PSOMCD: Particle Swarm Optimization Algorithm Enhanced with Modified Crowding Distance for Load Balancing in Cloud Computing

Bolin ZHOU[1], Jiao GE[2], RuiRui ZHANG[3]*
College of Computer Science and Engineering, Cangzhou Normal University; Hebei Cangzhou 061001, China[1, 3]
College of Physics and Information Engineering, Cangzhou Normal University; Hebei Cangzhou 061001, China[2]

***Abstract*—Effective load balancing in cloud computing architectures is crucial towards enhancing resource utilization, response times, and stability in the system. The present study proposes a new strategy with a Particle Swarm Optimization algorithm enhanced with Modified Crowding Distance (PSOMCD) to tackle task scheduling among Virtual Machines (VMs) in dynamic scenarios. The traditional PSO algorithm is supplemented by an enhanced crowding distance mechanism by PSOMCD to improve diversity in decision spaces and convergence to optimal solutions. The multi-objective fitness function addresses principal challenges in cloud computing, including load distribution, energy consumption, and throughput optimization. The performance of the algorithm is demonstrated in simulations, comparing its performance with other optimization techniques available in the literature. Results prove that PSOMCD provides better task allocation, improved load balancing, and decreased energy usage, thus effectively managing resources in dynamic and heterogeneous cloud ecosystems.**

***Keywords*—Cloud computing; load balancing; particle swarm optimization; crowding distance; task allocation**

I. INTRODUCTION

A. Background

Cloud computing is a revolution in modern computing that offers scalable, flexible, and economical access to computing resources such as processing power, storage, and applications via the internet [1]. It enables on-demand service hosting without investing in or managing physical infrastructure [2]. At the heart of cloud systems are large-scale data centers that consist of several Virtual Machines (VMs) that execute incoming tasks from users and applications [3]. As the need for cloud services grows staggeringly, delivering optimal performance and high availability becomes increasingly important [4]. Efficient load balancing is one of the most vital components in ensuring this, and it evenly splits workloads across available VMs to prevent bottlenecks and increase overall resource utilization [5].

Inefficient load balancing in cloud computing causes drastic performance degradation in the form of overloading of VMs, underutilization of resources, high energy consumption, and task execution delays [6]. Such load imbalances degrade the Quality of Service (QoS) and result in high operational expenditures and energy waste in data centers. Traditional load balancing techniques are inefficient in tackling the dynamic and heterogeneous nature of cloud workloads [7]. Due to the conventional load balancing methods being inefficient in handling the heterogeneous and dynamic nature of cloud workloads, meta-heuristic algorithms can resolve such complex optimization issues [8]. Their efficacy in successfully exploring high and nonlinear search spaces has made them highly suited to dynamic load balancing in contemporary cloud computing settings.

Machine learning techniques have been widely adopted across diverse fields, including business economics, to analyze and predict the impact of various influencing factors using models such as regression, decision trees, support vector machines, and neural networks [9, 10]. Similarly, hybrid meta-heuristic frameworks, such as GA-PSO, have effectively addressed multi-objective optimization problems in complex infrastructures like microgrids, where technical and economic factors must be balanced simultaneously [11]. These advancements underscore the adaptability and robustness of intelligent algorithms in managing dynamic resource allocation scenarios, including cloud environments.

B. Challenges in Cloud Load Balancing

Cloud computing setups are dynamic, distributed, and heterogeneous. The tasks arrive unpredictably, the resource demands evolve, and VMs vary in capacities and workloads [12]. The intrinsic complexity makes load balancing a non-trivial issue, as it must adapt to changing conditions continuously in real-time. Additionally, cloud resources are distributed over multiple data centers and networks, and, as a result, centralized coordination becomes more complex [13]. The task distribution in this distributed system should involve wise mechanisms capable of responding to changing workloads in real-time and considering resource availability and performance variability [14].

The second major challenge of load balancing in clouds is its multi-objective nature. The balancing strategy must balance the workload evenly, reduce makespan (total duration to finish all the jobs) and energy usage, and provide assurance for Service-Level Agreements (SLAs). Conventional algorithms, i.e., static heuristics or basic PSO, typically fall short in such environments because of their weak adaptability and premature convergence to poor-quality solutions [15]. Static methods cannot conform to dynamic workloads in real-time, while basic PSO may not maintain diversity in complex search spaces, leading to inefficient VM utilization. Therefore, more intelligent and adaptive algorithms are needed to overcome such limitations,

particularly those that can deal with multiple objectives and dynamic constraints concurrently.

C. Role of Meta-Heuristic Algorithms

Meta-heuristics are proposed as viable substitutes for complicated optimization problems in cloud computing, where conventional algorithms cannot deliver the scalability and flexibility as requirements [16]. One of many possibilities among these alternatives is Particle Swarm Optimization (PSO), a popular population-based search methodology inspired by birds' flocking or fish schooling behaviors. Every particle in PSO is a potential solution and searches through the space by changing its position based on individual and collective experience. The cooperation strategy allows PSO to effectively search and explore the solution space, making it a suitable candidate for dynamic load balancing in cloud computing.

However, standard PSO suffers from multimodal and multi-objective optimization challenges, e.g., those encountered in cloud task scheduling [17]. It can converge prematurely to local optima or lose diversity in solutions, mainly when dealing with complex or high-dimensional spaces. Advanced variants of PSO have been proposed to counter these flaws. One of those add-ons consists of the Modified Crowding Distance (MCD) mechanism. MCD aims to estimate decision and objective space solution density more accurately to improve solution diversity and pull the swarm away from local optima. With MCD, PSO becomes more robust and capable of simultaneously delivering convergence and diversity multiple objectives.

D. Contributions of the Study

This study discusses a new application of Modified Crowding Distance-based Particle Swarm Optimization (PSOMCD) to overcome challenging load balancing on cloud computing platforms. The new approach successfully balances exploration and exploitation to enable the algorithm to explore sufficiently while converging solutions to optima. In contrast to conventional PSO or static heuristics, PSOMCD can discover multiple high-quality task-VM mappings by maintaining solution diversity throughout optimization. Consequently, it's very suitable in dynamic and multimodal clouds, where workload distributions and resource conditions keep changing and require adaptable and robust scheduling methods.

The significant contributions of this research are: 1) the creation and deployment of a meta-heuristic methodology using PSOMCD for dynamic cloud load balancing, 2) the definition of an overall multi-objective fitness function that considers load variation, makespan, and energy consumption, and 3) rigorous performance comparison of the proposed algorithm through simulation-based experiments with traditional load balancing methods. The experiments validate that PSOMCD significantly improves VM load balancing, reduces energy consumption, and increases system throughput considerably. This study describes the first integration of PSOMCD with cloud computing, with new opportunities opening up in large-scale distributed computing system resource optimization.

The rest of the present study is arranged in the following manner: Section II reviews existing literature on task scheduling and load balancing techniques in cloud computing. Section III formalize the problem. Section IV presents the PSOMCD algorithm, detailing the modifications to the traditional PSO and introducing the crowding distance mechanism. The experimental setup, simulation results, and performance comparisons with other existing optimization algorithms are presented in Section V. Section VI discusses the implications of the findings, algorithm strengths, and potential limitations. Section VII summarizes the contributions of the study.

II. Related Work

Negi, et al. [18] developed a hybrid dynamic load balancing algorithm called CMODLB by combining supervised (artificial neural network), unsupervised (Bayesian optimization-based enhanced K-means clustering), and soft computing (interval type 2 fuzzy logic system) methods. The underloaded and overloaded clusters are created by initially partitioning VMs into clusters. Scheduling of user tasks is performed with a multi-objective TOPSIS-PSO algorithm to achieve effective workload balance.

Sefati, et al. [19] proposed a Grey Wolf Optimization (GWO)-based algorithm to improve load balancing through an assessment of resource reliability. The solution determines idle or busy nodes in the cloud infrastructure, measures individual thresholds for each node, and determines fitness functions accordingly. Evidence from CloudSim shows tremendous improvements in response times and overall cost-effectiveness concerning other methods.

Neelakantan and Yadav [20] introduced a Hybrid Krill Herd and Whale-based Deep Belief Neural Model (HKHW-DBNM) for cloud load balancing. The algorithm dynamically estimates loads between VMs and allocates tasks to balance resource usage optimally. It achieves multiple objectives, such as makespan optimization, resource usage improvement, and a decrease in load imbalance.

Kaviarasan, et al. [21] suggested a nature-inspired lion optimization algorithm with an enhanced solution approach to effectively handle load balancing in cloud computing. Intending to balance workload distribution, the approach enhances throughput and response time with fault tolerance and minimum task migration overhead. The algorithm performs better by balancing exploration and exploitation potential, preventing convergence to local optima.

Singhal, et al. [22] suggested a rock hyrax algorithm resolving local optimum issues and energy consumption in load balancing. Utilizing meta-heuristic methodologies, this process adjusts workload dynamically based on QoS measures. Experimentations reveal that the proposed approach minimizes makespan and energy significantly compared to conventional scheduling methods. Results highlight its viability towards achieving optimum resource allocation, better energy efficiency, and better system robustness under diverse workload scenarios.

Hayyolalam and Özkasap [23] proposed the CBWO algorithm by merging chaos theory with black widow optimization to address cloud load balancing. The method balances resource usage and energy consumption, with evaluation performed using CloudSim. Experimental results show significant makespan and energy consumption improvements compared with other meta-heuristics. The CBWO algorithm improves system performance by balancing

computational loads and reducing energy in dynamic cloud settings.

Hussain, et al. [24] introduced the Dynamic Enhanced Resource-Aware Load Balance Algorithm (DE-RALBA), fully considering VM computing powers to counteract load imbalance. Through its implementation and validation using standard datasets on CloudSim, DE-RALBA surpasses state-of-the-art load balancing algorithms with a remarkable improvement in resource utilization and makespan reduction. Findings demonstrate that DE-RALBA exhibits a significant performance gain, achieving optimum resource utilization in dynamically scheduled high-performance computing tasks.

Haris and Zubair [25] introduced a hybrid Battle Royale Deep Reinforcement Learning (BRDRL) algorithm based on Deep Reinforcement Learning and Battle Royale Optimization methods. The hybrid approach identifies the optimum underloaded VMs based on BRO and uses DRL to achieve efficient job transfer with consideration of cost-effectiveness, load balancing, and makespan minimization. Through experimental investigations using CloudSim simulation, higher throughput, response times, and makespan performance are achieved compared with state-of-the-art optimization-based load balancing algorithms.

Although different meta-heuristic and hybrid algorithms have tried to address load balancing in cloud computing, specific critical gaps persist, as highlighted by Table I. Existing methods are primarily unable to achieve an appropriate trade-off between exploration and exploitation for high variability and heterogeneity in clouds. Algorithms with a single objective or too much dependency on classic heuristics also suffer from issues of early convergence, local optima, or lack of diversity in solution space exploration.

Moreover, attempts by prior research are seldom directed towards maintaining multiple optimum solutions under dynamically changing conditions, as needed in task allocation under varying conditions. This study fills the gaps using the PSOMCD algorithm. The algorithm improves convergence and diversity, effectively handles multiple criteria for optimization (load deviation, energy usage, makespan), and adapts robustly to clouds, thereby contributing a new and all-encompassing solution towards balancing cloud loads.

TABLE I. COMPARATIVE ANALYSIS OF LOAD BALANCING ALGORITHMS

Reference	Considered algorithm	Performance metrics	Advantages	Disadvantages
[18]	Artificial neural network, Bayesian optimization-based enhanced K-means clustering, and interval Type 2 fuzzy logic system	Makespan, completion time, and resource utilization	Combines multiple hybrid techniques for optimized VM migration and resource utilization.	Computationally intensive due to algorithm complexity.
[19]	Grey wolf optimization	Response time and cost	Simple and effective at reducing response time and operational costs.	Risks premature convergence due to limited diversity.
[20]	Hybrid krill herd and whale-based deep belief neural model	Makespan, resource usage, and degree of imbalance	Efficiently balances load with strong multi-objective capabilities.	Parameter tuning is complex and computationally heavy.
[21]	Bio-inspired lion optimization	Throughput, response time, and task migration overhead	Effectively balances exploration-exploitation, reducing migration overhead.	Sensitive to initial parameters, affecting performance stability.
[22]	Rock hyrax	Makespan and energy consumption	Optimizes energy use while efficiently avoiding local maxima.	Limited scalability and reduced exploration under heavy loads.
[23]	Black widow optimization	Makespan, energy consumption, and resource utilization	Improves multiple metrics simultaneously with effective energy and resource optimization.	High computational complexity and sensitive parameter settings.
[24]	Dynamic enhanced resource-aware load balancing	Resource utilization and makespan	Significantly enhances resource utilization through dynamic scheduling.	Real-time monitoring increases computational overhead.
[25]	Battle royale deep reinforcement learning	Throughput, response time, and makespan	Integrates reinforcement learning for adaptive, efficient load balancing.	Complexity and high computational cost due to deep learning training.

III. PROBLEM FORMULATION

Cloud computing environments comprise numerous VMs, each responsible for efficiently scheduling and balancing tasks across available resources. Consider a set of VMs, $M = \{m_1, m_2, \ldots, m_m\}$, each characterized by multiple resources, including CPU, memory, disk, and bandwidth. Each VM request can thus be depicted as a multi-dimensional vector, representing resource load in each dimension. The cloud environment includes several homogeneous servers managed by a central server responsible for processing VM requests. Servers accept incoming VM requests if sufficient memory resources are available, generating corresponding VMs to accommodate these tasks. Given n independent tasks to execute on m available VMs, a scheduler is required to optimally allocate tasks, maintaining balanced resource utilization and maximizing accepted requests.

To evaluate VM loads, consider user parameters represented by $U(RM, RCPU, RS, RPHPU, C)$, where RM indicates memory requirements, RCPU represents required CPU resources, RS is the request size, RPHPU denotes requests per user group timestamp, and C signifies requests per minute. Resource allocation across VM groups leads to the computation of memory load (LM^i) for each VM i, given by Eq. (1):

$$LM^i = RM^i + \frac{V_{m_{pi}}}{V_{m_{mi}}} \times 100\% \quad (1)$$

where, $V_{m_{pi}}$ and $V_{m_{mi}}$ represent the available and total memory percentage in VM i, respectively, and RM^i denotes the remaining memory before task allocation.

Similarly, the CPU load (LC^i) is computed using Eq. (2):

$$LC^i = RC^i + \frac{V_{m_{ci}}}{V_{m_{mi}}} \times 100\% \tag{2}$$

where, $V_{m_{ci}}$ and $V_{m_{mi}}$ signify the available CPU capacity and total CPU in VM i, and RC^i denotes remaining CPU resources.

Combining memory and CPU load yields the overall VM load, calculated using Eq. (3):

$$OL^i = \alpha \times LM^i + \beta \times LC^i \tag{3}$$

With weighting factors α and β, satisfying $\alpha + \beta = 1$. The overall load on each host server j is computed by summing the VM loads as follows, using Eq. (4):

$$LH^j = \sum_{i=0}^{m_j} OL^{ji} = \sum_{i=0}^{m_j} \alpha \times LM^{ji} + \beta \times LC^{ji} \tag{4}$$

Where m_j denotes active VMs on host j. The mean load across all physical hosts p in the cloud network is determined using Eq. (5):

$$\begin{aligned} AL &= \frac{\sum_{j=0}^{p} LH^j}{p} = \frac{\sum_{j=0}^{p}\sum_{i=0}^{m_j} OL^{ji}}{p} \\ &= \frac{\sum_{j=0}^{p}\sum_{i=0}^{m_j}(\lambda_1 \times LM^{ji} + \lambda_2 \times LC^{ji})}{p} \end{aligned} \tag{5}$$

Thus, the first objective fitness function, capturing the absolute difference in load across hosts, is expressed using Eq. (6):

$$F_1 = \sum_{j=0}^{p} \left| LH^j - AL \right| \tag{6}$$

The second fitness function addresses energy consumption (EC) and makespan (MS). Energy consumption is calculated based on the active and idle states of each VM, while makespan represents the maximum completion time among all VMs. Given the task assignment, the binary variable U_{ij}, the execution time T_j for each VM j is calculated as follows, using Eq. (7) and Eq. (8):

$$U_{ij} = \begin{cases} 1, & if\ T_i\ is\ assigned\ to\ VM_j \\ 0, & otherwise \end{cases} \tag{7}$$

$$T_j = \sum_{i=1}^{n} U_{ij} \times TC_{ij} \tag{8}$$

where, TC_{ij} is the completion time of the i^{th} task on VM j is computed using Eq. (9):

$$TC_{ij} = \frac{L_i}{PS_j} \tag{9}$$

where, L_i denotes the length of task i in Million Instructions, and PS_j is the processing speed of j^{th} VM. The makespan measures the total execution time across all VMs calculated as follows, using Eq. (10):

$$MS = Max(T_j),\ \ 1 \le j \le m \tag{10}$$

The total energy consumption for each VM j is computed using Eq. (11):

$$F_2 = EC = \sum_{j=1}^{m} \left[\left(T_j \times \varepsilon_j + \left(MS - T_j \right) \varpi_j \right) \right] \times PS_j \tag{11}$$

where, ε_j and ϖ_j represent the energy (joules/Millions of Instructions) consumed in active and idle states, respectively.

The third fitness function calculates delay cost, calculated from the number of tasks received and processed in the cloud network using Eq. (12):

$$F_3 = \alpha_1 \times \frac{NIPT_i}{MNIPS} + \alpha_2 \times L_i \tag{12}$$

where, $NIPT_i$ is the number of instructions per task, $MNIPS$ is the processing unit's instructions per second, and L_i is the estimated penalty cost for delayed tasks. Weights α_1 and α_2 typically take values of 0.7 and 0.3, respectively.

The final weighted objective function integrates these three objectives into a weighted optimization problem as follows using Eq. (13):

$$F = \beta_1 \times F_1 + \beta_2 \times F_2 + \beta_3 \times F_3 \tag{13}$$

where, β_1, β_2, and β_3 are weight factors representing load balancing, energy efficiency, and delay cost importance, respectively. For optimization purposes, these weights are often set equally ($\beta_1 = 1, \beta_2 = 0.5, \beta_3 = 0.5$), ensuring balanced prioritization of all criteria. Thus, optimal load balancing is achieved by diminishing the combined fitness function according to predefined criteria weights.

IV. PROPOSED ALGORITHM

To effectively address the cloud computing load balancing problem, this study proposes a novel PSO algorithm enhanced by the MCD approach. The PSOMCD algorithm aims to optimally distribute tasks across VMs by considering multiple objectives: balancing resource utilization, reducing makespan, and minimizing energy consumption. The core innovation of PSOMCD is introducing an MCD mechanism combined with non-dominated sorting, thereby overcoming the limitations of conventional crowding distance measures. In traditional Crowding Distance (CD), crowding evaluations between solutions can be misleading due to inaccurate neighborhood identifications. Therefore, the proposed MCD strategy integrates an Affinity Propagation Clustering (APC) method, ensuring solutions with identical dominance levels are accurately grouped. For each solution i, crowding distances in the decision and objective spaces ($CD_{i,x}$ and $CD_{i,f}$) are calculated precisely, ensuring true proximity-based diversity using Eq. (14) and Eq. (15):

$$CD_{i,x} = \frac{|x_{i-1,1} - x_{i,1}| \cdot |x_{i,1} - x_{i+1,1}|}{\left(max|x_{i,1} - x_{i+1,1}|\right)^2} + \frac{|x_{i-1,2} - x_{i,2}| \cdot |x_{i,2} - x_{i+1,2}|}{\left(max|x_{i,2} - x_{i+1,2}|\right)^2} \quad (14)$$

$$CD_{i,f} = \frac{|f_{i-1,1} - f_{i,1}| \cdot |f_{i,1} - f_{i+1,1}|}{\left(max|f_{i,1} - f_{i+1,1}|\right)^2} + \frac{|f_{i-1,2} - f_{i,2}| \cdot |f_{i,2} - f_{i+1,2}|}{\left(max|f_{i,2} - f_{i+1,2}|\right)^2} \quad (15)$$

The MCD value for each solution is finalized based on a comparative assessment with average crowding values as follows, using Eq. (16):

$$MCD_i = \begin{cases} max\left(CD_{i,x}, CD_{i,f}\right), & if\ CD_{i,x} > CD_{avg,x}\ or\ CD_{i,f} \\ min\left(CD_{i,x}, CD_{i,f}\right), & otherwise \end{cases} \quad (16)$$

Next, the algorithm adopts a cosine similarity-based elite selection method to determine elite solutions from an External Archive (EXA), enhancing decision-space diversity. Initially, the external archive selects candidate elite solutions from non-dominated groups based on top MCD rankings. The optimal neighborhood solution $Nbest_i$ for a given particle is chosen through a cosine similarity-based tournament selection, defined as follows, using Eq. (17):

$$cos(A, B) = \frac{A.B}{|A| \times |B|} \quad (17)$$

Solutions with higher cosine similarity exhibit stronger directional relationships in the decision space, effectively preserving solution diversity and preventing clustering in local areas.

Finally, an offspring competition mechanism is introduced to prevent premature convergence further and maintain an effective exploration-exploitation balance. A candidate sampling set is created, consisting of the global best solution ($Gbest(t)$), historical best ($Pbest_i(t)$), and neighborhood optimal solution ($Nbest_i(t)$) using Eq. (18):

$$\text{sampleSet}_i(t) = [Gbest(t), Pbest_i(t), Nbest_i(t)] \quad (18)$$

Offspring particles are then generated through Gaussian sampling and mutation processes, calculated using Eq. (19):

$$O_i(t) = \mathcal{N}\left(\text{mean}\left(\text{sampleSet}_i(t)\right), \text{std}\left(\text{sampleSet}_i(t)\right) + \varepsilon\right) \quad (19)$$

The mutation may further diversify the offspring particles when a random number is lower than a predefined mutation probability (rm = 0.01) using Eq. (20):

$$O_i^d = x_{min}^d + rand.\left|x_{max}^d - x_{min}^d\right|, \ if\ rand < rm \quad (20)$$

where, x_{min}^d and x_{max}^d represent boundary conditions in the search space dimensions relevant to VM resource capacities.

By integrating these strategic mechanisms, PSOMCD effectively navigates complex decision-making and objective spaces. Consequently, it provides an efficient, robust solution for the multi-objective load balancing optimization problem formulated in cloud computing environments. Algorithm 1 presents the pseudocode of the proposed algorithm.

Algorithm 1 Pseudocode of PSOMCD

Inputs: Population P, population size ($popsize$), maximum iterations (T_{max})
Output: Optimal solution set stored in the External Archive (EXA)
Step 1: (Initialization)
Generate the initial population $P(0)$ randomly and evaluate each individual's fitness.
Step 2: (Archive preparation)
Initialize Personal Best Archive (PBA) and EXA.
For each individual i in the population, set $PBA(i)$ to the initial individual $P_i(0)$.
Initially set the external archive: $EXA = P(0)$
Step 3: (Main loop - iterative process)
For iteration $t = 1,2,\ldots,T_{max}$:
a) (Ranking and MCD calculation)
Assign non-dominated ranks and compute the MCD values for individuals in EXA using non-dominated sorting and MCD calculation.
Identify the global best individual G$best(t)$ as the top-ranked individual from the sorted EXA.
b) (Update global best)
If $t = 1$, explicitly perform the non-dominated sorting on EXA and select the first individual from sorted EXA as the initial global best G$best(1)$.
c) (Particle update)
For each individual i in the population (from 1 to $popsize$):
Update P$best_i(t)$ as the top-ranked individual from sorted $PBA(i)$.
Select neighborhood best N$best_i(t)$ from EXA using cosine similarity-based elite selection.
Update particle $P_i(t+1)$ positions and velocities according to PSO updating rules and then evaluate their fitness.
d) (Offspring generation and competition)
Generate offspring particles through the offspring competition mechanism, applying Gaussian sampling and mutation operations.
Evaluate offspring particles $O_i(t)$.
e) (Archive updates)
Add updated individuals $P_i(t+1)$ and offspring $O_i(t)$ into their respective personal best archives.
Recalculate non-dominated ranks and MCD values for each individual within $PBA(i)$.
f) (External archive and global best update)
Integrate current population $P(t+1)$ and offspring $O(t)$ into EXA.
Perform non-dominated sorting and recalculate MCD values for all solutions in EXA.
Update G$best(t+1)$ from sorted population $P(t+1)$.
Retain only the top $popsize$ individuals in EXA to ensure a controlled archive size.

Step 4: (Termination)
The process repeats until $t + T_{max}$, returning the final set of optimal solutions stored in EXA.

V. RESULTS

The effectiveness of the PSOMCD algorithm was tested using CloudSim 3.0.3, running on a computing platform comprising an Intel Core i7 processor, 8 GB RAM, a CPU at 3.4 GHz, and Windows 10 OS. Table II illustrates the simulation setup for the experiments. Two basic simulation scenarios were employed to verify the performance of the algorithm: first, by maintaining the VMs as a constant parameter equal to 100 (virtual running over 10 processors) and increasing the tasks incrementally up to 2000 in intervals of 50; second, by maintaining the task as a constant parameter equal to 1000 and changing VMs as a parameter between 10 and a maximum of 100 in intervals of 50.

Several performance indicators were utilized to analyze the PSOMCD algorithm in detail, such as makespan, energy utilization, standard deviation (representing load balance), throughput, the number of migrated tasks, task idle time, imbalance degree, and the time taken by VMs. As shown in Table III, the weightage of the fitness value was determined by practicing various values corresponding to the objective function.

TABLE II. DETAILS OF THE SIMULATION SETUP

Component	Quantity	Parameter	Specification
Hosts	20	Number of cores	6
		VM Monitor	Xen
		Bandwidth	15 GB/s
		RAM	16 GB
		Storage	4 TB
		MIPS	177,730
Virtual machines	10-100	VM monitor	Xen
		Number of processing elements	1
		Bandwidth	10GB
		Memory	1 GB/s
		Processor speed	9725 MIPS
Data center	1	Cost per storage unit	0.001
		Cost per memory unit	0.05
		Base cost	3
		VM monitor	Xen
		Operating system	Linux
		Architecture	X86

TABLE III. IMPACT OF WEIGHT PARAMETERS ON LOAD BALANCING PERFORMANCE

Weights			Performance metrics and convergence efficiency			
β_1	β_2	β_3	Energy consumption (KJ)	Standard deviation	Throughput (req/ms)	Makespan (ms)
1	0.9	0.9	310.12	0.378	9.12	9669.15
		0.8	299.53	0.377	8.68	9411.27
		0.7	294.88	0.363	8.51	9276.19
		0.6	290.58	0.415	8.33	9068.12
		0.5	281.05	0.392	8.09	8956.92
	0.8	0.9	251.63	0.375	8.85	9427.16
		0.8	244.11	0.363	8.71	9388.13
		0.7	230.96	0.341	8.35	9160.32
		0.6	225.06	0.313	8.13	9050.25
		0.5	213.67	0.299	7.89	8937.99
	0.7	0.9	297.17	0.352	8.47	9174.12

		0.8	290.54	0.318	8.35	9131.73
		0.7	288.35	0.335	8.61	9193.88
		0.6	284.18	0.310	8.05	9011.52
		0.5	277.32	0.301	7.94	8582.72
	0.6	0.9	266.05	0.309	8.27	9086.43
		0.8	261.15	0.299	8.01	8793.58
		0.7	257.43	0.286	7.86	8662.94
		0.6	251.02	0.257	7.34	8531.11
		0.5	246.51	0.236	7.41	8370.05
	0.5	0.9	241.48	0.273	7.32	8892.19
		0.8	234.35	0.175	7.17	8466.16
		0.7	225.26	0.098	6.71	8134.31
		0.6	217.17	0.083	5.76	7952.41
		0.5	175.78	0.069	5.43	7785.66

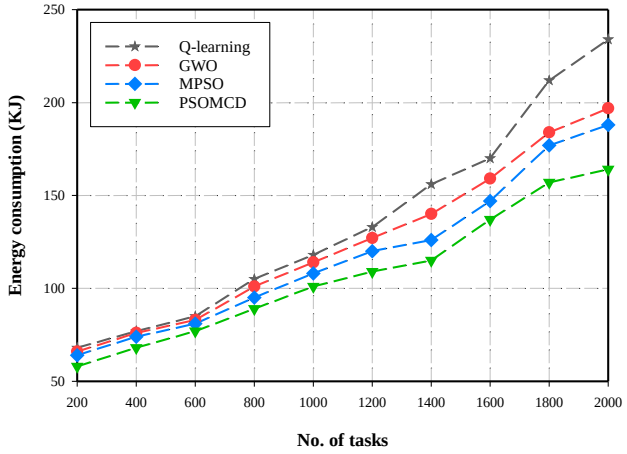

Fig. 1. Energy consumption with varying number of tasks for a fixed VM count

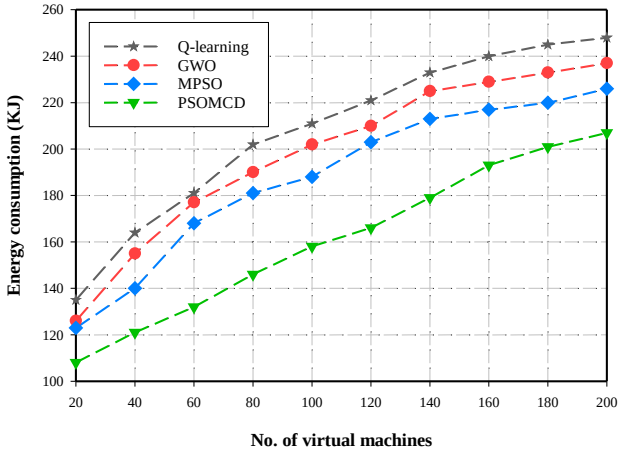

Fig. 2. Energy consumption with varying VMs for a fixed task count

Simulation results represented in Fig. 1 and Fig. 2 indicate that the PSOMCD algorithm requires less power than Q-learning, GWO, and MPSO algorithms in the execution of load balancing operations. Comparative figures for the makespan (response time) shown in Fig. 3 and Fig. 4 illustrate the efficiency of PSOMCD in the reduction of time taken for computation operations, thus enhancing the QoS provided by the system. Consequently, the findings present evidence of improved stability and performance by PSOMCD when computation workloads are balanced.

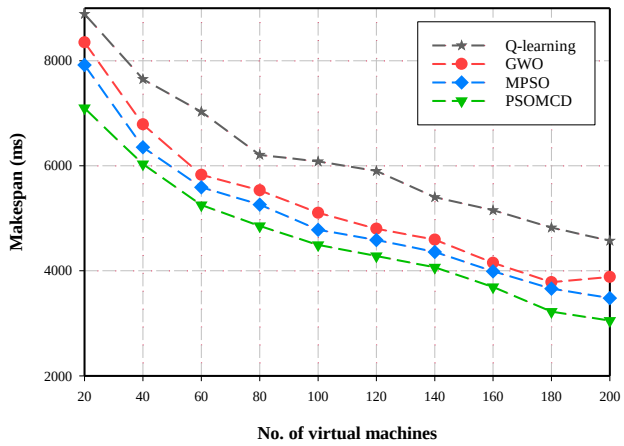

Fig. 3. Makespan with varying VMs for a fixed task count

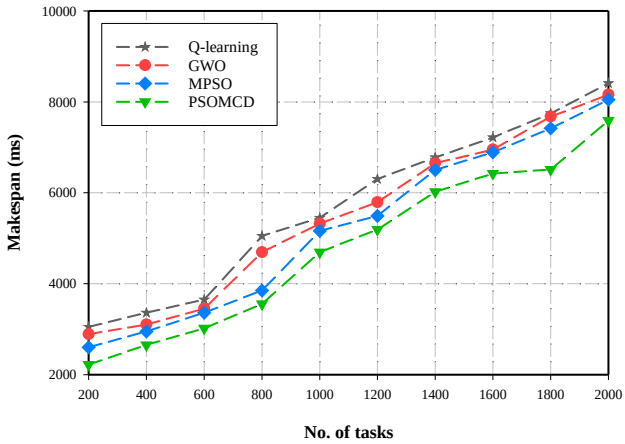

Fig. 4. Makespan with varying number of tasks for a fixed VM count

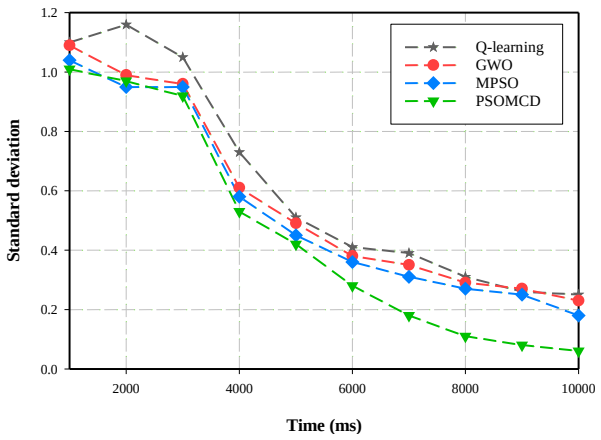

Fig. 5. Standard deviation comparison

Also, load balancing performance was assessed by measuring the standard deviation of resource use (Fig. 5). Low values in the standard deviation reflect higher load balancing performance, a metric in which PSOMCD performed much better than MPSO and Q-Learning, especially as simulation time elapsed. This performance advantage derives from the increased speed of PSOMCD in detecting and allocating optimal resources quickly. This balances computational loads and remaining resources among the VMs.

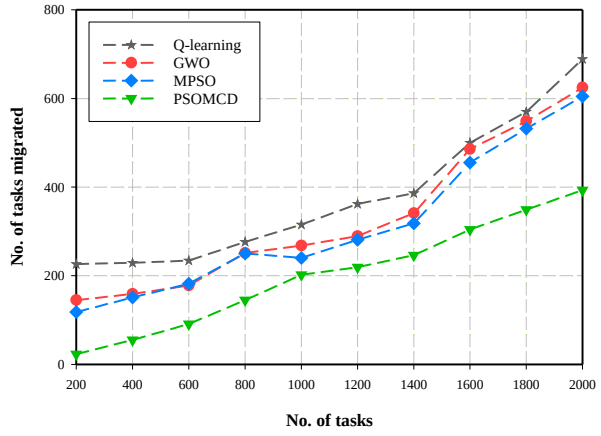

Fig. 6. Number of migrations comparison

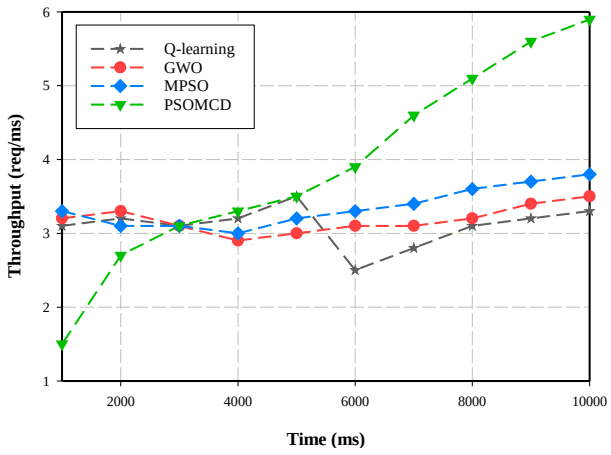

Fig. 7. Throughput comparison

Task migration, another key measure of effective resource utilization, was compared to the other (Fig. 6). PSOMCD had fewer task migrations, emphasizing the efficacy of resource pre-planning. Moreover, throughput analysis (Fig. 7) shows PSOMCD performed better than other approaches in externally managing resource demands with higher reliability in the cloud system, providing superior availability and responsiveness.

VI. Discussion

The performance evaluation demonstrates that PSOMCD effectively addresses the core challenges of dynamic task scheduling in cloud computing. The proposed algorithm maintains population diversity and avoids premature convergence by integrating the MCD mechanism with Particle Swarm Optimization. These enhancements result in improved resource utilization, reduced makespan, and lower energy consumption compared to benchmark methods such as MPSO, Q-learning, and GWO. The ability of PSOMCD to maintain lower task migration rates and higher throughput further underscores its robustness in adapting to fluctuating workloads.

Unlike traditional PSO-based or heuristic techniques, PSOMCD successfully balances multiple objectives simultaneously, thanks to its composite fitness function that considers load deviation, energy usage, and delay cost. This multi-objective orientation makes it highly suitable for modern cloud environments where service-level agreements and energy efficiency are equally critical. Moreover, incorporating elite selection via cosine similarity and offspring competition mechanisms enhances the algorithm's exploratory capabilities without sacrificing convergence performance.

While simulation results confirm PSOMCD's superiority, certain limitations warrant further investigation. The added complexity from clustering and diversity preservation strategies increases computational overhead, which may impact scalability in real-time or large-scale deployments. Additionally, the algorithm has been tested in a controlled CloudSim environment; future work should explore its performance in live cloud infrastructures and hybrid or multi-cloud systems. Adaptive parameter tuning and integration with reinforcement learning may further enhance responsiveness to evolving cloud conditions.

VII. Conclusion

This study introduced and tested a new meta-heuristic algorithm, PSOMCD, for load balancing in cloud computing environments. The PSOMCD framework integrated a modified crowding distance mechanism, affinity propagation clustering, a cosine similarity-based elite selection policy, and an offspring competition mechanism. All these advances resolved the issues related to early convergence, maintaining diversity in the solution, and the issues related to the complexity of multi-objective optimization in conventional algorithms.

A detailed simulation carried out on CloudSim 3.0.3 proved PSOMCD's performance advantage. Experimental settings considered varying numbers of tasks and VMs, and the findings proved considerable enhancement in load balancing, less power usage, lower makespan, a minor degree of imbalance, higher throughput, and reduced idle time over current algorithms such

as MPSO, Q-learning, and IPSO. Real-platform experiments also verified PSOMCD's practicality and resilience.

Future research will focus on extending PSOMCD to real-time and large-scale cloud environments, including hybrid and multi-cloud architectures. Integrating adaptive parameter tuning or reinforcement learning could enhance its responsiveness to dynamic workloads. Additionally, evaluating PSOMCD's performance under real-world constraints such as network latency, hardware failures, and user-driven demand variations will provide deeper insights into its practical deployment potential.

REFERENCES

[1] X. Zhang, "Optimizing scientific workflow scheduling in cloud computing: a multi-level approach using whale optimization algorithm," Journal of Engineering and Applied Science, vol. 71, no. 1, p. 175, 2024.

[2] H. Wang, K. J. Mathews, M. Golec, S. S. Gill, and S. Uhlig, "AmazonAICloud: proactive resource allocation using amazon chronos based time series model for sustainable cloud computing," Computing, vol. 107, no. 3, p. 77, 2025.

[3] A. Ullah and N. M. Nawi, "An improved in tasks allocation system for virtual machines in cloud computing using HBAC algorithm," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 4, pp. 3713-3726, 2023.

[4] A. Q. Khan, M. Matskin, R. Prodan, C. Bussler, D. Roman, and A. Soylu, "Cost modelling and optimisation for cloud: a graph-based approach," Journal of Cloud Computing, vol. 13, no. 1, p. 147, 2024.

[5] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," Ieee Access, vol. 9, pp. 41731-41744, 2021.

[6] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, vol. 23, no. 2, pp. 641-661, 2020.

[7] M. S. 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.

[8] P. Li, H. Wang, G. Tian, and Z. Fan, "Towards Sustainable Cloud Computing: Load Balancing with Nature-Inspired Meta-Heuristic Algorithms," Electronics, vol. 13, no. 13, p. 2578, 2024.

[9] 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.

[10] M. I. Al-Karkhi and G. Rządkowski, "Innovative Machine Learning Approaches for Complexity in Economic Forecasting and SME Growth: A Comprehensive Review," Journal of Economy and Technology, 2025.

[11] 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.

[12] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single - objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[13] F. Yunlong and L. Jie, "Incentive approaches for cloud computing: challenges and solutions," Journal of Engineering and Applied Science, vol. 71, no. 1, p. 51, 2024.

[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.

[15] M. S. R. Krishna and D. K. Vali, "Meta-RHDC: Meta Reinforcement Learning Driven Hybrid Lyrebird Falcon Optimization for Dynamic Load Balancing in Cloud Computing," IEEE Access, vol. 13, pp. 36550-36574, 2025.

[16] S. Jomah and A. S, "Meta-Heuristic Scheduling: A Review on Swarm Intelligence and Hybrid Meta-Heuristics Algorithms for Cloud Computing," in Operations Research Forum, 2024, vol. 5, no. 4: Springer, p. 94.

[17] P. Agarwal, R. Agrawal, and B. Kaur, "Multi-objective particle swarm optimization with guided exploration for multimodal problems," Applied Soft Computing, vol. 120, p. 108684, 2022.

[18] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: an efficient load balancing approach in cloud computing environment," The Journal of Supercomputing, vol. 77, no. 8, pp. 8787-8839, 2021.

[19] S. Sefati, M. Mousavinasab, and R. Zareh Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," The Journal of Supercomputing, vol. 78, no. 1, pp. 18-42, 2022.

[20] P. Neelakantan and N. S. Yadav, "An optimized load balancing strategy for an enhancement of cloud computing environment," Wireless Personal Communications, vol. 131, no. 3, pp. 1745-1765, 2023.

[21] R. Kaviarasan, G. Balamurugan, R. Kalaiyarasan, and Y. Venkata Ravindra Reddy, "Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm," E-prime-advances in electrical engineering, electronics and energy, vol. 6, p. 100326, 2023.

[22] S. Singhal et al., "Energy Efficient Load Balancing Algorithm for Cloud Computing Using Rock Hyrax Optimization," IEEE Access, 2024.

[23] V. Hayyolalam and Ö. Özkasap, "CBWO: A Novel Multi-objective Load Balancing Technique for Cloud Computing," Future Generation Computer Systems, vol. 164, p. 107561, 2025.

[24] A. Hussain, M. Aleem, A. U. Rehman, and U. Arshad, "DE-RALBA: dynamic enhanced resource aware load balancing algorithm for cloud computing," PeerJ Computer Science, vol. 11, p. e2739, 2025.

[25] M. Haris and S. Zubair, "Battle Royale deep reinforcement learning algorithm for effective load balancing in cloud computing," Cluster Computing, vol. 28, no. 1, p. 19, 2025.