International Journal of Applied Mathematics

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A HYBRID LOAD BALANCING APPROACH IN HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM USING ARTIFICIAL BEE COLONY OPTIMIZATION AND BHILL CLIMBING

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

[1] Department of Computer Science & Engineering, KLE Technological University, Vidyanagar, Hubballi, 580031, India

e-mail: vidya_handur@kletech.ac.in

[2]Visvesvaraya Technological University, Belagavi,590018, India

e-mail:sld@vtu.ac.in

*Corresponding author: vidya_handur@kletech.ac.in

Abstract

With the advancement in technology and the increasing number of Internet users, it has become necessary to manage servers to prevent overloading or underloading. This requires balancing the load in the network to efficiently utilize resources. In this study, a novel load balancing approach is proposed that augments Artificial Bee Colony (ABC) optimization with a mutation operator and a β-Hill Climbing local improver for task-to-node assignment. The method integrates peer-guided neighbours to produce good partial allocations, employs diversity-preserving micro mutations to avoid premature convergence, and uses a bottleneck-aware β-hill climber that repeatedly moves work from the current max-load node to under-loaded nodes, selecting the best improving micro move at each step. The fitness function reduces makespan, thereby improving resource utilization, throughput, and fairness index. When compared with other existing metaheuristic methods ELBABCEβHC, HDWOA-LBM, and WWO-ACO, the proposed method achieves an improved average resource utilization of 93%, with a reduced average makespan, higher throughput, and fair task allocation to nodes. Across heterogeneous workloads, the algorithm consistently enhances results while scaling favorably as the number of nodes or tasks increases.

1. Introduction

Artificial Bee Colony (ABC) is a population-based metaheuristic inspired by the foraging behavior of honeybees. It alternates three roles—employed, onlooker, and scout—to explore and exploit candidate solutions with greedy replacement. The existing study report introduced the core dynamics and highlighted ABC's few hyperparameters and ease of adaptation across domains [1][16]. Subsequent formalization in Journal of Global Optimization provided empirical evidence of competitive performance against contemporary methods and clarified implementation details widely reused in later variants [2]. Comparative studies then examined parameter effects and convergence behavior across diverse test functions, noting sensitivity to settings like population size and abandonment limit, and risks of stagnation on complex landscapes [3].

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In a distributed computing network, the optimization target is not a single scalar but a service profile: short makespan, high resource utilization, fair allocation across heterogeneous nodes, and adequate throughput under dynamic loads [15,17]. Surveys of load-balancing and scheduling methods consistently list these four criteria as primary objectives and constraints for multi-tenant platforms, motivating metaheuristics that can trade off among them under heterogeneity and bursty arrivals [4,18]. Despite ABC's simplicity, three challenges recur in prior literature. First, exploration–exploitation balance: basic neighbour updates are axis-aligned and can prematurely converge without diversity control; tuning "limit" and colony size mitigates but does not remove this risk [2,3]. Second, problem encoding: mapping ABC to discrete scheduling needs task-move operators and feasibility repairs, or penalties, to respect capacity and precedence; poor encodings inflate makespan and depress utilization [4]. Third, multi-objective handling: naive scalarization can bias toward makespan and starve fairness or throughput; robust formulations and local improvement steps are required to keep all four targets in view [4]. These gaps motivate hybridization that injects controlled variability and structured local search.

The proposed study is to improve load balancing on the four operational criteria—makespan, resource utilization, fairness, and throughput—while retaining ABC's low complexity. Mutation adds directed diversity to escape stagnation. β-hill climbing supplies improvement-only refinements that convert exploratory moves into consistent gains on multi-objective composites.The contributions of the study are to design an ABC variant with a mutation operator and a β-hill-climbing local search tailored to discrete task to node scheduling, to define a fitness construction that balances makespan and utilization, with maintained fairness, and throughput without collapsing to a single metric and to evaluate against strong baselines on heterogeneous workloads to justify the proposed study.

The structure of the paper is organized as follows: Section 2 presents the related ABC and load balancing work. In section 3 system and optimization model are discussed. Section 4 presents the proposed method. Section 5 details the simulation setup. Section 6 reports results and discussions. The paper is concluded in Section 7.

2. Literature review

Zeedan et al. in [5] proposed Hybrid ABC with an enhanced β-hill-climbing local search for cloud load balancing. Results show lower response time and processing cost, higher resource utilization than Round-Robin/Throttled/Active-Monitoring in CloudAnalyst. The algorithm exhibits stronger exploitation without losing exploration. The limitations include extra tuning and runtime overhead.

Li & Wang in [6] designed a Modified ABC that injects GA crossover/mutation and gbest-guided neighbourhood search. The results demonstrated improved resource utilization and energy/cost as compared to GA, HHO, ACO, and vanilla ABC. The advantage is that the premature convergence is delayed due to diversity. Its drawback is that the algorithm is sensitive to parameter settings.

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Ladj et al.in [7] proposed a novel ABC for coordinated production-maintenance scheduling in permutation flowshops. The results show that shorter makespan than variable-neighbourhood search on Taillard benchmarks. The advantage of the method is that discrete ABC adapts well to joint sequencing however it does not track throughput or fairness; the domain differs from heterogeneous clouds.

Hu et al. in [8] mentioned a Constructive-destructive neighbour search drives ABC (CDS-ABC) for variable-speed green hybrid flow-shop. The performance metrics measured are reduced makespan and energy, better IGD statistics than peers. The advantage of the method is that the targeted local moves increase solution quality. The limitation is that the added operators increase complexity and may raise compute time.

Karim et al. in [9] proposed EASA-MORU using dung-beetle optimization for cloud scheduling. The outcomes show lower makespan, higher resource utilization and throughput than several metaheuristics, but slower compute time in some loads. The major contribution is that the algorithm exhibits robust load balancing under rising request counts. However, it is computationally intensive at scale.

Kaplan et al. in [10] show a comparative study of metaheuristics for task scheduling under normal and DDoS. In the study, ABC demonstrated fast execution but lower resource utilisation and no explicit gains in fairness. The advantage is that it is a stable scheduling under attack in certain settings. The resource utilization and fairness remain concerns.

Rajammal et al. in [11] have discussed Predictive graph networks and adaptive neural scheduling; ABC was used as a baseline. The results show higher throughput and lower makespan than ABC, PSO, ACO across cloudlet and VM scales. The strength is in its predictive allocation that produces enhanced throughput at load. The limitation is that the results depend on tuned CPU and memory weights.

Jiao GE et al. in [12] have mentioned Hybrid ABC–Bat for edge–cloud resource allocation with DAG-modelled dependencies. The parameters measured ensure reduced makespan and improved resource utilization. The algorithm balances exploration–exploitation via periodic solution exchange. The concern in the algorithm is DAG modelling and parameterization overhead, and the fair distribution of load is not evaluated.

JongBeom Lim in [13] proposed an agent-based adaptive scheduling in edge–cloud. The results demonstrate fewer SLO violations and reported fairness improvements using Jain's index; better task completions at higher CPU utilization. The limitation is that it is an edge-centric evaluation.

Zhang et al. in [14] proposed a method known as Runtime Queue Scheduling with fairness. The results have shown improved fairness and throughput on switches while keeping latency bounded. The advantage of the method is that it raises Jain-style fairness without large throughput loss. The limitation is that it is network-centric.

Lilhore U.K. et al. in [19] presented an innovative hybrid optimisation method combining Water Wave Optimization and Ant Colony Optimization for dynamic resource allocation. The

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

results include reduced response times, increased resource efficiency, and lower operational expenses.

Krishnan, Ramya et al. in [20] have discussed a hybrid dingo and whale optimisation algorithm-based load balancing mechanism for achieving effective load balancing to maximise throughput, reliability, and resource utilization in cloud environments. The method adopts the mimicking of hunting characteristics of the dingo (equivalent to tasks) and VMs as their prey to be hunted.

Zeedan Maha et al. in [21] have proposed a load balancing technique based on artificial bee colony and β-Hill climbing to improve the performance metrics: response time, processing cost, and resource utilization. The experiments are carried out for different user groups.

3. System Model

Load balancing refers to assigning incoming tasks to computational nodes, considering their current loads without overloading or underloading of nodes. In the proposed study, the load balancing problem is modelled as: In a fully distributed heterogeneous system, given ‘n’ number of tasks and ‘m’ number of nodes, the ‘n' tasks have to be fairly distributed among ‘m’ nodes, considering the current load and capacity of the nodes. The nodes are heterogeneous in nature, each with a distinct architecture, speed, and capacity C. The parameters for assigning the tasks to nodes include the speed, indicating the number of tasks executed per unit of time, measured in Million Instructions Per Second (MIPS), and capacity, referring to the queue length. The tasks are heterogeneous, varying in length, quantified in Millions Instructions (MI).

3.1 Optimization Model for Load Balancing

In the proposed study, the load balancing problem is modelled as a Maximization problem. The objective is to optimally allocate ‘n’ tasks to ‘m’ nodes to minimize the makespan and maximize the resource utilization.

Let M_i be the unique identification of each node, C_i be the memory capacity, μ_i be the processing speed TL_i be the current load of node M_i for i=1,2,….,m.

Makespan: It is the time elapsed from the start of the first task to the end of the last task.

Let MS_i denote the makespan of node ‘i’. MS_i is calculated as shown in Eq.1

$$MS_i = \max_i ECT_i \qquad \forall \text{ i =1, 2, 3, ..., m} \tag{1}$$

where $ECT_i = \frac{TL_i}{\mu_i} \quad \forall\, i = 1,2,\ldots m$ indicating how long is the work queued.

The average makespan, AvgMS, is calculated as in Eq. 2

$$AvgMS = \frac{1}{m}\sum_{i=1}^{m} MSi \tag{2}$$

The objective is to minimize AvgMS and is represented as F1 as shown in Eq (3)

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$F_1 = \text{minimize AvgMS}(TL_1, TL_2, \ldots, TL_m) \quad (3)$$

Resource Utilization: This is a measure of how much of the available resources are currently being utilised. Load balancing should ensure that resources are utilised proportionally to their capabilities, maximising resource utilization. It is measured as shown in Eq (4)

$$RU_i = \frac{TL_i}{C_i} \quad \forall \ i = 1, 2, 3, \ldots, m \quad \text{subject to constraint } 0 < RU \leq 1 \quad (4)$$

The average resource utilization is measured as in Eq(5)

$$AvgRU = \frac{1}{m}\sum_{i=1}^{m} RUi \quad (5)$$

The objective is to maximize the $AvgRU$ and is represented as F2 as shown in Eq(6)

$$F2 = \text{maximize AvgRU}(TL_1, TL_2, \ldots, TL_m) \quad (6)$$

Load Balancing as a Maximization Problem:

Combining both objectives, load balancing can be expressed as a maximization problem as shown in Eq. (7).

$$\text{Maximize } F = F2 - F1 \quad (7)$$

4 Proposed Methodology

Load balancing involves distributing workloads across computational nodes to maintain system performance. The proposed study combines Artificial Bee Colony Optimization with a mutation operator and β-hill climbing (MABO-βHC) to improve makespan, resource efficiency, fairness, and throughput in task assignment to computational nodes.

Initially, tasks are received by the computational node. The node monitors its current load to decide whether the task has to be executed locally or remotely. If the current load is within the threshold, the task is carried out locally; otherwise, it is executed remotely. Let L_i be the load on a computational node M_i, let T_i be the threshold value of the node

$$\text{Task Execution} = \begin{cases} L_i \leq T_i & executed\ locally \\ L_i > T_i & executed\ remotely \end{cases} \quad \forall \ i=1,2,\ldots.m$$

To execute the task remotely, the remote node selection and allocation are based on the MABO-βHC algorithm proposed in the study.

4.1 Load Balancing using Hybrid MABO-βHC

ABC is a population-based optimizer inspired by honey bees' foraging. Each food source is a candidate solution; its nectar denotes fitness, indicating the quality of the solution. A colony of

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

bees iteratively improves solutions through exploration by finding new sources and exploitation by refining the good ones. The combined algorithm works as follows:

Step 1: Initialization of Population size with random task-to-node assignment.

Step 2: a) Employed bees phase**:** Each bee owns a food source denoted as x_i, explores its neighbourhood according to Eq. (8)

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}), \quad \emptyset_{ij} \sim U[-1,1], k \neq i \qquad \text{Eq (8)}$$

b) Mutant operator (diversity): with prob p_{mut}, apply K micro-edits by assigning one task to a random node.

c) β Hill Climbing: iteratively edits the solution to increase the fitness value

Step 3: Onlooker bees: Observe employed bees' dances, which represent the fitness, and probabilistically(p_i) select good sources to exploit such that p_i α fitness(x_i).

Step 4: Scout bees: Replace stagnant sources with random new ones to inject diversity.

The detailed algorithm is given in Algorithm 1 and Algorithm 2

Algorithm 1 Neighbor with Mutation

Input: Tasks $\{w_t\}_{t=1}^{n}$, VM capacities $\{c_i\}_{i=1}^{m}$, parameters λ, ε, mutation (p_{mut}, K), Beta (α, β), local budget $M_{\max}$.
Output: Helper routines Eval(·), Utility(·), NeighborWithMutation(·), βHillClimb(·).
function Eval(a) ▷ Per-VM loads and summary metrics
  $MS_i \leftarrow \sum_{t:a(t)=i} \frac{w_t}{c_i}$ for all i
  $C_{\max} \leftarrow \max_i MS_i$, AvgMS $\leftarrow \frac{1}{m}\sum_i MS_i$
  if $C_{\max} = 0$ **then**
    AvgRU $\leftarrow 0$
  else
    AvgRU $\leftarrow \frac{\sum_i MS_i}{m\, C_{\max}}$
  end if
  return $(\{MS_i\}, C_{\max}, \text{AvgMS}, \text{AvgRU})$
end function
function Utility(a; $\{a_j\}$) ▷ Maximization utility
  obtain AvgMS(a) and AvgRU(a) via Eval(a)
  get $\min_j \text{AvgMS}(a_j)$ and $\max_j \text{AvgMS}(a_j)$ from population $\{a_j\}$
  $\widetilde{\text{AvgMS}}(a) \leftarrow \frac{\text{AvgMS}(a) - \min_j \text{AvgMS}(a_j)}{\max_j \text{AvgMS}(a_j) - \min_j \text{AvgMS}(a_j) + \varepsilon}$
  return $U(a) = \lambda\, \text{AvgRU}(a) - (1-\lambda)\, \widetilde{\text{AvgMS}}(a)$
end function
function NeighborWithMutation(x, $peer$) ▷ Peer-guided edit + diversity
  $v \leftarrow x$; pick random task t; set $v(t) \leftarrow peer(t)$
  if rand() < 0.25 **then**
    swap VMs of two random tasks in v
  end if
  if rand() $< p_{\text{mut}}$ **then**
    for $r = 1$ **to** K **do**
      if rand() < 0.5 **then**
        reassign one random task to a random VM
      else
        swap VMs of two random tasks
      end if
    end for
  end if
  return v
end function
function βHillClimb(x; $\{a_j\}$) ▷ Targeted tail relief
  $cur \leftarrow x$; $M \leftarrow 1 + \lfloor \text{Beta}(\alpha, \beta)\, M_{\max} \rfloor$
  for $s = 1$ **to** M **do**
    $(\{MS_i\}, \_, \_, \_) \leftarrow$ Eval(cur); $i_{\max} \leftarrow \arg\max_i MS_i$
    $U_{cur} \leftarrow$ Utility(cur; $\{a_j\}$); $best \leftarrow \varnothing$; $bestImp \leftarrow 0$
    for $q = 1$ **to** 12 **do** ▷ micro-move tries
      propose y by moving/swapping one task *from* $i_{\max}$ to a candidate VM
      $U_y \leftarrow$ Utility(y; $\{a_j\}$)
      if $U_y - U_{cur} > bestImp$ **then**
        $bestImp \leftarrow U_y - U_{cur}$; $best \leftarrow y$
      end if
    end for
    if $best = \varnothing$ **then**
      break
    else
      $cur \leftarrow best$
    end if
  end for
  return cur
end function

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm 2 MABC-βHC

Input: Same problem data and parameters as Alg. 1; population size SN, cycles C, abandonment *limit*.
Output: Best assignment $a^\star$ with $(C_{\max}, \text{AvgMS}, \text{AvgRU})$.
Initialization: create SN random assignments $\{a_i\}$; trials$[i] \leftarrow 0$; evaluate all; set $a^\star = \arg\max_i \textsc{Utility}(a_i; \{a_j\})$
for *cycle* $= 1$ **to** C **do**
  recompute utilities $U(a_i)$ and update population bounds for normalization
  // Employed-bee phase
  for $i = 1$ **to** SN **do**
    pick peer $k \neq i$; $v \leftarrow \textsc{NeighborWithMutation}(a_i, a_k)$
    $v \leftarrow \beta\textsc{HillClimb}(v; \{a_j\})$
    if $\textsc{Utility}(v; \{a_j\}) \geq \textsc{Utility}(a_i; \{a_j\})$ **then**
      $a_i \leftarrow v$; trials$[i] \leftarrow 0$;
      if $\textsc{Utility}(a_i; \{a_j\}) > \textsc{Utility}(a^\star; \{a_j\})$ **then** $a^\star \leftarrow a_i$
      end if
    else
      trials$[i] \leftarrow$ trials$[i] + 1$
    end if
  end for
  // Onlooker-bee phase
  $p_i \propto \epsilon + \textsc{Utility}(a_i; \{a_j\})$; normalize $\{p_i\}$
  for $t = 1$ **to** SN **do**
    $i \leftarrow$ RouletteSelect$(\{p_i\})$; pick peer $k \neq i$
    $v \leftarrow \textsc{NeighborWithMutation}(a_i, a_k)$; $v \leftarrow \beta\textsc{HillClimb}(v; \{a_j\})$
    if $\textsc{Utility}(v; \{a_j\}) \geq \textsc{Utility}(a_i; \{a_j\})$ **then**
      $a_i \leftarrow v$; trials$[i] \leftarrow 0$;
      if $\textsc{Utility}(a_i; \{a_j\}) > \textsc{Utility}(a^\star; \{a_j\})$ **then** $a^\star \leftarrow a_i$
      end if
    else
      trials$[i] \leftarrow$ trials$[i] + 1$
    end if
  end for
  // Scout-bee phase
  for $i = 1$ **to** SN **do**
    if trials$[i] \geq$ limit **then**
      reinitialize a_i randomly (repair if needed); trials$[i] \leftarrow 0$
    end if
  end for
end for
return $a^\star$

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4. Simulation Setup

The experiment is carried out in a heterogeneous environment. The parameters and their values are given in Table 1. The simulations are carried out in the Cloudism simulator.

Scenario 1: Here, the simulations are conducted with the task size varying from 1000 to 5000 in steps of 1000 and nodes set to 200.

Scenario 2: In this scenario, simulations are conducted with a constant task size of 5000, while the number of nodes varies from 40 to 200 in steps of 40.

Table 1 Simulation Parameters

Type	Parameters	Values
Node characteristics	Number of Nodes	40-200
	Node memory capacity	4096MB
	Node speed	2000 – 6000 MIPS
Task characteristics	Number of tasks	1000-5000
	Type of task	Independent, Atomic, Non-preemptive
ABC–Mutation–βHC parameters	Colony size (SN)	20
	Max cycles	50
	Abandonment limit	5 X No. of nodes
	probability p_m	0.08
	K	2
	α	2.0
	β	5.0

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

5. Results and Discussions

Load balancing is crucial in situations where some of the nodes get overloaded causing load imbalance in the system. The study proposes a novel methodology combining ABC and βHill climbing.The performance metrics measured are the makespan and resource utilization. The throughput and the fairness index are also measured.

Throughput: It is a measure of total number of tasks completed per unit of time, as shown in Eq.(9)

$$T = \frac{Total\ tasks\ completed}{Total\ time\ taken} \tag{9}$$

Fairness Index: It is a measure of even distribution of tasks to the nodes proportional to their capacities as shown in Eq.(10)

$$JFI = \frac{(\sum_{i=1}^{m} L_i)^2}{m\,(\sum_{i=1}^{m} L_i^2)} \tag{10}$$

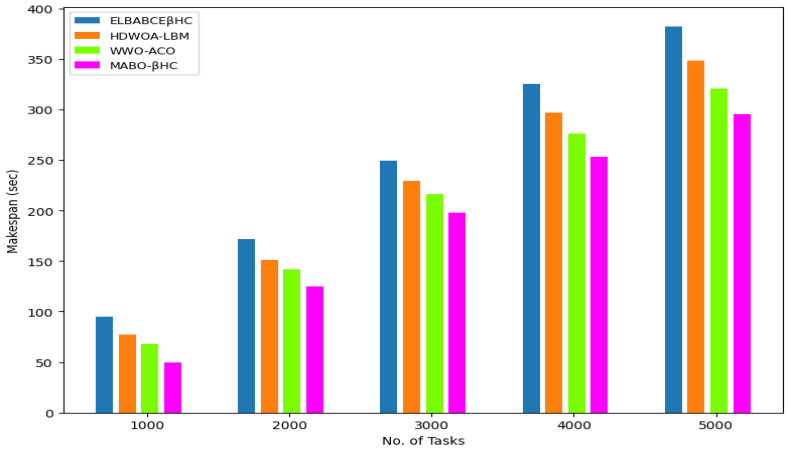

Fig. 1 Comparison of Makespan for varying tasks

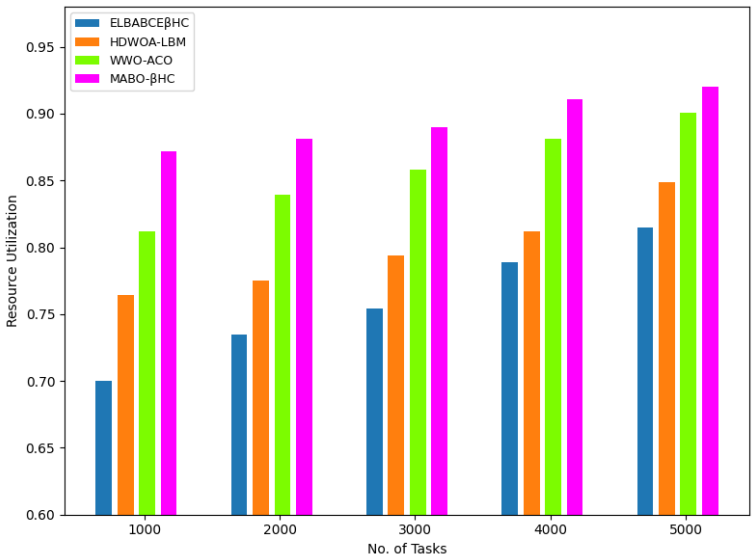

Fig. 2 Comparison of Resource Utilization for varying tasks

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

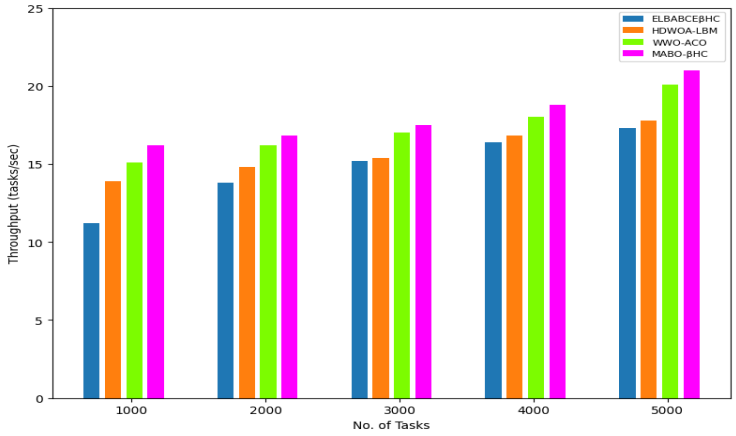

Fig. 3 Comparison of Throughput for varying tasks

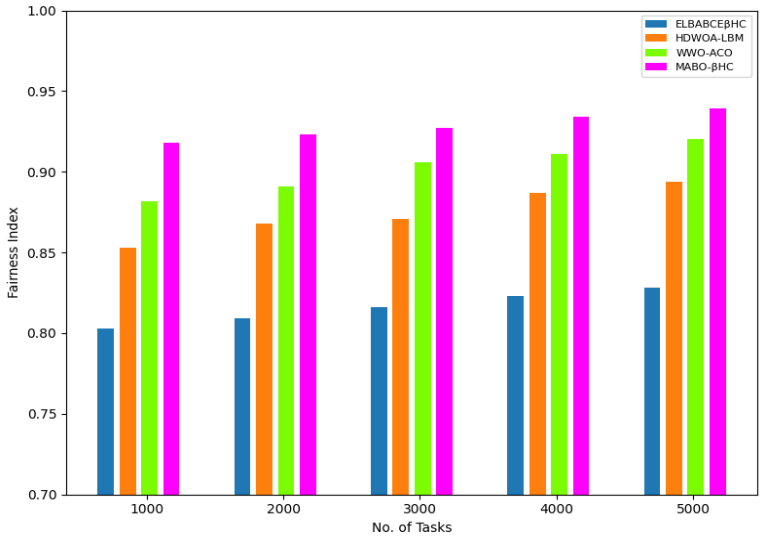

Fig. 4 Comparison of Fairness Index for varying tasks

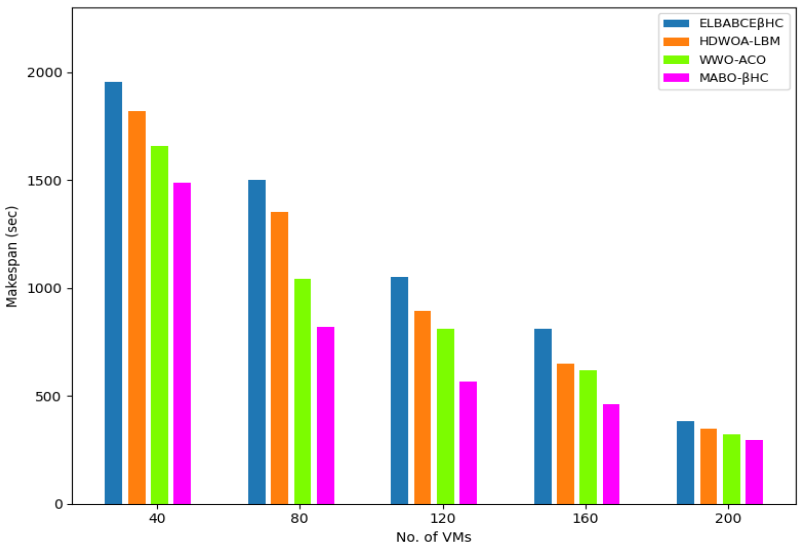

Fig. 5 Comparison of Makespan for varying nodes

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

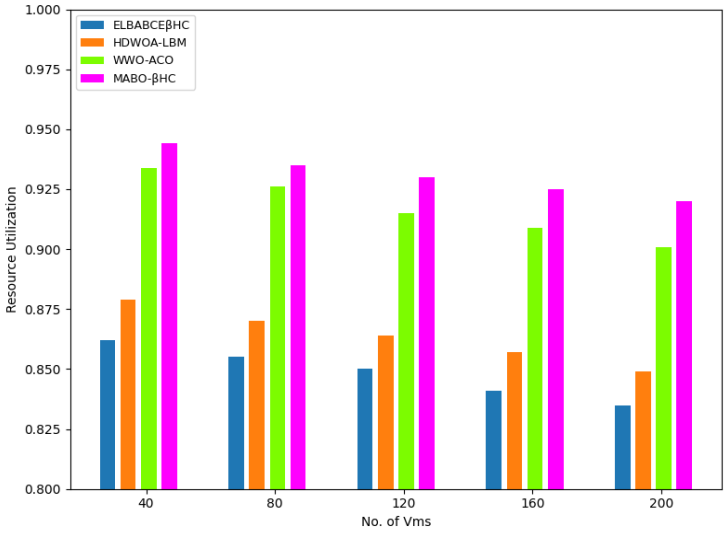

Fig.6 Comparison of Resource Utilization for varying nodes

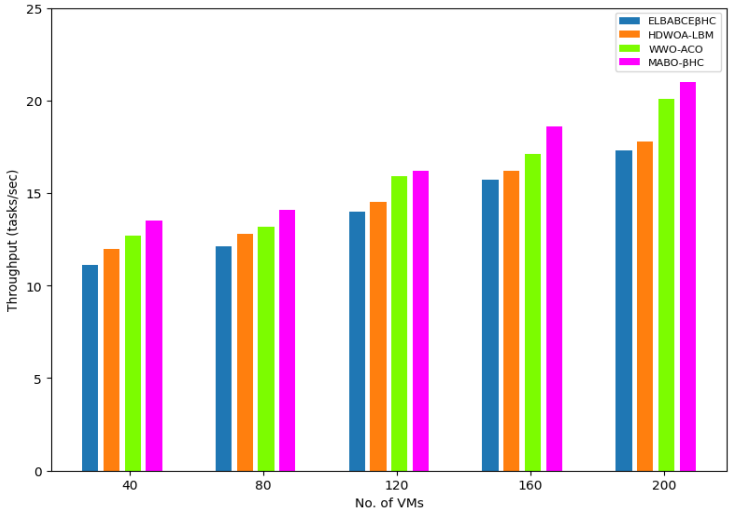

Fig. 7 Comparison of Throughput for varying nodes

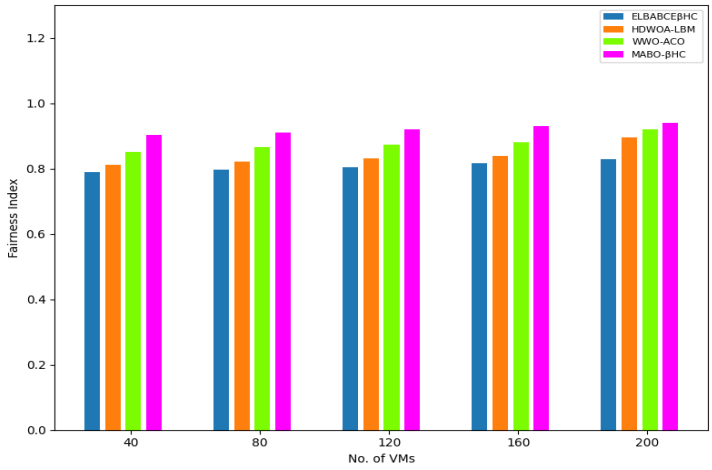

Fig. 8 Comparison of Fairness index for varying nodes

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figures 1 to 8 show the comparison of performance metrics between the existing study and the proposed study. From the graphs, it is clear that the proposed method shows improvements in the results.

Fig. 1 illustrates the comparative analysis of makespan. As shown in the graph, β-HC targets the critical node at each local step; among several candidate micro-moves, the proposed algorithm identifies the best node to minimize the makespan. Fig. 2 provides a graphical representation that shows improvements in resource utilization. The peer-guided edits and mutations distribute loads across nodes with lower variance in loads, resulting in fewer idle tails at the end and higher RU than other existing methods considered in the study. Fig. 3 is a comparative analysis of throughput measured for a varying number of tasks, showing that the proposed study gives improved throughput. The peer-guided edits and mutation explore better placement of tasks to nodes, resulting in a good start point to β-HC to finish the task. Micro mutations maintain diversity so that the colony does not collapse onto a suboptimal mapping of tasks to nodes. Also, the short, frequent and local improvements α & β keep the makespan small, raising the throughput.

Fig. 4 illustrates the fair allocation of tasks to computational nodes, clearly indicating that the proposed methodology yields good results. The micro-moves in β-hill climbing reduce the load variance by migrating the task from bottleneck nodes to underloaded nodes, resulting in an increase in fair allocation. Further, the mutation prevents the population from settling into sub-optimal patterns.

Figures 5 to 8 represent the comparative analysis of the proposed study and the existing methods for varying numbers of nodes. Fig. 6 shows the comparison of makespan for different methods and from the figure it is clear that the proposed algorithm yields less makespan. As the number of nodes grows, there are more underloaded target nodes to receive work from the current overloaded node. The utility explicitly rewards for better fitness value, so onlooker roulette automatically allocates more trials to candidates already exhibiting shorter tails. The β-hill climbing step explicitly selects the best improvement among multiple micro-moves in each cycle, so the critical path is more trimmed as nodes increase, reducing the makespan.

Fig. 6 shows the comparative analysis of Resource Utilization between the proposed study and the existing methods. The proposed method improves the resource utilization due to the strength in the algorithm. β-hill climbing moves the load from the max-load node to the underloaded nodes while the peer-guided neighbours make good placements. The utility rewards the higher resource utilization and lower makespan so onlooker roulette and greedy acceptance concentrate searches around schedules that activate new nodes and equalizes loads, preventing underloading of nodes. Mutation and multiple micro-moves per cycle explore many node assignments; the best improvements are chosen to improve the resource utilization.

Figures 7 and 8 show the comparative analysis of throughput and fairness index of the proposed method and the existing methods. As shown in the graphs, the throughput and the fair allocation of tasks to nodes is improved with the proposed method for varying numbers of nodes. With more nodes, β-hill climbing has more underloaded nodes from the current overloaded node,

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

yielding reductions in makespan, which directly increases the throughput. During each cycle the moves from max load to low loads are prioritized, thus increasing the fairness index. As the utility rewards the lower makespan and resource utilization, the onlooker sampling and the greedy acceptance allocate effort to candidates that enhance throughput and equalize loads on nodes, strengthening the algorithm as the node count increases.

6. Conclusion

In the study, a novel method is proposed by combining the Artificial Bee Colony Optimization and Hill Climbing along with Mutation, for optimal task to node allocation to solve load balancing. The novelty of the algorithm lies in peer-guided exploration, diversity-preserving mutation that prevents premature convergence and β-hill climbing to overcome overloads in the system. The alignment between the operators and the objectives yields gains across the nodes, thus the makespan decreases by offloading load from the max-load node; resource utilization increases by equalizing loads and reducing underloading; throughput increases as the makespan shrinks and fairness improves through systematic variance reduction per-node loads. The simulations indicate that the method scales favourably with the number of nodes by creating more profitable local moves, which the proposed algorithm explores through utility selection and greedy acceptance. Overall, ABC-βM delivers fast, robust convergence to balanced schedules under heterogeneous workloads, offering a practical, low-overhead optimizer for load balancing.

References

[1] D. Karaboga, “An Idea Based some on Honey Bee Swarm for Numerical Optimization,” Technical Report TR06, 2005.
[2] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm,” Journal of Global Optimization, 2007. SpringerLink
[3] D. Karaboga and B. Akay, “A comparative study of Artificial Bee Colony algorithm,” Applied Mathematics and Computation, 2009.

[4] D. A. Shafiq et al., “Load balancing techniques in cloud computing environment,” Journal of King Saud University – Computer and Information Sciences, 2022

[5]Zeedan, Maha; Attiya, Gamal; and El-Fishawy, Nawal (2023) "Enhanced Load Balancing Based on Hybrid Artificial Bee Colony with Enhanced β-Hill climbing in Cloud," *Mansoura Engineering Journal*: Vol. 48, Issue 3, Article 11. Available at: https://doi.org/10.58491/2735-4202.3098

[6] Qian LI, Xue WANG “Modified Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environments”, International Journal of Advanced Computer Science and Applications, Vol. 15, No. 5, 2024

[7] Ladj, A., Benbouzid-Si Tayeb, F., Dahamni, A., Benbouzid, M. “An Artificial Bee Colony Algorithm for Coordinated Scheduling of Production Jobs and Flexible Maintenance in

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Permutation Flowshops", *Technologies* 2024, *12*, 45. https://doi.org/10.3390/technologies12040045

[8] Hu D, Wu Y, Qiu L, Mi J, Yang Y. "Constructive-destructive neighbour search drives artificial bee colony algorithm for variable speed green hybrid flowshop scheduling problem", Sci Rep. 2025 Mar 20;15(1):9671. doi: 10.1038/s41598-025-93582-5. PMID: 40113816; PMCID: PMC11926210.

[9]K. Karim F, Ghorashi S, Alkhalaf S, H. A. Hamza S, Ben Ishak A, Abdel-Khalek S,"Optimizing makespan and resource utilization in cloud computing environment via evolutionary scheduling approach", PLoS ONE, 2024,19(11): e0311814. https://doi.org/10.1371/journal.pone.0311814

[10]Kaplan, F.; Babalik, A. Performance Analysis of Cloud Computing Task Scheduling Using Metaheuristic Algorithms in DDoS and Normal Environments. *Electronics* **2025**, *14*, 1988. https://doi.org/10.3390/electronics14101988

[11] Rajammal, K., Chinnadurai, M. Dynamic load balancing in cloud computing using predictive graph networks and adaptive neural scheduling. *Sci Rep* **15**, 22181 (2025). https://doi.org/10.1038/s41598-025-97494-2

[12] Jiao GE, Bolin ZHOU and Na LIU, "Hybrid Artificial Bee Colony and Bat Algorithm for Efficient Resource Allocation in Edge-Cloud Systems" International Journal of Advanced Computer Science and Applications (IJACSA), 16(2), 2025. http://dx.doi.org/10.14569/IJACSA.2025.01602101

[13] JongBeom Lim," Adaptive Scheduling Based on Intelligent Agents in Edge-Cloud Computing Environments",Journal of Internet Technology Vol. 25 No. 4, July 2024, DOI: https://doi.org/10.70003/160792642024072504011

[14] Hong Zhang, Zhenge Xu, Feixue Han, Fuliang Li, Qing Li,"RTQS: Real-time flow fairness queue scheduling policy on router devices", Computer Networks, Volume 269,2025, 111443, ISSN 1389-1286,https://doi.org/10.1016/j.comnet.2025.111443.

[15]Minghao Tong, Zhenhua Peng, Qin Wang, "A hybrid artificial bee colony algorithm with high robustness for the multiple travelling salesman problem with multiple depots", Expert Systems with Applications, Volume 260,2025,125446, ISSN 0957-4174,https://doi.org/10.1016/j.eswa.2024.125446.

[16]VS Handur, S L Deshpande,"Artificial bee colony optimization-based load balancing in distributed computing systems—a survey",Smart Trends in Computing and Communications: Proceedings of SmartCom 2022,Pages 733-740

[17] R. Marakumbi Prakash S. Handur Vidya,"Response time analysis of dynamic load balancing algorithms in Cloud Computing",2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4).

Received: August 08, 2025

Volume 38 No. 7s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[18]Vidya S Handur, Santosh L Deshpande,"Particle swarm optimization for load balancing in distributed computing systems–a survey", Turkish Journal of Computer and Mathematics Education (TURCOMAT), Volume 12, Issue 1S, Pages 257-265

[19]Lilhore U.K., Simaiya S., Prajapati Y.N. et al. "A multi-objective approach to load balancing in cloud environments integrating ACO and WWO techniques", Sci Rep 15, 12036 (2025). https://doi.org/10.1038/s41598-025-96364-1

[20]Krishnan, Ramya & Ayothi, Senthilselvi. (2023). Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment. Transactions on Emerging Telecommunications Technologies. 34. 10.1002/ett.4760.

[21]Zeedan, Maha; Attiya, Gamal; and El-Fishawy, Nawal (2023) "Enhanced Load Balancing Based on Hybrid Artificial Bee Colony with Enhanced β-Hill climbing in Cloud," *Mansoura Engineering Journal*: Vol. 48: Iss.3,Article11. Available at: https://doi.org/10.58491/2735-4202.3098

Received: August 08, 2025