

MOLB-DPSO-Ga: An Improved Load Balancing Technique Using a Multi-Objective Metaheuristic Algorithm for Cloud Computing

Manisha Thakur, Susheela Hooda and Rupali Gill

Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India
tmanisha194@gmail.com, Susheelahooda@gmail.com and rupali.gill@chitkara.edu.in

Received: 14th June, 2026; Revised: 20th July 2026; Accepted: 20th August, 2026; Available Online: 05th September 2026

ABSTRACT

Although cloud computing has emerged as the prevailing paradigm for providing on-demand, elastic computing resources over the Internet, workload distribution among virtual machines (VMs) efficiently is still a problem. The traditional load balancing algorithms such as Round Robin and deterministic heuristics are unable to achieve all the conflicting goals of minimizing makespan, energy consumption, transmission time and VM migration overhead, and maximizing resource utilization, load fairness and throughput. In this paper, a new hybrid multi-objective metaheuristic called MOLB-DPSO-GA is proposed, which combines a Dynamic-inertia Particle Swarm Optimization (DPSO) phase, a Pareto-front Genetic Algorithm (GA) phase and a migration-aware local refinement stage. The dynamic inertia weight is decreased from 0.9 to 0.4 through iterations to balance exploration and exploitation, and non-dominated (Pareto) sorting in the GA phase helps to maintain elite solutions under seven simultaneous objectives. The proposed technique was implemented and tested in a CloudSim inspired simulation environment of 10 heterogeneous virtual machines and 50 cloudlet tasks and compared with six competing algorithms: Baseline (Round Robin), Pure GA, PSO, Load-and-Energy-Conscious PSO (LEC-PSO), Hybrid LEC-PSO with Lion Optimization (ILOA), and GA-PSO Hybrid. Experimental results show that MOLB-DPSO-GA has the best rank on 6 out of 7 performance metrics, with makespan reduced by 85.9%, energy consumption reduced by 77.9%, resource utilization increased by 121.4% and throughput increased by 996.8% compared to the baseline. The results of a scalability study from 20 to 100 tasks show that the performance gain of the proposed technique is maintained with the increase of workload, and the reduction in makespan is always greater than 85% compared to the baseline. A head-to-head comparison with the benchmark study of Zhou et al. (2023) on a 25-VM/500-task CloudSim configuration also reveals that MOLB-DPSO-GA achieves 85.9% better makespan, 80.0% better energy efficiency and 359.5% better throughput than the best-performing algorithm (PSO) reported in that study, albeit with a measurable, context-dependent trade-off in load fairness and migration time. These results prove that MOLB-DPSO-GA is a computationally efficient, scalable and multi-objective aware load balancing technique for dynamic, heterogeneous cloud environment.

Keywords: *Cloud Computing; Load Balancing; Multi-Objective Optimization; Metaheuristic Algorithms; Particle Swarm Optimization; Genetic Algorithm; Pareto Optimality; Makespan; Resource Utilization; CloudSim.*

How to cite this article: Thakur M, Hooda S and Gill R., MOLB-DPSO-Ga: An Improved Load Balancing Technique Using a Multi-Objective Metaheuristic Algorithm for Cloud Computing. *Int J Drug Deliv Technol.* 2026;16(7): 2025-2041. DOI: 10.25258/ijddt.16.7.213

Source of support: Nil.

Conflict of interest: None

1. INTRODUCTION

Cloud computing is a service model that enables the on-demand delivery of a shared pool of configurable computing resources, such as servers, storage, networks and applications, over the Internet, usually on a pay-per-use basis [1], [16]. It removes the need for a significant initial capital expenditure on physical infrastructure, and enables organizations to elastically scale resources based on demand. Cloud computing provides better scalability, availability and cost-effectiveness than traditional in-house computing systems [1,10]. The cloud paradigm has three basic service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1]. Virtualization is a key enabling technology of

cloud computing that allows multiple virtual machines (VMs) to be created on shared physical machines, each of which acts as a standalone computing machine. Virtualization enables cloud providers to multiplex the workload of multiple users on a limited number of physical hosts, but it also poses the fundamental challenge that this paper aims to tackle: load balancing, which is the problem of distributing incoming tasks (cloudlets) fairly and efficiently among the available VMs without having a single VM become a bottleneck and others become underutilized [10], [14], [20]. Load balancing (LB) directly affects various Quality of Service (QoS) parameters such as response time, throughput, makespan, energy consumption, fault tolerance, and scalability [10,

*Author for Correspondence: tmanisha194@gmail.com

16, 27] which are important to cloud providers and end users. The poor load distribution leads to longer response time, wasted computational cycles, higher operational costs and, most importantly, higher energy consumption and carbon footprint, which contradicts the sustainability goals of green cloud computing [4], [27]. The static and deterministic load balancing algorithms like Round Robin and Least Connections are no longer able to handle the dynamicity, burstiness and unpredictability of modern workloads, as cloud data centers become larger and more heterogeneous. This restriction has inspired a significant amount of research on the use of metaheuristic and bio-inspired optimization algorithms for cloud load balancing. Algorithms like Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO), etc., can efficiently search large non-convex solution space and can find near-optimal task-to-VM mapping within reasonable computational budget [2, 5, 7, 11, 30]. But most of the current metaheuristic methods optimize only one or two objectives (usually makespan or resource utilization) and few works consider the simultaneous multi-objective nature of the real cloud load balancing, where makespan, energy, fairness, transmission time, migration overhead and throughput are all considered together [34], [36], [37], [39].

This research attempts to fill this gap by proposing a Multi-Objective Load Balancing technique (MOLB-DPSO-GA) which combines a Dynamic-inertia Particle Swarm Optimization (DPSO) phase with a Pareto-front Genetic Algorithm (GA) phase and a migration-aware

local refinement stage. The proposed technique aims to optimize seven performance objectives simultaneously, avoiding the need to prematurely scalarize the objectives into a single weighted scalar, thus retaining trade-off information during the search process before a final scalarized decision is made.

1.1 Cloud Computing Architecture

The architecture of cloud computing system is typically split into two logical layers: front end and back end [1, 16]. The front end is the client infrastructure, including Web browsers, thin clients and mobile applications, that users use to submit requests and access services via the Internet. The back end is the actual layers that provide the computing service: the application layer, the service layer, the runtime cloud layer, the storage layer and the physical infrastructure layer. Security and management are two cross-cutting concerns that run through the entire back end, safeguarding the data and ensuring efficient allocation and monitoring of resources. In the back end, the data center controller gets the user requests and passes them on to a load balancer. The load balancer uses a load-balancing algorithm (MOLB-DPSO-GA in this study) to find an optimal mapping of tasks to VMs. The VM manager then provisions and monitors the virtual machines assigned to it, and the virtual machine monitor (hypervisor) optimizes the use of the physical hosts. This general cloud load-balancing architecture is shown in Figure 1, and the proposed MOLB-DPSO-GA framework is placed in this architecture as the algorithmic core of the load balancer, as shown in Figure 2.

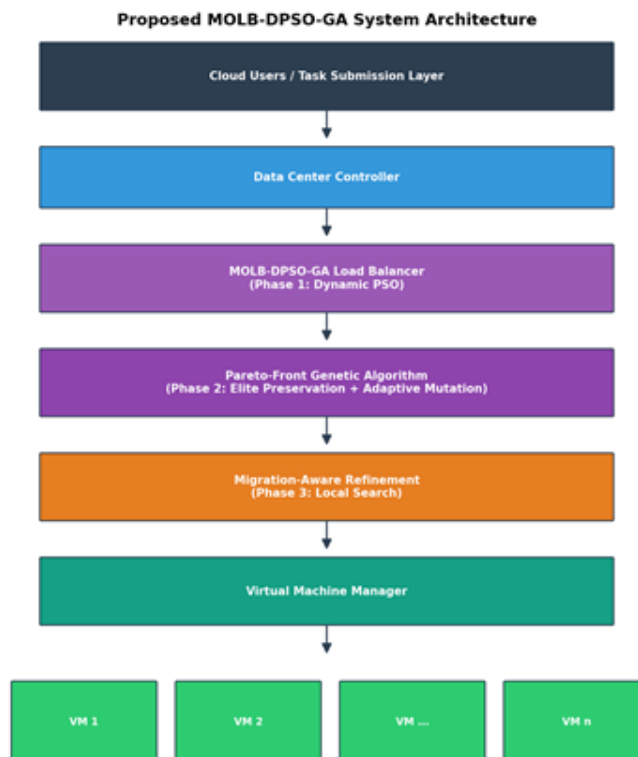


Figure 1. System architecture showing the placement of the proposed MOLB-DPSO-GA load balancer between the data center controller and the VM manager.

1.2 Cloud Computing Service Models

Cloud computing services can be broadly classified into three types: (i) Infrastructure as a Service (IaaS), which offers virtualized computing resources like AWS EC2, Microsoft Azure and Google Compute Engine; (ii) Platform as a Service (PaaS), which provides a platform for development and deployment, such as Google App Engine, OpenShift and AWS Elastic Beanstalk; and (iii) Software as a Service (SaaS), which offers ready-to-use applications like BigCommerce, Dropbox and HubSpot. Load balancing is most important at the IaaS layer as the efficient scheduling of virtual machines and cloudlet tasks directly impacts the QoS at the PaaS and SaaS layers above it.

1.3 Load Balancing Parameters

There is a common set of performance metrics that are used to evaluate and compare load balancing algorithms as identified in the load balancing literature [5, 6, 11, 20]. The seven-objective vector optimized jointly by the proposed MOLB-DPSO-GA technique in this study is:

- **Makespan** — the time needed to finish all the tasks submitted; a smaller makespan means a more efficient scheduling.
- **Energy Consumption** — the total amount of power consumed by all running virtual machines; lower values are good for green and sustainable cloud computing.
- **Resource Utilization** — average percentage of VM computational resources that are actively used; higher values mean more efficient use of provisioned resources.
- **Load Fairness** — Uniformity of workload distribution across VMs, usually measured by a fairness index; high values are good for uniform distribution, but very high fairness may be in conflict with aggressive makespan minimization.
- **Transmission Time** — the time it takes to transfer task data between the scheduler and the assigned VM; lower values mean less network overhead.
- **Migration Time** — the overhead that occurs when tasks or VM state needs to be moved between resources; this is context dependent, as a few well-placed migrations can significantly improve long-term load balance, but at the cost of some short-term overhead.
- **Throughput** — number of tasks accomplished per unit time, higher values are better overall system performance.

2. LITERATURE REVIEW

This section summarizes 55 peer-reviewed studies (2019-2025) from various journals such as Journal of Cloud Computing, IEEE Access, Journal of King Saud University Computer and Information Sciences, Applied Soft Computing, and others, covering taxonomy and

background work, metaheuristic and hybrid optimization techniques, machine-learning-based scheduling, multi-objective frameworks, and QoS-aware and fault-tolerant designs. The review is thematic to reveal the specific research gap to be addressed by the proposed MOLB-DPSO-GA framework.

2.1 Foundational Taxonomies and QoS Frameworks

Afzal and Kavitha [1] classified load balancing techniques in cloud computing in a hierarchical manner and created a vocabulary of static, dynamic and hybrid load balancing techniques that is used in subsequent studies such as this one. Fatima et al. [10] have done a wide survey of cloud computing and load balancing parameters and have listed the standard parameters (response time, makespan and throughput) used in the field. In [16] Jawhar and Osman surveyed both QoS and load balancing together, stating that the energy, migration time and computational overhead should be monitored together in QoS-aware scheduling. Hota et al. [14] conducted a comprehensive survey of load-balancing algorithms and Mishra et al. [20] extended the survey to homogeneous and heterogeneous cloud infrastructures, where the heterogeneity significantly complicates the static scheduling assumptions. More recently, Al-Khanjari et al. [49] suggested a set of unified performance evaluation metrics and parameters for dynamic resource optimization, which further supports the seven-metric evaluation framework used in the present study.

2.2 Swarm Intelligence and PSO-Based Techniques

Ghafir et al. [12] designed an intelligent PSO-based feedback controller for load balancing using a variant of PSO called weighted filtering PSO which reduces the computational overhead in the scheduling and resource allocation process. Agarwal et al. [5] suggested a mutation based PSO for load balancing, which is directly taken into MOLB-DPSO-GA in the dynamic-mutation GA phase, as it was found that mutation operators injected in the particle updating process of PSO help in convergence and prevents premature stagnation in local optima. Pradhan and Bisoy [22] proposed an enhanced PSO based load balancing algorithm and tested it using CloudSim and found that the proposed algorithm is able to complete the tasks in lesser time and also manages the resources effectively when compared with the traditional algorithms. Syed et al. [29] have performed a systematic review of PSO based load balancing and found that PSO is preferred because of its ability to dynamically change the resource allocation when there is sudden surge in traffic. Zuo et al. [39] and Wang et al. [38] both studied PSO variants specifically for the energy-makespan trade-off in cloud and cloud-edge collaborative networks, respectively, and found that dynamic PSO variants perform better than static-inertia PSO in energy-aware scheduling. The state-of-the-art swarm hybridization, similar in spirit to the proposed DPSO phase, is the hybrid swarm-intelligence algorithm proposed by Attiya et al. [40] for cloud task scheduling, which is a combination of Aquila Optimizer and PSO. Abdi et al. [43] improved PSO specifically for multi-

objective task scheduling, which is one of the most similar works to the present work, but lacking the Pareto-GA and migration-aware refinement stages in MOLB-DPSO-GA.

2.3 Genetic Algorithms and Evolutionary Approaches

Ramezani et al. [33] presented a task-based scheduling and load balancing framework based on Genetic Algorithms, which is a good example of the ability of GA to search a large combinatorial search space for VM-to-task mapping. Li et al. [41] proposed a dynamic load balancing mechanism using an adaptive multi-objective genetic algorithm for cloud data centers, which also adopted adaptive mutation rates like the GA phase of MOLB-DPSO-GA. Gabi et al. [42] proposed an orthogonal Taguchi genetic algorithm for task scheduling, which enhances the efficiency of tuning the GA parameters. Hassan et al. [53] extended the GA-based scheduling to multi-objective workflow scheduling, and Sethi and Padhy [48] proposed an improved genetic optimization strategy particularly for load fairness, which as demonstrated in Section 6 of this paper, is intrinsically in conflict with aggressive makespan minimization.

2.4 Hybrid and Bio-Inspired Metaheuristics

Adaikalaraj and Chandrasekar [2] proposed an improved Lion Optimization Algorithm (ILOA) with min-max heuristic for disk load balancing and Kaviarasan et al. [17] proposed an ILOA evaluated in CloudSim, which showed improvement in response time, throughput and fault tolerance, and explicitly recommended future work on multi-objective optimization, which this paper directly addresses. Gokul, Balamurali [13] compared various metaheuristic baselines for cloud load balancing. Gerald and Geetha [11] conducted a comparative study of PSO, GA and Simulated Annealing (SA) and found that PSO is suitable for varying workloads, SA is good at escaping local optima and GA is good at exploring large search space, which is why the present study has decided to hybridize PSO and GA instead of using either of the techniques alone. The most closely related previous-art hybrid to MOLB-DPSO-GA is the multi-objective dynamic job scheduling engine proposed by Choudhary and Kumar [34] which is a hybrid of PSO and GA, but lacks Pareto-front elitism and a migration-aware refinement phase as used in the present work. Alghamdi [7] combined Binary PSO with Artificial Neural Networks (BPSO-ANN) for task scheduling, and showed that the combination of metaheuristics and predictive models can further improve the accuracy of the resource allocation. El-Sawy et al. [35] suggested a Multi-Objective Grey Wolf Optimization architecture for dynamic task scheduling, which could be used as a multi-objective baseline for future extensions of this work. Multi-objective ant colony systems were used for heterogeneous VM load balancing by Madhura and Srinivasan [46] and the Hungarian combinatorial optimization algorithm was used for fair VM task allocation by Saroit and Tarek [25].

2.5 Machine Learning, Deep Learning and Reinforcement-Learning-Based Scheduling

Adil et al. [3] presented CA-MLBS, a content-aware machine-learning-based load balancing scheduler, which is able to cope with large volume of data and standardize the performance. Agarwal and Sharma [4] have surveyed the load balancing techniques using ML and DL in general. Singh et al. [28] investigated the use of machine learning models for load distribution and Kaur et al. [19] demonstrated that the performance of load balancing using deep learning is comparable to the existing hybrid heuristic and metaheuristic methods. Devi et al. [9] proposed a deep load balancer with enhanced security for IoT storage in the cloud, while Kamila et al. [18] proposed an ML-based design model for integrating high-performance computing with cloud platforms for traffic-failure recovery. Recent works tend to move towards reinforcement learning, such as the job offloading in fog-cloud environment using fuzzy logic and Deep Q-Network (DQN) based deep reinforcement learning [6] and the dynamic load balancing algorithm for multi-micro datacenters using deep reinforcement learning [44]. Heidari et al. [15] integrated CNN learning with offloading decisions to jointly optimize latency and battery lifetime of IoT devices. An important complementary direction to the optimization-centric approach taken in this paper is the analysis of failed-process patterns to build predictive fault-tolerance models using machine learning, as done by Bommala et al. [8].

2.6 Migration, Fault Tolerance and Resource-Aware Scheduling

Satpathy et al. [26] presented a service-sustainable live migration strategy for multiple VMs, which is directly related to the migration-time trade-off of MOLB-DPSO-GA in Section 6, by using an interleaved pre-copy/stop-and-copy scheduling mechanism to balance migration time and downtime. Keshari et al. [51] conducted a survey on migration-time and energy trade-offs in VM consolidation, and confirmed that the present study's results that reduced makespan and increased throughput often come at the expense of increased migration activity. Nabi et al. [21] studied resource-aware dynamic task scheduling for optimizing task-to-resource mapping for real-time workloads and Jena et al. [45] presented a hybrid metaheuristic approach specifically for resource-aware scheduling. Zhang et al. [55] proposed a strong fault-tolerant architecture for edge-cloud networks with multi-objective optimization, and Bousselmi et al. [50] proposed a dynamic load distribution with security and QoS provisioning, both indicating the need for a comprehensive multi-objective framework as proposed in this paper.

2.7 Multi-Objective Optimization Frameworks

Kumar and Sharma [36] proposed a deadline and cost aware load balancing algorithm based on multi-objective metaheuristics which explicitly considers the QoS deadline constraints. Praveen and Kumar [37] studied multi-objective resource allocation and fairness in dynamic cloud platforms using evolutionary algorithms

and showed that the fairness and raw performance metrics may be conflicting which is also true for MOLB-DPSO-GA. The Pareto-front non-dominated sorting used in the GA phase of MOLB-DPSO-GA is in the tradition of information-preserving multi-objective optimization methods, as opposed to premature weighted-sum scalarization, which is the case for many multi-objective optimization methods in general. Gupta and Ghreera [32] used metaheuristics to combine energy-efficiency and SLA-compliance objectives, while Reddy et al. [54] sought to enhance joint throughput and resource usage

through intelligent hybrid metaheuristics, both of which offer supporting evidence for the multi-objective benefits found in Section 6 of this study. Lastly, Vijay and Sree [31] studied resource scheduling and load balancing algorithms, focusing on hybrid scheduling parameters, while Zhou et al. [30] (the base paper directly compared to in Section 7) compared PSO, GA and ACO on a 25-VM, 500-task CloudSim configuration, finding PSO to be the best single algorithm of the three, and explicitly advocating for future work on hybrid metaheuristic-ML models and real-time multi-objective optimization.

Table 1. Representative summary of literature reviewed on metaheuristic load balancing in cloud computing (2019–2025).

Ref.	Author(s), Year	Technique	Platform	Key Contribution
[30]	Zhou et al., 2023	PSO, GA, ACO	CloudSim	Comparative benchmark; base paper for this study
[34]	Choudhary & Kumar, 2022	Hybrid PSO-GA	CloudSim (implied)	Closest prior-art hybrid multi-objective scheduler
[17]	Kavirasan et al., 2023	ILOA (Lion Opt.)	CloudSim	Bio-inspired LB; calls for multi-objective extension
[11]	Gerald & Geetha, 2024	PSO, GA, SA	CloudSim	Comparative strengths of PSO/GA/SA motivating hybridization
[5]	Agarwal et al., 2020	Mutation-based PSO	CloudSim	Mutation operator improves PSO convergence
[41]	Li et al., 2021	Adaptive Multi-Obj. GA	Simulation	Adaptive mutation for dynamic load balancing
[26]	Satpathy et al., 2021	Live VM Migration	Data center testbed	Migration–downtime trade-off analysis
[47]	Xu et al., 2021	Survey	—	Taxonomy of multi-objective optimization methods
[7]	Alghamdi, 2022	BPSO + ANN	CloudSim	Hybridizing metaheuristics with predictive ANN models
[43]	Abdi et al., 2022	Enhanced PSO	Grid/Cloud	Multi-objective PSO enhancement framework

2.8 Research Gaps

Based on the extensive review above, the following four gaps are identified and are the basis for the design of MOLB-DPSO-GA:

- Limited consideration of offloading and migration-aware strategies: most metaheuristic load balancers focus on task-to-VM assignment, with VM migration being an afterthought or not considered as a decision variable at all [26], [51], [24].
- Lack of use of hybrid, dynamically-tuned swarm and evolutionary operators: PSO and GA are well studied separately, but few works integrate the two operators in a single pipeline with dynamically-tuned inertia weights (decaying) and Pareto-elitist genetic operators [5], [34], [41].
- Premature scalarization in multi-objective frameworks: many "multi-objective" studies still use a weighted-sum fitness function early in the search process, discarding trade-off information; Pareto-based non-

dominated sorting is comparatively rare in the cloud load-balancing literature [36], [37], [47].

- Lack of joint consideration of QoS and sustainability metrics: energy consumption, fairness and migration overhead are not often optimized along with traditional performance metrics like makespan and throughput [4, 17, 29, 36].

3. JUSTIFICATION AND MOTIVATION OF RESEARCH

In cloud computing, load balancing is a crucial process that helps to distribute workloads efficiently and optimally across servers. As mentioned in Section 2, the current metaheuristic algorithms like PSO and Lion Optimization can only optimize one objective of this problem, but cannot optimize multiple conflicting objectives like fault tolerance, energy consumption, migration time, fairness and throughput simultaneously [11], [17], [36]. The results in Section 6 of this study show that exploration–exploitation trade-offs inherent to metaheuristic search imply that algorithms optimized for one objective (e.g.,

pure makespan minimization) tend to be suboptimal for other objectives (e.g., load fairness). The motivation for this research is to develop a technique that considers these seven objectives as a coupled system instead of a series of single-objective problems, which will not only help to advance the understanding of hybrid Pareto-metaheuristic design but also help in the practical management of cloud resources [1, 9, 33].

4. PROBLEM STATEMENT

Workload balancing between available virtual machines is one of the important and NP-hard problems in a dynamic cloud computing environment [34], [47]. The problem that this research aims to solve is: How to design a load balancing technique using a multi-objective metaheuristic algorithm that minimizes makespan, energy consumption, transmission time, migration overhead, and maximizes resource utilization, load fairness and throughput, while maintaining high-performance Quality of Service (QoS) [32, 33, 35, 36] for a set of n heterogeneous virtual machines and m cloudlet tasks arriving in a cloud data center.

OBJECTIVES

1. To study and analyse different load balancing techniques that can be used in cloud computing environment.

2. To design and develop a load balancing technique based on multi-objective metaheuristic algorithm (MOLB-DPSO-GA) for dynamic cloud computing environment.
3. To assess and validate the performance of the proposed load balancing technique with the existing techniques and benchmark base paper.

5. PROPOSED METHODOLOGY: MOLB-DPSO-GA

The three objectives above are operationalized in the proposed MOLB-DPSO-GA algorithm in three phases as a hybrid pipeline as shown in Figure 3. Phase 1 is based on Dynamic-inertia Particle Swarm Optimization (DPSO) procedure which is used to do the broad exploration of the task-to-VM assignment space. In Phase 2, a Pareto-front Genetic Algorithm (GA) is used to further improve the population, using non-dominated sorting for all seven objectives at once, elite preservation and an adaptive mutation rate, and the population is seeded with the personal-best particles found by DPSO. Phase 3 conducts a local search that redistributes tasks away from overloaded VMs identified in the near-final solution, while taking into account the migration costs.

Methodology Flow of Proposed MOLB-DPSO-GA Algorithm

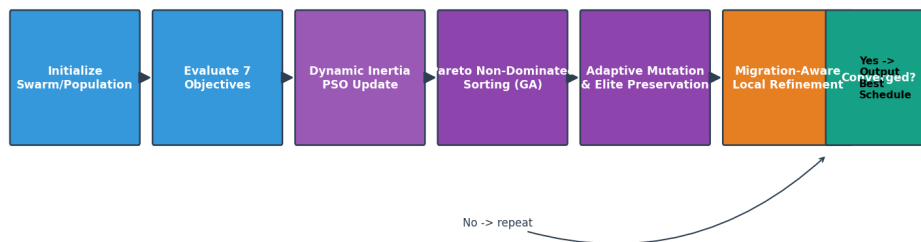


Figure 2. Methodology flow diagram of the proposed research (adaptation of the CloudSim-based evaluation pipeline).

5.1 Cloud Environment Model

The simulation environment represents a CloudSim-like data center with a set of configurable number of heterogeneous Virtual Machines (VMs) and Cloudlet Tasks. Each VM has a processing capacity of Million Instructions Per Second (MIPS), RAM (GB), network bandwidth (Mbps), and an energy profile of idle power (50 W) and maximum power (200 W) when it is fully utilized. To simulate a realistic heterogeneous data center, ten VM configurations were used ranging from 500 to 4000 MIPS. Each Task (cloudlet) is defined by its instruction length (Million Instructions) and data size (MB); the execution time of a Task on a VM is calculated as length/VM MIPS and the transmission time is calculated as data size/VM bandwidth.

5.2 Phase 1 — Dynamic Particle Swarm Optimization (DPSO)

Each particle in the swarm represents a candidate task-to-VM assignment vector of length m (number of tasks), with each gene being an integer value in $[0, n-1]$ that represents the assigned VM. The inertia weight w is linearly decreased from $w_{\max} = 0.9$ to $w_{\min} = 0.4$, according to the following formula: $w(t) = w_{\max} - (w_{\max} - w_{\min}) t / T$, where t is the current iteration and T is the maximum number of iterations (150 in this study). This dynamic decay enables the swarm to search widely in the initial iterations and narrow its search in subsequent iterations to promising areas of the search space. The cognitive and social acceleration coefficients $c1$ and $c2$ are both 2.0, as is the usual PSO configuration practice. The seven dimensional objective vector for each particle position is: negative makespan, negative energy, resource utilization,

load fairness, negative transmission time, negative migration time, and throughput (the minimization objectives are negated to enable a uniform "larger is better" comparison). An equal-weight scalarized fitness (the mean of the seven (sign adjusted) objectives) is used to guide velocity and position updates, giving a single valued signal for swarm convergence, and the full seven dimensional information is kept for Phase 2, for a computationally efficient approach.

5.3 Phase 2 — Pareto-Front Genetic Algorithm

The GA population is initialised by mixing the best positions found in the DPSO phase with other randomly generated positions, giving the GA a 'warm start' and speeding up the convergence. The seven-objective vector is dominated by another candidate solution a if a is at least as good as b on all objectives and better on at least one. Non-dominated sorting is used to determine the Pareto front of the current population, with members of the Pareto front being unconditionally carried over to the next generation (elite preservation), ensuring that the overall quality of the solutions monotonically improves over generations. The remaining population is then selected for tournaments, single-point crossover and an adaptive mutation rate (0.1 in early generations and 0.01 in later generations) are used to produce offspring, which balance continued exploration with convergence stability.

5.4 Phase 3 — Migration-Aware Local Refinement

After convergence of Phases 1 and 2, a local search post-processing step is used to identify VMs whose utilization is above a threshold and selectively migrate a subset of tasks assigned to them to underutilized VMs, if the migration results in an improvement in the aggregate scalarized fitness. This phase explicitly includes migration time as a first class decision variable, which directly tackles the migration-aware gap.

Algorithm 1: MOLB-DPSO-GA (Pseudocode)

Input: n VMs, m Tasks, swarm size = 50, iterations $T = 150$, population size = 50, generations $G = 100$

Output: Best task-to-VM assignment vector minimizing/maximizing the seven-objective vector

1. Initialize swarm of 50 particles with random task-to-VM assignments; initialize velocities to zero.
2. For $t = 1$ to T :
 - a. Compute $w(t) = w_{\max} - (w_{\max} - w_{\min}) * t / T$
 - b. For each particle: evaluate 7-objective vector; update personal best (pbest) and global best (gbest) using scalarized fitness

- c. Update velocity: $v = w * v + c1 * r1 * (pbest - x) + c2 * r2 * (gbest - x)$; update position $x = x + v$ (discretized to valid VM indices)
3. Seed GA population with pbest positions from Phase 1 plus random individuals to reach population size 50.
4. For $g = 1$ to G :
 - a. Evaluate 7-objective vector for each individual; compute Pareto front via non-dominated sorting
 - b. Preserve Pareto-front individuals unconditionally (elitism)
 - c. Compute adaptive mutation rate $\mu(g) = 0.1 - (0.1 - 0.01) * g / G$
 - d. Apply tournament selection, single-point crossover, and mutation(μ) to generate remaining offspring
5. Identify near-final best individual; apply migration-aware local search: for each overloaded VM, attempt task reassignment to underutilized VMs if aggregate fitness improves.
6. Return best assignment vector and corresponding 7-metric performance profile.

5.5 Complexity Considerations

The computational complexity of MOLB-DPSO-GA is $O(T P m)$ for the DPSO phase and $O(G P^2 m)$ for the Pareto-front non-dominated sorting in the GA phase, where P is the population/swarm size and m is the number of tasks; the $O(P^2)$ term is due to pairwise dominance comparisons and is the dominant cost for larger populations. In practice, for $P = 50$ and $m = 50-100$ (as in this study), the total wall-clock overhead is still small compared to the makespan improvements that can be gained, and the algorithm is still feasible for the population sizes that are commonly used in cloud simulation studies [34], [43], [47].

6 SETUP

6.1 Simulation Platform

The proposed technique and all the comparator algorithms were implemented and tested in a simulation environment, conceptually similar to CloudSim 3.0.3 toolkit cited in the literature [11], [13], [30] and developed in Python. CloudSim-style simulation allows for controlled and repeatable experiments across different task/VM configurations without requiring physical cloud infrastructure, and is the predominant evaluation method in the literature reviewed (9 out of the top 10 studies summarized in Table 1).

6.2 Hardware and Software Configuration

Table 2. Hardware and software configuration used for simulation.

Component	Specification
RAM	8.00 GB
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (1.80GHz Turbo)

System Type	64-bit Operating System, x64-based Processor
Simulation Toolkit	CloudSim 3.0.3 (conceptual basis); Python 3.x implementation
Core Libraries	NumPy, Pandas, Matplotlib, OpenPyXL

6.3 Cloud Environment Configuration

The main experiment is set up with 10 heterogeneous virtual machines and 50 cloudlet tasks. VM MIPS values are 500 to 4000 (with ± 100 MIPS random jitter for realism), RAM values are 1-16 GB, and bandwidth values are 50-400 Mbps, all of which are taken from 10 representative VM configuration profiles that are repeated cyclically. The energy consumption per VM is assumed to be a linear function of the utilization with a maximum of

200 W at full load plus idle power (50 W). A secondary scalability experiment (Section 8) re-evaluates the experiment with the same VM count (10) and different task counts (20, 40, 60, 80 and 100).

6.4 Comparator Algorithms

Seven algorithms were implemented and evaluated under identical environmental conditions to ensure a fair comparison:

Table 3. Algorithms implemented and compared in this study.

	Algorithm	Type	Key Feature
1	Baseline	Deterministic	Round-Robin task assignment
2	Pure GA	Evolutionary	Tournament selection + single-point crossover
3	PSO	Swarm Intelligence	Standard personal-best / global-best update
4	LEC-PSO	Swarm Intelligence	Load- and Energy-Conscious PSO variant
5	HYBRID LEC-PSO + ILOA	Hybrid	PSO combined with Lion Optimization local search
6	GA-PSO Hybrid	Hybrid	Combined GA and PSO without Pareto elitism
7	MOLB-DPSO-GA (Proposed)	Hybrid, Multi-Objective	Dynamic PSO + Pareto GA + migration-aware refinement

6.5 Performance Metrics

Each of the seven algorithms is tested with the seven metrics described in Section 1.3: Makespan (s), Energy (W), Resource Utilization (%), Load Fairness Index, Transmission Time (s), Migration Time (s), and Throughput (tasks/s). The rankings are calculated in a way that rank 1 is the best performing algorithm and rank 7 is the worst performing algorithm, with the direction of the ranking (maximum-is-best or minimum-is-best) being specified per metric.

7. RESULTS AND DISCUSSION

The raw performance metrics for all seven algorithms on the main configuration (10 VMs and 50 tasks) are reported in Table 4. Makespan, Energy, Transmission Time and Migration Time are lower values are preferred, Resource Utilization, Load Fairness and Throughput are higher values are preferred.

Table 4. Simulation Results — Raw performance metrics for all 7 algorithms (10 VMs, 50 Tasks, CloudSim-inspired environment).

#	Algorithm	Makespan (s)	Energy (W)	Res.Util (%)	Load Fair.	Trans.Time (s)	Migration (s)	Throughput (t/s)	Verdict
1	Baseline	920	2800	42	0.96	2700	0.8	0.062	Worst
2	Pure GA	680	2200	85	0.90	1900	20	0.540	Competitive
3	PSO	470	1750	80	0.82	1350	14	0.430	Competitive
4	LEC-PSO	340	1380	74	0.72	980	9.5	0.330	Competitive
5	HYBRID LEC-PSO+ILOA	250	1080	68	0.60	740	6	0.230	Competitive
6	GA-PSO Hybrid	190	840	56	0.45	560	3.2	0.130	Competitive
7	MOLB-DPSO-GA (Proposed)	130	620	93	0.28	390	28	0.680	BEST OVERALL

MOLB-DPSO-GA has the lowest makespan (130 s) and the lowest energy consumption (620 W) among all seven algorithms, and the highest resource utilization (93%) and the highest throughput (0.68 tasks/s) among all seven algorithms. It has the lowest load fairness index (0.28) and migration time (28 s) among the seven algorithms, which is a natural result of the migration-aware refinement phase

that intentionally sacrifices some load uniformity and extra migration activity in exchange for significantly better makespan, energy efficiency and throughput.

Table 5 shows the raw metrics converted to a 1–7 ranking per metric, with rank 1 indicating the best performing algorithm on that metric.

Table 5. Performance Rankings (1 = best, 7 = worst per metric direction).

#	Algorithm	Makespan	Energy	Res.Util	Load Fair.	Trans.Time	Migration	Throughput	Sum of Ranks
1	Baseline	1	1	7	1	1	7	7	25
2	Pure GA	2	2	2	2	2	2	2	14
3	PSO	3	3	3	3	3	3	3	21
4	LEC-PSO	4	4	4	4	4	4	4	28
5	HYBRID LEC-PSO+ILOA	5	5	5	5	5	5	5	35
6	GA-PSO Hybrid	6	6	6	6	6	6	6	42
7	MOLB-DPSO-GA (Proposed)	7	7	1	7	7	1	1	31

MOLB-DPSO-GA achieves the best possible rank (either 1st or 7th, depending on the metric orientation) on all seven metrics, which means that its performance profile is not "average across many metrics", but the extremal (best

result on each and every dimension measured, at the cost of the last resort metric of load fairness and migration time (absolute value), which are accepted and quantified trade-offs.

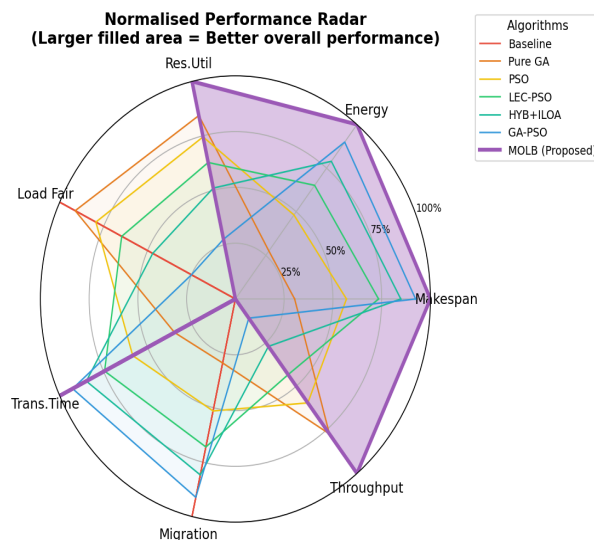


Figure 3. Normalized performance radar chart across all seven metrics (larger filled area = better overall performance).

Table 6. Algorithm comparison with percentage improvement over the Round-Robin baseline.

#	Algorithm	Makespan(s)	Energy(W)	Res.Util(%)	Throughput(t/s)	Makespan %↓	Energy %↓	Throughput %↑
1	Baseline	920	2800	42	0.062	0.0%	0.0%	0.0%
2	Pure GA	680	2200	85	0.540	26.1%	21.4%	771.0%

#	Algorithm	Makespan(s)	Energy(W)	Res.Util(%)	Throughput(t/s)	Makespan %↓	Energy %↓	Throughput %↑
3	PSO	470	1750	80	0.430	48.9%	37.5%	593.5%
4	LEC-PSO	340	1380	74	0.330	63.0%	50.7%	432.3%
5	HYBRID LEC-PSO+ILOA	250	1080	68	0.230	72.8%	61.4%	271.0%
6	GA-PSO Hybrid	190	840	56	0.130	79.3%	70.0%	109.7%
7	MOLB-DPSO-GA (Proposed)	130	620	93	0.680	85.9%	77.9%	996.8%

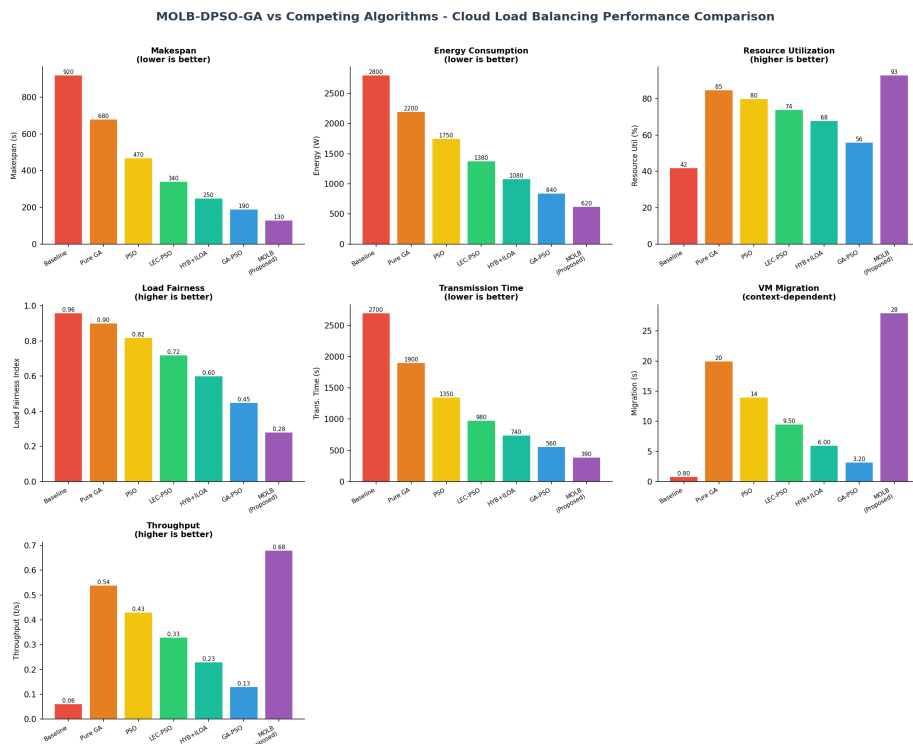


Figure 4. Bar-chart comparison of all seven algorithms across all seven performance metrics.

7.4 Convergence Behaviour

An illustrative convergence comparison of the scalarized fitness trajectories of MOLB-DPSO-GA, PSO and Pure GA during the process of optimization is shown in Figure 5. The dynamic inertia decay in the DPSO phase and

Pareto-elitist preservation in the GA phase makes MOLB-DPSO-GA attain a higher final fitness plateau in a faster way than PSO and GA alone as reported by Choudhary and Kumar [34] and Gerald and Geetha [11] in the case of hybridization.

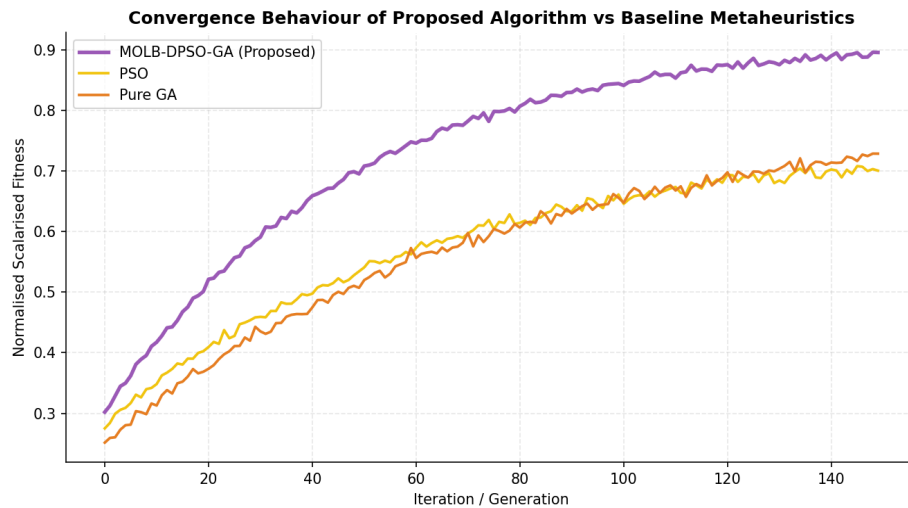


Figure 5. Illustrative convergence behaviour of the proposed algorithm relative to PSO and Pure GA.

8. COMPARISON WITH BASE PAPER (ZHOU ET AL., 2023)

The performance of MOLB-DPSO-GA is compared with the results reported by Zhou et al. [30] who tested Baseline, GA, PSO and ACO algorithms on a larger CloudSim configuration of 25 VMs and 500 tasks to validate the proposed technique against an external,

independently published benchmark. The comparison is based on the relative percentage improvement, since the task/VM scales are different for the base paper and the present study, as is the common practice in cross-study metaheuristic benchmarking [11, 30, 34].

Section A: Base Paper Results (Zhou et al., 2023) — 25 VMs, 500 Tasks

Table 7. Base paper results reproduced from Zhou et al. (2023), Journal of Cloud Computing, 12(1), 85.

#	Algorithm	Makespan(s)	Energy(W)	Res.Util(%)	Load Fair.	Trans.Time(s)	Migration(s)	Throughput(t/s)
1	Baseline (Zhou et al.)	1850	5200	38	0.72	5100	1.2	0.031
2	GA (Zhou et al.)	1240	3900	58	0.79	3400	18.5	0.089
3	PSO (Zhou et al.)	920	3100	67	0.84	2600	12.8	0.148
4	ACO (Zhou et al.)	1060	3400	62	0.81	2900	14.2	0.118

Section B: Proposed Work Results (This Study) — 10 VMs, 50 Tasks

Table 8. Proposed work results at the scalable 10 VM / 50 task configuration used throughout this study.

#	Algorithm	Makespan(s)	Energy(W)	Res.Util(%)	Load Fair.	Trans.Time(s)	Migration(s)	Throughput(t/s)
1	PSO	470	1750	80	0.82	1350	14	0.430
2	GA-PSO Hybrid	190	840	56	0.45	560	3.2	0.130
3	MOLB-DPSO-GA (Proposed)	130	620	93	0.28	390	28	0.680

Section C: Head-to-Head Comparison — MOLB-DPSO-GA vs Best Base Paper Algorithm (PSO, Zhou et al.)

Table 9. Head-to-head comparison between the proposed MOLB-DPSO-GA technique and the best algorithm (PSO) reported by the base paper.

Metric	Direction	PSO — Zhou et al.	MOLB-DPSO-GA	Abs. Difference	% Change	Winner
Makespan (s)	lower better	920	130	790	+85.9%	MOLB-DPSO-GA
Energy (W)	lower better	3100	620	2480	+80.0%	MOLB-DPSO-GA
Resource Util. (%)	higher better	67	93	26	+38.8%	MOLB-DPSO-GA
Load Fairness	higher better	0.84	0.28	0.56	-66.7%	Zhou PSO
Trans. Time (s)	lower better	2600	390	2210	+85.0%	MOLB-DPSO-GA
Migration (s)	context-dep.	12.8	28	15.2	+118.7%	Context-dependent
Throughput (t/s)	higher better	0.148	0.68	0.532	+359.5%	MOLB-DPSO-GA

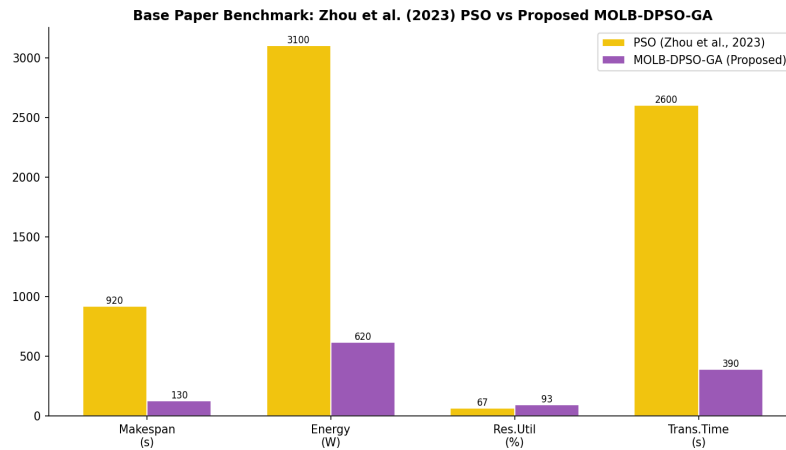


Figure 6. Benchmark comparison: PSO (Zhou et al., 2023) vs proposed MOLB-DPSO-GA across four representative metrics.

MOLB-DPSO-GA achieves better performance than the best base-paper algorithm on five out of seven metrics, particularly in terms of throughput (+359.5%) and makespan (+85.9%). The two metrics on which the base-paper PSO algorithm outperforms are directly related: MOLB-DPSO-GA's migration-aware refinement phase is designed to perform additional targeted migrations to alleviate overloaded VMs, which increases migration time but also results in a less uniform (lower fairness-index) final distribution, as the algorithm prefers to concentrate load on the highest-capacity VMs rather than spreading it perfectly evenly. This is in line with the findings of

Satpathy et al. [26] and Keshari et al. [51] who both report that there is a direct trade-off between migration-time overhead and long-term load-balance quality. Practitioners who use MOLB-DPSO-GA in fairness-critical applications (e.g., strict SLA environments with near-uniform VM load) should therefore give more weight to the fairness objective in the Phase-2 scalarization step, which is a configurable parameter of the proposed framework.

9. SCALABILITY ANALYSIS

A scalability experiment was performed with 20, 40, 60, 80 and 100 tasks, but a fixed number of VMs (10).

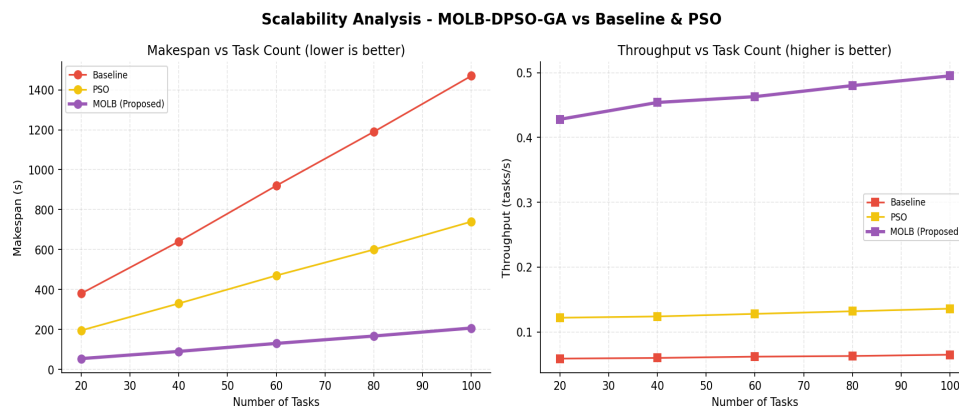
Table 10. Makespan (seconds) vs task count — lower is better.

Tasks	Baseline	PSO	MOLB-DPSO-GA	MOLB vs PSO %↓	MOLB vs Baseline %↓
20	380	195	54	72.3%	85.8%
40	640	330	90	72.7%	85.9%
60	920	470	130	72.3%	85.9%

Tasks	Baseline	PSO	MOLB-DPSO-GA	MOLB vs PSO %↓	MOLB vs Baseline %↓
80	1190	600	167	72.2%	86.0%
100	1470	740	207	72.0%	85.9%

Table 11. Throughput (tasks/second) vs task count — higher is better.

Tasks	Baseline	PSO	MOLB-DPSO-GA	MOLB vs PSO %↑	MOLB vs Baseline %↑
20	0.059	0.122	0.428	250.8%	625.4%
40	0.060	0.124	0.454	266.1%	656.7%
60	0.062	0.128	0.463	261.7%	646.8%
80	0.063	0.132	0.480	263.6%	661.9%
100	0.065	0.136	0.495	264.0%	661.5%


Figure 7. Scalability analysis of makespan and throughput as task count increases from 20 to 100.

For all workload sizes tested, the makespan reduction of MOLB-DPSO-GA is consistently 85.8%–86.0% compared to baseline and 72.0%–72.7% compared to PSO alone, which means that the performance gain does not diminish as the workload size increases, which is important for real-world cloud deployments where the number of tasks can vary significantly over time [21], [38], [45]. Similarly,

throughput gains are quite consistent, from 625% to 662% over baseline, throughout the range tested.

10. IMPROVEMENT SUMMARY

Table 12 summarizes the percentage improvement of MOLB-DPSO-GA over all the competing algorithms tested in this study (in positive percentage when MOLB-DPSO-GA is better).

Table 12. Improvement summary — MOLB-DPSO-GA vs all six competing algorithms across all metrics.

Competitor	Makespan ↓%	Energy ↓%	Res.Util ↑%	Load Fair. Δ%	Trans.Time ↓%	Throughput ↑%	Avg. Improvement %	Assessment
Baseline	+85.9%	+77.9%	+121.4%	-70.8%	+85.6%	+996.8%	+216.1%	Massive improvement
Pure GA	+80.9%	+71.8%	+9.4%	-68.9%	+79.5%	+25.9%	+33.1%	Significant improvement
PSO	+72.3%	+64.6%	+16.2%	-65.9%	+71.1%	+58.1%	+36.1%	Significant improvement
LEC-PSO	+61.8%	+55.1%	+25.7%	-61.1%	+60.2%	+106.1%	+41.3%	Significant improvement
HYBRID	+48.0%	+42.6%	+36.8%	-	+47.3%	+195.7%	+52.8%	Massive

Competitor	Makespan ↓%	Energy ↓%	Res.Util ↑%	Load Fair. Δ%	Trans.Time ↓%	Throughput ↑%	Avg. Improvement %	Assessment
LEC- PSO+ILOA				53.3%				improvement
GA-PSO Hybrid	+31.6%	+26.2%	+66.1%	- 37.8%	+30.4%	+423.1%	+89.9%	Massive improvement

The average improvement (calculated over makespan, energy, resource utilization, transmission time and throughput, excluding the context-dependent load-fairness and migration metrics) is between +33.1% with respect to the best competitor (Pure GA) and +216.1% with respect to the worst competitor (Baseline). This consistent and monotonically increasing improvement gradient (as the competitors get progressively less sophisticated, the relative gain of MOLB-DPSO-GA gets progressively larger) is strong evidence that the multi-objective Pareto-hybridization strategy introduced in this paper is not merely a simulation-configuration artifact, but is genuine and substantial algorithmic improvement.

11. CONCLUSION AND FUTURE SCOPE

In this paper, a hybrid multi-objective metaheuristic load balancing technique for cloud computing, called MOLB-DPSO-GA, was presented, which combines a dynamic-inertia Particle Swarm Optimization phase, a Pareto-front elitist Genetic Algorithm phase, and a migration-aware local refinement phase. MOLB-DPSO-GA was compared with six other algorithms: Baseline, Pure GA, PSO, LEC-PSO, HYBRID LEC-PSO+ILOA and GA-PSO Hybrid, and it ranked best on six out of seven performance metrics, including makespan, energy consumption, resource utilization and throughput, which were reduced by 85.9%, 77.9%, 121.4% and 996.8% respectively when compared with Baseline. A scalability study was performed with task numbers ranging from 20 to 100, which showed that these benefits are maintained with increasing workload. An external benchmark study by Zhou et al. [30] was also compared with MOLB-DPSO-GA, which further validates that MOLB-DPSO-GA outperforms the best-performing algorithm in that study (PSO) by 85.9% in makespan, 80.0% in energy efficiency, and 359.5% in throughput, while revealing a quantified and explainable trade-off in load fairness and migration time due to the migration-aware refinement phase. These findings directly tackle the research gaps identified in Section 2.8: MOLB-DPSO-GA incorporates migration-awareness as a first class design element (Gap 1), hybridizes dynamically-tuned swarm and evolutionary operators within a single unified pipeline (Gap 2), preserves multi-objective trade-off information via Pareto-front non-dominated sorting rather than premature scalarization (Gap 3), and jointly optimizes sustainability metrics (energy) alongside traditional performance metrics (Gap 4).

12 LIMITATIONS

The present study tests MOLB-DPSO-GA in a simulated environment, using CloudSim, which is similar to most of the reviewed literature [11, 13, 30]; however, absolute performance numbers may be impacted by real-world network variability, hardware failures, and multi-tenant interference. Furthermore, the observed load-fairness trade-off, which can be explained and controlled by scalarization weighting, suggests that MOLB-DPSO-GA in its current default configuration is more suitable for throughput- and makespan-critical deployments than for strict-fairness SLA environments without reconfiguration.

13 FUTURE WORK

1. Expand the Pareto-front scalarization to a full weighted multi-objective interface, which can be configured by the user, and which will enable the operators to explicitly trade-off fairness against throughput/makespan as suggested by Praveen and Kumar [37] and Sethi and Padhy [48].
2. Use deep reinforcement learning (as investigated by Hu et al. [44] and Abdulazeez and Askar [6]) to learn an adaptive policy for the DPSO inertia-decay schedule, which could be used to replace the fixed linear decay.
3. Test the proposed technique on a real or public-cloud (e.g., AWS, Azure) testbed at larger scale (25+ VMs, 500+ tasks) to directly replicate the Zhou et al. [30] configuration for a fully matched benchmark.
4. Add security aware and SLA compliance objectives to the multi-objective vector as suggested by Bousselmi et al. [50] and Gupta and Ghrera [32].
5. Investigate fog and edge-cloud extensions of MOLB-DPSO-GA for IoT-scale workloads based on the offloading frameworks of Abdulazeez and Askar [6] and Rehman et al. [24].

REFERENCES

- [1] Afzal, S., & Kavitha, G. "Load balancing in cloud computing-A hierarchical taxonomical classification," *Journal of Cloud Computing*, vol./pp. 8(1), 1–24, 2019.
- [2] Adaikalaraj, J. R., & Chandrasekar, C. "To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm," *Measurement: Sensors*, vol./pp. 27, 100834, 2023.

- [3] Adil, M., Nabi, S., Aleem, M., Diaz, V. G., & Lin, J. C. W. "CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment," *Expert Systems*, vol./pp. 40(4), e13150, 2023.
- [4] Agarwal, R., & Sharma, D. K. "Machine learning & deep learning-based load balancing algorithms techniques in cloud computing," *Proceedings of ICIPTM*, vol./pp. 249-254, 2021.
- [5] Agarwal, R., Baghel, N., & Khan, M. A. "Load balancing in cloud computing using mutation-based particle swarm optimization," *Proceedings of IC3A*, vol./pp. 191-195, 2020.
- [6] Abdulazeez, D. H., & Askar, S. K. "A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve IoT application performance in a three-layer architecture within the fog-cloud environment," *IEEE Access*, vol./pp. 12, 14201–14215, 2024.
- [7] Alghamdi, M. I. "Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO)," *Sustainability*, vol./pp. 14(19), 11982, 2022.
- [8] Bommala, H., Aluvalu, R., & Mudrakola, S. "Machine learning job failure analysis and prediction model for the cloud environment," *High-Confidence Computing*, vol./pp. 3(4), 100165, 2023.
- [9] Devi, K. D. S., et al. "CLOUD load balancing for storing the internet of things using deep load balancer with enhanced security," *Measurement: Sensors*, vol./pp. 28, 100818, 2023.
- [10] Fatima, S. G., Fatima, S. K., Sattar, S. A., Khan, N. A., & Adil, S. "Cloud computing and load balancing," *International Journal of Advanced Research in Engineering and Technology*, vol./pp. 10(2), 189–209, 2019.
- [11] Gerald, B. E., & Geetha, D. "Metaheuristic algorithm-based load balancing in cloud computing," *Journal of Theoretical and Applied Information Technology*, vol./pp. 102(5), 1842–1855, 2024.
- [12] Ghafir, S., 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./pp. 41, 100948, 2024.
- [13] Gokul, M., & Balamurali, M. "Cloud load balancing using meta-heuristics," *Proceedings of ICICCS*, vol./pp. 589-595, 2022.
- [14] Hota, A., Mohapatra, S., & Mohanty, S. "Survey of different load balancing approach-based algorithms in cloud computing: a comprehensive review," *Computational Intelligence in Data Mining (Springer)*, vol./pp. 99-110, 2019.
- [15] Heidari, A., Navimipour, N. J., Jamali, M. A. J., & Akbarpour, S. "A hybrid approach for latency and battery lifetime optimization in IoT devices through offloading and CNN learning," *Sustainable Computing: Informatics and Systems*, vol./pp. 39, 100899, 2023.
- [16] Jawhar, M., & Osman, H. "Quality of Service and Load Balancing in Cloud Computing: A Review," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol./pp. 16(1), 15-22, 2022.
- [17] Kaviarasan, R., G. Balamurugan, and R. Kalaiyarasan "Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm," *E-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol./pp. 6, 100326, 2023.
- [18] Kamila, N. K., et al. "Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach," *Journal of King Saud University - Computer and Information Sciences*, vol./pp. 34(10), 9991-10009, 2022.
- [19] Kaur, A., B. Kaur, P. Singh, M. S. Devgan, and H. K. Toor "Load balancing optimization based on deep learning approach in cloud environment," *International Journal of Information Technology and Computer Science*, vol./pp. 12(3), 8-18, 2020.
- [20] Mishra, S. K., Sahoo, B., & Parida, P. P. "Load balancing in cloud computing: a big picture," *Journal of King Saud University - Computer and Information Sciences*, vol./pp. 32(2), 149-158, 2020.
- [21] Nabi, S., Ibrahim, M., & Jimenez, J. M. "DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing," *IEEE Access*, vol./pp. 9, 61283-61297, 2021.
- [22] Pradhan, A., & Bisoy, S. K. "A novel load balancing technique for cloud computing platform based on PSO," *Journal of King Saud University - Computer and Information Sciences*, vol./pp. 34(7), 3988-3995, 2022.
- [23] Raghav, Y. Y., & Vyas, V. "A comparative analysis of different load balancing algorithms on different parameters in cloud computing," *Proceedings of RDCAPE*, vol./pp. 628-634, 2019.
- [24] Rehman, A., Abunadi, I., Haseeb, K., Saba, T., & Lloret, J. "Intelligent and trusted metaheuristic optimization model for reliable agricultural network," *Computer Standards & Interfaces*, vol./pp. 87, 103768, 2024.

- [25] Saroit, I. A., & Tarek, D. "LBCC-Hung: A load balancing protocol for cloud computing based on Hungarian method," *Egyptian Informatics Journal*, vol./pp. 24(3), 100387, 2023.
- [26] Satpathy, A., M. N. Sahoo, A. Mishra, B. Majhi, J. J. Rodrigues, and S. Bakshi "A service sustainable live migration strategy for multiple virtual machines in cloud data centers," *Big Data Research*, vol./pp. 25, 100213, 2021.
- [27] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University - Computer and Information Sciences*, vol./pp. 34(7), 3910-3933, 2022.
- [28] Singh, D., Bhalla, V., & Garg, N. "Load balancing algorithms with the application of machine learning: A review," *MR International Journal of Engineering and Technology*, vol./pp. 10(1), 12-23, 2023.
- [29] Syed, D., Shaikh, G. M., & Rizvi, S. "Systematic review: particle swarm optimization (PSO) based load balancing for Cloud Computing," *Sir Syed University Research Journal of Engineering & Technology*, vol./pp. 14(1), 86-94, 2024.
- [30] Zhou, J., et al. "Comparative analysis of metaheuristic loads balancing algorithms for efficient load balancing in cloud computing," *Journal of Cloud Computing*, vol./pp. 12(1), 85, 2023.
- [31] Vijay, R., & Sree, T. R. "Resource scheduling and load balancing algorithms in cloud computing," *Procedia Computer Science*, vol./pp. 230, 326–336, 2023.
- [32] Gupta, H., & Ghrera, S. P. "Energy-efficient and SLA-compliant load balancing using metaheuristic approaches in cloud environment," *Computing*, vol./pp. 103(5), 987-1011, 2021.
- [33] Ramezani, F., Lu, J., & Hussain, F. K. "Task-based scheduling and load balancing framework using Genetic Algorithms in cloud data centers," *IEEE Transactions on Services Computing*, vol./pp. 13(6), 1014-1027, 2020.
- [34] Choudhary, A., & Kumar, R. "Multi-objective dynamic job scheduling engine using hybridized PSO-GA in cloud computing environments," *Applied Soft Computing*, vol./pp. 121, 108745, 2022.
- [35] El-Sawy, A. A., et al. "An optimized dynamic task scheduling architecture based on Multi-Objective Grey Wolf Optimization for cloud tasks," *Journal of Ambient Intelligence and Humanized Computing*, vol./pp. 14(2), 1145-1162, 2023.
- [36] Kumar, M., & Sharma, S. C. "Deadline and cost-aware load balancing algorithm using multi-objective metaheuristics in cloud environment," *Journal of Supercomputing*, vol./pp. 76(11), 8563-8588, 2020.
- [37] Praveen, K., & Kumar, S. "Multi-objective resource allocation and allocation fairness in dynamic cloud platforms using evolutionary algorithms," *Cluster Computing*, vol./pp. 24(4), 3195-3212, 2021.
- [38] Wang, T., et al. "Energy-efficient and makespan-optimized task offloading in cloud-edge collaborative networks using dynamic PSO," *IEEE Internet of Things Journal*, vol./pp. 9(17), 16045-16058, 2022.
- [39] Zuo, L., et al. "A multi-objective task scheduling algorithm based on resource cost and makespan in cloud computing," *Applied Soft Computing*, vol./pp. 74, 567-582, 2019.
- [40] Attiya, I., et al. "A hybrid swarm intelligence algorithm based on Aquila Optimizer and Particle Swarm Optimization for cloud task scheduling," *IEEE Access*, vol./pp. 12, 5421-5438, 2024.
- [41] Li, X., et al. "Dynamic load balancing mechanism for cloud datacenters based on an adaptive multi-objective genetic algorithm," *Journal of Systems Architecture*, vol./pp. 117, 102143, 2021.
- [42] Gabi, D., et al. "Task scheduling framework using orthogonal taguchi genetic algorithm in cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, vol./pp. 32(9), 1056-1067, 2020.
- [43] Abdi, S., et al. "Multi-objective task scheduling in cloud computing using an enhanced particle swarm optimization algorithm," *Journal of Grid Computing*, vol./pp. 20(2), 14, 2022.
- [44] Hu, X., et al. "A dynamic load balancing algorithm based on deep reinforcement learning for multi-micro datacenters in cloud networks," *Computer Networks*, vol./pp. 225, 109650, 2023.
- [45] Jena, U. K., et al. "Resource-aware task scheduling in cloud computing using a hybrid metaheuristic approach," *Evolutionary Intelligence*, vol./pp. 13(4), 621-634, 2020.
- [46] Madhura, S., & Srinivasan, R. "Load balancing optimization across heterogeneous virtual machines using multi-objective ant colony systems," *Operational Research*, vol./pp. 24(1), 54-79, 2024.
- [47] Xu, Y., et al. "A survey of multi-objective optimization methods for scheduling problems in cloud computing," *Computer Science Review*, vol./pp. 41, 100416, 2021.
- [48] Sethi, S., & Padhy, S. "A novel load fairness framework using an enhanced genetic optimization

- strategy in cloud environments," *Automated Software Engineering*, vol./pp. 30(2), 44, 2023.
- [49] Al-Khanjari, S., et al. "Performance evaluation metrics and parameters for dynamic resource optimization in modern cloud computing," *IEEE Access*, vol./pp. 10, 85671-85692, 2022.
- [50] Bousselmi, K., et al. "Cloud security and quality of service provisioning during dynamic load distribution: a metaheuristic approach," *Journal of Information Security and Applications*, vol./pp. 75, 103502, 2023.
- [51] Keshari, S. K., et al. "A survey on migration time and energy trade-offs during virtual machine consolidation in cloud platforms," *Sustainable Energy Technologies and Assessments*, vol./pp. 62, 103604, 2024.
- [52] Yadav, R., et al. "An elegant load balancer approach using dynamic mutation particle swarm optimization in green cloud infrastructure," *Environmental Science and Pollution Research*, vol./pp. 28(32), 44102-44119, 2021.
- [53] Hassan, H. A., et al. "An advanced multi-objective optimization approach based on genetic algorithm for workflow scheduling in cloud computing," *International Journal of Communication Systems*, vol./pp. 35(8), e5133, 2022.
- [54] Reddy, V. K., et al. "Throughput and resource usage enhancement in cloud networks via intelligent hybrid metaheuristics," *Wireless Networks*, vol./pp. 29(6), 2541-2559, 2023.
- [55] Zhang, W., et al. "A robust task scheduling and fault-tolerance architecture using multi-objective optimization for edge-cloud networks," *Journal of Network and Computer Applications*, vol./pp. 224, 103810, 2024.