International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(4s) 2026 pp. 708-721
https://doi.org/10.70917/ijcisim-2026-2560
Article

Dynamic and Fair Load Balancing in Heterogeneous Distributed Systems using Hybrid Optimization

Vidya S Handur[1], Santosh L. Deshpande[2]

[1]Department of Computer Science and Engineering, KLE Technological University, India, E-mail: vidya_handur@kletech.ac.in
[2]Department of Computer Science and Engineering, Visvesvaraya Technological University, India, E-mail: sld@vtu.ac.in

Abstract: With the advancement in technology, the need for performance optimisation and efficient load balancing remains challenging in heterogeneous distributed systems due to dynamic workload variations and differences in resource capabilities. To achieve scalable task allocation in heterogeneous environments, a hybrid swarm intelligence-based load-balancing algorithm is proposed that uses PSO and ACO. The proposed work leverages PSO's rapid global exploration to obtain high-quality tasks. The work presented in this paper leverages PSO's rapid global exploration to generate high-quality task-to-node mappings and ACO-based local refinement to improve workload equity and eliminate any remaining imbalances. A multi-objective optimization model for dynamic workload adjustment is constructed, in which task completion time, system cost, and load variance are minimized while Jain's fairness index is maximized. The overhead of task migration and response to changes in workload is also minimized. Static and dynamic workload experiments demonstrate the improved performance of the proposed hybrid PSO-ACO workload distribution compared to traditional load balancing methods. The evaluation results show that the proposed load balancing achieves greater fairness while maintaining similar convergence across varying node counts, swarm sizes, and workloads, with reductions in task completion time and system cost. To verify the feasibility of our proposed method, dynamic workload experiments are carried out.

Keywords: Dynamic Load Balancing, Heterogeneous Distributed Systems, Swarm Intelligence, Fairness Optimization

1. INTRODUCTION

In recent years, distributed computing environments, cloud computing, grid systems, and edge-fog infrastructures have advanced rapidly, increasing the need for efficient and fair load-balancing mechanisms. Since they consist of nodes with different computational, memory, and communication capabilities, modern distributed systems are fundamentally heterogeneous. Additionally, the task arrival frequencies and computing demands vary across application workloads, making load balancing challenging and critical. The goal of load balancing is to enhance system performance by distributing computing tasks across resources, maximising resource utilisation, and preventing performance degradation caused by overloaded or underloaded nodes.

Existing load balancing schemes, such as least-connection and round-robin, are lightweight but lack heterogeneity awareness in large-scale and dynamic environments, leading to unfairness and inefficiencies in resource utilization. Adaptive and intelligent load balancing methods have been extensively studied to address these limitations. Among different approaches, swarm intelligence algorithms have attracted much attention due to their high-quality optimisation capabilities, decentralized architecture, and self-adaptive features. Models based on swarm intelligence that incorporate natural behaviors such as ant foraging and bird swarming are well-suited for distributed decision-making problems. Particle Swarm Optimization (PSO) has been used in load balancing because of its global search ability and fast convergence. However, in highly heterogeneous situations, PSO may prematurely converge and fail to effectively improve local solutions. Similarly, the Ant Colony Optimization (ACO) method has demonstrated good performance in discrete optimization problems with pheromone- based learning, but its convergence rate is often slower than PSO, especially in large- scale systems. The limitations of hybrid swarm intelligence systems can be

overcome by integrating the complementary strengths of different algorithms, as recent research suggests. In this context, combining the global exploration capabilities of PSO with the local exploitation abilities of ACO presents a promising direction for enhancing both load balancing effectiveness and fairness. Nevertheless, existing hybrid methods often neglect fairness and workload dynamics, focusing instead on static workloads or optimising for specific performance characteristics. Additionally, excessive overhead from task migration may occur due to frequent re-optimisation in dynamic scenarios, ultimately degrading overall system efficiency. Motivated by these observations, this study proposes an equitable workload balancing approach for heterogeneous distributed systems using a hybrid PSO–ACO algorithm. The proposed method treats load balancing as a multi-objective optimisation problem, initially maximising fairness via Jain's fairness index while also considering cost minimization, task completion time, and workload variance simultaneously. ACO is employed to improve solutions and achieve balanced workload distribution among nodes, while PSO facilitates rapid exploration of the solution space to identify promising task-node assignments. A novel dynamic workload adaptation mechanism is introduced, accounting for time-varying task arrivals and performance fluctuations across nodes, thereby enhancing practicality. The framework, which combines warm- start PSO and local ACO refinement, continuously monitors system imbalance in hyperspace and gradually rebalances instead of re- optimising entirely. This approach maintains system fairness while significantly reducing computational overhead and job migration.

. Comprehensive experimental comparisons under both static and dynamic workload scenarios demonstrate the effectiveness of the proposed hybrid PSO–ACO approach. Results show that completion time, fairness, system cost and convergence behaviour can universally be enhanced over traditional load balancing algorithms, without a binary operation for all the benchmarks. Sensitivity analysis further demonstrates the robustness of the proposed approach to varying swarm sizes, job volume, and number of nodes.

The major contributions of this paper can be summarized as follows:

(i) A novel multi-objective hybrid PSO–ACO load balancing model designed specifically for distributed heterogeneous systems.

(ii) A runtime re-scheduling strategy that improves rebalancing between different workloads.

(iii)Comprehensive experimentation that demonstrates scalability and robustness, which also involves dynamic workload scenarios and sensitivity analysis.

The paper is organized as follows: Section 2 presents the related work. Section 3 outlines the system model and problem formulation. Section 4 details the algorithm. Section 5 presents an adaptive dynamic load balancing mechanism. Section 6 discusses the results. Section 7 concludes the paper.

2. RELATED WORK

Recent work acknowledges that in the modern load balancer, multiple objectives such as response time, energy usage, operational cost and fairness must be optimized.

Particle Swarm Optimization (PSO) remains a widely used method for task scheduling and load balancing because of its simplicity and strong global search capability. Recent works extend PSO by combining it with prediction techniques (e.g., LSTM) or with other metaheuristics to handle heterogeneity and time-varying demand. Simaiya et al. in [1] integrate deep learning with PSO and GA for dynamic workload provisioning, showing improved host utilization and reduced provisioning delay, while Ghafir [2] proposed intelligent PSO controllers for VM scheduling and migration to optimize load balancing in cloud environments. These PSO-centric contributions demonstrate that augmenting PSO with workload prediction or complementary optimizers significantly improves dynamic provisioning and makespan under realistic workloads. ACO continues to be attractive for discrete task-to-node assignment problems due to its pheromone-guided local search properties. Recent studies apply ACO for QoS-aware service composition and VM placement, with multi-pheromone and mutation operations enhancements to avoid local optima and improve convergence speed in multi-cloud settings [3][4]. Hybrid schemes that place ACO as a local refinement stage after a global optimizer have shown particular promise in reducing SLA violations while keeping solution stability. Hybrid metaheuristics that combine PSO with ACO or with other global or local methods are increasingly common. Studies report that such hybrids balance exploration and exploitation, produce better makespan and cost trade-offs, and reduce load variance compared to single-method baselines. Examples include DPSO-GA, PSO-GWO hybrids, WWO-ACO hybrids, and direct PSO-ACO integrations developed for cloud and edge deployments. [5]-[8]. These hybridizations are particularly effective when the optimization objective is multi-criteria.

Recent work on modern load balancing recognizes that several, sometimes conflicting, goals should be pursued: response time, energy and monetary cost, and fairness. Some recent works have introduced multi-objective WWO-ACO and HA-PSO hybrids that include energy or cost terms in fitness functions and report tangible savings in energy consumption and SLA satisfaction [9][10]. It is important to note that integrating learning with supervised or reinforcement-based swarm or metaheuristic methods remains an open area. Recent works demonstrate that DRL-boosted hybrids, such as DRL & ACO-WWO, converge to new workloads faster than pure metaheuristics and can support online parameter tuning for schedulers [11][12]. This trend motivates the use of warm-start and adaptive parameter techniques in dynamic re-optimization. Some recent works combine prediction, localized refinement and warm-start optimisation to enable quick responses to load surges [14][15]. Empirical studies have been conducted at a large scale to compare hybrid metaheuristics across different dimensions: makespan, energy, fairness, and convergence speed. All of these studies conclude that a hybrid scheme is better performing than classical heuristics on realistic trace-driven workloads [16][17]. In general, the literature shows a linear evolution from basic heuristic and single metaheuristic methods to hybrid techniques and adaptive strategies for dynamic, heterogeneous environments. Hybrid swarm intelligence techniques, which combine the advantages of global and local search with adaptives, have demonstrated strong potential in load balancing problems towards fair distribution, low adjustment latency and minimal degradation [19]-[30]. This is the motivation for this combined PSO–ACO methodology, along with the dynamic workload adjustment mechanism introduced in this work.

3. SYSTEM MODEL AND LOAD REPRESENTATION

The distributed system studied in this work is a network of computing nodes with different processing capabilities. Arriving tasks to the system have diverse computational complexities and running times. Multiple tasks can be handled by each node, but if a single node becomes overloaded, it significantly impacts system performance. The goals of the load balancing problem are to assign the load based on the capacity of the nodes and to optimise the total task execution. The objective function is designed by encapsulating variations in load and task completion time, enabling an algorithm to consider both performance and fairness.

The distributed system in the study consists of generic computing nodes and independent tasks. Every node is characterized by a processing power, and every task requests to be processed. Once the nodes are assigned tasks, the total load on a node can be represented as in (1)

$$L_i = \sum_{j=1}^{M} x_{ij} T_j \tag{1}$$

where x_{ij} is a binary decision variable such that:

$$x_{ij} = \begin{cases} 1, if\ the\ task\ j\ assigned\ to\ node\ i\ , \\ 0,\ \ Otherwise \end{cases}$$

The average load across all nodes is then defined as given by Eq. (2)

$$\overline{L} = \frac{1}{N} \sum_{i=1}^{N} L_i \tag{2}$$

A well-balanced system aims to keep each node's load as close as possible to this average, still considering the node's processing capacity.

3.1 Fairness Measurement

Fairness is one of the key considerations in heterogeneous systems, and non-uniform task allocation results in persistent bottlenecks. Jain's fairness index is used to calculate the degree of fairness, and is defined as follows in (3)

$$FI = \frac{\left(\sum_{i=1}^{N} L_i\right)^2}{N \sum_{i=1}^{N} {L_i}^2} \quad (3)$$

where *FI* is a positive number between 0 and 1. The closer the value to 1, the more balanced the load among nodes, and the lower it is means less balance.

3.2 Objective Function Formulation

In a heterogeneous distributed system, it is necessary to consider several objectives for load balancing. First, the load among nodes should be as equal as possible to maintain fairness. On the other hand, it is desirable to minimize the total task completion time for performance enhancement. To achieve both goals, the following combined cost function is defined by (4)

$$F = \alpha \sum_{i=1}^{N} \left(L_i - \overline{L}\right)^2 + \beta \sum_{j=1}^{M} T_j^{comp} \quad (4)$$

where, $\left(L_i - \overline{L}\right)^2$ is penalizes load imbalance,T_j^{comp} denotes the completion time of the task j,α and β are weighting factors that balance fairness and performance. Minimizing this function leads to task allocations that are both fair and efficient.

4. HYBRID PSO-ACO LOAD BALANCING APPROACH

The proposed approach is combined with two phases: global optimization using PSO and local refinement using ACO.

4.1 Global Search Using PSO

Each particle in the PSO process represents a possible assignment of tasks to a node. The particle's location is the set of nodes responsible for each task, and its fitness is based on load balance and execution time. Particles move by learning their best position so far and by tracking the best solution explored by the swarm. This provides the algorithm with a fast search phase that spreads across the search space and leads to promising areas of load distribution. PSO excels at finding good global solutions but can miss local minima that are highly consequential in the fairness case, which is why ACO is incorporated.

4.2 Local Refinement Using ACO

In the second stage, when PSO finds a good solution, ACO refines it. In this stage, artificial ants dynamically reallocate tasks mainly from heavily loaded nodes to lightly loaded ones. Pheromone trails are updated based on the quality of the refined solutions, supporting task-to-node mappings that enhance fairness and reduce response time. This local improvement technique enables the algorithm to correct errors that PSO cannot fix on its own. With each iteration of the algorithm, pheromone amounts drive ants towards a progressively fairer and more efficient distribution of effort.

4.3 Hybrid Optimization Flow

The hybrid algorithm alternates between PSO-based global updates and ACO-based local refinements. PSO ensures rapid convergence to high-quality solutions, while ACO improves solution robustness and fairness. The algorithm terminates when further improvements become negligible or when a predefined iteration limit is reached.

In the PSO phase, each particle represents a possible task-to-node assignment. The position of particle k at iteration t is denoted by $X_k(t)$ and its velocity by $V_k(t)$. Particle velocities are updated according to (5):

$$\boldsymbol{V_k}(t+1) = w\,\boldsymbol{V_k}(t) + c_1 r_1(\mathbf{pbest}_k - \mathbf{X}_k(t)) + c_2 r_2(\mathbf{gbest} - \mathbf{X}_k(t)) \quad (5)$$

where w is the inertia weight controlling exploration, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are random values in [0,1],*pbest* is the best solution found by particle *k*, and *gbest* is the best solution found by the entire swarm.

The particle position is then updated as given in (6):

$$\boldsymbol{X}_k(t+1) = \boldsymbol{X}_k(t) + \boldsymbol{V_k}(t+1) \tag{6}$$

Through these updates, PSO rapidly explores the search space and identifies globally promising load distributions.

4.3.1 Ant colony optimization for local refinement

Although PSO can reach close to the optimal solution, it may miss local improvements in fairness. To this end, ACO is employed as a post-processing step. In ACO, ants probabilistically build task assignments based on pheromone intensity and heuristic desirability, which is a function of node capacity and current load. The probability of assigning the task to the node is given in (7)

$$P_{ij} = \frac{\tau_{ij}^{\alpha}\ \eta_{ij}^{\beta}}{\sum_{k=1}^{N} \tau_{ij}^{\alpha}\ \eta_{ij}^{\beta}} \tag{7}$$

After ants complete their solutions, pheromone levels are updated using (8)

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij} \tag{8}$$

where ρ is the pheromone evaporation rate, and $\Delta\tau_{ij}$ is proportional to improvements in fairness and reduction in load variance.

4.4 Hybrid PSO-ACO Optimization Strategy

Some synergy is achieved by the combination of PSO and ACO. Initially, PSO uses the objective function F shown in Eq. (4) to search for a globally efficient set of task allocations, followed by ACO with its task-balancing feature, which finds an optimal set by relocating tasks from overloaded nodes to underloaded ones. This can be mathematically expressed as shown in (9)

$$X^* = ACO(PSO(\boldsymbol{X})) \tag{9}$$

where X^* denotes the final refined allocation. This sequential optimization ensures both fast convergence and improved fairness.

5. EXPERIMENTAL SETUP

To measure the performance of the hybrid PSO–ACO approach, a synthetic simulation environment was developed that is designed as closely as possible to real atomic heterogeneous distributed systems. The experimental environment is designed to reflect node heterogeneity, workload diversity, and dynamic task assignment.

Static Workload

The distributed system comprises four groups of 20 computing nodes, including both heterogeneous and homogeneous types, as shown in Table 1, based on power. Node capacities were modelled using relative computational units, as real-world CPU speeds and resources differ.

Table 1. Node Characteristics

Node Group	Number of Nodes	Processing Capacity (Units)
Low-end nodes	6	1–3
Medium-end nodes	8	4–6
High-end nodes	6	7–10

This configuration ensures realistic heterogeneity, with faster nodes expected to handle larger workloads while slower nodes handle lighter tasks

Workload Model

A total of 100 independent tasks were generated for each simulation run. Task sizes were randomly assigned to represent diverse computational demands, as shown in Table 2. Tasks are assumed to be non-preemptive and independent, allowing clear evaluation of load balancing efficiency without task dependency interference

Table 2. Task characteristics and their computational demand

Task Category	Number of Tasks	Computational Demand
Small	35	10–30 units
Medium	40	31–60 units
Large	25	61–100 units

5.1Algorithm Parameters

To ensure fairness in comparison, all swarm intelligence algorithms were executed under identical stopping conditions and iteration limits. The algorithm's parameters were defined as shown in Table 3 for PSO and ACO.

Table 3. Algorithms Parameter Values

PSO Parameters	ACO Parameters
Parameter	**Value**
Swarm size	30 particles
Inertia weight (w)	0.9 to 0.4
Cognitive coefficient (c1)	2.0
Social coefficient (c2)	2.0
Maximum iterations	200

In the hybrid PSO–ACO approach, PSO was executed first to obtain a globally optimized solution, followed by ACO for local refinement at each iteration cycle. The proposed hybrid algorithm was compared against the following baseline methods:(i) Round-Robin Scheduling (RR) – static and non-adaptive, (ii) Standalone ACO local search-based load balancing, (iii)Classical PSO – global search-based optimization, (iv)Proposed Hybrid PSO–ACO. All algorithms were implemented under identical simulation conditions to ensure unbiased

Metric	RR	ACO	PSO	Hybrid PSO–ACO
Completion Time (ms)	105.3	92.4	87.1	70.2
Fairness Index	0.78	0.85	0.88	0.97
System Cost	120	105	98	80
Load Variance	38.6	26.9	21.4	9.8

Table 4. Comparative performances among different algorithms

comparison. The considered Performance Metrics measured are performance indicators: (i)Average Task Completion Time (ms), (ii)Fairness Index (Jain's Index) and (iii)System Cost, defined as the cumulative normalized resource usage across nodes.

5.2 Comparative Results and Analysis

The experimental results are shown in Table 4, where the hybrid PSO–ACO approach achieved the lowest average completion time. Compared to classical PSO, the improvement is approximately 19.4%**,** while the reduction compared to round-robin exceeds 33%**.** The fairness index of 0.97 demonstrates that the proposed hybrid approach distributes load almost uniformly across nodes. This gain is in large part assisted by ACO-based local optimizations that transfer load from overloaded nodes. A low-cost system indicates optimal resource usage. Compared with the other algorithms, combining two migration strategies and two local computations, the overhead incurred in load movement and underactivity on high-load nodes can be minimized. The results show that the load imbalance is much less for the hybrid approach, indicating better load balancing of heterogeneous systems.

5.2.1 Sensitivity Analysis

Even though the hybrid PSO–ACO algorithm achieves good results under baseline configuration, it is also necessary to know how efficient it is considering different system settings. Thus, a sensitivity analysis is performed to estimate the effects of three parameters: swarm size, number of tasks, and number of heterogeneous nodes. These parameters affect scale, convergence dynamics, and equity in distributed systems. All sensitivity tests were carried out by changing a single parameter at a time and keeping other parameters unchanged. Averages were taken over multiple simulation runs to ensure convergence.

5.2.2 Effect of Swarm size

Swarm size determines the number of candidate solutions explored in each iteration. A small swarm may converge quickly but risks premature convergence, whereas a very large swarm increases computational overhead without proportional gains. The configuration was tested over the swarm sizes of 10, 20, 30, 40, the number of tasks was 100, and the total number of nodes was 20. The outcomes are shown in Table 5. It is observed that as the swarm size increases, both fairness and completion time improve noticeably up to a swarm size of 30. This improvement is due to enhanced exploration capability, allowing PSO to identify better global solutions while ACO refines them locally. Beyond this point, performance gains become marginal, while computational cost increases. Therefore, a swarm size of 30 offers an optimal balance between solution quality and computational efficiency.

Table 5. Effect of Swarm Size

Swarm Size	Completion Time (ms)	Fairness Index	System Cost
10	78.6	0.91	92
20	73.9	0.95	85
30	**70.2**	**0.97**	**80**
40	69.8	0.97	79

5.2.3 Effect of number of Tasks

The number of tasks reflects workload intensity in the distributed system. Increasing task count tests the scalability of the load balancing algorithm under heavier system load. The performance of the configuration was evaluated for a swarm size of 30 against task counts in the range of 50, 100, 150, and 200, with total node count set to 20. The results are presented in Table 6. The results show that the completion time of the task rises with the increasing workload, the fairness is maintained to be high, even when more tasks are considered, indicating that the hybrid PSO–ACO algorithm shows improved performance.

Table 6. Effect of Number of Tasks

Number of Tasks	Completion Time (ms)	Fairness Index	Load Variance
50	42.8	0.98	6.3
100	**70.2**	**0.97**	**9.8**

150	96.4	0.95	14.7
200	128.9	0.93	21.6

5.2.4 Effect of the number of Heterogeneous nodes

The number of nodes influences the load balancing complexity. Due to the increasing number of nodes, more heterogeneity and decision space can be obtained. Configuration is tested on a swarm size of 30 and task sizes of 100 with a total number of nodes 10,20,30,40. Results obtained are presented in Table 7. With the increase in the number of nodes, the completion time will decrease, because more computing resources are offered. The fairness index is also high for all the configurations, suggesting that the algorithm still distributes tasks well even under a more heterogeneous system.

Table 7. Effect of Number of Nodes

Number of Nodes	Completion Time (ms)	Fairness Index	System Cost
10	82.5	0.94	91
20	70.2	0.97	80
30	68.9	0.96	78
40	67.4	0.95	77

From the sensitivity analysis, the following are the observations: (i)Swarm size has a large impact on the convergence quality until an upper limit is reached, beyond which there is a small improvement. (ii)Task load affects completion time while fairness is not deteriorated that much, indicating scalability. (iii)Using node count does improve overall system performance, which is capable of a balanced load distribution. These outcomes indicate that the hybrid PSO–ACO algorithm is not fine-tuned to a specific performance but operates well for various system settings. The sensitivity analysis has shown that the proposed hybrid methodology is applicable for real-life heterogeneous distributed systems where workload and resources can change over time. By choosing an intermediate size of swarms, and scaling to system size, the algorithm is useful in cloud, grid and edge environments with no strong parameter retuning.

5.3 Interpretation of Sensitivity Graphs

In Figure 1(a), the change in average task completion time is depicted with respect to the swarm size for the hybrid PSO–ACO algorithm. When the population size is small, solutions are not thoroughly explored, resulting in suboptimal task–node mappings and longer completion times. It is observed that between swarm sizes of 10 and 30, the time taken suddenly decreases. This improvement is due to the larger swarm's better global exploration in PSO, which leads to improved load balancing. Above a swarm size of 30, the time saved drops off, indicating saturation, which shows that an overly large swarm has less of a positive impact on performance compared to the computational overhead incurred.

The fairness index for different swarm sizes is illustrated in Figure 1(b). Even with the smallest swarm size, fairness is still low because of early convergence and lack of variety in solutions. Fairness continues to improve with increased swarm size; it does so increasingly as more and more agents are added to the swarm, most of which occurs at swarm sizes of 30 and 40. This suggests that if there is enough particle diversity, the workload can be successfully balanced between different nodes, demonstrating the stability of the hybrid PSO–ACO algorithm after reaching a sufficient exploration ability.

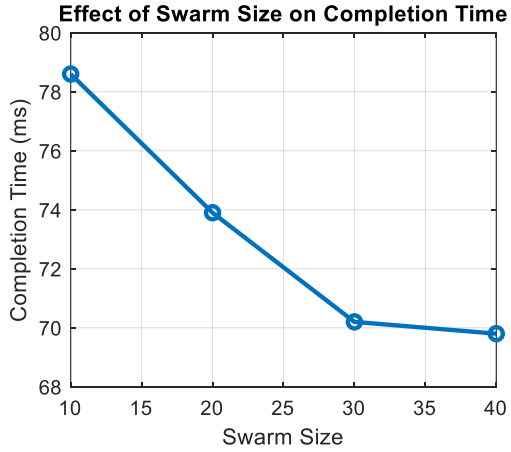
The effect of volume on completion time is shown in Figure 1(c). As the task count increases, throughput increases approximately linearly. This behaviour can be seen as an effect of the increased computational demand on the system. Even with this large workload, the hybrid algorithm is still able to maintain reasonable growth in its completion time, which indicates that it remains scalable. The lack of sudden degradation also validates that the load is well balanced by the algorithm even under high traffic rates.

Figure 1(d) shows how fairness changes with increasing task load. Although fairness gradually decreases with task load, it does not drop below 0.93 even at the highest workload. This indicates that the hybrid PSO–ACO can continue to avoid serious load imbalance. In Figure 1(e), the mean completion time for the one-probability problem

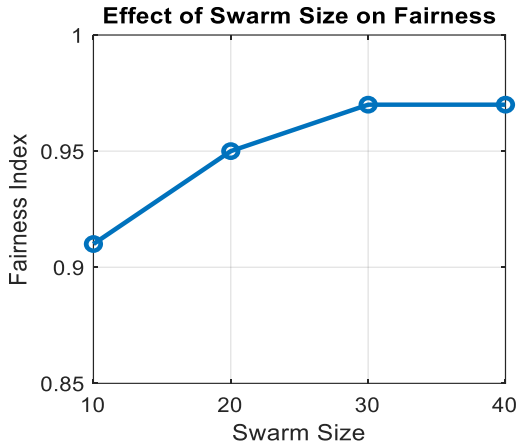
on heterogeneous nodes is investigated. Increasing the number of nodes predictably reduces completion time by providing more processing power. The gain is more significant from 10 to 20 nodes, but it then slows. This suggests that while parallelism improves with more nodes due to the gain in resources, it is cost-effective when the resources are sufficiently available to handle the workload efficiently.

The correlation between the number of nodes in a system and its cost is shown in Figure 1(f). With the number of nodes, the system cost reduces gradually. This decrease is due to a more uniform load balancing that reduces the overload penalties and idle times. The gradual downward fall reflects the good adaptability of hybrid PSO–ACO to large-scale systems without unnecessary overhead, and thus its applicability to large heterogeneous distributed systems.

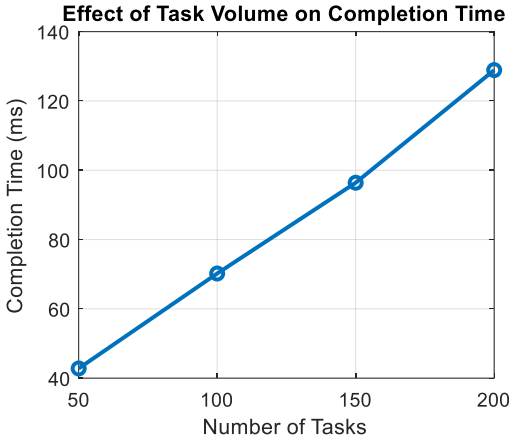
Overall, sensitivity analysis graphs have shown that the proposed hybrid PSO–ACO load balancing algorithm can be considered as one of the effective approaches and is not dependent on a particular operating point. Performance gains reach saturation at some parameter settings, which yield optimal configurations empirically. More importantly, fairness is consistently high, implying that the algorithm can provide equitable load distribution and scale well with system size and load intensity.



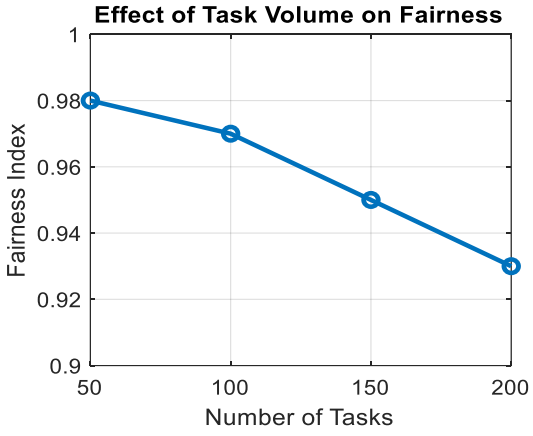
(a)



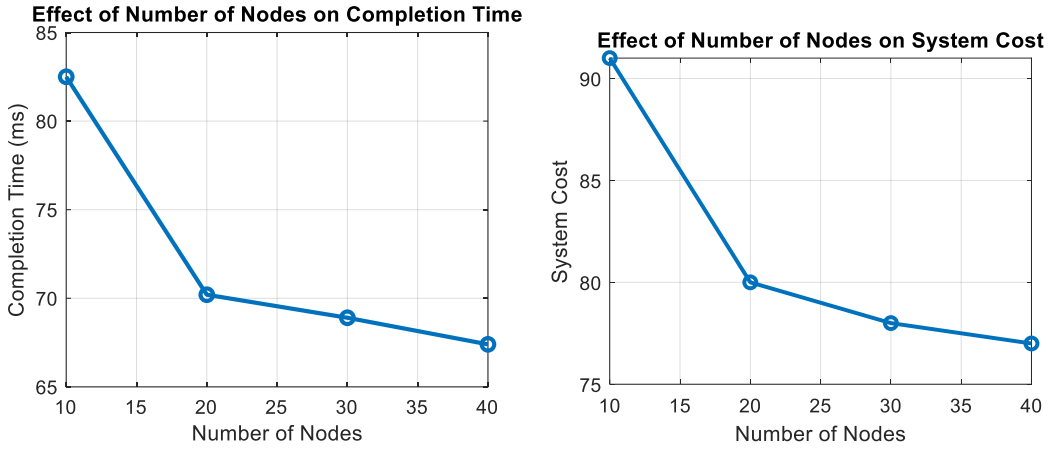
(b)



(c)



(d)



(e) (f)

Figure 1 (a) - (f) Performance variation characteristics in the sensitivity measure by hybrid PSO-ACO

5.4 Dynamic Workload Adaptation Extension

In real heterogeneous distributed systems, workloads are rarely static. Task arrival rates fluctuate, node performance may vary due to background processes, and nodes may join or leave the system dynamically. A static load-balancing strategy may therefore become suboptimal over time. To address this limitation, a dynamic workload adaptation mechanism is integrated into the proposed hybrid PSO–ACO framework. The objective of this extension is to continuously monitor system state and trigger partial re-optimisation only when necessary, thereby maintaining fairness and efficiency while minimizing reconfiguration overhead.

Dynamic System Model

Let:$T(t)$ denote the set of active tasks at time t,$R(t)$ denote the set of available nodes at time t,$L_j(t)$ represent the load on node j at time t, $\lambda(t)$ represent the task arrival rate, the node capacity 'C 'is updated dynamically as given by (10):

$$C_j(t) = C_j * \delta_j^t \tag{10}$$

where $\delta j(t) \in [0,1]$ captures temporary performance degradation due to contention or failures.

Workload Change Detection

Dynamic adaptation is triggered only when a significant imbalance is detected. The system continuously monitors the load deviation factor $\Delta(t)$ as given by (11)

$$\Delta(t) = \frac{max_j L_j(t) - min_j L_j(t)}{\frac{1}{M}\sum_{j=1}^{M} L_j(t)} \tag{11}$$

A re-optimization event is triggered if: $\Delta(t) > \theta$; where θ is a predefined imbalance threshold.

Incremental Fitness Function

In the study, an incremental fitness function is introduced instead of full re-optimization, as given by (12)

$$f_{dyn} = \alpha 1 \cdot \Delta T_{comp} + \alpha 2 \cdot \Delta Cost + \alpha 3 \cdot Var_{load} \tag{12}$$

where:ΔT_{comp} is the change in completion time;$\Delta Cost$ is the change in the system cost,Var_{load} represents current load variance. This formulation ensures minimal task migration while restoring fairness.

5.5 Dynamic Hybrid PSO–ACO Strategy

Instead of restarting the optimization from the beginning, the dynamic framework performs:(i) Warm-start PSO using the previous global best solution, (ii) reduced swarm size to limit computation, (iii) Localized ACO refinement applied only to overloaded nodes.

This approach significantly reduces response time while preserving solution quality. The complete pseudo-code is given as extended pseudo-code.

Extended Pseudocode: Dynamic Hybrid PSO–ACO Load Balancing

Input

- Current task set $T(t)$; Current node set $R(t)$; Previous best solution $X*(t-1)$
- Imbalance threshold θ ; Maximum adaptation iterations I_{dyn}

Step 1 Continuous Monitoring
- 1.1 Monitor node loads $L_j(t)$ and task arrivals.
- 1.2 Compute imbalance factor $\Delta(t)$

Step 2 Adaptation Trigger
- 2.1 If $\Delta(t) \leq \theta$, maintain current allocation
- 2.2 otherwise, initiate dynamic re-optimization

Step 3 Warm-Start PSO
- 3.1 Initialize particles around X*$(t-1)$
- 3.2 Run PSO for I_{dyn} iterations using f_{dyn}
- 3.3 Reduce swarm size and inertia weight

Step 4 Local ACO Refinement
- 4.1 Identify overloaded nodes
- 4.2 Apply ACO only to tasks mapped to these nodes
- 4.3 Update pheromone trails locally.

Step 5 Migration Control
- 5.1 Limit task migration count using $M_{max} = \gamma \cdot | T(t) |$
- 5.2 Apply refined task assignment

Step 6 Update System State
- 6.1 Update X*(t)
- 6.2 Precompute performance metrics.

Return to the monitoring phase, Step 1

There are several key advantages of Dynamic Extension: (i)reduces full re-computation (ii) handles workload spikes gracefully, (iii) preserves fairness during system fluctuations, (iv) reduces migration overhead, (v)Suitable for cloud, fog, and edge environments.

6. EXPERIMENTAL RESULTS WITH DYNAMIC WORKLOAD

Further, to test the performance of the proposed dynamic hybrid PSO–ACO-based load-balancing approach, another set of experiments is conducted on time-varying workloads. These experiments emulate well-known distributed working conditions, characterized by task arrival rates that undergo sudden changes and fluctuate throughout time, as well as node performance degradation. Results of static hybrid PSO–ACO are compared with those of dynamic hybrid PSO–ACO to emphasize the value of re-optimisation.

Experimental Setup for Dynamic Workload

Parameters	**Values**
Number of heterogeneous nodes	20
Initial number of tasks	100
Workload pattern	bursty arrivals every 50-time units
Task arrival rates	low → medium → high
Imbalance threshold θ	0.25
Maximum migration limit	10% of active tasks

The performance was evaluated in terms of (i) Average task completion time, (ii)Jain's fairness index, (iii)system cost, and (iv)task migration overhead. The resulting performances for each approach are demonstrated in Table 8. The dynamic hybrid PSO–ACO improves the static methods in terms of all performance measures. After a load surge, the completion time of the dynamic hybrid method is approximately 10% lower than that of the static hybrid method due to the timely balancing of the load. The fairness index also goes up to 0.97, which means that it can achieve close to fair load distribution even if at a fluctuating factor. More importantly, the migration overhead is significantly reduced, which indicates that incremental re-optimization reduces task movement.

Table 8. Performance Comparison under Dynamic Workload

Algorithm	**Completion Time (ms)**	**Fairness Index**	**System Cost**	**Migration Overhead (%)**
Static PSO	102.6	0.86	108	18.4
Static ACO	95.3	0.89	101	15.7
Static Hybrid PSO–ACO	82.4	0.93	90	13.2
Dynamic Hybrid PSO–ACO	**74.1**	**0.97**	**81**	**6.8**

The influence of workload intensity on dynamic performances is illustrated in Table 9. Workloads with higher intensities cause completion time to be higher (on account of increased system utilization), but fairness always exceeds 0.95, thus attesting to the soundness of our dynamic adaptation mechanism. The managed load variance increase suggests that the system manages to prevent extreme load imbalances also during high-load periods.

Table 9. Impact of Workload Intensity on Dynamic Performance

Workload Level	**Avg. Tasks**	**Completion Time (ms)**	**Fairness Index**	**Load Variance**
Low Load	60	46.5	0.98	5.2
Medium Load	100	74.1	0.97	9.8
High Load	160	109.3	0.95	18.6

The results of the response time comparison are illustrated in Table 10. Dynamic hybrid PSO–ACO is 3× roughly more adaptable than the static algorithm re-execution. This enhancement is due to warm-start initialization of the PSO and locality-based refinements by ACO, which greatly reduces computational burden and job migration.

Table 10. Adaptation Response Time Comparison

Algorithm	**Re-optimization Time (ms)**	**Avg. Migrations**
Static Hybrid PSO–ACO (Full Re-run)	42.8	19
Dynamic Hybrid PSO–ACO	**14.6**	**7**

On the other hand, dynamic behavior manifests better performance with any deviation from static. In the presence of a workload spike, the static algorithm allows prolonged imbalance before re-optimization. This implies that completion time increases and degradation in fairness is experienced. In contrast, dynamic hybrid PSO–ACO quickly detects the imbalance and makes localized adjustments to restore the stability of systems in a few iterations. Such behaviour shows that the developed framework is adequate for real-time and large-scale systems. Several important observations have been made from Dynamic Experiments: (i) Rapid Adaptation: The discovery from considering the Dynamic PSO–ACO in workload-dependent experimentation lies in the fact that it responds promptly with minimum cost. (ii) Sustained Fairness: Fairness of workloads never drops from unity regardless of how bursty the inputs have been. (iii) Reduced Migration Cost: The small tasks' migration limits unnecessary movement. (iv)

Scalability: under high load, it only exhibits a balanced degradation in performance. From the aforementioned results, the dynamic hybrid PSO–ACO algorithm is suitable for cloud computing, edge computing, and heterogeneous systems based on the unpredictable nature of workloads. The mathematical findings indicate that even in the least favourable conditions, fairness and efficiency are maintained.

7. CONCLUSION

A novel PSO-ACO-based Fair Load Balancing for HDSs has been proposed and introduced in this work. The method is specifically developed for dealing with the major concerns in distributed systems, such as load imbalance, disparity of resources and dynamic variation in workloads. The hybrid PSO–ACO algorithm takes advantage of PSO's effective global search for a good set of task–node mappings, and then conducts ACO-based local optimization to enhance fairness that still remains after PSO-supported mapping. A fitness model with a multi-objective function of completion time, system cost, and load-balance variance can prevent performance from being at the expense of fairness. In addition, a dynamic workload adaptation scheme was proposed to mitigate time-varying task arrivals and fluctuating operations of nodes via warm-start re-optimization and localized rebalancing. The extensive experimental evaluation with static and dynamic workloads shows that the proposed approach is more effective than conventional load balancing techniques, standalone PSO and ACO solutions, as well as other static hybrids. The hybrid PSO–ACO cuts down task completion time, raises Jain's fairness index, decreases system cost and considerably reduces load variance. Sensitivity analysis shows that the proposed algorithm is robust with respect to different swarm sizes, task volumes, and node numbers, while experiments under dynamic workload conditions show that the proposed method is able to quickly react dynamically with little overhead in terms of task migrations. In general, the results support that integration of global and local swarm intelligent behaviours is a successful and scalable approach for equitable load balancing in heterogeneous distributed environments.

Although the developed hybrid PSO–ACO algorithm performs well, there are a few promising future research directions. First, the algorithm can be generalised by considering energy-aware and carbon-aware objectives to achieve sustainable scheduling on green data centres and edge environments. Second, one can design some adaptive parameter tuning methods utilizing reinforcement learning or self-adaptive control to avoid manual parameter selection. Another noteworthy generalization concerns dealing with node failures and network disturbances, in which prediction models could be able to anticipate task redistribution before performance drops. The model can also be extended to multi-cloud (geo-distributed) settings, wherein the communication latency and data locality may significantly matter. Furthermore, combining priority-aware and deadline-constrained task scheduling would increase the applicability to real-time systems. Finally, the future work will be to implement the above approach for real distributed platforms and to experiment with large-scale benchmarks alongside trace-driven workloads in order to validate its phenomenology. Such extensions would help in consolidating the hybrid PSO–ACO framework as a viable and trustworthy strategy for emerging distributed computing systems.

References

1. S. Simaiya, S. D. B., and A. Kumar, "A hybrid cloud load balancing and host utilization prediction method using deep learning and PSO–GA (DPSO–GA)," Scientific Reports, vol. 14, no. 1, pp. 1–18, 2024.
2. S. Ghafir, "Intelligent PSO-based feedback controller for virtual machine scheduling and migration in cloud computing," Journal of Cloud Computing, vol. 13, no. 1, pp. 1–16, 2024.
3. N. Sharma and R. Patel, "Improved ant colony optimization-based service composition for multi-cloud environments," Journal of Cloud Computing, vol. 13, no. 1, pp. 1–20, 2024.
4. U. K. Lilhore, S. Tanwar, R. Sharma, and N. Kumar, "Hybrid DRL-enhanced ACO–WWO for efficient resource allocation and load balancing in cloud computing," International Journal of Computational Intelligence Systems, vol. 18, no. 1, pp. 1–15, 2025.
5. D. Syed, M. Hussain, and S. Ahmad, "IMH-LB: A scalable improved multi-objective hybrid load balancing algorithm for cloud environments," Journal of Supercomputing, vol. 81, no. 2, pp. 1–25, 2025.
6. K. Rajammal, R. Anand, and S. Kannan, "Dynamic load balancing in cloud computing using hybrid PSO–GWO," Scientific Reports, vol. 15, no. 1, pp. 1–19, 2025.
7. S. Mohapatra, S. Mohanty, H. K. Nayak, and M. K. Mallick, "DPSO: A hybrid Dragonfly–PSO approach for load balancing in cloud computing environments," EAI Endorsed Transactions on Internet of Things, vol. 10, no. 1, pp. 1–12, 2024.
8. M. Singal, "A hybrid PSO–Whale optimization approach for efficient cloud load balancing," IEEE Transactions on Sustainable Computing, vol. 9, no. 1, pp. 45–57, 2024.
9. A. M. Baydoun and A. S. Zekri, "Towards efficient VM placement: A two-stage ACO–PSO approach for green cloud infrastructure," International Journal of Computer Networks and Communications, vol. 17, no. 4, pp. 1–18, 2025.

10. J. Yao, X. Li, and Y. Zhang, “Hybrid cloud load balancing and host utilization prediction using deep learning and swarm intelligence,” Scientific Reports, vol. 14, no. 1, pp. 1–17, 2024.
11. U. K. Lilhore, R. Sharma, and N. Kumar, “Multi-objective hybrid WWO–ACO optimization for load balancing in cloud data centres,” Scientific Reports, vol. 15, no. 1, pp. 1–21, 2025.
12. M. Al-Hawari and A. Al-Dulaimi, “Hybrid PSO–ACO framework for optimizing edge–cloud deployment of large-scale AI workloads,” Future Generation Computer Systems, vol. 153, pp. 145–158, 2025.
13. P. K. Singh and R. Kumar, “Energy-aware hybrid PSO–GWO scheduling for heterogeneous cloud environments,” Cluster Computing, vol. 28, no. 1, pp. 1–15, 2025.
14. R. Verma and A. K. Sharma, “Dynamic workload provisioning using hybrid DPSO–GA in cloud computing,” Journal of Parallel and Distributed Computing, vol. 185, pp. 75–88, 2024.
15. S. Patel, M. Shah, and D. Mehta, “PSO–JSO hybrid metaheuristic for adaptive load balancing in distributed systems,” Informatica, vol. 49, no. 1, pp. 89–103, 2025.
16. T. Nguyen and H. Tran, “Hybrid metaheuristic optimization for cloud load balancing: Performance and fairness evaluation,” IEEE Access, vol. 12, pp. 42110–42125, 2024.
17. Y. Li, J. Wang, and Z. Chen, “Energy-efficient multi-objective hybrid optimization for cloud task scheduling,” Engineering Applications of Artificial Intelligence, vol. 130, pp. 106–121, 2025.
18. A. Roy and S. Das, “Hybrid ACO–GA based load balancing strategy for hybrid cloud systems”, International Journal of Computational Intelligence and Applications, vol. 23, no. 2, pp. 1–20, 2024.
19. Elsakaan, N., Amroun, K. “A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment”. J Supercomput 80, 13434–13474, 2024. https://doi.org/10.1007/s11227-024-05990-5
20. L. Datta, "A Novel Hybrid Scheduling Algorithm for Load Balancing in Cloud Computing," 2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2025, pp. 709-715, doi: 10.1109/ICESC65114.2025.11212592.
21. U.K. Jena, P.K. Das, M.R. Kabat,” Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment”, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 6, Part A,2022, Pages 2332-2342
22. Ajay Kumar Dubey and Vimal Mishra, “ Design and analysis of novel hybrid load balancing algorithm for cloud data centre”, International Journal of Grid and Utility Computing Vol. 16, No. 4,pp 349-362,2025
23. A. S. Srikar, V. P. Naik and S. Bhaskaran, "SDN-based Hybrid Load Balancing Algorithm," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205630
24. A. Senthilselvi, V. S. Varshini, E. Reena Sharan, T. Shruthi Shree, B. J. Chelliah and S. Senthil Pandi, "Load-balancing in Cloud Computing Environment using Hybrid Particle Swarm Optimization and Ant Colony Optimization Algorithm," 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT), Faridabad, India, 2024, pp. 755-760, doi: 10.1109/ICAICCIT64383.2024.10912094.
25. Narwal, A. Resource Utilization Based on Hybrid WOA-LOA Optimization with Credit-Based Resource-Aware Load Balancing and Scheduling Algorithm for Cloud Computing. J Grid Computing 22, 61,2024. https://doi.org/10.1007/s10723-024-09776-0
26. Pathania, N., Singh, B., Batra, I. et al. FireBat: a hybrid approach for load balancing in cloud computing. J Cloud Comp 15, 5,2026. https://doi.org/10.1186/s13677-025-00805-1
27. Zhou, J., Lilhore, U.K., M, P. et al. “Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing”, J Cloud Comp 12, 85 ,2023, https://doi.org/10.1186/s13677-023-00453-3
28. Serda H, Bogdan F., “Exploring PSO, ACO, and a Hybrid PSACO Approach: A Comparative Study for System Parameter Tuning”, 9th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, Tokyo, Japan, April 26 - 27, 2025
29. D. Pal, P. Verma, D. Gautam and P. Indait, "Improved optimization technique using hybrid ACO-PSO", 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 277-282, doi: 10.1109/NGCT.2016.7877428
30. Yao, J., Luo, X., Li, F. et al. “Research on hybrid strategy Particle Swarm Optimization algorithm and its applications”. Sci Rep 14, 24928,2024, https://doi.org/10.1038/s41598-024-76010-y
31. Dapu Li, Kenli Li, Jie Liang, Aijia Ouyang,” A hybrid particle swarm optimization algorithm for load balancing of MDS on heterogeneous computing systems”, Neurocomputing, Volume 330,2019, Pages 380-393, https://doi.org/10.1016/j.neucom.2018.11.034.
32. Sanaj, M.S., Horng, M.F., Shankar, S.S. et al. “Energy-Efficient Task Scheduling in Cloud Computing with EcoTaskOpt: A Hybrid Ant Colony and Particle Swarm Optimization Approach”, Int J Comput Intell Syst 19, 64, 2026. https://doi.org/10.1007/s44196-025-01095-w