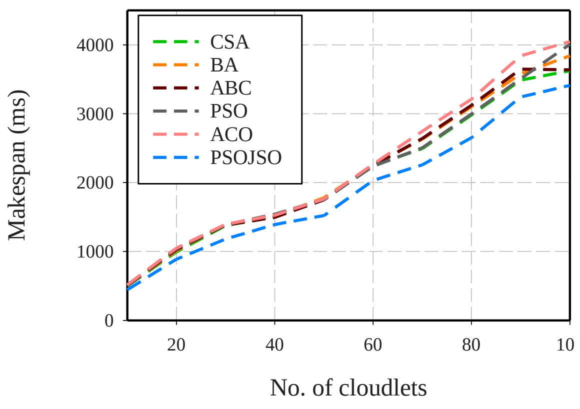

Figure 7: Makespan results for 10 VMs

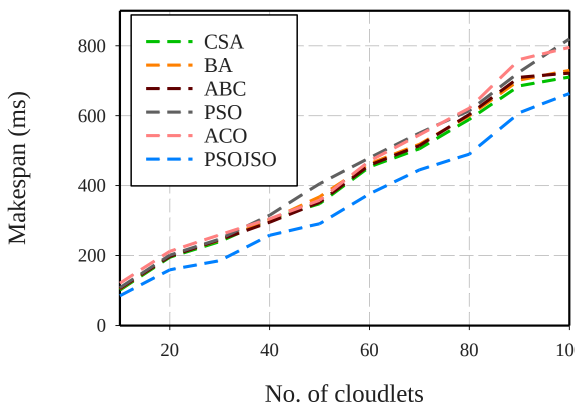

Figure 8: Makespan results for 50 VMs

a greener computing philosophy by saving energy while optimizing resource utilization.

6 Discussion

The proposed PSOJSO algorithm demonstrated superior performance in critical metrics, such as makespan, energy efficiency, resource utilization, and response time, when compared to widely used metaheuristics, including ACO, PSO, ABC, BA, and CSA. These improvements can be directly attributed to three key features.

- Dynamic time-control mechanism: This component enables the algorithm to dynamically shift between exploration (PSO phase) and exploitation (JSO phase) phases adaptively, thereby preventing early convergence, a common issue in standalone PSO, and improving solution diversity throughout the optimization process.
- Nonlinear time-varying coefficients: The inertia weight and acceleration coefficients are modulated over time using nonlinear decay functions. This ensures that early iterations favor global search, while later stages intensify local search, enhancing convergence stability and precision.
- Chaotic logistic initialization: By seeding the population using a chaotic map, the algorithm ensures a well-distributed initial search space, reducing the risk of stagnation and improving coverage in high-dimensional problems.

Compared to the state-of-the-art hybrid algorithms, such as HDWOA-LBM, MOABCQ, PPSO-DA, and ACOTS, the developed PSOJSO method shows better exploration and exploitation balance, mainly under dynamic and diverse cloud workload conditions. For one, although MOABCQ uses reinforcement learning to promote scheduling, there is no adaptive parameter control, and energy efficiency is not tackled directly. HDWOA-LBM prioritizes reliability and throughput but is unsuitable for scalability and optimizing energy. PPSO-DA and ACOTS have balanced resource allocation but high computational complexity or restricted metric consideration (e.g., throughput only). PSOJSO, however, introduces a nonlinear time-varying design of the coefficients, a chaotic initialization method, and dynamic switching between PSO and JSO phases so that PSOJSO can achieve better results along several QoS dimensions, including makespan, energy consumption, response time, and imbalance degree. These innovations enable PSOJSO to generalize and scale better under multi-resource settings and complete the gaps of other techniques.

PSOJSO maintains robust performance as the number of tasks increases, with makespan remaining stable even as workload intensity grows. This suggests good scalability in multi-VM, multi-data center environments. Additionally, since the method does not rely on problem-specific heuristics, it is generalizable to other cloud computing scenarios such as fog/edge computing or task offloading in IoT.

7 Conclusion

This paper presented the PSOJSO hybrid load-balancing algorithm for resource allocation problems and workload management for cloud computing environments. This integrated solution utilizes the exploration of PSO and exploitation of JSO to achieve an optimal trade-off, thereby preventing premature convergence and enhancing solution quality. It also utilized dynamic scaling, adaptive coefficients, and nonlinear inertia weights for improved resource utilization and efficiency. PSOJSO obtained stable makespan without increasing workloads, maximized throughput by reducing VM idle time, and decreased energy consumption due to adaptive resource scaling. A detailed performance evaluation based on CloudSim confirmed that the designed PSOJSO algorithm outperformed existing algorithms, including BA, PSO, ABC, ACO, and CSA, in all key QoS metrics, namely, makespan, response time, and energy efficiency. These enhancements would result in lower operational costs, higher user satisfaction, and improved system reliability.

References

[1] A. Gou, "Cloud-Computing-Enabled Transformer Architecture for the Design of Functional Clothing Structures," *Informatica,* vol. 49, no. 11, 2025, doi: https://doi.org/10.31449/inf.v49i11.6763.

[2] S. Sun, J. Dong, Z. Wang, X. Liu, and L. Han, "An on-demand collaborative edge caching strategy for edge–fog–cloud environment," *Computer Communications,* vol. 228, p. 107967, 2024, doi: https://doi.org/10.1016/j.comcom.2024.107967.

[3] K. Saidi and D. Bardou, "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities," *Cluster Computing,* vol. 26, no. 5, pp. 3069-3087, 2023, doi: https://doi.org/10.1007/s10586-023-04098-4.

[4] W. Yao, Z. Wang, Y. Hou, X. Zhu, X. Li, and Y. Xia, "An energy-efficient load balance strategy based on virtual machine consolidation in cloud environment," *Future Generation Computer Systems,* vol. 146, pp. 222-233, 2023, doi: https://doi.org/10.1016/j.future.2023.04.014.

[5] A. Kermani *et al.*, "Energy management system for smart grid in the presence of energy storage and photovoltaic systems," *International Journal of Photoenergy,* vol. 2023, no. 1, p. 5749756, 2023, doi: https://doi.org/10.1155/2023/5749756.

[6] M. B. Bagherabad, E. Rivandi, and M. J. Mehr, "Machine Learning for Analyzing Effects of Various Factors on Business Economic," *Authorea Preprints,* 2025, doi: https://doi.org/10.36227/techrxiv.174429010.09842200/v1.

[7] C. Vijaya and P. Srinivasan, "Multi-objective meta-heuristic technique for energy efficient virtual machine placement in cloud data centers," *Informatica,* vol. 48, no. 6, 2024, doi: https://doi.org/10.31449/inf.v48i6.5263.

[8] S. Ghafir, M. A. Alam, F. Siddiqui, and S. Naaz, "Load balancing in cloud computing via intelligent PSO-based feedback controller," *Sustainable Computing: Informatics and Systems,* vol. 41, p. 100948, 2024, doi: https://doi.org/10.1016/j.suscom.2023.100948.

[9] B. Pourghebleh, A. Aghaei Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," *Cluster Computing,* vol. 24, no. 3, pp. 2673-2696, 2021, doi: https://doi.org/10.1007/s10586-021-03294-4.

[10] M. Ahmadi *et al.*, "Optimal allocation of EVs parking lots and DG in micro grid using two-stage GA-PSO," *The Journal of Engineering,* vol. 2023, no. 2, p. e12237, 2023, doi: https://doi.org/10.1049/tje2.12237.

[11] Z. Jafari, A. Habibizad Navin, and A. Zamanifar, "Task scheduling approach in fog and cloud computing using Jellyfish Search (JS) optimizer and Improved Harris Hawks optimization (IHHO) algorithm enhanced by deep learning," *Cluster Computing,* pp. 1-25, 2024, doi: https://doi.org/10.1007/s10586-024-04347-0.

[12] Y. Gao, B. Yang, S. Wang, G. Fu, and P. Zhou, "A multi-objective service composition method considering the interests of tri-stakeholders in cloud manufacturing based on an enhanced jellyfish search optimizer," *Journal of Computational Science,* vol. 67, p. 101934, 2023, doi: https://doi.org/10.1016/j.jocs.2022.101934.

[13] K. Shao, H. Fu, and B. Wang, "An efficient combination of genetic algorithm and particle swarm optimization for scheduling data-intensive tasks in heterogeneous cloud computing," *Electronics,* vol. 12, no. 16, p. 3450, 2023, doi: https://doi.org/10.3390/electronics12163450.

[14] G. Annie Poornima Princess and A. Radhamani, "A hybrid meta-heuristic for optimal load balancing in cloud computing," *Journal of grid computing,* vol. 19, no. 2, p. 21, 2021, doi: https://doi.org/10.1007/s10723-021-09560-4.

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

[16] A. Thakur and M. S. Goraya, "RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment," *Simulation Modelling Practice and Theory,* vol. 116, p. 102485, 2022, doi: https://doi.org/10.1016/j.simpat.2021.102485.

[17] K. Ramya and S. Ayothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment," *Transactions on Emerging Telecommunications Technologies,* vol. 34, no. 5, p. e4760, 2023, doi: https://doi.org/10.1002/ett.4760.

[18] A. Narwal, "Resource Utilization Based on Hybrid WOA-LOA Optimization with Credit

Based Resource Aware Load Balancing and Scheduling Algorithm for Cloud Computing," *Journal of Grid Computing,* vol. 22, no. 3, p. 61, 2024, doi: https://doi.org/10.1007/s10723-024-09776-0.

[19] J. P. Gabhane, S. Pathak, and N. M. Thakare, "A novel hybrid multi-resource load balancing approach using ant colony optimization with Tabu search for cloud computing," *Innovations in Systems and Software Engineering,* vol. 19, no. 1, pp. 81-90, 2023, doi: https://doi.org/10.1007/s11334-022-00508-9.

[20] S. Singhal *et al.*, "Energy Efficient Load Balancing Algorithm for Cloud Computing Using Rock Hyrax Optimization," *IEEE Access,* 2024, doi: http://dx.doi.org/10.1109/ACCESS.2024.3380159.

[21] A. S. Karuppan and N. Bhalaji, "Efficient load balancing strategy for cloud computing environment with African vultures algorithm," *Wireless Networks,* pp. 1-17, 2024, doi: https://doi.org/10.1007/s11276-024-03810-5.

[22] R. Mishra and M. Gupta, "DRABC-LB: A Novel Resource-Aware Load Balancing Algorithm Based on Dynamic Artificial Bee Colony for Dynamic Resource Allocation in Cloud," *SN Computer Science,* vol. 5, no. 2, p. 233, 2024, doi: https://doi.org/10.1007/s42979-023-02570-x.

[23] J.-S. Chou and D.-N. Truong, "A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean," *Applied Mathematics and Computation,* vol. 389, p. 125535, 2021, doi: https://doi.org/10.1016/j.amc.2020.125535.