International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I9P204

Volume 71 Issue 9, 36-45, September 2023

Original Article

An Adaptive Load Balancing using Particle Swarm Optimization for Cloud Task Scheduling

Chaitanya Udatha[1], Gondi Lakshmeeswari[2]

[1,2]*Department of Computer Science and Engineering, GITAM (Deemed to be University), Andhra Pradesh, India.*

[1]*Corresponding Author : chaitanyasreeram15@gmail.com*

Received: 20 May 2023 Revised: 29 July 2023 Accepted: 15 August 2023 Published: 03 September 2023

Abstract - With *cutting-edge services available via subscription, the usage of cloud technology is rapidly increasing in daily life using advanced services. Efficient management of services and delivering them on demand require proper scheduling of resources and requests. Load balancing aware scheduling techniques are employed to accomplish this, which distribute requests uniformly among resources and optimize resource utilization. In cloud computing, load balancers are essential for balancing workloads on resources. They ensure even workload distribution across all resources by transferring workloads from overloaded to underloaded resources. The proposed Load Balancing Improved Multi-Objective Particle Swarm Optimization (LBIMOPSO) technique aims to manage load uniformly and allocate tasks to the best-suited virtual machines. It is a robust optimization technique that considers multiple objective functions simultaneously and effectively balances workloads in a cloud computing environment. However, according to an existing survey, there is an improvement in makespan performance compared to proper load balancing across virtual machines. Therefore, the proposed LBIMOPSO algorithm improves resource utilization, makespan, and load balance deviation compared to traditional swarm-intelligence-based ant colony and particle swarm optimization algorithms.*

Keywords - *Cloud computing, Improved particle swarm optimization, Load balance deviation, Makespan, Resource utilization.*

1. Introduction

In the present era, the cloud plays a pivotal role in academics, businesses, and industries—the cloud pools resources from data centers in various regions worldwide [1]. The resources, including high-performance computing elements with storage, networking devices, databases, development environments, and applications, are provided as three essential cloud services: infrastructure, platform, and software. Through the cloud environment, users can leverage computing resources, comprising software applications, data storage, and processing power, provided as a service through the internet. A third-party provider typically maintains and manages the cloud infrastructure, enabling customers to increase or decrease their resources per their requirements. The essential characteristic of the cloud environment is its virtual nature. The Cloud Service Provider (CSP) delivers services on a subscription basis to the cloud users. Properly managing cloud services is a crucial task for CSPs, leading to increased attention towards proper scheduling and managing load to harness the beneficial features of cloud computing fully. The scheduling of tasks involves selecting the appropriate Virtual Machine (VM) for execution while meeting quality of service constraints, such as energy consumption, response time, resource utilization, and makespan, while adhering to service level agreements [2,3]. The performance of the cloud is enhanced via load balancing, leading to satisfaction for the cloud end user and provider, as it generates economic benefits for both parties. The research gap in cloud computing lies in effectively integrating and synchronizing task scheduling and load balancing, two vital aspects of cloud resource management. Although closely related, they represent distinct dimensions for optimizing cloud performance and resource utilization. A crucial area of research is the seamless fusion of load balancing and task scheduling algorithms. Load balancing ensures an even workload distribution among available resources to avoid bottlenecks and overloading. At the same time, task scheduling focuses on efficiently assigning tasks to resources for optimal completion time and resource usage. Integrating these aspects can lead to a more comprehensive approach, considering both workload distribution and task assignment, ultimately enhancing overall performance.

Servers, applications, and networks can perform load balancing at different levels based on the types and locations of resources being managed [4]. When a user submits a task, the task requires computational resources to execute the request. However, before mapping the task to a VM, the load is divided uniformly across the VMs. The classification of task allocation to VMs into two levels.

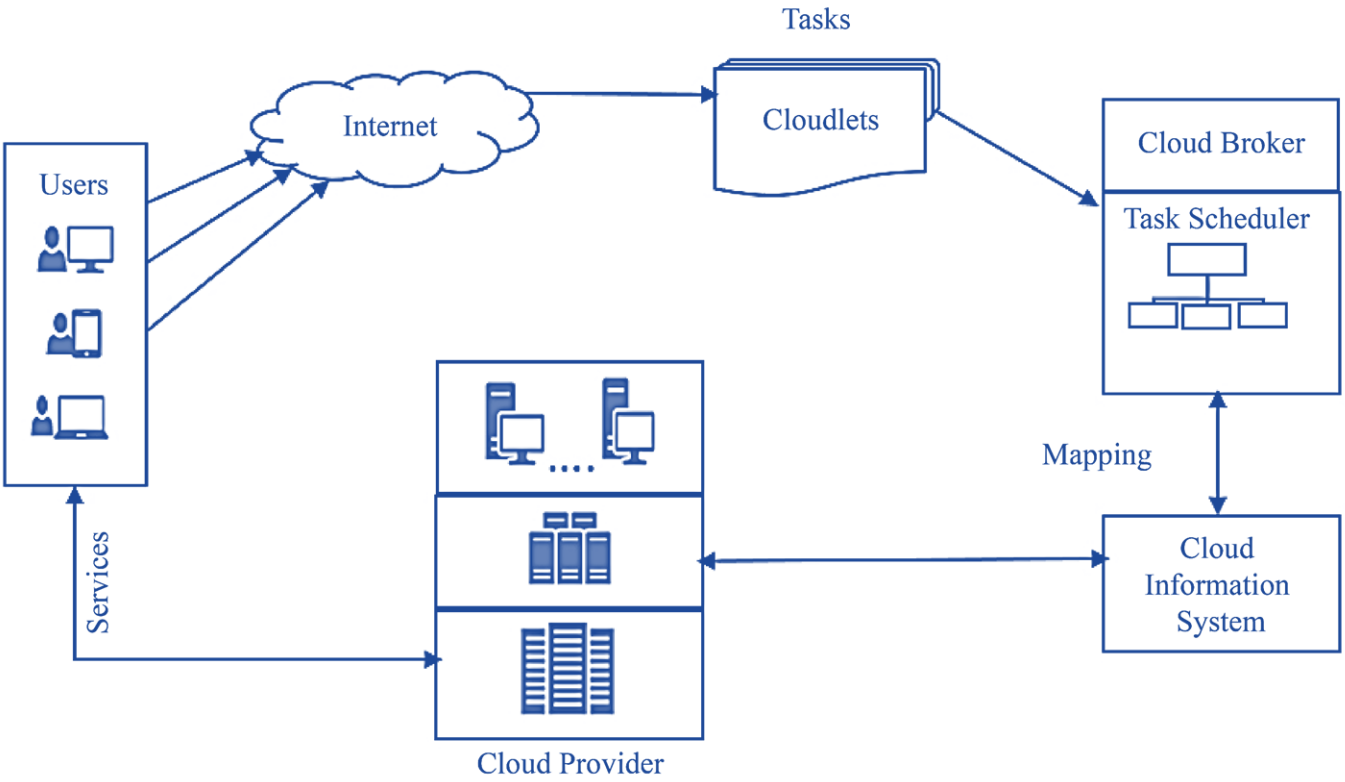

Fig. 1 Load balancing based task scheduling model

The primary level involves assigning tasks to VMs on different host machines based on their load. The HDDB technique distributes VMs to the most appropriate hosts based on membership and CPU usage value [5]. In this approach, VMs migrated from one host to another. The next level of load balancing occurs within a host among VMs, i.e., tasks are scheduled to suitable virtual machines depending on the VM load, and tasks move from heavily loaded VMs to underloaded ones. The proposed algorithm falls under the second level but without migrating tasks from overloaded to underloaded resources while balancing the tasks on each VM during the scheduling process. Task-level load balancing typically has a lower overhead than VM-level load balancing because it does not require creating and managing additional VMs. Task-level load balancing can optimize the utilization of resources by allocating only the necessary resources to each task. In contrast, VM-level load balancing may result in overprovisioning or under-provisioning of requests for each VM.

The significant contribution of the proposed research work is as follows:

- Tasks are heterogeneous, independent, and non-preemptive. The sequential scheduling technique executes tasks on the same VM when multiple tasks are assigned.
- The proposed load balancing aware scheduling using the improved PSO algorithm for heterogeneous cloud environments to address multi-objective optimization, i.e., maximum resource utilization and minimization of makespan and load balance deviation.
- The probability-based fitness function was incorporated into the individual influence component to attain the optimal global solution.
- The LBIMOPSO algorithm's effectiveness is evaluated compared to conventional ACO and PSO algorithms.

The remaining part of this paper is organized into four sections as follows:

Section II represents a survey on scheduling and load-balancing strategies and discusses a load-balancing-based scheduling environment. Section III explains the problem formulation and parameter estimation of the proposed LBIMOPSO approach. Section IV presents the simulation results of the proposed LBIMOPSO algorithm to the traditional swarm intelligence-based ant colony and particle swarm optimization algorithms using the cloudSim framework. Finally, Section V concludes the proposed work and future direction improvements.

2. Related Work and Background

In cloud environments, a wide range of static and dynamic algorithms have been proposed by researchers; these algorithms utilize various techniques such as heuristics, meta-heuristics, and combined approaches to ensure the provision of high-quality service. Meta-heuristic algorithms

have gained popularity due to their ability to provide solutions in large search spaces and solve complex problems [6]. The following survey focuses on examining popular evolutionary and swarm-intelligence algorithms.

Using the BGA algorithm, Gulbaz, Rohail, et al. proposed to search for the best global solution in a solution space based on genetic principles [7,8,9]. The algorithm considers the quality-of-service parameters to achieve Service Level Agreements (SLAs) but needs improvement regarding task priority and dependency.

Nabi, Said, et al. used various inertia weight strategies for PSO to improve the load balancing of tasks [10,11]. However, all task instances should have an improved resource utilization ratio.

RATSA algorithm is a task scheduling approach using reliability to reduce network failure rates [12]. It is a two-level process, identifying the most utilized host and scheduling tasks to VM on that host based on the probability of fitness function.

Elmagzoub, M. A. et al. provide surveys on various swarm intelligence algorithms [13]. Researchers can extend their work on load balancing using these algorithms by considering the pros and cons of each algorithm. The whale optimization method uses Multiple objective fitness values to balance the load, but only two objectives are satisfied, and the operation cost objective needs to perform better [14,15].

An ACO (Ant Colony Optimization) with a SWIM approach was proposed for scheduling and load balancing [16]. This approach combines three parameters: CPU, memory, and bandwidth utilization rate, which is considered a load-balancing factor among virtual machines. A higher value of load deviation indicates more imbalance in the load, but it does not consider the makespan parameter for comparison.

Numerous studies on scheduling strategies in cloud computing using different meta-heuristics optimization strategies, along with their benefits and drawbacks, are discussed [17,18,19]. Researchers can enhance their exploration of future research directions by carefully considering the challenges identified in existing surveys. Hosseinzadeh, Mehdi, et al. proposed a levy flight principle-based BOA algorithm to improve algorithm efficiency and avoid local optima and fast convergence problems [20]. However, it is a single-objective algorithm used for placing data in edge computing, and it can be improved to handle workflow scheduling with multiple objectives.

The primary mechanism of load balancing with task scheduling is depicted in Figure. 1. The users can submit their requests to the cloud through any heterogeneous smart device connected to the internet. These requests are called tasks or cloudlets in the cloud environment. The cloud broker processes the tasks through the cloud information system to the requested resources in the data center. The cloud broker implements an effective scheduling algorithm to map the requests to the resources. Load balancing with scheduling employs a robust algorithm to uniformly allocate requests to available resources by managing enterprise demand. It is essential for accommodating resources on demand and maximizing the utilization of available resources.

Load balancing is crucial for applications that need to scale and be flexible as they grow. It prioritizes tasks that require immediate execution and those that need more time compared to other jobs. With balanced workloads, CSPs can optimize cloud services for economic benefits and user satisfaction. Load balancing aims to distribute tasks evenly among available resources to optimize resource utilization and reduce system response time. The task can get the necessary resources using the proposed LBIMOPSO algorithm; otherwise, beyond their needs results in resource waste. Load balancing is one of the primary objectives of task scheduling by considering a load parameter.

3. Proposed Methodology

3.1. Improved PSO Model and Description

PSO is a nature-inspired self-adaptive algorithm based on the social behaviours of animal groups or bird flocks to carry out a global search for food sources [21]. The bird searching for food with the flock randomly can improve its search, while sharing information within the group improves its chances of survival. The benefit of sharing the best information can aid a flock in locating the finest hunting grounds. The PSO algorithm comprises the number of particles and generations, where generations represent the iterations needed to explore search space to achieve an optimal solution. Every generation in the swarm comprises a group of particles, each representing a specific solution. Each generation particle possesses its unique position, traveling velocity, personal best, global best, and inertia value. The inertia value is used to govern particle movement. At the same time, personal and global best represent the best solutions locally and globally across the swarm of particles.

During each iteration, particle velocity and position are updated based on factors such as inertia weight and the values of pBest and gBest, which are adjusted based on the optimal fitness value. $X_i^{(t)}$ denotes the position vector of particles within a multidimensional search space while $U_i^{(t)}$ representing their velocity vector.

The traditional formulas for the PSO algorithm are as follows [22]:

$$U_i^{(t+1)} = \omega_i * U_i^{(t)} + c_1 * r_1 * \left(X_{pbesti}^{(t)} - X_i^{(t)} \right) + c_2 * r_2 \left(X_{gbesti}^{(t)} - X_i^{(t)} \right) \quad (1)$$

$$X_i^{(t+1)} = X_i^{(t)} + U_i^{(t+1)} \quad (2)$$

PSO is popular due to algorithm simplicity and ease of understanding, fast convergence, quick implementation, and very few parameters, but it is easily trapped into local optima. So, improved PSO is an ideal solution for efficiently scheduling tasks while considering the uniform load on each resource [23].

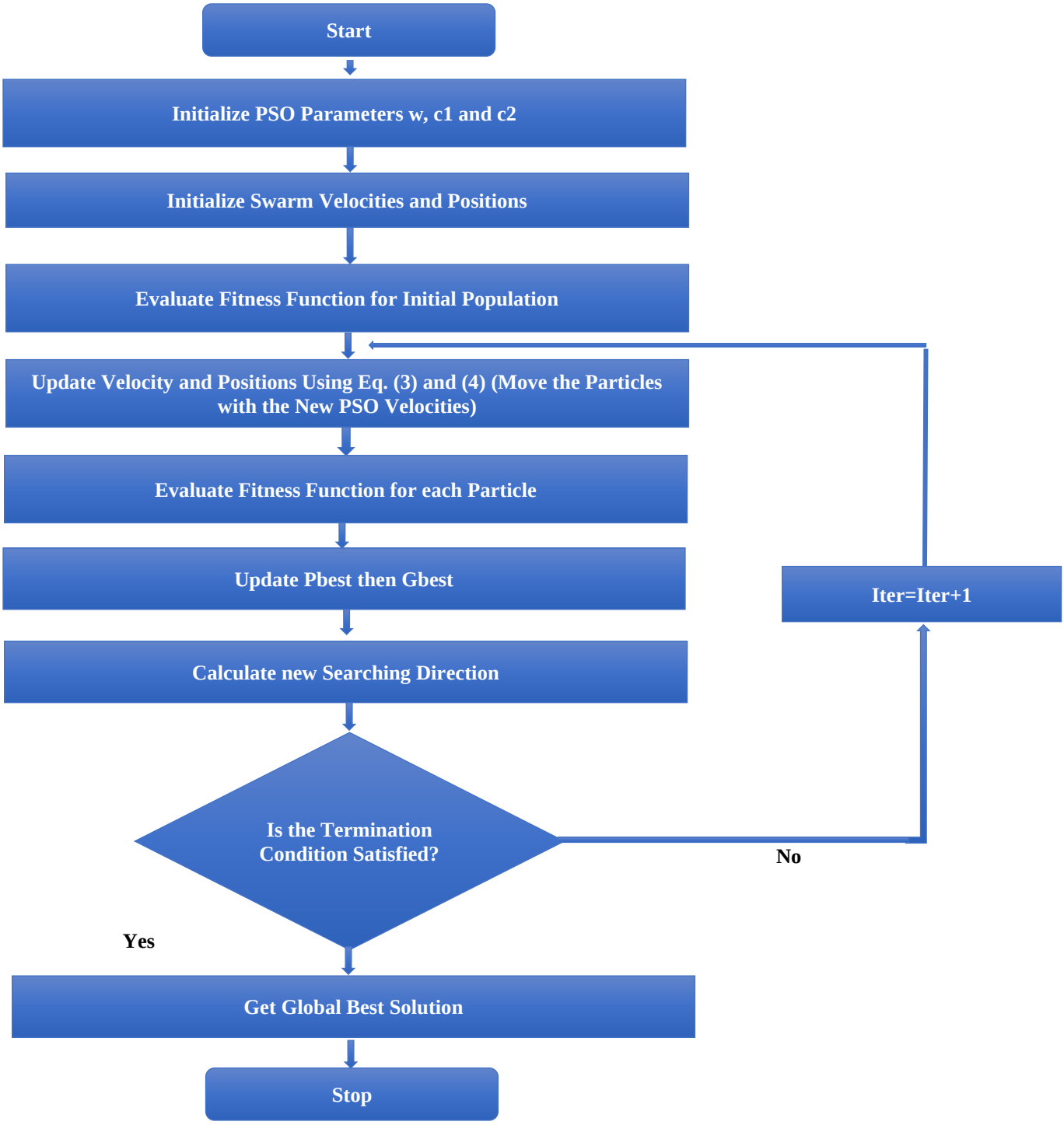

Fig. 2 Improved PSO algorithm

The formulas for the improved PSO algorithm are as follows:

$$U_{i_{new}}^{(t+1)} = \omega_i * U_i^{(t)} + (1 - P_i) * c_1 * r_1 * \left(X_{pbesti}^{(t)} - X_i^{(t)} \right) + * c_2 * r_2 \left(X_{gbesti}^{(t)} - X_i^{(t)} \right) \quad (3)$$

$$X_{i_{new}}^{(t+1)} = X_i^{(t)} + U_{i_{new}}^{(t+1)} \quad (4)$$

$$\omega_i = \frac{Iter_{max} - Iter_i}{Iter_{max}} \quad (5)$$

ω_i indicates inertia weight decreases with the iteration number to explore search space more efficiently.

$$P_i = \frac{Fit_{gbest,i}}{\sum(Fit_{lbest,i})} \quad (6)$$

P_i indicates the probability of global fitness to local fitness to avoid the problem of local trap [24]. The improved PSO algorithm is represented in the Figure. 2.

3.2. Task Scheduling Problem Description and Definitions

The term cloud represents a repository of virtual resources; it consists of data centers located in different regions; each data center consists of numerous hosts and, in turn, consists of various flavors of Virtual Machines (VMs) based on the choice of customers. These resources are provided to the users on a subscription basis. When multiple requests arrive at a data center, the number of resources is always less than the user requests. The cloud broker efficiently maps the task to a suitable resource machine to optimize resource utilization using a scheduling algorithm. The optimized way of distributing N requests to M resources is an NP-hard problem.

The scheduling environment is depicted in Figure 1, and user requests are generated from heterogeneous client devices called tasks in the cloud environment. The cloud broker mediates between the tasks and resources through an information system. Cloud resources are represented as virtual machines, and their unit is generally represented in a Million Instructions Per Second (MIPS). Each task is represented by a task ID and its corresponding length. All tasks are considered independent, and every task will be executed on only one VM.

The processing time of each task on every resource must be determined, represented in the ET matrix, to establish the optimal scheduling strategy for N tasks ({T1, T2, …Tn}) on M VMs ({VM1, VM2, … VMm}).

$$ET = \begin{bmatrix} \frac{T_1}{VM_1} & \cdots & \frac{T_1}{VM_m} \\ \vdots & \ddots & \vdots \\ \frac{T_n}{VM_1} & \cdots & \frac{T_n}{VM_m} \end{bmatrix}$$

The overall task execution time is calculated using equation (7), in which T_{Len_i} denotes the i[th] task length in MI (million instructions) and is divided by the computing power of j[th] VM.

$$f_1 = TET_{ij} = \sum_{ij}^{nm} \frac{T_{Len_i}}{VM_{MIPS_j}} \quad (7)$$

The makespan denotes the latest task completion time on PT_{VM_i}, which represents the processing time of the VM.

$$f_2 = Make\ span = max\{PT_{VM_1}, PT_{VM_2}, .., PT_{VM_m}\} \quad (8)$$

Table 1. Parameters of scheduling environment

LBIMOPSO Algorithm / Cloud Parameters	Initial Values / Values Range
Population	50
Iterations	1000
ω	0.9
α	0.05
β	0.7
γ	0.25
c1	2.0
c2	1.49
r1, r2	[0,1]
Tasks	100-500
Task Size	1100 MI -2000 MI
VMs	5
VM Processing rate	500 MIPS-900 MIPS

The DI is the degree of VM imbalance, is determined by computing the disparity between the maximum and minimum processing times of VMs and the average execution time as indicated in equation (9)

$$ATET = \frac{TET_{ij}}{m} \quad (9)$$

$$f_3 = DI = \frac{f_2 - min\{PT_{VM_1}, PT_{VM_2}, ..., PT_{VM_m}\}}{ATET} \quad (10)$$

The load balancing measure is determined by applying equation (12), in which PT_{VM_j} represents the sum of the processing times of all virtual machines.

$$PT_{VM_j} = \sum_{i=1}^{n} PT_{ij} \quad (11)$$

$$f_4 = LB = \sqrt{\frac{\sum_{j=1}^{m} \left(PT_{VM_j} - ATET \right)^2}{m}} \quad (12)$$

The load balancing value impacts the makespan and tends to increase as the number of tasks increases [25]. Table. 1 list the parameters of the improved PSO algorithm and cloud simulation environment. These parameters are used in the LBIMOPSO algorithm to schedule the tasks of VMs using

3.3. Proposed LBIMOPSO Algorithm

The Load Balancing based Improved Multi-Objective Particle Swarm Optimization (LBIMOPSO) is suggested for self-adaptive load balancing based on scheduling the user tasks on a heterogeneous cloud environment to maximize resource utilization and balance the load uniformly on each resource to avoid overprovision and under-provision of resources during the scheduling process. This section introduces a multi-objective improved PSO algorithm for load balancing and task scheduling.

The proposed LBIMOPSO algorithm is an extension of MOHIPSO, and it is suggested for scheduling tasks with a uniform load on heterogeneous VMs in a cloud environment. To improve resource utilization and decrease the load balancing deviation among VMs. The proposed algorithm is shown below.

Algorithm 1: Pseudocode for LBIMOPSO

```
Input:  Heterogeneous type of N Tasks and M VMs
Output: Load on each VM is balanced equally based on
        MIPS
Start:
Sorting Tasks in ascending order and Vms in descending order
Calculating execution time matrix, i.e. Task vs VM
Initialize the particle's position and velocity randomly
For i: 1 to max iterations
   For j: 1 to population size
     calculate the fitness function using (13)
Locate the local best and modify the particle's velocity accordingly by (3)
     Update the particle's position using (4)
   End
  Find the updated optimal personal best(pbest_i)
End
Output the swarm global best with the positioning of tasks on Vms with uniform load
End
```

In the proposed scheduling algorithm, the tasks are arranged in ascending order, VMs in descending order and calculate the execution time matrix. Cloud broker schedules tasks on VMs using an improved PSO algorithm. In this algorithm, each particle position is initialized randomly with VM number, and velocity is randomly assigned between the value [0,1] to start the search in the Solution space. In each iteration, the velocity and position of the particle are updated based on a multi-objective fitness function to explore the search space.

The equations (7) to (12) are considered for constructing the multi-objective fitness function, as represented in equation (13).

Table 2. Task load on virtual machines

Task Load % on VM					
No. of Tasks	**VM No.**	**VM MIPS**	**PSO**	**ACO**	**LBIMOPSO (Proposed)**
100	0	500	120.87	108	99.59
	1	600	119.5	98	99.48
	2	700	104.6	95.5	99.56
	3	800	80.69	93.6	99.62
	4	900	73.17	102	99.54
200	0	500	125.11	110	99.47
	1	600	111.67	104	99.62
	2	700	103.92	101	99.51
	3	800	83.21	93.6	99.43
	4	900	74.68	88.9	99.54
300	0	500	131.15	97	99.47
	1	600	113.73	108	99.62
	2	700	95.22	104	99.51
	3	800	82.15	103	99.43
	4	900	76.14	86.9	99.54
400	0	500	135.98	108	99.43
	1	600	109.09	108	99.52
	2	700	95.51	90.7	99.51
	3	800	82.83	92	99.5
	4	900	74.94	99.8	99.55
500	0	500	134.23	105	99.46
	1	600	110.71	97.2	99.51
	2	700	95.88	92.6	99.48
	3	800	82.38	100	99.51
	4	900	75	102	99.43

$$fitness = min\{\alpha f_1 + \beta f_2 * e^{f_4} + \gamma f_3\} \quad (13)$$

The minimum fitness value is calculated based on four objective functions, f_1, f_2, f_3, and f_4, and these functions are represented in (7), (8), (10), and (12), respectively. Where parameters α, β, and γ refer to the weight component of each objective function represented in Table I. The sum of weight components is always equal to value one, and these values always lie in the range of [0,1]. The process iteratively searches for the global best value and continues to the maximum number of iterations. Once the tasks are allocated to virtual machines using the algorithm's global best solution, the resource utilization is evaluated as a performance metric. The average resource utilization specifies the percentage of VM utilization expressed in equation (14).

$$ARUR = \frac{TET}{m \times Makespan} \times 100 \quad (14)$$

4. Results and Comparison

The LBMOIPSO algorithm's performance is evaluated against the conventional PSO and ACO algorithms. The tasks are considered in the range of [100, 500] with the task size range of [1100, 2000] measured in MI to be scheduled on five heterogeneous resources with a computing power range of [500, 900] measured in MIPS.

Table 2 represents the VM load of each computing resource compared to the proposed and existing PSO and ACO algorithms. The VM load is calculated using (15).

$$VM_{jLoad} = \frac{\sum_{i=1}^{n} ET_{ij}}{ATET \times VM_{jMIPS}} \times 100 \qquad (15)$$

VM_{Loadj} is computed as the sum of scheduled tasks executed on VM_j to the average total execution time of all tasks multiplied with VM_j computing power and multiplied by 100 to give $VM_{jload.}$ The proposed algorithm distributes the load equally on each resource based on the computing power of the VM. The load on each VM does not exceed the value of 100, which indicates that no VM is overloaded, and the value of the VM is not below the value of 90, indicating that no VM is underloaded and all resources are utilized effectively.

However, the state-of-the-art algorithms do not balance the VM load equally if the value of the VM load is above 100, indicating that the VM is overloaded.

The proposed algorithm's performance is evaluated using three metrics: makespan, resource utilization, and load balance deviation. Table 3 represents **a** comparison of makespan, average resource utilization and load balance deviation among ACO, PSO and LBIMOPSO.

Table 3. Comparison of makespan, average resource utilization and load balance deviation

No. of Tasks	Makespan			Average Resource Utilization			Load Balance Deviation		
	PSO	ACO	LBIMOPSO (Proposed)	PSO	ACO	LBIMOPSO (Proposed)	PSO	ACO	LBIMOPSO (Proposed)
100	56.03	48.59	44.94	82.37	92.05	98.88	9.12	2.24	0.23
200	115.76	99.8	89.35	82.37	92.05	98.88	9.12	2.24	0.23
300	182.93	145.03	133.89	79.67	90.55	99.56	17.02	6.58	0.23
400	253.03	194.45	178.68	75.93	92.54	99.7	28.31	9.68	0.22
500	312.06	234.6	223.04	73.35	92.3	99.61	39.88	13.11	0.35

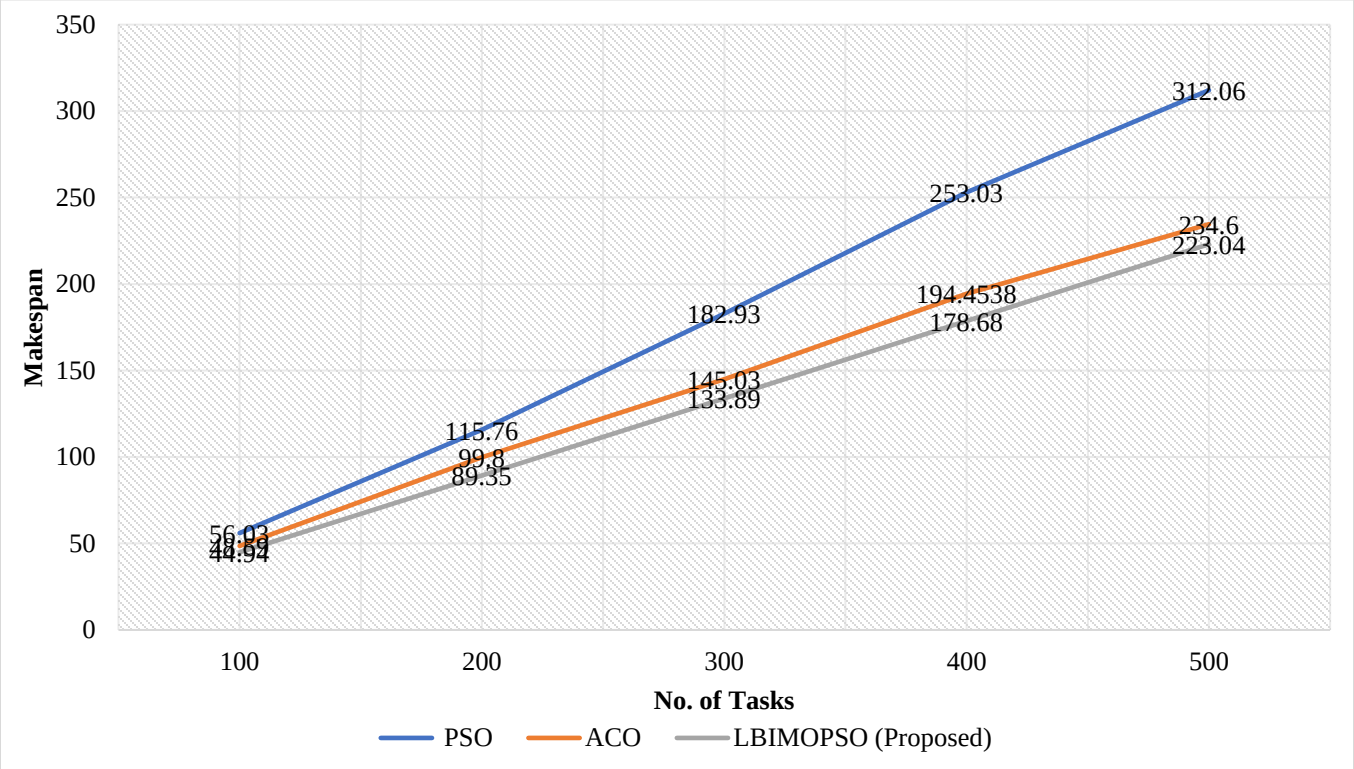

Fig. 3 Comparison of the makespan value

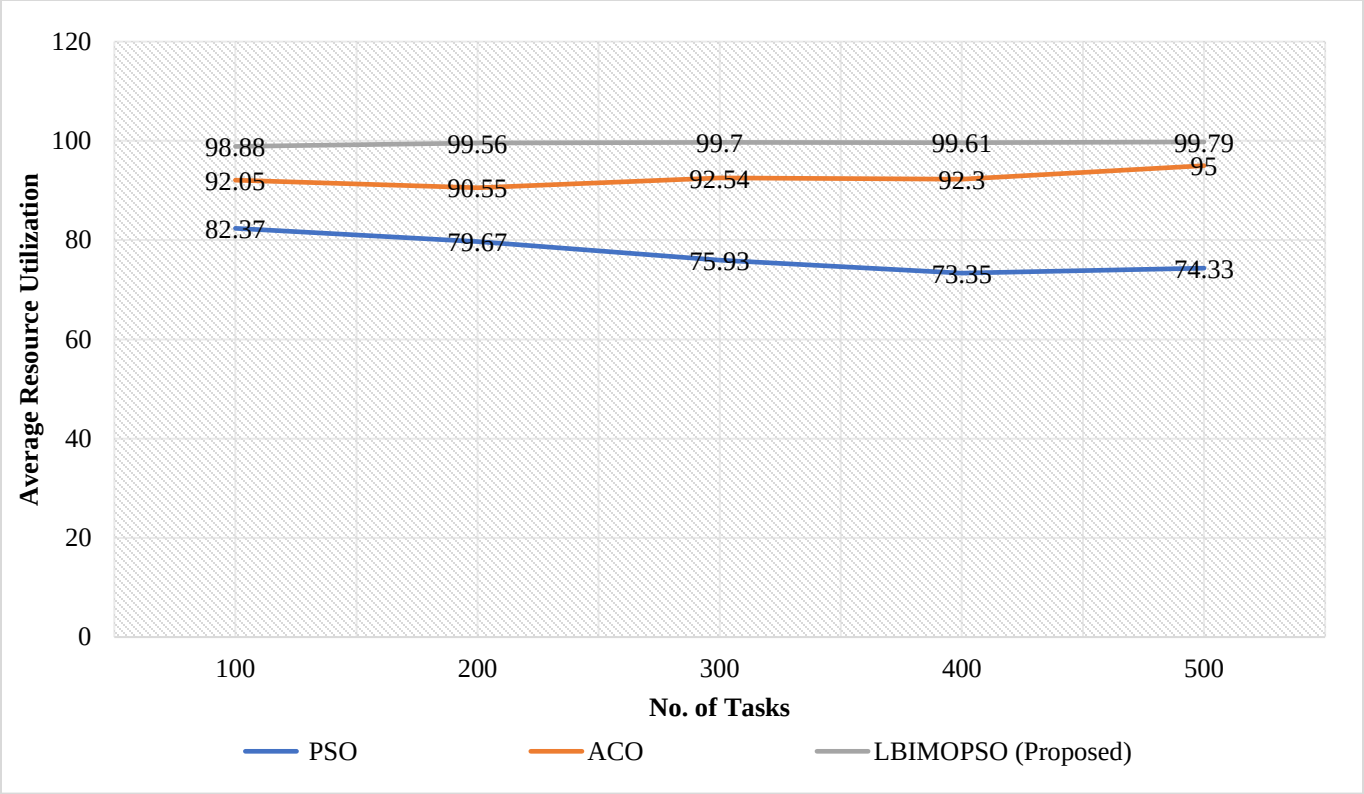

Fig. 4 Comparison of average resource utilization

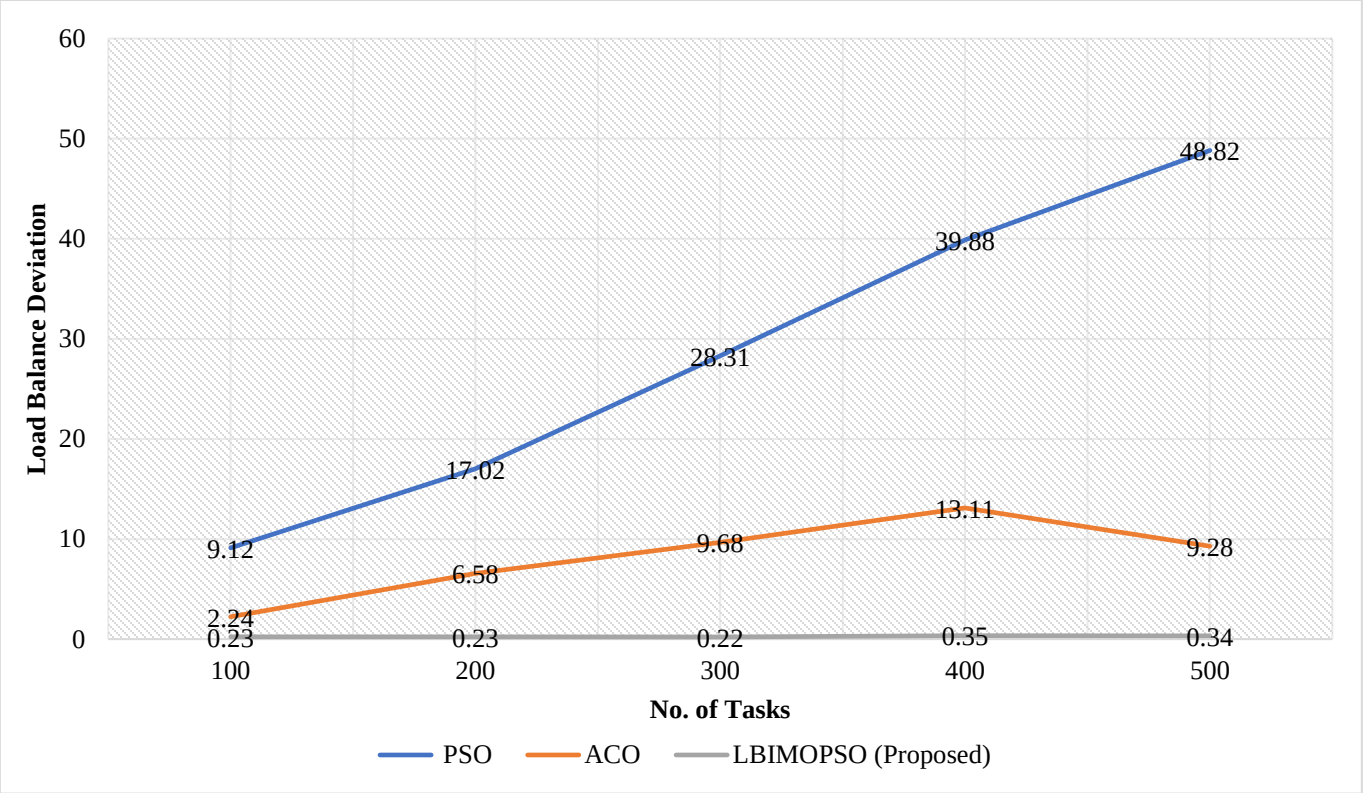

Fig. 5 Comparison of load balance deviation

The results are in Figure. 3 illustrates the comparison of makespan for tasks ranging from 100 to 500. The findings indicate that, in contrast to the standard PSO and ACO algorithms, the proposed LBIMOPSO algorithm significantly decreases makespan. Furthermore, the results reveal that makespan values improve as the number of tasks increases.

Figure 4 plots the average resource utilization among VMs against the number of tasks. The proposed algorithm indicates that maximum utilization of resources is achieved; that is, the rate of ARU is around 99% on all heterogeneous resources. The proposed algorithm is an efficient scheduling strategy and distributes tasks uniformly among all VMs concerning the increasing number of tasks compared to the ACO and PSO algorithms.

In Fig. 5, the load balance deviation of VMs to tasks indicates that the load balance deviation increases with the traffic for existing algorithms as the number of tasks increases. The low value of load balance deviation indicates a balanced load on all VMs, and a higher value indicates more load imbalance on VMs.

The proposed algorithm also increases load deviation with the increasing number of tasks, and this load balance deviation value is low compared to the other algorithms. The load deviation value of ACO and PSO is directly proportional to the increasing number of tasks, indicating an exorbitant load imbalance on VMs with increasing traffic.

5. Conclusion and Future Work

The LBIMOPSO method is a load balancing with task scheduling technique that efficiently distributes tasks among computing resources to ensure optimal resource utilization and prevent overload in the cloud environment. It employs an improved PSO algorithm to uniformly distribute tasks to VMs and manage the load equally on heterogeneous resources based on various task lengths. The conventional PSO technique will likely suffer from local optimality, leading to premature convergence. An improved PSO algorithm alters the particle's position. It extends the scope of searching area in the solution space to improve the local optimality by considering load balancing deviation as one of the multiple objectives. The proposed algorithm provides better results than traditional PSO and ACO regarding minimum makespan and load balance deviation and maximum resource utilization among VMs.

The proposed algorithm has the potential for future extensions to consider additional parameters of the cloudlet, such as cost, heterogeneous input and output file sizes, bandwidth, and task latency, which will help to achieve energy-efficient task scheduling.

References

[1] Asan Baker Kanbar, and Kamaran Faraj, “Region Aware Dynamic Task Scheduling and Resource Virtualization for Load Balancing in IoT-Fog Multi-Cloud Environment,” *Future Generation Computer Systems*, vol. 137, pp. 70-86, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mohit Kumar et al., “A Comprehensive Survey for Scheduling Techniques in Cloud Computing,” *Journal of Network and Computer Applications*, vol. 143, pp. 1-33, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Aida A. Nasr et al., “A New Online Scheduling Approach for Enhancing QOS in Cloud,” *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 424-435, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida, “Load Balancing in Cloud Computing: A Big Picture,” *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp. 149-158, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Aparna Joshi, and Shyamala Devi Munisamy, “Evaluating the Performance of Load Balancing Algorithm for Heterogeneous Cloudlets Using HDDB Algorithm," *International Journal of System Assurance Engineering and Management*, vol. 13, no. 1, pp. 778-786, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Essam H. Houssein et al., “Task Scheduling in Cloud Computing Based on Meta-Heuristics: Review, Taxonomy, Open Challenges, and Future Trends,” *Swarm and Evolutionary Computation*, vol. 62, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] G Rohail Gulbaz et al., "Balancer Genetic Algorithm—A Novel Task Scheduling Optimization Approach in Cloud Computing," *Applied Sciences*, vol. 11, no. 14, p. 6244, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zhi-Hui Zhan et al., "Load Balance Aware Genetic Algorithm for Task Scheduling in Cloud Computing," *Simulated Evolution and Learning*, vol. 8886, pp. 644-655, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xueliang Fu et al., "Task Scheduling of Cloud Computing Based on Hybrid Particle Swarm Algorithm and Genetic Algorithm," *Cluster Computing*, pp. 1-10, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Said Nabi et al., "AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing," *Sensors*, vol. 22, no. 3, p. 920, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Said Nabi, and Masroor Ahmed, "PSO-RDAL: Particle Swarm Optimization-Based Resource-and Deadline-Aware Dynamic Load Balancer for Deadline Constrained Cloud Tasks," *The Journal of Supercomputing*, vol. 78, pp. 4624-4654, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Aida Amini Motlagh, Ali Movaghar, and Amir Masoud Rahmani, "A New Reliability-Based Task Scheduling Algorithm in Cloud Computing," *International Journal of Communication Systems*, vol. 35, no. 3, pp. e5022, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] M. A. Elmagzoub et al., "A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment," *Electronics*, vol. 10, no. 21, p. 2718, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Nupur Jangu, and Zahid Raza, "Improved Jellyfish Algorithm-Based Multi-Aspect Task Scheduling Model for IoT Tasks Over fog Integrated Cloud Environment," *Journal of Cloud Computing*, vol.11, no. 98, pp.1-21, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lina Ni et al., "GCWOAS2: Multiobjective Task Scheduling Strategy Based on Gaussian Cloud-Whale Optimization in Cloud Computing," *Computational Intelligence and Neuroscience*, vol. 2021, pp. 1-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Gang Li, and Zhijun Wu, "Ant Colony Optimization Task Scheduling Algorithm for SWIM Based on Load Balancing," *Future Internet*, vol. 11, no. 4, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] R. Ghafari, F. Hassani Kabutarkhani, and N. Mansouri, "Task Scheduling Algorithms for Energy Optimization in Cloud Environment: A Comprehensive Review," *Cluster Computing*, vol. 25, no. 2, pp. 1035-1093, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Seyed Salar Sefati, Maryamsadat Mousavinasab, and Roya Zareh Farkhady, "Load Balancing in Cloud Computing Environment using the Grey Wolf Optimization Algorithm Based on the Reliability: Performance Evaluation," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 18-42, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Abhikriti Narwal, and Sunita Dhingra, "Load Balancing using Enhanced Multi-Objective with Bee Colony Optimization in Cloud Networks," *Pertanika Journal of Social Science and Humanities*, vol. 28, no. 3, pp. 1049-1061, 2020. [Google Scholar] [Publisher Link]

[20] Mehdi Hosseinzadeh et al., "Improved Butterfly Optimization Algorithm for Data Placement and Scheduling in Edge Computing Environments," *Journal of Grid Computing*, vo1. 19, no. 14, pp. 1-27, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Mohit Agarwal, and Gur Mauj Saran Srivastava, "A PSO Algorithm Based Task Scheduling in Cloud Computing," *International Journal of Applied Metaheuristic Computing*, vol. 10, no. 4, pp. 1-17, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Dineshan Subramoney, and Clement N. Nyirenda, "Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments," *IEEE Access*, vol. 10, pp. 117199-117214, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Almothana Khodar et al., "Design Model to Improve Task Scheduling in Cloud Computing Based on Particle Swarm Optimization," *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*, pp. 345-350, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[24] Aashish Kumar Bohre, Ganga Agnihotri, and Manisha Dubey, "Hybrid Butterfly Based Particle Swarm Optimization for Optimization Problems," *First International Conference on Networks & Soft Computing*, pp. 172-177, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[25] Sobhanayak Srichandan, Turuk Ashok Kumar, and Sahoo Bibhudatta, "Task Scheduling for Cloud Computing using Multi-Objective Hybrid Bacteria Foraging Algorithm," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 210-230, 2018. [CrossRef] [Google Scholar] [Publisher Link]