Educational Administration: Theory and Practice
2021, 27(4), 1361-1367
ISSN: 2148-2403
https://kuey.net/

Research Article

An Optimized Cloud Load Balancing Approach Using Hybrid DE-ABC Algorithms

Shameer A P[1]*, Minimol V K[2]

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

Citation: Shameer A P et al., (2021), An Optimized Cloud Load Balancing Approach Using Hybrid DE-ABC Algorithms, *Educational Administration: Theory and Practice*, 27 (4) 1361-1367
Doi: 10.53555/kuey.v27i4.9763

ARTICLE INFO

ABSTRACT

Cloud computing is a fast-growing emerging field that users can access a diverse services —such as data storage, software applications, and servers—via the Internet. It enables organizations to utilize remote resources provided by various service providers on a pay-as-you-go basis. This model reduces the need for extensive on-site infrastructure, allowing businesses to manage large-scale data and applications virtually through the cloud. Load balancing is the potential process of assigning or allocating the load among the different virtual machines existing in the data center. The workload entering into the cloud computing environment need to be significantly allocated to the resources, such that each share is responsible for sharing an equal amount of loads at any particular point of time. The performance of the cloud environment completely depends on the degree to which the resources are equally shared, since imbalance in load leads to deterioration in the network efficiency. This proposed work is the detailed view of the DE-ABC-Load Balancing (DE-ABCA-LB) scheme presented for effective and efficient load balancing in cloud computing framework. This study presents a mathematical model along with the parameters used to design the fitness function that supports the DE-ABC-LB approach for effective load balancing among virtual machines in a cloud environment. It also details the experimental setup and analyzes the performance of the suggested work DE-ABC-LB method under varying conditions, including different task volumes, instruction lengths, and numbers of virtual machines.

1. INTRODUCTION TO DE-ABC-LB SCHEME

The DE-ABC-LB approach is based on the hybridization of Differential Evolution (DE) and Artificial Bee Colony (ABC) metaheuristic algorithm for mutual elimination of their shortcoming in order to facilitate significant load balancing between virtual machines in cloud environment. The DE-ABCA-LB approach incorporates several parameters in the formulation of its fitness function, including node processing time (HPT) , individual VM computational time , average VM PT, load standard deviation (Load SD), standard normal deviation for VMs, and VM availability. These factors collectively contribute to achieving efficient load balancing. The primary goal of the proposed DE-ABCA-LB method is to ensure equitable workload distribution across various network paths, thereby minimizing the chances of both over-utilization and under-utilization of cloud resources. Fair load balancing is achieved by assessing if the current task count on a virtual server is relatively less than that of other virtual servers in the cloud environment. It also minimizes the computation time of hosts and total response duration by estimating and comparing the PT incurred by VM currently processing the tasks and the mean PT of remaining VMs in order to verify whether the difference is exceeds the permissible limit greater value of load balancing threshold. The load balancing threshold in the proposed DE-ABCA-load balancing methods plays a crucial role in influencing the standard deviation, thereby helping to maintain an effective balance of workloads within the cloud environment.

2. LOAD BALANCING STRATEGIES

The load balancing approaches contributed for cloud computing environment is classified into two based on the current or previous knowledge utilization as depicted in Figure 1.

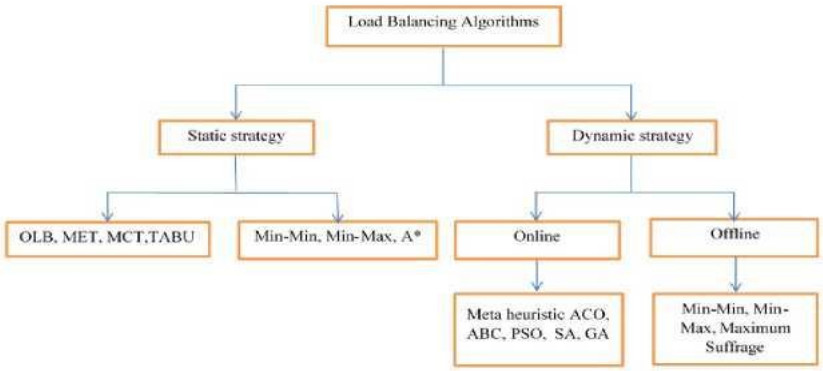

Figure 1: Load balancing strategies

3. THE MATHEMATICAL MODEL OF DE-ABCA-LB SCHEME

In the DE-ABCA-LB scheme, the cloud computing environment comprises of set of VMs with associated tasks for processing. The complete set of virtual machines is considered to be parallel and unrelated with non-pre-emptive independent tasks scheduled to them. In other works, task processing on a virtual machine is uninterruptible. Thus, the proposed model assumes that the failure does not happen. This cloud computing environment is considered to comprise of a collection of data centers, the data centers in turn consists of hosts and each host includes a collection of 'n' Virtual Machines. The individual data center consists of VM LB (Load Balancer). This load balancer is responsible for identifying an appropriate host and suitable VM from the selected host for the objective of allocating the subsequent task by computing some specific metrics. The metrics used for task allocation such PT of host, mean PT of all hosts, PT of Virtual Machines, Mean PT of all Virtual Machines, Load SD, Standard normal deviate of VM and availability of Virtual Machines for the computation of fitness function.

4. PSEUDO CODE THE DE-ABCA-LB APPROACH

This section outlines of the DE-ABCA-LB approach, which is structured around two key functions. The first function is designed to prioritize virtual machines (VMs) to avoid overloading by assigning tasks based on their instruction lengths. This prioritization ensures that tasks are allocated to suitable VMs according to their processing capabilities. The second key function aims to maintain load equilibrium among VMs by implementing an allocation policy that restricts task assignments, thereby preventing excessive load on any single VM. The pseudo code of the proposed approach is given below table.

Step 1: *set i* <—1
Step 2: $Min_{PT} \leftarrow M_{ax\_value}$.Integer (the highest computational time of hosts is defined as the highest integer value recorded)
Step 3: For each entry of host 'i' in State $_{Host\_List}$.
Step 4 When host 'i' is in an available state then
Step 5: If ($Host_{pt(i)} < Host_{pt(min)}$) then
Step 6: Set $Host_{pp\ (MIN)} = Host_{pF\ (i)}$
Step 7: Set i ←*Current_host_count.*
Step 8: End of all if conditional block
Step 9: End of all for loop block.
Step10: Set *j* ←1.
Step 11: $Min_{vm} \leftarrow$ *Max_value.Integer* (To identify the minimum, the number of VMs is first assigned the greatest integer value)
Step 12: For every host 'i' listed in $State_{VM\_List}$.
Step 13: If virtual machine 'j' is active, then
Step 14: If (Min_{VMY} (J) <$VM_{(Min\text{-}count)}$) then
Step 15: Set $Host_{pt(MIN)} = Host_{py}$ (i)
Step 16: Set j←*Current_ VMt_count.*
Step 17: End of all if conditional block
Step 18: for If *j*<—1 then
Step 19: Add the task to the waiting list until one of the VMs becomes available, else
Step 20: Record that they arrived task is assigned to VM (j)
Step 21: Allocation details, including the current processing time of hosts and VMs, as well as the number of tasks being handled, are updated to verify the availability of resources.
Step 22: The task is released from the VM upon completion of its execution.
Step 23: The system updates de-allocation metrics such as the processing time of the host and VM, and the number of active tasks, to reassess resource availability.
Step 24: End If

5. PERFORMANCE ANALYSIS

The proposed DE-ABCA-LB approach was evaluated through simulation experiments using CloudSim, a widely adopted framework for modeling and modeling task scheduling in extensive cloud computing systems. In this study, CloudSim was utilized to replicate the allocation and management of computing resources and virtual machines, allowing for a comprehensive assessment of the DE-ABCA-LB scheme's effectiveness. The simulation environment was configured with 25 virtual machines, 15 data centers, and task loads ranging from 50 to 1000. Task lengths used in the evaluation varied between 2000 and 10000 Million Instructions (MI). Additionally, the key parameters utilized in the simulation of the DE-ABCA-LB approach are summarized in Table 1.

Type	Parameter	Value
Data Center	VM Scheduler	Time-Shared
	Number of Hosts	02-04
	Number of Data Centers	10
Virtual Machine (VM)	Cloudlet Scheduler	Time-Shared
	Bandwidth	500-1000
	Required Number of Processor Elements	01-02
	Speed of the Processor	4000-2000 MIPS
	Number of VMs	50
	Memory Space Available in Each VM	256-2018 Mb
Cloudlets (Tasks)	Number of Tasks	100-1000
	Task Length	2000-20000

Table 1: Simulation Parameters Used for proposed approach

This section examines the performance efficiency of the DE-ABCA-LB approach through a comprehensive set of analytical evaluations, emphasizing the following key areas:

i) average response time across different task volumes and instruction lengths,
ii) comparison of average response time, instruction length, and execution time against traditional algorithms,
iii) analysis of the same metrics in comparison with swarm intelligence-based algorithms,
iv) Task migration count increases with more virtual machines when the total number of tasks remains the same
v) task migration count as the number of tasks increases with a fixed number of virtual machines.

5.1 Performance Evaluation Using Mean Response Time with Task Varying

This section investigates the effectiveness of the DE-ABCA-LB approach by examining the mean response time across varying numbers of tasks and instruction sizes within a cloud infrastructure. Figures 2 and 3 illustrate the performance of the DE-ABCA-LB method, showcasing the mean response time (in seconds) as the "number of tasks and instruction" sizes (in bytes) change. When the instruction length is set to 2000, the "mean response time" of the proposed scheme increases from 7.24 seconds to 19.21 seconds as the task count rises from 100 to 1000. Likewise, the mean RT of the DE-ABCA-LB approach with executable instruction length of 8000 is proved to increase from 7.46 seconds to 40.64 seconds with increasing tasks. Further, the mean RT of the DE-ABCA-LB approach with "executable instruction" length of 14000 is proved to increase from 7.82 seconds to 71.24 seconds with increasing tasks. Furthermore, when the "executable instruction" length is set to 20,000, the mean RT of the proposed work is observed to rise consistently from 8.12 seconds to 98.42 seconds as the number of tasks steadily increases.
nder

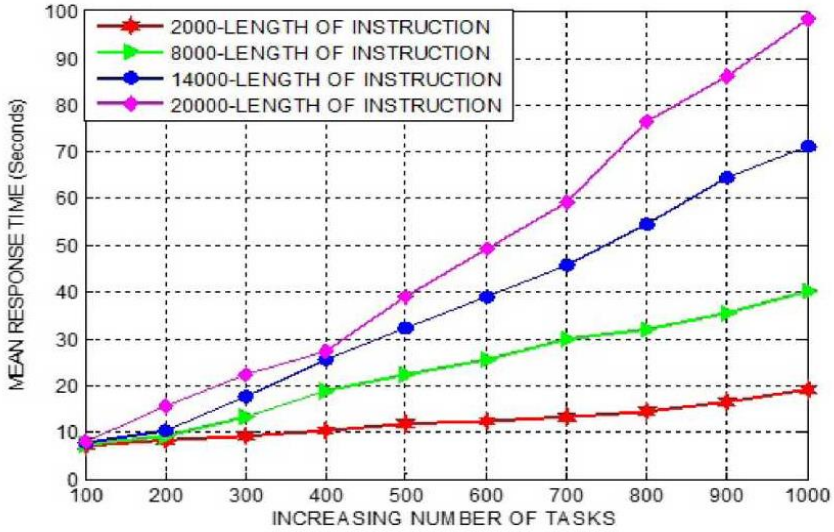

Figure 2: Mean RT under Varying #of Task

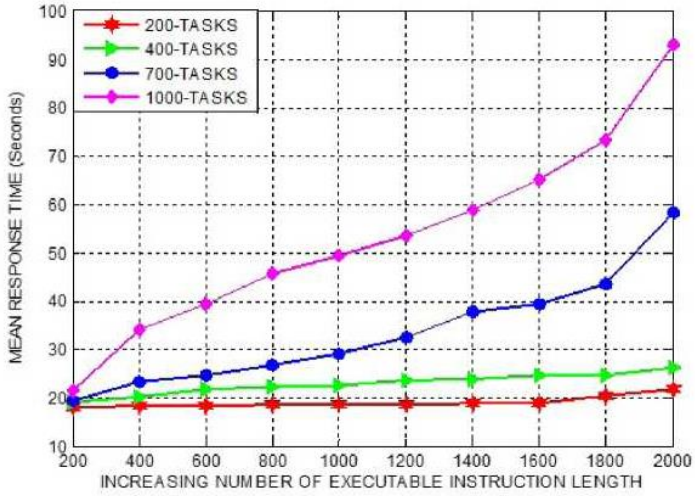

Figure 3: Mean RT under variety Execution Instruction Length

This increase in the response time under different executable instruction length independent to the increase in the tasks submitted to the workflow is mainly due the significant increase in the system load. Figure 3 illustrates the trend in average "response time" for the proposed work as the number of executable instructions increases. For 200 tasks, the response time rises from 18.42 seconds to 21.76 seconds as the instruction length ranges from 200 to 2000. The number of tasks is increased to 400, the response time similarly grows, starting at 19.21 seconds and reaching 25.12 seconds. With 700 tasks, a more substantial increase is observed, with response times escalating from 18.44 seconds to 58.42 seconds. In the case of 1000 tasks, the delay becomes even more pronounced, growing from 21.32 seconds to 91.28 seconds. This consistent increase in response time, irrespective of the task volume, is attributed to the design of the DE-ABCA-LB scheme, particularly its policies for resource allocation and de-allocation, as well as its configured threshold parameters.

6. COMPARATIVE STUDY WITH TRADITIONAL SCHEMES

This section presents a comparative evaluation of the DE-ABCA-LB scheme against three widely-used load balancing strategies: Dynamic Load Balancing (DLB), "Throttled Load Balancing Algorithm (TLBA)", and "Weighted Round Robin (WRR)". The analysis is based on two key parameters—task count and executable instruction length—as both significantly affect system performance. Task volume is incremented from 200 to 2000, with corresponding "instruction lengths" ranging from 1000 to 10000. A threshold value of 0.1 is used, as this point yielded the lowest standard deviation during performance evaluation. As illustrated in Figure 4, the "mean response time" across all approaches is analyzed for varying task loads. The DE-ABCA-LB method demonstrates superior performance, achieving reductions in mean response time of approximately 11.32%, 13.33%, and 14.52% when evaluated against DLB, TLBA, and WRR, respectively.

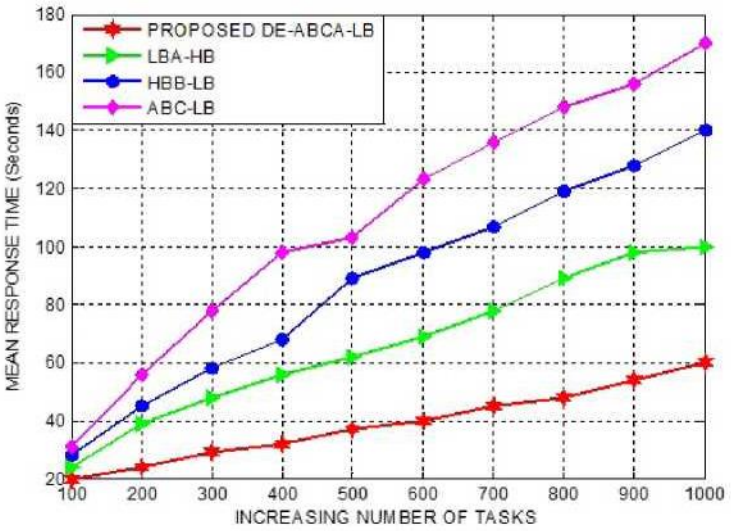

Figure 4: Mean RT under Increasing # of Tasks

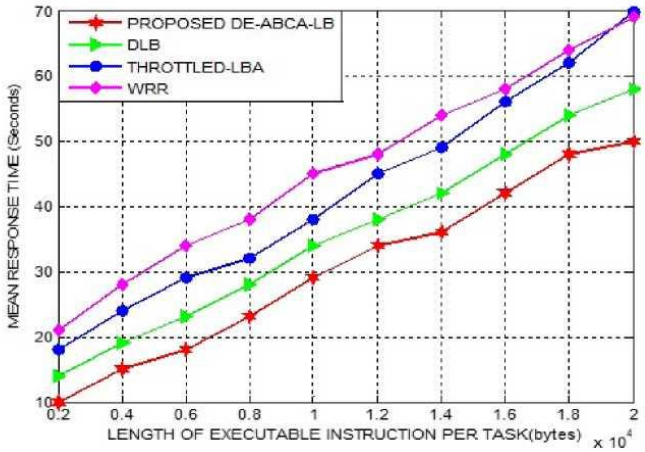

Figure 5: Mean RT under Different Executable Instruction Length

The enhanced performance in mean response time achieved by the DE-ABCA-LB algorithm is largely due to its capacity to adapt to varying loads across virtual machines (VMs), respond to dynamic changes in VM availability, and efficiently allocate tasks to the least burdened VMs. As shown in Figure 5, the average response times of DE-ABCA-LB, DLB, Throttled-LBA, and WRR are compared across increasing executable instruction lengths ranging from 1000 to 10000. The findings indicate that DE-ABCA-LB consistently outperforms the other methods, lowering response time by approximately 11.41%, 12.41%, and 13.61% relative to DLB, Throttled-LBA, and WRR, respectively. This performance gain can be credited to the algorithm's ability to prevent both overloading and underutilization of VMs. Furthermore, as depicted in Figure 6, the average execution time for the same set of algorithms is evaluated with task counts ranging from 100 to 1000. The DE-ABCA-LB technique consistently outperforms the alternatives, achieving execution time reductions of 12.11%, 13.47%, and 15.21% over DLB, Throttled-LBA, and WRR, respectively. This enhanced performance is largely due to the scheme's efficient use of computational resources, enabling a more balanced and optimized distribution of workload across available VMs.

7. COMPARATIVE INVESTIGATION WITH DIFFERENT SCHEMES

The DE-ABCA-LB approach is evaluated against three swarm based load balancing algorithms: "Honey Bee-based Load Balancing (LBA-HB)", "Honey Bee Behavior-inspired Load Balancing (HBB-LB)", and the ABC-LB approach. The analysis considers varying numbers of tasks, ranging from 200 to 2000, and "executable instruction lengths" from 1000 to 10000, with a threshold of 0.1. Figure 6 presents the average response times for "LBA-HB, HBB-LB, and ABC-LB" under these conditions. The DE-ABCA-LB approach demonstrates an ability to reduce mean response times by up to 11.22%, 12.56%, and 13.78% relative to the "LBA-HB, HBB-LB, and ABC-LB" schemes, respectively. This improvement is largely attributed to the efficient allocation of requests to virtual machines, which ensures that the variance in processing times remains within a predefined limit.

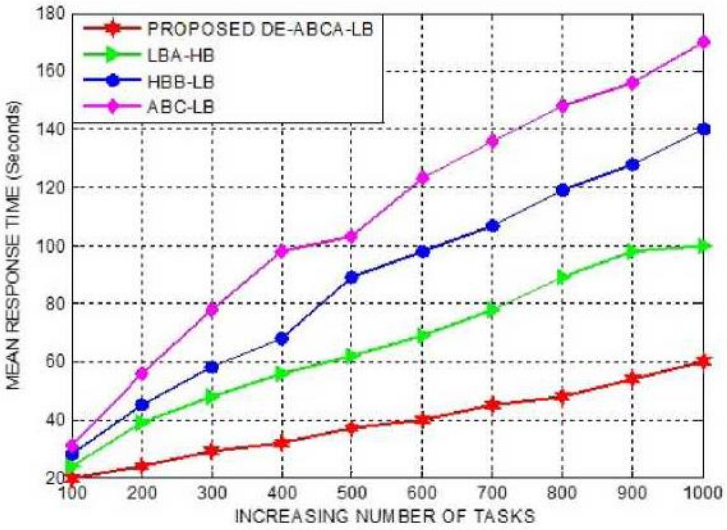

Figure 6: Mean RT under Increasing #Tasks

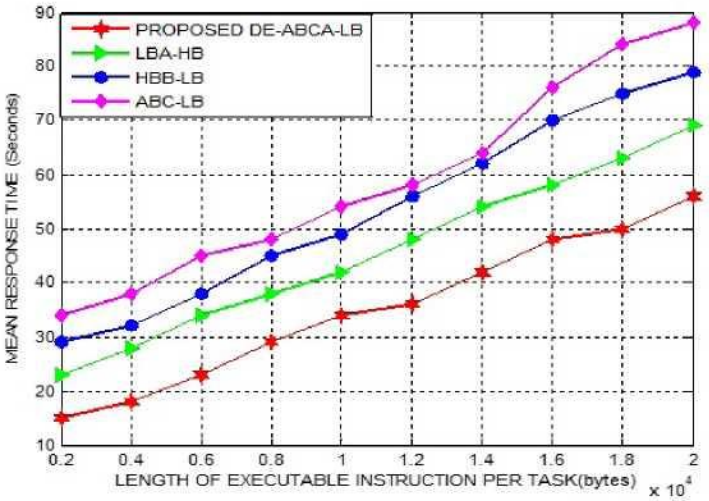

Figure 7: # executable instruction length varying

Figure 7 illustrates the average response times for the "DE-ABCA-LB, LBA-HB, HBB-LB, and ABC-LB" algorithms, as the executable instruction length ranges from 1000 to 10000. The DE-ABCA-LB consistently outperforms the others, with response time reductions of approximately 9.02%, 10.12%, and 12.74% compared to LBA-HB, HBB-LB, and ABC-LB, respectively. This advantage stems from its ability to minimize load distribution fluctuations across virtual machines, thus reducing variability in VM utilization. This efficiency is due to its adaptive threshold mechanism, which helps ensure a balanced workload distribution by controlling load variations across VMs.

8. CONCLUSION

This study introduced the DE-ABCA-LB load balancing method, which combines the strengths of Differential Evolution (DE) and Artificial Bee Colony (ABC) methods to efficiently allocate virtual machines (VMs) and hosts for incoming tasks in a cloud environment. The DE-ABCA-LB scheme exhibits significant advantages in reducing mean response time. When tested on task volumes ranging from 100 to 1000, it achieved reductions in response time of 8.02%, 10.12%, and 12.84% with respect to the LBA-HB, HBB-LB, and ABC-LB schemes, respectively. Similar positive results were obtained when the executable instruction length varied from 1000 to 10000, underscoring the scheme's stable performance across different workloads. Additionally, the scheme significantly improved mean execution time, outperforming the LBA-HB, HBB-LB, and ABC-LB algorithms by 11.54%, 12.36%, and 14.64%, respectively, as task volumes increased from 100 to 1000. In terms of task migration efficiency, DE-ABCA-LB showed notable effectiveness. With the number of VMs increasing from 2 to 10, the scheme reduced task migrations by 5.52%, 5.81%, and 6.84% compared to LBA-HB, HBB-LB, and ABC, respectively. Under task volumes ranging from 100 to 1000 with a stable number of VMs, task migrations decreased by 3.85%, 4.68%, and 5.32%, respectively. These outcomes demonstrate the robustness and flexibility of the DE-ABCA-LB scheme in managing load distribution and minimizing overhead in dynamic cloud environments.

REFERENCES

1. **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*.
2. **Boukerche A.**, **Guan S.**, and **De Grande R. E.**, (2018), A Task-Centric Mobile Cloud-Based System to Enable Energy-Aware E*ffi*cient O*ffl*oading, *IEEE Transactions on Sustainable Computing*, **3**(4), 248–261.
3. **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.
4. **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.
5. **Du J.**, **Zhao L.**, **Feng J.**, and **Chu X.**, (2018), Computation O*ffl*oading and Resource Allocation in Mixed Fog/Cloud Computing Systems with Min-Max Fairness Guarantee, *IEEE Transactions on Communications*, **66**(4), 1594–1608.
6. **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).
7. **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.
8. **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.
9. **Ab Wahab M. N.**, **Nefti-Meziani S.**, and **Atyabi A.**, (2015), A Comprehensive Review of Swarm Optimization Algorithms, *PloS one*, **10**(5), e0122827.
10. **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.
11. 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
12. 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*
13. 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
14. 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