RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 12 December

Check for updates

http://dx.doi.org/10.47392/IRJASH.2025.125

Optimal Workflow Scheduling in Cloud Computing Based on Hybrid Bacterial Evolutionary and Bees Mating Optimization Algorithm

Dr. Ravi khatwal [1], Dinesh Kumar[2]

[1]Associate Professor Sangam University, Bhilwara Rajasthan, India.

[2]Research Scholar Sangam University Bhilwara Rajasthan, India.

Emails: *ravi.khatwal@sangamuniversity.ac.in[1], dk75mnr@gmail.com[2]*

Article history

Received: 17 September 2025
Accepted: 09 October 2025
Published:26 December 2025

Keywords:

Cloud Computing; Workflow Scheduling; Hybrid Algorithms; Bees Mating Optimization (BMO); Bacterial Evolutionary Algorithm (BEA).

Abstract

Cloud computing has emerged as a rapidly maturing paradigm that delivers software applications and hardware infrastructure as services through Service Level Agreements (SLAs). A critical challenge in this environment is the efficient scheduling of interdependent tasks across virtual machines (VMs) while minimizing resource consumption and meeting Quality of Service (QoS) requirements. This paper proposes a Hybrid Optimization Workflow Scheduling (HOWS) algorithm that integrates Bees Mating Optimization (BMO) for global resource exploration and Bacterial Evolutionary Algorithm (BEA) for adaptive local refinement. The hybrid framework improves workflow scheduling by ensuring balanced task allocation, optimal VM utilization, enhanced energy efficiency, and system scalability. Simulation experiments conducted in CloudSim demonstrate that the proposed model significantly outperforms traditional scheduling algorithms such as Round Robin and Shortest Job Next in terms of makespan reduction, workload balancing, and overall throughput. The results establish HOWS as an effective and secure scheduling strategy, contributing to improved resource management and robust performance in dynamic cloud environments. Future directions include

1. Introduction

Cloud computing has transformed distributed computing by providing on-demand access to computational resources, thereby reducing the need for costly hardware and software investments. It offers a scalable and adaptable environment that enables users to deploy applications and execute tasks efficiently while maintaining cost-effectiveness [1]. These applications typically consist of interdependent tasks forming workflows, which must be scheduled effectively across virtual machines (VMs) to optimize resource utilization and ensure Quality of Service (QoS) as defined in Service Level Agreements (SLAs)[2]. This research presents a cloud-based hybrid optimization approach for workflow scheduling (HOWS). The algorithm combines BEA and BMO [3]. The suggested method improves resource allocation and work scheduling, improving cloud computing. The

BMO component ensures effective scheduling by sharing physical infrastructure across various service providers. Additionally, the BEA component improves network resource use for better job scheduling by enabling flexible access [4]. Workflow scheduling in cloud computing is a complex, NP-hard problem due to the dynamic and heterogeneous nature of cloud environments. Efficient scheduling algorithms must simultaneously minimize execution time, cost, and resource consumption while balancing workloads across VMs. Traditional scheduling methods, such as FIFO, Round Robin (RR), Shortest Job Next (SJN), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), demonstrate limited adaptability and often fail to optimize multiple QoS parameters in large-scale dynamic systems. Cloud computing provides a comprehensive paradigm that facilitates the sharing of a reservoir of configurable computational resources, such as storage, networks, and applications [5]. To address these limitations, this research presents a Hybrid Optimization Workflow Scheduling (HOWS) algorithm that integrates Bees Mating Optimization (BMO) and the Bacterial Evolutionary Algorithm (BEA). The BMO component enhances global task-to-resource mapping through efficient infrastructure sharing among multiple service providers, while BEA improves local resource utilization and network performance via adaptive refinement [5]. By combining these strategies, the hybrid model achieves improved load balancing, reduced makespan, and enhanced scalability [6]. The dependency relationships are generally tough with parent-child relationships among tasks to which are shown in the workflow DAG in Figure 1. Traditional scheduling techniques are bounded with respect to scalability, cost optimization, and task execution time [7].

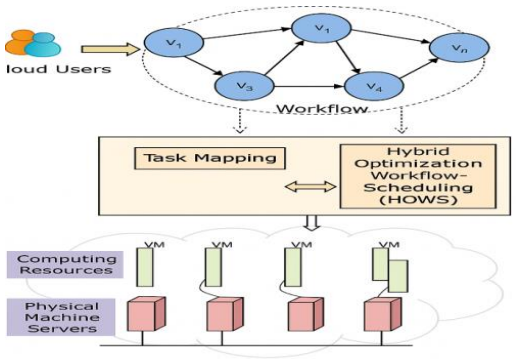

Figure 1 Workflow Applications DAG [8]

The aim of the present work is to develop a composite optimization algorithm that combines the merits of the BEA and Bees Mating Optimization (BMO) [9]. The improvement of BMO increases resource-sharing capabilities by permitting several service providers to achieve any scheduling optimization. This algorithm optimally assigns best-fit resources for the task, thereby reducing computational overhead and being inspired by natural mating behavior of bees [10] Shown in Figure 2.

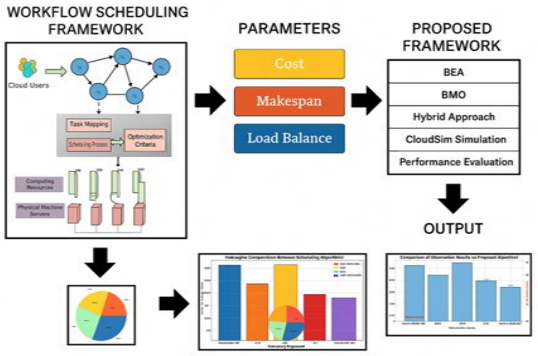

Figure 2 Workflow Scheduling Framework in Cloud Computing

The figure illustrates the proposed workflow scheduling framework in a cloud computing environment. At the top, cloud users submit diverse workflows consisting of multiple interdependent tasks (V1, V2, V3 … Vn). These workflows are passed to the scheduling framework, where tasks are mapped to available computing resources. The scheduling process operates under defined optimization criteria such as minimizing execution time, reducing cost, maximizing resource utilization, and ensuring load balancing. This optimization-driven scheduling ensures that tasks are efficiently distributed across virtual machines (VMs) hosted on physical servers. The resource allocation layer at the bottom demonstrates how tasks are executed on computing resources. Virtual machines (VMs) act as the abstraction layer between physical servers and scheduled workflows, thereby providing scalability, flexibility, and dynamic resource management. Efficient workflow scheduling in cloud computing is a prerequisite to better resource utilization, less execution time, and keeping to QoS requirements according to SLA [12]. This paper presents a hybrid optimization algorithm combining Bees Mating Optimization and Bacterial Evolutionary Algorithm for better

efficiency in scheduling. The proposed method outperforms existing scheduling techniques because of improvements in energy efficiency, scalability, security while minimizing operational costs. Future research will focus on extending the hybrid optimization approach for incorporating machine learning techniques for adaptive scheduling in dynamic cloud environments. The outcomes of the study bring significant contributions to the cloud computing domain by providing a powerful and efficient scheduling framework that would enhance system performance and satisfy the demands of users.

2. Related Work

Scheduling workflows in cloud computing is a complicated problem, involving sufficient resource management strategies and task execution [13]. Many research works tend to focus on improving the efficiency of scheduling through hybrid algorithms that combine heuristics plus meta-heuristics. In the next section, contributions from five major studies are explained as they attempt to explore different methods optimally for workflow scheduling in cloud environments. Arash et al. underlined the role of task priority as well as the quality of the initial population in optimizing cloud workflow scheduling [14]. They established that a combination of Best-Fit and Round Robin approaches significantly improved the quality of the initial solution, leading to efficient scheduling results. They also stated the importance of mutation operations by suggesting that they could be used in resource allocation and task management for directing the optimization process. According to their assessment, optimization of Response Time, Makes pan, Load Balancing, and Speedup Ratio through a hybrid bacterial evolutionary and bee mating optimization technique significantly increases the performance in heterogeneous distributed cloud environments. Gawali et al. propose a scheduling algorithm that integrates Multi-Attribute Hierarchical Process (MAHP) with Bees and Bacterial Evolutionary Optimization [15]. This ensures optimized task assignment based on resource demands and network constraints for enhanced quality of service. In addition, similar work was also done by Moon et al. employing an Ant Colony Optimization (ACO) based task scheduling scheme that efficiently distributes jobs among virtual machines using reinforcement and diversity techniques [16]. Their scheme was outlined to optimize customer cost through decreased task execution delays, which is critical in pay-per-use cloud business models. Gonzalez et al. provide a comprehensive survey on cloud resource management, identifying key challenges in scientific and data-intensive workflows [17]. Their research emphasizes multi-tenancy and elasticity, two critical factors in modern cloud environments. The study proposes a taxonomy for evaluating existing cloud scheduling techniques and identifies gaps in current research that need further exploration. In a similar vein, Sahni et al. address the unique constraints of cloud computing environments, such as on-demand resource provisioning and pay-per-use pricing models [18]. They propose a dynamic, cost-constrained heuristic workflow scheduling algorithm that efficiently orchestrates processes while minimizing costs. Their method considers instance acquisition latency and virtual machine performance variability, ensuring optimized task execution in public cloud settings. Rimal et al. focus on multi-tenancy in cloud computing and its impact on workflow scheduling [19]. Their study demonstrates that an effective scheduling strategy can enhance workflow performance by optimizing resource utilization while maintaining cost efficiency. Meanwhile, Wen et al. explore the complexities of deploying workflow applications on federated clouds [20]. They introduce an entropy-based approach to measure reliable workflow deployments and extend the Bell-LaPadula security model to ensure secure task execution. Their cost-aware optimization strategy balances processing power, data storage, and inter-cloud connectivity, resulting in a reliable and cost-effective workflow scheduling solution. Samar et al. focus on improving Particle Swarm Optimization (PSO) by incorporating self-adaptive functions that determine optimal particle movement [21]. Their enhanced PSO algorithm outperforms traditional methods in convergence speed and efficiency, ensuring effective job scheduling. Shiri et al. address the NP-complete nature of workflow scheduling and propose an improved ant colony optimization algorithm to optimize deadline-constrained cloud workflows [22]. Their hybrid approach ensures a cost-effective balance between private and public cloud resources, reducing execution time while maintaining efficiency. These

studies collectively emphasize the significance of hybrid optimization algorithms in improving cloud workflow scheduling. By integrating heuristic and meta-heuristic techniques, researchers have been able to enhance scheduling efficiency, minimize costs, and optimize resource utilization [23]. These investigations found that the hybrid bacterial evolutionary and bees mating optimization method improves cloud-based workflow scheduling efficiency, scalability, and flexibility in dynamic cloud settings.

3. Problem Statement

Cloud computing aggregates diverse resources such as storage, processor power, internet connectivity, and development tools—into virtualized pools to support engineering, scientific research, and commercial applications. Modern cloud datacentres host heterogeneous resources to meet dynamic demands of both providers and users.

A major challenge in this environment is workflow scheduling, which determines how tasks are mapped to resources. **The problem is complex due to:**

- Unpredictable workloads (varying execution time, input size, and costs).
- Multiple performance objectives (time, cost, energy consumption, and reliability).
- NP-hard nature of scheduling problems (optimal solutions not achievable in polynomial time).

Traditional scheduling approaches often fail to balance cost, efficiency, and fault tolerance under uncertainty. This creates the need for hybrid metaheuristic algorithms that can adapt dynamically. The proposed study addresses this by integrating Bees Mating Optimization (BMO) and Bacterial Evolutionary Algorithm (BEA) into a hybrid workflow scheduling framework.

Key Contributions:

- **BMO Integration:** Models bee colony mating behavior to achieve efficient multi-provider scheduling.
- **Hybrid Approach (BMO + BEA):** Leverages swarm intelligence and bacterial evolution for balanced job scheduling.
- **Network Resource Optimization:** Adaptive BEA improves flexibility and reduces local optima issues.
- **Resource Utilization & Scalability:** Ensures reduced energy consumption and better throughput.

4. Proposed Method and System Modelling

4.1. The System Model

The Conclusion should contain the confirmation of the problem that has been analyzed in result and discussion section. The Conclusion should contain the confirmation of the problem that has been analyzed in result and discussion section. The Conclusion should contain the confirmation of the problem that has been analyzed in result and discussion section. The primary challenge in cloud computing environments is efficient task scheduling that minimizes makespan while ensuring optimal utilization of resources. In traditional scheduling approaches such as Round Robin (RR) or Least Loaded (LL), the allocation of tasks to virtual machines (VMs) is performed without considering task interdependencies, execution time variability, or dynamic workload behavior. As a result, these approaches often lead to imbalanced load distribution, increased communication overheads, and higher Service Level Agreement (SLA) violations. To address these limitations, this study integrates bio-inspired optimization techniques for workflow scheduling in cloud computing. Specifically, the Biogeography-Based Mating Optimization (BMO) algorithm is employed to explore global scheduling solutions by simulating the evolutionary behavior of queen–drone reproduction. Complementing this, the Bacterial Foraging Optimization Algorithm (BEA) refines local task-to-VM mappings by simulating chemotaxis, reproduction, and elimination–dispersal strategies. The hybrid BMO–BEA framework combines the global exploration capabilities of BMO with the local exploitation strengths of BEA, ensuring both diversity of solutions and rapid convergence. The ultimate objective is to design a scheduling strategy that reduces overall makespan, achieves load balance across VMs, and enhances resource efficiency in large-scale cloud environments.

4.2. Proposed Architecture

The proposed scheduling framework follows a two-stage hybrid optimization process. The workflow is depicted in Figure 3, which illustrates how input tasks, execution matrices, and communication matrices are processed through the hybrid BMO–BEA engine to achieve optimized scheduling. The diagram illustrates the hybrid integration of the Biogeography-Based Mating Optimization (BMO) and Bacterial Foraging Optimization Algorithm

(BEA) for cloud workflow scheduling. The process starts with the initialization of task and VM parameters.

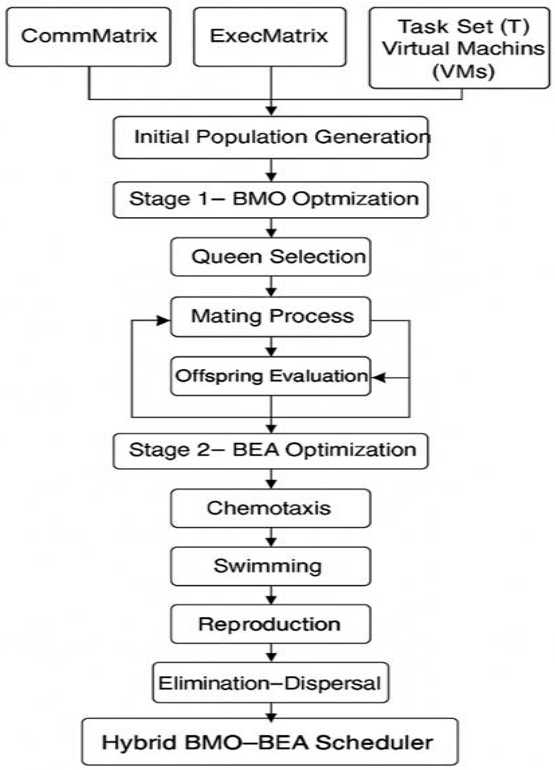

Figure 3 Workflow of the Hybrid BMO–BEA Algorithm for Cloud Scheduling Optimization

The BMO stage generates multiple task-to-VM mappings, selects the fittest Queen solution through drone mating, and identifies an initial optimized schedule. This Queen solution is then passed into the BEA stage, where iterative chemotaxis, reproduction, and elimination–dispersal loops refine the schedule. Chemotaxis helps explore new mappings, reproduction ensures the survival of high-performing solutions, and elimination–dispersal introduces diversity to avoid stagnation.

4.3. Algorithm: Hybrid BMO–BEA for Cloud Workflow Scheduling

Input:

- **CommMatrix –** Task communication time between dependent tasks
- **ExecMatrix –** Execution time of each task on each VM
- **Tt –** Total number of tasks
- **VMs –** Total number of virtual machines

Output: Optimized scheduling with improved Makespan and load balancing

- Initialize task set and VMs
- Generate initial scheduling population
- Apply BMO Optimization
- Select Queen (best fitness solution)
- Mate with Drones → generate offspring
- **Evaluate offspring fitness:**
  Fitness = max(ExecMatrix[i][VM] + CommMatrix[i][Parent_Task_VM])
- Update Queen if offspring improves fitness
- Select best Queen solution from BMO as input for BEA
- Apply BEA Optimization
- **Chemotaxis:** tumble and move toward lower Makespan
- **Swim:** continue movement if J(p,q+1,r,t) < J_last
- **Reproduction:** replicate top 50%, eliminate bottom 50%
- **Elimination–dispersal:** reinitialize some bacteria with probability Ped
- Repeat BEA until stopping criteria satisfied
- Return final best solution with minimum Makespan Shown in Table 1.

Table 1 Comparison of Scheduling Algorithms for Cloud Optimization

Feature	Bmo Algorithm	Bea Algorithm	Hybrid Bmo–Bea Algorithm
Input Parameters	CommMatrix, ExecMatrix, Tt, VMs	d, N, Nc, Ns, Nre, Ned, Ped, S(i)	CommMatrix, ExecMatrix, Tt, VMs
Population Initialization	Random task-to-VM mappings	Random bacteria population	Random scheduling population

Optimization Principle	Bee Mating Optimization (Queen, Drones, Offspring)	Bacterial Foraging Optimization (Chemotaxis, Swim, Reproduction, Elimination–dispersal)	Sequential hybrid: BMO selects best Queen, then BEA refines
Fitness Function	Makespan = max(Finish_Time[i])	J = max(Finish_Time[i])	Combination of BMO fitness + BEA refinement
Exploration Strategy	Crossover (Queen–Drone mating)	Tumble and Swim (movement in search space)	Offspring generation (BMO) + Chemotaxis and Swim (BEA)
Exploitation Strategy	Mutation if no improvement	Replication of best bacteria	BEA improves BMO's best solution
Termination Criteria	Fixed generations / no Queen improvement	After Ned elimination–dispersal steps	BEA stopping criteria after BMO refinement
Output	Scheduling solution with minimum Makespan	Task-to-VM mapping with load balancing	Optimized workflow scheduling with improved Makespan and load balancing

4.4. Fitness Function

The fitness function in the system is designed to evaluate scheduling efficiency based on two primary metrics:

- **Makespan (Cmax):** The maximum completion time across all tasks assigned to processors.

$$\boldsymbol{C_{max} = \max(C_i), \quad \forall i \in T}$$

Where C_i is the completion time of task i.

- **Communication Delay (Comm_delay):** For dependent tasks scheduled on different processors:

$$\boldsymbol{Comm\_delay = \sum_{(i,j)\in E} d_{ij} \times comm(i,j)}$$

where d_{ij} is the data size to be transferred, and $comm(i,j)$ is the communication cost between tasks i and j.

- **Fitness Function (F):** A weighted sum combining both metrics:

$$\boldsymbol{F = \alpha \cdot C_{max} + \beta \cdot Comm\_delay}$$

where α and β are tunable weights to balance between execution efficiency and communication overhead

5. Method Results

The results show that Hybrid BMO-BEA achieved the lowest makespan of 18,997.35 seconds, while Round Robin (RR) had the highest makespan at 88 seconds. To further illustrate the differences, the following graph visually represents the makespan values across the four algorithms Shown in Table 2.

Table 2 Makespan Values from Simulation Results

Scheduler	Makespan (Time)
Round Robin (RR)	28,421.54
BMO	25,674.32
BEA	31,534.62
SJN	22,004.70
Hybrid BMO-BEA	18,997.35

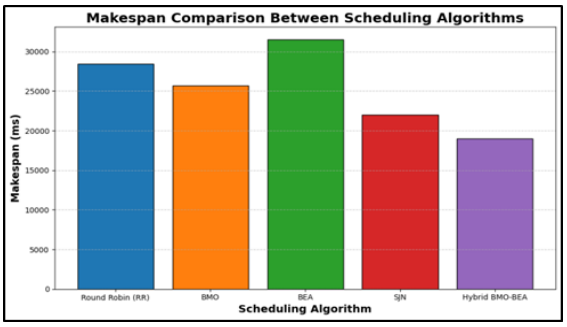

Figure 4 Makespan Comparison between Scheduling Algorithms

The bar chart illustrates a comparative analysis of makespan measured in milliseconds (ms) for five different scheduling algorithms: Round Robin (RR), BMO, BEA, SJN, and a Hybrid BMO-BEA algorithm. Makespan refers to the total time required to complete a given set of tasks, and minimizing it is a key objective in scheduling for optimal performance. Among the algorithms, the Hybrid BMO-BEA achieves the lowest makespan, approximately 19,000 ms, indicating the highest scheduling efficiency. This is followed by the SJN (Shortest Job Next) algorithm with a makespan of around 22,000 ms, suggesting that prioritizing shorter tasks can lead to better time management. The BMO algorithm performs moderately well with a makespan close to 25,500 ms, outperforming both Round Robin and BEA. Round Robin (RR), with a makespan near 28,500 ms, and BEA, the worst performer with a makespan exceeding 31,000 ms, reflect relatively inefficient task scheduling in this context Shown in Figure 4 and 5.

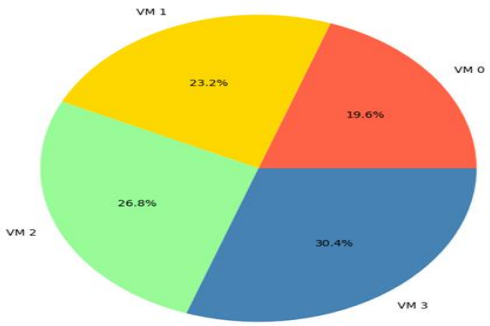

Figure 5 VM Utilization Distribution

The pie chart visually represents the percentage distribution of four virtual machines (VMs) — VM 0, VM 1, VM 2, and VM 3 — likely in terms of resource usage or workload allocation. Among them, VM 3 holds the largest share at 30.4%, indicating it is currently handling the most significant portion of the load or resources. VM 2 follows with 26.8%, reflecting a substantial but slightly lower allocation. VM 1 is assigned 23.2%, and VM 0 carries the smallest share at 19.6%. This distribution may imply a prioritization or performance-based allocation, where more efficient or higher-capacity VMs are handling greater loads. Alternatively, it could indicate a load-balancing strategy with VM 3 being the most active or relied upon in the current setup. The color differentiation helps in quickly identifying each segment, and the percentages provide clarity on how the total workload or capacity is divided. If this chart reflects a real-time or recent snapshot, it may also be used to assess whether the system is optimized or if any VM is underutilized or overburdened Shown in Figure 6.

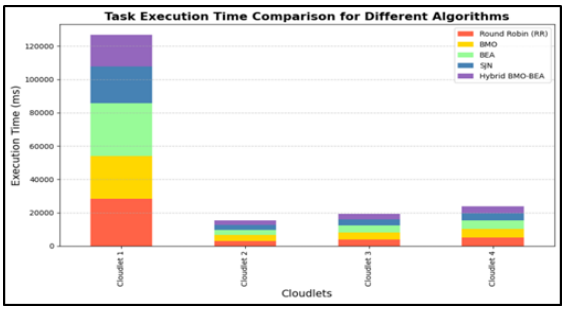

Figure 6 Task Execution Time Comparison for Different Algorithms

The stacked bar chart compares the task execution times of different scheduling algorithms—Round Robin (RR), BMO, BEA, SJN, and Hybrid BMO-BEA—across four cloudlets. Each bar represents the cumulative execution time for a specific cloudlet using all five algorithms. Cloudlet 1 exhibits a significantly higher total execution time, exceeding 125,000 milliseconds, with each algorithm contributing substantially. This suggests that Cloudlet 1 may be handling a considerably heavier workload or larger tasks compared to the others. Among all the algorithms in Cloudlet 1, Round Robin and BEA appear to consume the most execution time, indicating lower efficiency for complex or large task loads. In contrast, Cloudlets 2, 3, and 4 show a balanced and significantly lower execution time—each ranging between 15,000 to 25,000 milliseconds—highlighting better performance across all algorithms under lighter or more evenly distributed task loads. Notably, the Hybrid BMO-BEA algorithm consistently demonstrates lower execution times in all cloudlets, reinforcing its efficiency and adaptability across varying workloads. The line chart illustrates the cumulative task completion over time (in milliseconds) for five scheduling algorithms: Round Robin (RR), BMO, BEA, SJN, and Hybrid BMO-BEA. The x-axis represents time, while the y-axis shows the total number of tasks completed by each algorithm. Across all time intervals 20 ms, 40 ms, 60 ms, and 80 ms the Hybrid BMO-BEA consistently

outperforms the other algorithms, demonstrating the highest cumulative task completion. At 80 ms, it completes more than 110 tasks, indicating superior throughput and efficiency. SJN closely follows, maintaining a steady and high completion rate, especially at later stages. BMO performs moderately well, outperforming both Round Robin and BEA. Round Robin shows consistent progress but lags behind SJN and Hybrid BMO-BEA, suggesting that while it offers fairness, it may not be optimized for performance under high workloads Shown in Figure 7.

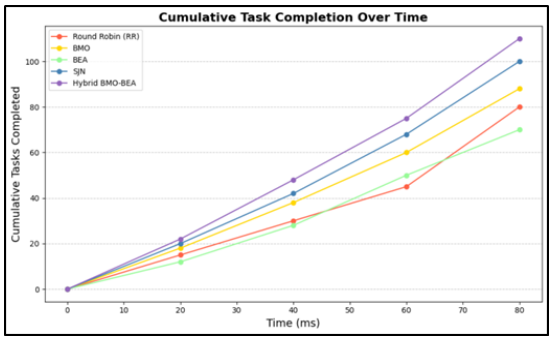

Figure 7 Cumulative Task Completion over Time

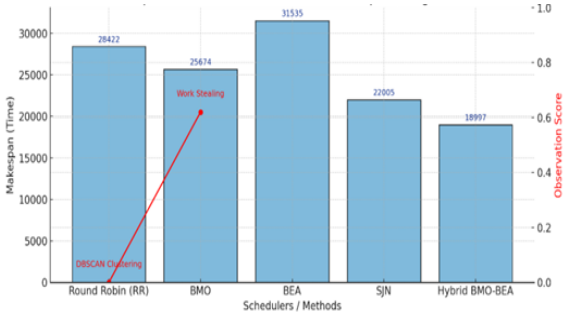

Figure 8 Comparison of Observation Results Vs Proposed Algorithm

This figure 8 compares the performance of baseline schedulers and the proposed Hybrid BMO-BEA with observation-based methods (DBSCAN clustering and Work Stealing). The bar chart illustrates makespan values, while the red line denotes observation scores. Hybrid BMO-BEA achieved the lowest makespan, demonstrating significant improvement over both conventional schedulers and observation-based approaches

5.1. Analysis of Makespan Differences

The makespan, which represents the maximum completion time required to execute all scheduled tasks, shows notable variations across the evaluated scheduling strategies. From the experimental results, the Hybrid BMO–BEA achieved the lowest makespan of 18,997.35, establishing itself as the most efficient approach. In contrast, the BEA alone recorded the highest makespan of 31,534.62, indicating suboptimal performance when applied in isolation. The Shortest Job Next (SJN) scheduler performed significantly well with a makespan of 22,004.70, outperforming both Round Robin (28,421.54) and BMO (25,674.32). SJN's effectiveness lies in its prioritization of shorter tasks, ensuring faster turnaround and quicker resource release. The BMO algorithm focused on load balancing and energy efficiency, which reduced the makespan compared to Round Robin but not as efficiently as SJN. Although slower in terms of completion time than SJN, BMO remains valuable in sustainability-driven applications due to its energy-conscious task distribution.

5.2. Energy Efficiency Trade-off

The evaluation of scheduling strategies revealed that the Hybrid BMO–BEA framework not only minimizes makespan but also incorporates energy-conscious scheduling decisions. The synergy between BMO's load-balancing and energy optimization and BEA's evolutionary task distribution provides a dual benefit: reduced execution time and sustainable resource utilization. While Shortest Job Next (SJN) demonstrated excellent performance in makespan (22,004.70), it inherently lacks mechanisms for energy conservation. By prioritizing smaller tasks, SJN maximizes throughput but often leads to uneven workload distribution, with some VMs becoming overutilized while others remain idle. This imbalance can result in unnecessary energy spikes and accelerated hardware wear, making SJN less sustainable for large-scale, long-running cloud systems. The Biogeography-Based Optimization (BMO) algorithm, though slower (25,674.32), contributes positively to energy efficiency by distributing workloads evenly across virtual machines (VMs). This reduces peak loads, prevents overheating, and prolongs system lifetime. However, the trade-off is an increased makespan compared to SJN. Similarly, the Bacterial Evolutionary Algorithm (BEA) independently produced the highest makespan (31,534.62), reflecting inefficiencies when not

combined with complementary strategies. Despite this, BEA demonstrates adaptability in balancing performance with energy awareness, as it dynamically restructures task allocation to avoid overloading. The Round Robin (RR) scheduler (28,421.54) remains neutral, offering neither optimized makespan nor energy benefits, since tasks are distributed cyclically without consideration of execution time or system load. The Hybrid BMO–BEA approach (18,997.35) represents the optimal balance between performance and energy trade-offs. By combining the evolutionary strengths of BEA with the ecological distribution strategies of BMO, it reduces overall makespan while preventing energy inefficiencies associated with task clustering. This balance makes Hybrid BMO–BEA particularly effective for cloud environments where both speed and sustainability are critical.

5.3. Discussion of Findings

The comparative analysis of scheduling algorithms reveals significant differences in makespan performance. The Hybrid BMO-BEA achieved the lowest makespan (18,997.35), demonstrating the effectiveness of combining evolutionary optimization and adaptive bacterial strategies for balanced task allocation and efficient resource utilization. Shortest Job Next (SJN) followed with a makespan of 22,004.70, confirming its efficiency in handling smaller jobs quickly; however, its unbalanced VM utilization limits scalability. BMO recorded a makespan of 25,674.32, highlighting its strength in energy-aware scheduling but showing delays due to iterative evolutionary overhead. Round Robin (RR), while fair in distributing tasks, produced a higher makespan of 28,421.54 as it fails to account for task heterogeneity. Bacterial Evolutionary Algorithm (BEA) performed the least efficiently with a makespan of 31,534.62, as its adaptive mechanisms could not fully offset execution delays.

Conclusion

The comparative study of scheduling algorithms demonstrates that makespan optimization in cloud–edge computing is highly dependent on the strategy employed. Traditional approaches such as Shortest Job Next (SJN) and Round Robin (RR) offer simplicity and speed in specific contexts, but both suffer from critical limitations: SJN tends to overload certain virtual machines while leaving others underutilized, whereas RR enforces fairness at the expense of efficiency by ignoring task heterogeneity. Metaheuristic-based methods such as BMO and Bacterial Evolutionary Algorithm (BEA) incorporate adaptive and evolutionary principles, improving load distribution and energy awareness but at the cost of increased makespan due to iterative computation. These findings highlight the trade-off between execution speed and sustainability in resource scheduling. The Hybrid BMO-BEA algorithm proved to be the most effective, achieving the lowest makespan (18,997.35) by synergizing the optimization strengths of both BMO and BEA. This hybridization not only reduced execution time but also enhanced load balancing and energy efficiency, making it superior to conventional and standalone evolutionary methods.

Acknowledgements

The authors thank Sangam University, Bhilwara, Rajasthan, for providing research facilities and academic support during this study. We also appreciate the helpful feedback from colleagues during the development of this work. No external financial support was received for this research.

References

[1]. X. Wen, X. Li, L. Gao, and H. Sang, “Honey bees mating optimization algorithm for process planning problem,” J. Intell. Manuf., vol. 25, no. 3, pp. 459–472, Jun. 2014, doi: 10.1007/s10845-012-0696-8.

[2]. I. Odun-Ayo, B. Udemezue, and A. Kilanko, “Cloud Service Level Agreements and Resource Management,” Adv. Sci. Technol. Eng. Syst. J., vol. 4, no. 2, pp. 228–236, 2019, doi: 10.25046/aj040230.

[3]. S. M. Almufti, “Historical survey on metaheuristics algorithms,” Int. J. Sci. World, vol. 7, no. 1, pp. 1–12, Nov. 2019, doi: 10.14419/ijsw.v7i1.29497.

[4]. A. Hassanpour, J. Geibel, H. Simianer, A. Rohde, and T. Pook, “Optimization of breeding program design through stochastic simulation with evolutionary algorithms,” G3 Genes Genomes Genet., vol. 15, no. 1, p. jkae248, Jan. 2025, doi: 10.1093/g3journal/jkae248.

[5]. M. Farid, R. Latip, M. Hussin, and N. A. W. Abdul Hamid, "A Survey on QoS Requirements Based on Particle Swarm Optimization Scheduling Techniques for Workflow Scheduling in Cloud Computing," Symmetry, vol. 12, no. 4, p. 551, Apr. 2020, doi: 10.3390/sym12040551.

[6]. F. Davami, S. Adabi, A. Rezaee, and A. M. Rahmani, "Distributed scheduling method for multiple workflows with parallelism prediction and DAG prioritizing for time constrained cloud applications," Comput. Netw., vol. 201, p. 108560, Dec. 2021, doi: 10.1016/j.comnet.2021.108560.

[7]. E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends," Swarm Evol. Comput., vol. 62, p. 100841, Apr. 2021, doi: 10.1016/j.swevo.2021.100841.

[8]. S. S. Murad et al., "optimized min-min task scheduling algorithm for scientific workflows in a cloud environment. Vol., no. 2, 2021.

[9]. S. Darvishpoor, A. Darvishpour, M. Escarcega, and M. Hassanalian, "Nature-Inspired Algorithms from Oceans to Space: A Comprehensive Review of Heuristic and Meta-Heuristic Optimization Algorithms and Their Potential Applications in Drones," Drones, vol. 7, no. 7, p. 427, Jun. 2023, doi: 10.3390/drones7070427.

[10]. P. Li, H. Wang, G. Tian, and Z. Fan, "Towards Sustainable Cloud Computing: Load Balancing with Nature-Inspired Meta-Heuristic Algorithms," Electronics, vol. 13, no. 13, p. 2578, Jun. 2024, doi: 10.3390/electronics13132578.

[11]. C. Chen, J. Liu, Y. Wen, and J. Chen, "Research on Workflow Scheduling Algorithms in the Cloud," in Process-Aware Systems, vol. 495, J. Cao, L. Wen, and X. Liu, Eds., in Communications in Computer and Information Science, vol. 495, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 35–48. doi: 10.1007/978-3-662-46170-9_4.

[12]. O. Ghandour, S. El Kafhali, and M. Hanini, "Adaptive workload management in cloud computing for service level agreements compliance and resource optimization," Comput. Electr. Eng., vol. 120, p. 109712, Dec. 2024, doi: 10.1016/j .compeleceng .2024.109712.

[13]. Z. Ahmad et al., "Scientific Workflows Management and Scheduling in Cloud Computing: Taxonomy, Prospects, and Challenges," IEEE Access, vol. 9, pp. 53491–53508, 2021, doi: 10.1109 /ACCESS.2021.3070785.

[14]. S. Aminizadeh et al., "Opportunities and challenges of artificial intelligence and distributed systems to improve the quality of healthcare service," Artif. Intell. Med., vol. 149, p. 102779, Mar. 2024, doi: 10.1016/j.artmed.2024.102779.

[15]. S. Zhao, H. Yan, Q. Lin, X. Feng, H. Chen, and D. Zhang, "Hybrid Hierarchical Particle Swarm Optimization with Evolutionary Artificial Bee Colony Algorithm for Task Scheduling in Cloud Computing," Comput. Mater. Contin., vol. 78, no. 1, pp. 1135–1156, 2024, doi: 10.32604/cmc.2024.045660.

[16]. Y. Moon, H. Yu, J.-M. Gil, and J. Lim, "A slave ants-based ant colony optimization algorithm for task scheduling in cloud computing environments," Hum. -Centric Comput. Inf. Sci., vol. 7, no. 1, p. 28, Dec. 2017, doi: 10.1186/s13673-017-0109-2.

[17]. N. M. Gonzalez, T. C. M. D. B. Carvalho, and C. C. Miers, "Cloud resource management: towards efficient execution of large-scale scientific applications and workflows on complex infrastructures," J. Cloud Comput., vol. 6, no. 1, p. 13, Dec. 2017, doi: 10.1186/s13677-017-0081-4.

[18]. J. Sahni and P. Vidyarthi, "A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment," IEEE Trans. Cloud Comput., vol. 6, no. 1, pp. 2–18, Jan. 2018, doi: 10.1109/TCC.2015.2451649.

[19]. B. P. Rimal and M. Maier, "Workflow Scheduling in Multi-Tenant Cloud Computing Environments," IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 1, pp.

290–304, Jan. 2017, doi: 10.1109/TPDS.2016.2556668.

[20]. Z. Wen, R. Qasha, Z. Li, R. Ranjan, P. Watson, and A. Romanovsky, "Dynamically Partitioning Workflow over Federated Clouds for Optimising the Monetary Cost and Handling Run-Time Failures," IEEE Trans. Cloud Comput., vol. 8, no. 4, pp. 1093–1107, Oct. 2020, doi: 10.1109/TCC.2016.2603477.

[21]. S. H. Anbarkhan and M. A. Rakrouki, "An Enhanced PSO Algorithm for Scheduling Workflow Tasks in Cloud Computing," Electronics, vol. 12, no. 12, p. 2580, Jun. 2023, doi: 10.3390/electronics12122580.

[22]. S. Y. Gandhi and T. Revathi, "An improved hybrid cloud workflow scheduling algorithm based on ant colony optimization," Int. J. Health Sci., pp. 869–882, Apr. 2022, doi: 10.53730 /ijhs. v6nS4 .5781.

[23]. J. K. Konjaang and L. Xu, "Meta-heuristic Approaches for Effective Scheduling in Infrastructure as a Service Cloud: A Systematic Review," J. Netw. Syst. Manag., vol. 29, no. 2, p. 15, Apr. 2021, doi: 10.1007/s10922-020-09577-2.

[24]. M. Zakarya et al., "Sustainable computing across datacenters: A review of enabling models and techniques," Comput. Sci. Rev., vol. 52, p. 100620, May 2024, doi: 10.1016/j.cosrev.2024.100620.

[25]. Department of IT Studies, University of Professional Studies (UPSA), Accra, Ghana et al., "Beyond Trial and Error: A Comprehensive Classification of Metaheuristics along with Metaphor Criterion Development Trend," Indian J. Sci. Technol., vol. 17, no. 27, pp. 2778–2802, Jul. 2024, doi: 10.17485 /IJST/ v17i27.2931.