Educational Administration: Theory and Practice
2022, 28(2), 293 - 301
ISSN: 2148-2403
https://kuey.net/
Research Article

Adaptive Cloud Load Balancing Using Opposition-Learning- Artificial Bee Colony Optimization

Shameer A P[1*], Haseeb V V[2]

[1*]Department of Computer Science, NAM College Kallikkandy, Kannur, Kerala, India, 670693; shameerap@gmail.com
[2]Department of Computer Science, NAM College Kallikkandy, Kannur, Kerala, India, 670693, haseebvvs@gmail.com

Citation: Shameer A P, et.al (2022). Adaptive Cloud Load Balancing Using Opposition- Learning- Artificial Bee Colony Optimization, *Educational Administration: Theory and Practice*, 28(2) 293 - 301
Doi: 10.53555/kuey.v28i02.9815

ARTICLE INFO

ABSTRACT

The increasing complexity and scale of cloud computing environments demand intelligent strategies for managing workloads across distributed resources. Load balancing plays a pivotal role in ensuring that computational tasks are efficiently allocated to available servers, thereby improving performance, reducing latency, and maintaining service reliability. This paper introduces an enhanced metaheuristic approach for cloud load balancing by combining the Artificial Bee Colony (ABC) algorithm with Opposition Learning (OL). While ABC provides a biologically inspired mechanism for optimization, its performance may degrade in high-dimensional or rapidly changing scenarios due to premature convergence. The integration of OL enhances the exploration capability of the algorithm by considering opposite candidate solutions during the search process, increasing the chances of escaping local optima. This research presents a comprehensive overview of the OLABCA-LB framework, proposed as an effective solution for dynamic load balancing within cloud environments. The study introduces a detailed mathematical model along with the key parameters integrated into the fitness function design, which collectively guide the balanced allocation of computational tasks across physical hosts and virtual machines. The paper also outlines the simulation environment, describing the setup and configuration used to evaluate the proposed algorithm. Performance analysis is conducted under varying conditions, including different task volumes, instruction lengths, and scales of virtual machine deployment, to validate the robustness and scalability of the OLABCA-LB approach.

1. INTRODUCTION

The rapid evolution of cloud computing has fundamentally transformed the paradigm of distributed computing by providing scalable and elastic resource provisioning tailored to varying user needs. Cloud environments host numerous applications and services with diverse workloads, often under dynamic and unpredictable user demands. In such settings, load balancing emerges as a crucial technique to ensure the efficient utilization of cloud resources, maintain service quality, and reduce operational costs. Cloud load balancing refers to the strategic distribution of incoming tasks or network traffic across multiple computing instances or servers in a cloud infrastructure. Its primary objective is to prevent resource bottlenecks, avoid overloading specific nodes, and maintain high availability and reliability of services. An effective load balancing mechanism not only improves system throughput and response time but also plays a vital role in energy efficiency and cost optimization by minimizing idle or underutilized resources. By enabling scalable resource allocation via the internet, cloud computing has transformed the way organizations handle data management, processing, and storage. A key challenge in cloud computing is effectively managing load balancing—this entails distributing tasks uniformly across various computing nodes to enhance resource efficiency, reduce latency, and maintain system dependability. To address this, bio-inspired optimization algorithms have gained increasing attention. Among these, the ABC algorithm, inspired by the foraging behavior of honey bees, stands out for its simplicity and effectiveness in handling complex optimization problems. However, standard ABC can sometimes suffer from slow convergence and premature stagnation in local optima. To overcome these limitations, the integration of OL into the ABC algorithm has been proposed. OBL enhances the algorithm's exploration ability by simultaneously evaluating a candidate solution and it's opposite, thereby accelerating convergence and improving the chances of finding a global optimum. The OLABCA-LB framework is developed through the integration of OBL with the ABC metaheuristic, aiming to enhance both the initial population diversity and the exploration–exploitation balance within the search space. This hybrid method is specifically designed to optimize load balancing across virtual machines in cloud environments. By embedding the principles of opposition learning, the algorithm strengthens its ability to explore potential solutions more effectively and adaptively during the optimization process. In this work, the fitness function is carefully constructed to incorporate multiple critical parameters—energy consumption, makespan, response time, data center operational cost, and the degree of resource imbalance. By considering these factors, the proposed approach achieves a more robust and efficient allocation of cloud workloads, supporting dynamic and scalable resource management.

The proposed OLABCA-LB framework is designed to address three key objectives in cloud resource management:

1. It facilitates effective load distribution by classifying VMs into overloaded and under loaded categories, ensuring balanced task allocation across the infrastructure.
2. It emphasizes energy efficiency within data centers by identifying opportunities to reduce power consumption, ultimately lowering operational costs.
3. It identifies underutilized VMs within the data center, enabling a dynamic transition of these machines from an active state to a low-power or sleep mode when appropriate.

The proposed approach also demonstrates the ability to accelerate convergence speed while enhancing the solution quality, thereby enabling more effective and timely load balancing within the cloud infrastructure. Furthermore, it introduces a strategy for defining upper and lower threshold values based on key workload characteristics. These thresholds act as indicators for detecting overutilization and underutilization of VMs, determined by analyzing the volume and distribution of incoming tasks. This dynamic thresholding mechanism supports more responsive and informed resource management, ensuring that the system adapts efficiently to fluctuating workloads.

2. FRAMEWORK OF THE OLABCA-LB SYSTEM

The implementation of the OLABCA-LB system involves the following structured steps.

Step 1: Generate an initial population using random initialization techniques.

Step 2: Derive the opposite population by applying the generalized OL approach.

Step 3: Select the top **NS** solutions based on fitness from the initial population **P**, combine them with the opposition population **OP**, and treat the merged set as the updated initial population.

Step 4: This phase integrates the Generalized OB with Differential Evolution (DE) to enhance global optimization efficiency. Accordingly, when a randomly generated value rand (0,1) is less than the predefined opposition probability **O**P, the opposition population corresponding to the search space is generated using the GODE strategy.

Step 5: In this stage, the onlooker bee selects a promising solution by evaluating the fitness probability, which is influenced by the fitness values obtained by each employed bee. Upon generating a new candidate solution, the onlooker bee updates its current solution if the newly generated one offers better fitness.

Step 6: During the onlooker bee phase, an enhanced strategy is applied that merges the search equation with the mutation mechanism from DE. At this point, the modification rate is dynamically adapted according to the current iteration count. This dynamic adjustment promotes greater exploration during the early stages of the algorithm and enhances exploitation as the algorithm approaches convergence.

Step 7: Any solution that remains unchanged over a predefined number of iterations is considered abandoned. This solution is then reinitialized and passed to the scout bee for exploration. If an update for the abandoned solution is detected, it is replaced with a randomly generated solution during the current iteration.

Step 8: The algorithm terminates once the stopping condition is met; otherwise, the process loops back to Step 2 for continued optimization.

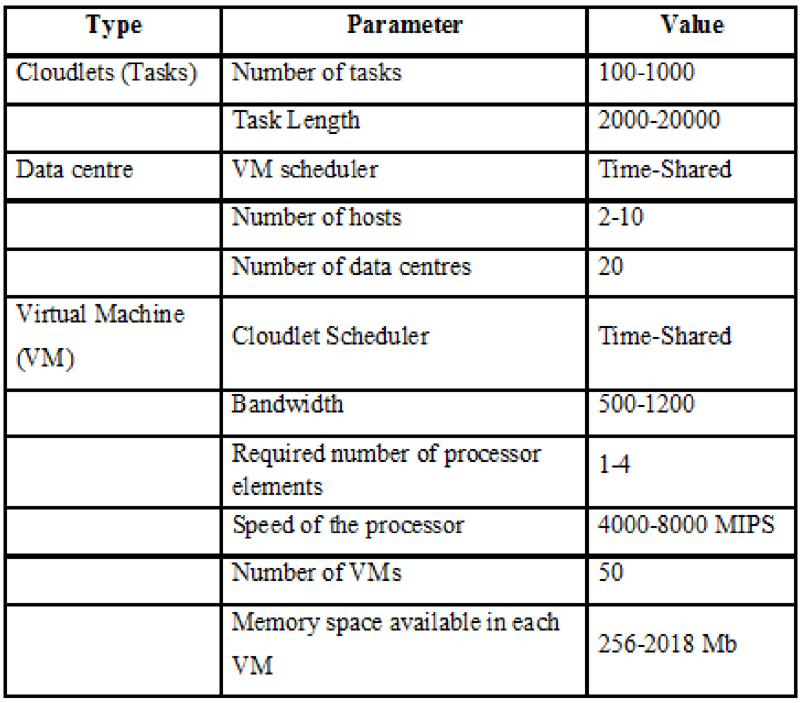
3. ALGORITHM OF THE OLABCA-LB SCHEME

The OLABCA-LB system employs a multi-objective approach to manage the distribution and redistribution of either new and existing tasks across suitable VMs or hosts. A fundamental aspect of task allocation involves adhering to key constraints, notably ensuring that a VM's workload exceeds a defined upper limit once a task is assigned. When a significant number of VMs are available, the task scheduling process also considers task deadlines to enhance overall performance. Furthermore, the system facilitates task migration from VMs experiencing high loads to those with lower loads, with this decision being guided by the task's deadline requirements or its anticipated completion time. Specifically, when dealing with tasks that are expected to take a long time to finish, the system prioritizes VMs that are currently handling tasks with the earliest deadlines. Conversely, for tasks with average expected completion times, preference is given to VMs that are processing tasks with moderate to longer deadlines. VMs are dynamically grouped based on current load conditions. Two distinct groups are formed: overloaded and under loaded VM groups determined using the objective function. Tasks are moved from VMs within the overloaded group, and these VMs temporarily

cease processing tasks until a suitable VM is identified for reassignment in subsequent iterations. Simultaneously, under loaded VMs are assigned pending tasks or those designated for reallocation. This process of task redistribution continues iteratively until all under loaded VMs have received tasks.
The OLABCA-LB framework is designed to achieve three primary goals:
i) To enhance load balancing by categorizing VMs into overloaded and underloaded groups.
ii) To improve energy efficiency by decreasing energy consumption and operational costs within the data center.
iii) To identify underutilized VMs that are suitable for transitioning from active to sleep states.
The system also incorporates dynamic management of VM power states -ON/OFF- based on their utilization levels and predefined energy thresholds. Virtual machines are automatically transitioned to a sleep mode when their workload falls below a specified lower limit and are reactivated when the load exceeds an upper threshold. This dynamic state management optimizes energy consumption and operational efficiency within the data center.

4. SIMULATION SETUP

The performance of the OLABCA-LB scheme was rigorously assessed through simulations conducted using the CloudSim toolkit. CloudSim played a pivotal role in modeling and simulating various operations, enabling a comprehensive study of the scheme's effectiveness under dynamic and resource-variant cloud conditions. To evaluate scalability and adaptability, the OLABCA-LB scheme was tested across diverse configurations involving multiple hosts, data centers, and VMs. The simulation setup included 20 data centers, 100 virtual machines (VMs), and a dynamic workload comprising between 500 and 1000 tasks. For performance analysis, the instruction lengths of tasks were defined within the range of 2,000 to 20,000 MI, simulating heterogeneous workloads. Additionally, the critical simulation parameters employed for evaluating the OLABCA-LB scheme are detailed in Table 1, providing a clear foundation for replicability and benchmarking



Type	Parameter	Value
Cloudlets (Tasks)	Number of tasks	100-1000
	Task Length	2000-20000
Data centre	VM scheduler	Time-Shared
	Number of hosts	2-10
	Number of data centres	20
Virtual Machine (VM)	Cloudlet Scheduler	Time-Shared
	Bandwidth	500-1200
	Required number of processor elements	1-4
	Speed of the processor	4000-8000 MIPS
	Number of VMs	50
	Memory space available in each VM	256-2018 Mb

Table 1: Parameters Used for the proposed Scheme

5. SIMULATION OUTCOMES AND ANALYSIS

This part of the study evaluates the performance of the OLABCA-LB load balancing scheme by analyzing how the mean response time is affected by variations in both the number of tasks and the length of executable instructions within a cloud computing setup. Figures 1 and 2 present the response trends observed across different configurations. When the instruction length is set to 1000 bytes, the average response time rises from 7.21 seconds with 200 tasks to 22.32 seconds with 2000 tasks. Increasing the instruction length to 7000 bytes leads to a response time range of 7.85 to 44.37 seconds over the same task interval. For instruction lengths of 15000 bytes and 20000 bytes, the mean response time increases from 8.02 to 85.31 seconds and 7.02 to 101.21 seconds, respectively, as the task volume increases. This upward trend in response time across larger task loads and instruction sizes is primarily influenced by the threshold-based mechanism embedded in the OLABCA-LB scheme. These thresholds are designed to detect over-utilization and under-utilization in cloud resources, impacting how tasks are scheduled and managed as the system load grows.

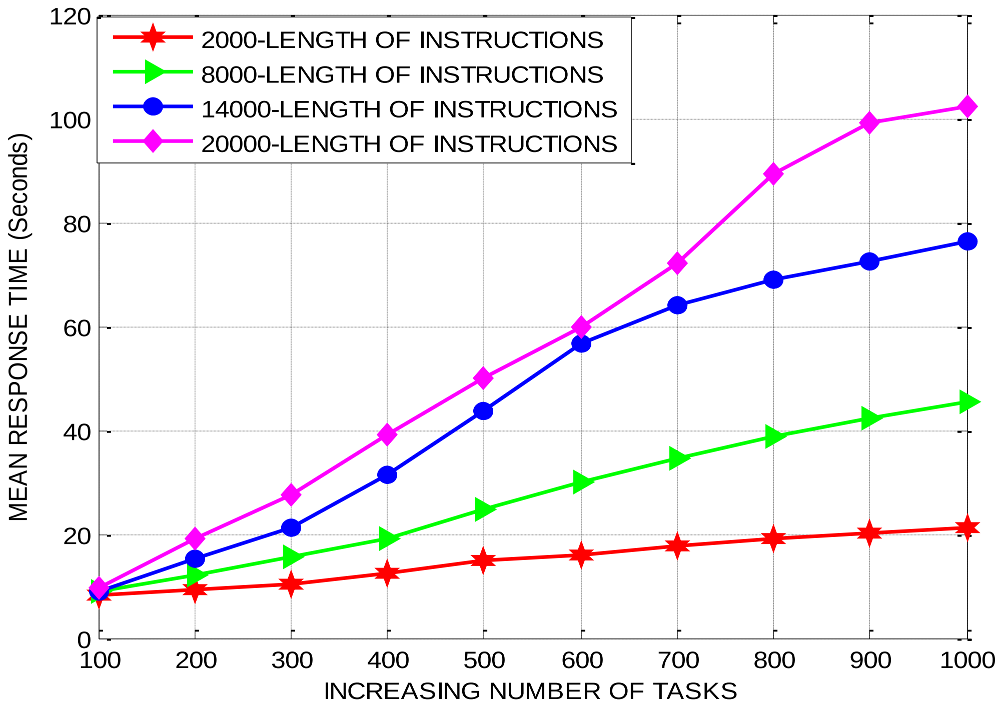


Figure 1: Mean Response Time under number of tasks

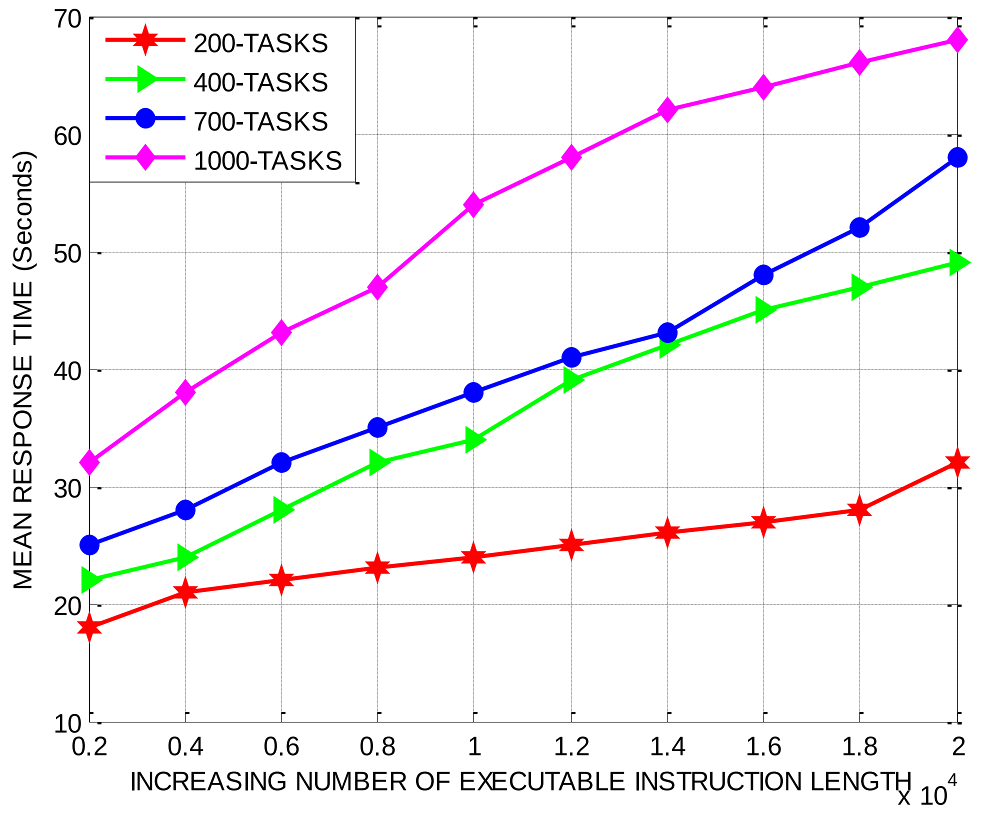


Figure 2: Standard deviation under different threshold values

In Figure 2, the OLABCA-LB scheme's average response time increases as the executable instruction length grows, depending on the number of tasks being processed. When handling 100 tasks, the response time rises from 18.01 seconds with a 100-instruction length to 31.22 seconds with a 1000-instruction length. Similarly, for 300 tasks, the response time increases between 21.35 seconds and 42.11 seconds. With 600 tasks, the response time escalates from 26.02 to 63.31 seconds, and for 1000 tasks, it extends from 33.74 to 91.44 seconds. This pattern is due to the adaptive nature of the OLABCA-LB scheme's load balancing process, which adjusts according to instruction length by using set threshold values to ensure system effectiveness.

5.2 Performance over benchmarked swarm intelligent schemes

This section delivers a comparative performance analysis of the proposed OLABCA-LB with three established swarm intelligence-based load balancing approaches: ABC-LB, PSO-LB, and DE-LB. The evaluation considers two key variables—the number of tasks and the length of executable instructions. The experimental setup involves varying the number of tasks from 100 to 1000, with the instruction length proportionally increasing from 1000 to 10000. A fixed threshold of 0.1 was consistently applied during the evaluation process. As illustrated in Figure 3, OLABCA-LB consistently records a lower average response time compared to its counterparts. Specifically, it achieves performance improvements of 11.24%, 11.55%, and 11.98% when measured against PSO-LB, DE-LB, and ABC-LB, respectively. These enhancements are largely attributed to the incorporation of adaptive upper and lower threshold strategies, which enable more effective workload distribution across virtual machines, thereby reducing latency and improving system responsiveness.

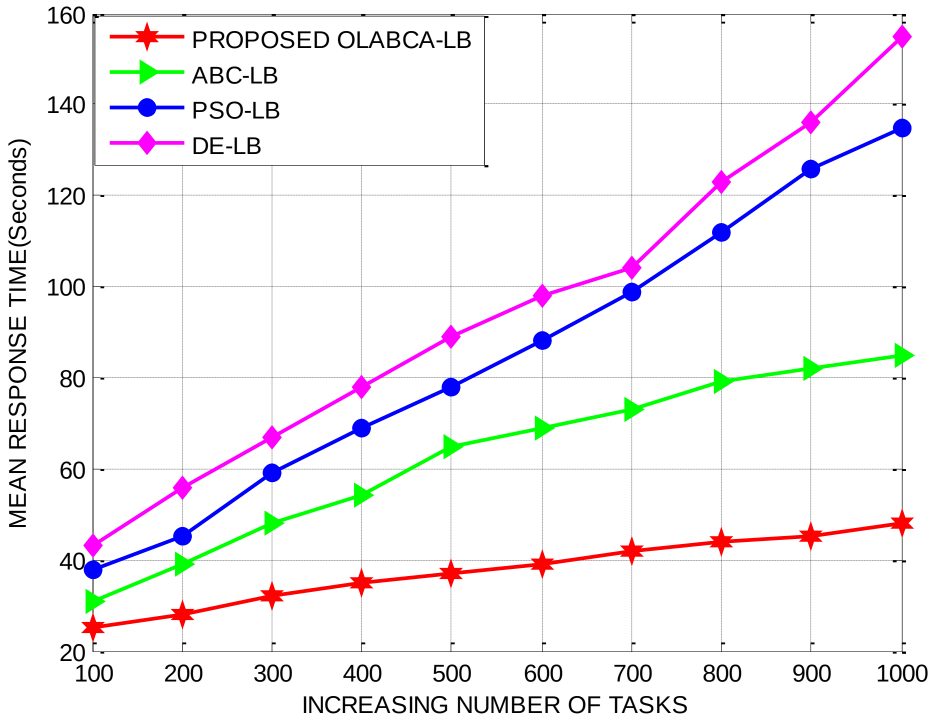


Figure 3: Average Response Time Across Varying Task Loads

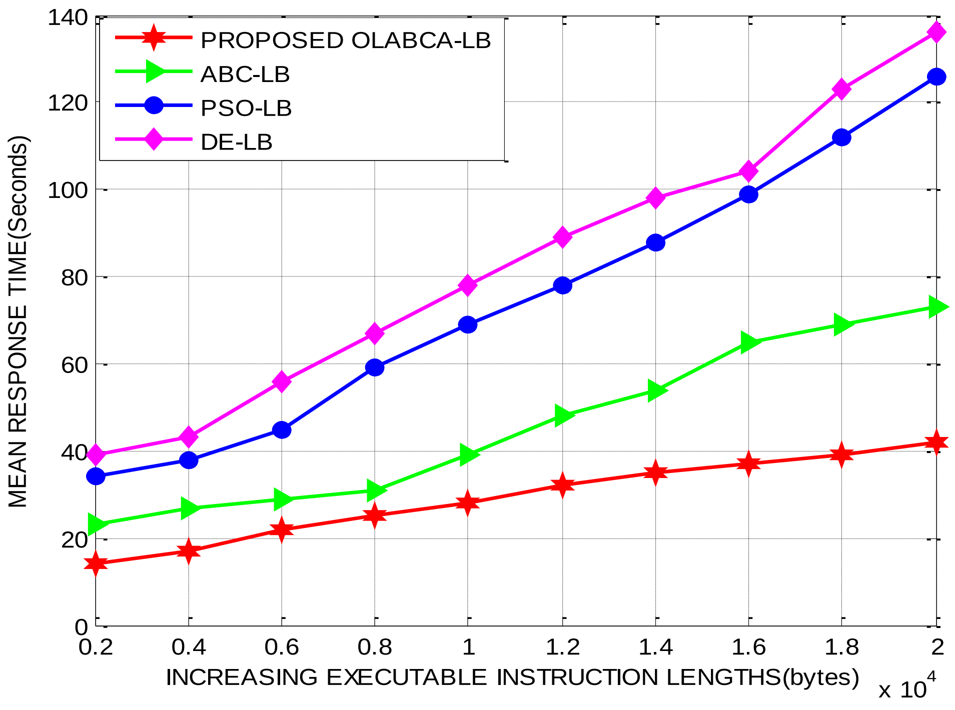


Figure 4: Instruction Length on Average Response Time

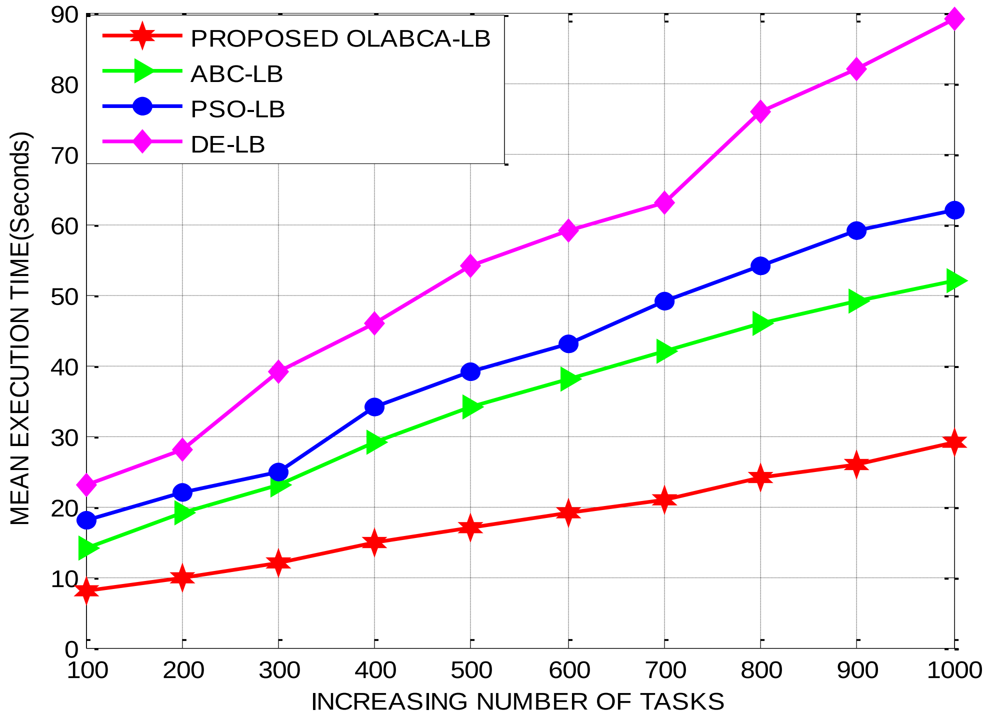


Figure 5: Proposed Mean response time under number of tasks

Figure 4 illustrates the average response time performance of the OLABCA-LB, ABC-LB, PSO-LB, and DE-LB algorithms across a range of instruction lengths, varying from 1000 to 10000. The data clearly shows that the OLABCA-LB approach consistently delivers superior efficiency, with reductions in mean response time by 7.88%, 8.14%, and 9.44% in comparison to ABC-LB, PSO-LB, and DE-LB, respectively. This improvement stems from the algorithm's ability to dynamically manage and minimize standard deviation in load allocation, resulting in more balanced task distribution among virtual machines. In Figure 5, the analysis shifts focus to the average execution time, when the task load expands from 200 to 2000. The OLABCA-LB work again demonstrates its effectiveness, achieving execution time savings of 9.04%, 10.33%, and 12.56% over ABC-LB, PSO-LB, and DE-LB respectively. These gains are primarily attributed to the implementation of reactive thresholding mechanisms, which intelligently adjust upper and lower bounds to mitigate load imbalance and improve overall system throughput.

The effectiveness of the OLABCA-LB strategy is further evaluated by observing how the number of task migrations changes as the number of VMs increases. As shown in Figures 6 and 7, which depict the results for task loads of 100 and 500 respectively, the proposed method consistently outperforms alternative strategies across different VM configurations. The results reveal that OLABCA-LB substantially reduces task migration, even as the number of VMs grows. This improvement stems from the algorithm’s use of adaptive upper and lower thresholds for resource availability, which enhances the flexibility and efficiency of VM allocation and de-allocation during load balancing. In the scenario involving 100 tasks, OLABCA-LB achieves significant reductions in task migrations—outperforming ABC-LB, PSO-LB, and DE-LB by 7.22%, 8.76%, and 9.54%, respectively. These findings highlight the method’s capacity to maintain optimal task distribution while avoiding unnecessary migrations, contributing to greater system efficiency and stability.

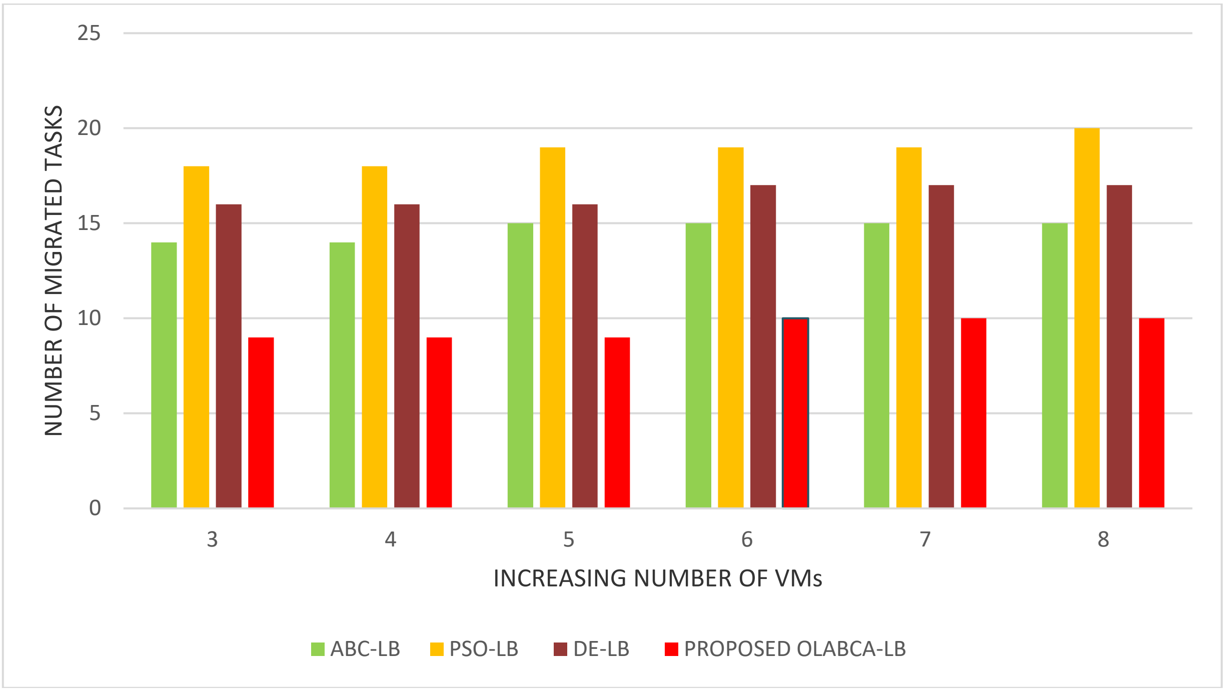


Figure 6: Task Migrations Under Varying VM Count (200 Tasks)

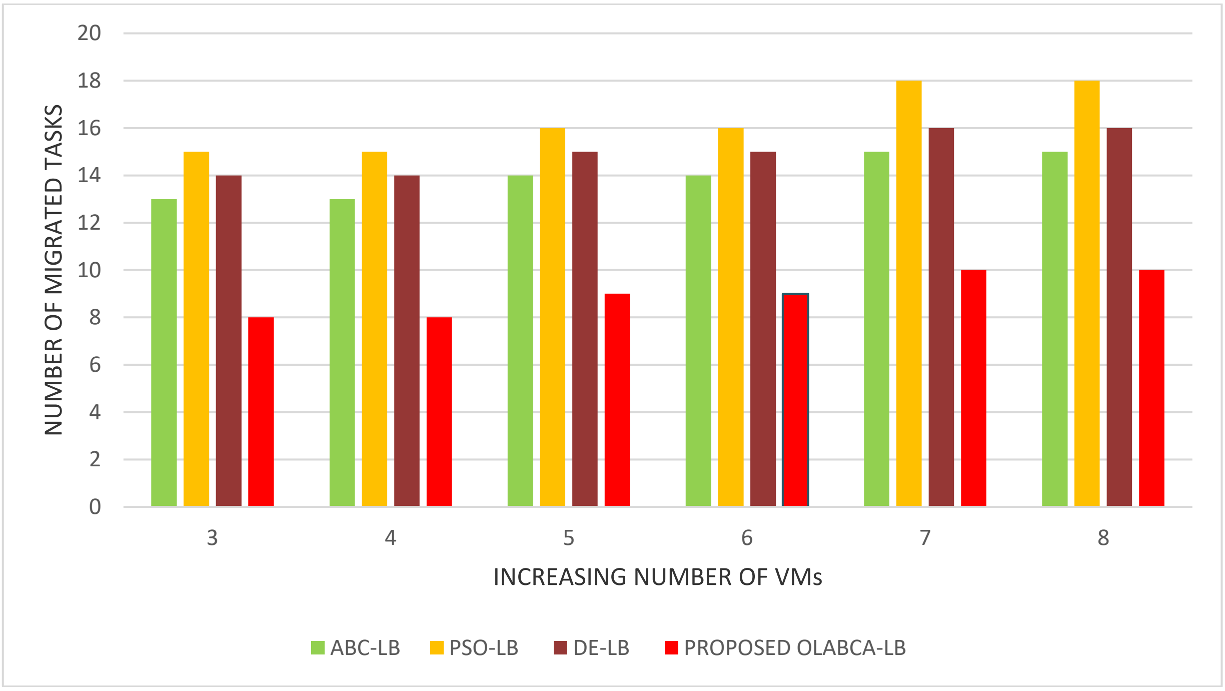


Figure 7: Task Migrations Under Varying VM Count (400 Tasks)

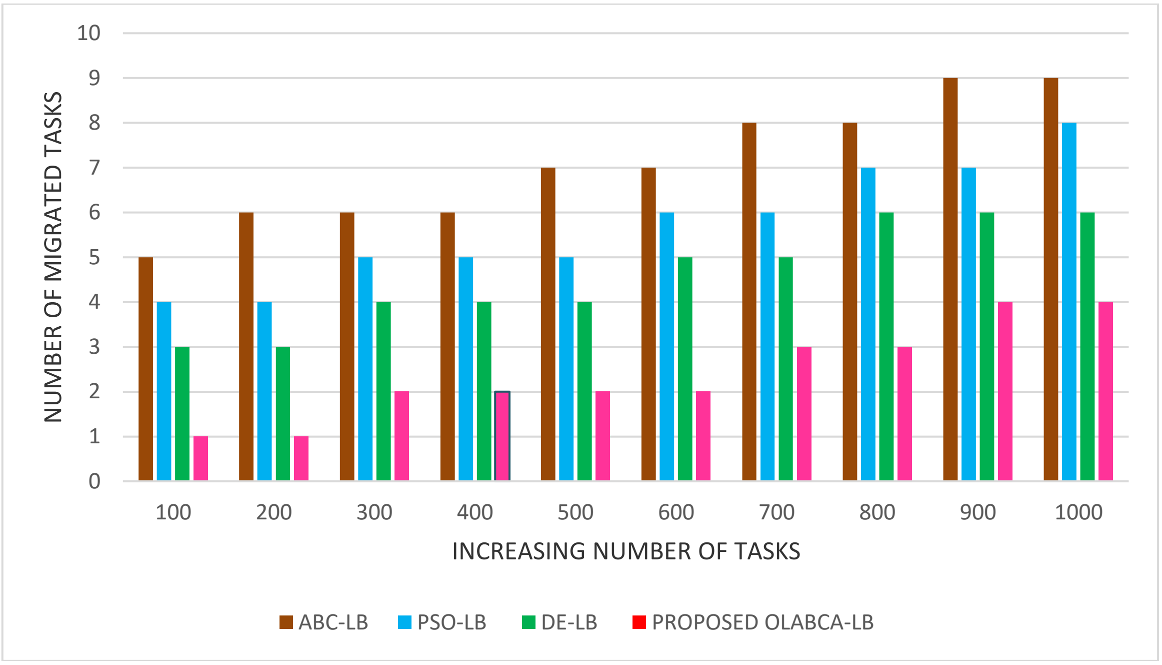


Figure 8: Migration of Tasks with Increasing Task Count (VMs = 2)

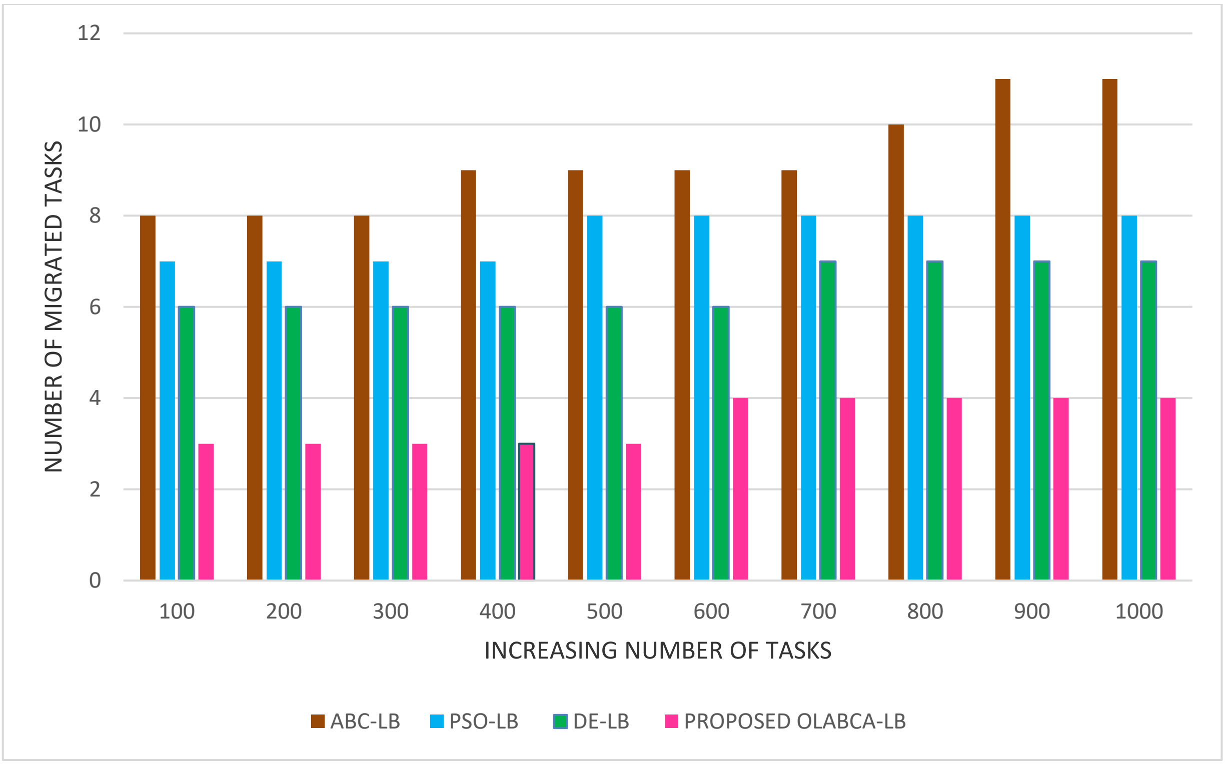


Figure 9: Migration of Tasks with Increasing Task Count (VMs = 4)

To assess the performance of the OLABCA-LB scheme, an analysis was conducted on task migration behavior as the overall task volume increased. Figures 8 and 9 present the results for two configurations, using 2 and 4 VMs, respectively. The findings highlight the OLABCA-LB scheme's ability to substantially lower the number of task migrations across both scenarios, even under growing workload conditions. This notable improvement stems from the scheme's adaptive strategy for allocating and releasing resources, which dynamically reacts to load balancing thresholds. When operating with 2 VMs, OLABCA-LB achieves a reduction in task migrations by 4.77%, 5.08%, and 6.84% contrasted with the ABC-LB, PSO-LB, and DE-LB methods, respectively. The advantage becomes even more pronounced with 4 VMs, where reductions of 5.92%, 6.94%, and 7.86% are observed against the same benchmark algorithms.

CONCLUSION

This proposed work has provided a comprehensive overview of the OLABCA-LB load balancing framework, which integrates Oppositional Learning with the Artificial Bee Colony optimization technique to enable dynamic and adaptive task allocation across hosts and virtual machines. Experimental results evaluating the mean response time across task loads ranging from 100 to 1000 demonstrate that OLABCA-LB consistently outperforms baseline algorithms, reducing response time by 7.86%, 8.06%, and 9.68% when compared to ABC-LB, PSO-LB, and DE-LB, respectively. Furthermore, tests using varying instruction lengths—from 2,000 to 10,000 units—highlight the scheme's effectiveness, yielding response time improvements of 7.62%, 8.04%, and 10.44% against the same set of comparative methods. In terms of mean execution time, as task volume scales from 100 to 1000, the OLABCA-LB approach achieves reductions of 7.94%, 8.04%, and 10.22%, reinforcing its efficiency over ABC-LB, PSO-LB, and DE-LB. In addition, the scheme significantly minimizes task migration. For scenarios where the number of VMs increases from 2 to 10 under different task loads, OLABCA-LB reduces migrations by 5.86%, 6.74%, and 7.62%. Similarly, when the task count is increased from 100 to 1000 under varying VM configurations, the scheme demonstrates further reductions of 3.22%, 4.68%, and 5.31%, showcasing its adaptability and optimization strength relative to existing baseline methods.

REFERENCES

1. A.P. Shameer and A.C. Subhajini (2024) OABC scheduler: a multi-objective load balancing-based task scheduling in a cloud environment, "*Int. J. Advanced Intelligence Paradigms, Vol. 27, Nos. 3/4, 2024*
2. A.P. Shameer and A.C. Subhajini (2019) Quality of Service Aware Resource Allocation Using Hybrid Opposition-Based Learning-Artificial Bee Colony Algorithm. "Journal of Computational and Theoretical Nanoscience Vol. 16, 588–594, 2019
3. Shameer A P, Haseeb V V, Minimol V K, Reshma P K, Aneesh Kumar K(2023) Enhanced Cloud Load Balancing With MPSOA- LB: A Multi-Objective PSO Approach for Dynamic Task Allocation and Performance Optimization," International Journal of Intelligent Systems and Applications in Engineering IJISAE, 2023, 11(1), 445–458
4. Shameer A P and Minimol V K, (2021) ,"An Optimized Cloud Load Balancing Approach Using Hybrid DE-ABC Algorithms". Educational Administration: Theory and Practice 2021, 27(4), 1361-1367
5. Alnusairi T. S., Shahin A. A., and Daadaa Y., (2018), Binary PSOGSA for Load Balancing Task Scheduling in Cloud Environment, *Arxiv Preprint Arxiv:1806.00329*.
6. Boukerche A., Guan S., and De Grande R. E., (2018), A Task-Centric Mobile Cloud-Based System to Enable Energy-Aware Efficient Offloading, *IEEE Transactions on Sustainable Computing*, 3(4), 248–261.
7. Canali C., Chiaraviglio L., Lancellotti R., and Shojafar M., (2018), Joint Minimization of the Energy Costs From Computing, Data Transmission, and Migrations in Cloud Data Centers, *IEEE Transactions on Green Communications and Networking*, 2(2), 580–595.
8. Chaudhary D. and Kumar B., (2018), A New Balanced Particle Swarm Optimisation for Load Scheduling in Cloud Computing, *Journal of Information & Knowledge Management*, 17(01), 1850009.
9. Du J., Zhao L., Feng J., and Chu X., (2018), Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems with Min-Max Fairness Guarantee, *IEEE Transactions on Communications*, 66(4), 1594–1608.
10. Fahim Y., Rahhali H., Hanine M., Benlahmar E.-H., Labriji E.-H., Hanoune M., and Eddaoui A., (2018), Load Balancing in Cloud Computing Using Meta-Heuristic Algorithm, *Journal of Information Processing Systems*,14(3).
11. Gai K., Qiu M., Zhao H., and Sun X., (2017), Resource Management in Sustainable Cyber-Physical Systems Using Heterogeneous Cloud Computing, *IEEE Transactions on Sustainable Computing*, 3(2), 60–72.
12. Acharya J., Mehta M., and Saini B., Particle Swarm Optimization Based Load Balancing in Cloud Computing, *In International Conference on Communication and Electronics Systems)*. IEEE, 2016, 1–4.
13. Ab Wahab M. N., Nefti-Meziani S., and Atyabi A., (2015), A Comprehensive Review of Swarm Optimization Algorithms, *PloS one*, 10(5), e0122827.
14. Abdelmaboud A., Jawawi D. N., Ghani I., Elsafi A., and Kitchenham B., (2015), Quality of Service Approaches in Cloud Computing: A Systematic Mapping Study, *Journal of Systems and Software*, 101, 159–179.
15. A.P. Shameer and A.C. Subhajini (2017) Optimization Task Scheduling Techniques on Load Balancing in Cloud Using Intelligent Bee Colony Algorithm." International Journal of Pure and Applied Mathematics ", Volume 116 No. 22 2017, 341-352
16. Sengathir Janakiraman, M. Deva Priya."Improved Artificial Bee Colony UsingMonarchy Butterfly Optimization Algorithmfor Load Balancing (IABC-MBOA-LB) in Cloud Environments", Journal of Network and Systems Management, Volume 29 , Issue 4 Oct 2021