International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/223686 Volume 10, Issue 6, November – December (2023)

RESEARCH ARTICLE

A Load Balancing Aware Task Scheduling using Hybrid Firefly Salp Swarm Algorithm in Cloud Computing

Pankaj Jain
Department of Computer Science and Engineering, Banasthali Vidyapith, Niwai, India.
mail2pankajjain@gmail.com

Sanjay Kumar Sharma
Department of Computer Science and Engineering, Banasthali Vidyapith, Niwai, India.
ssanjay@banasthali.in

Received: 21 July 2023 / Revised: 13 November 2023 / Accepted: 30 November 2023 / Published: 30 December 2023

Abstract – Cloud computing is an evolutionary computational model which provides on-demand scalable and flexible resources by the pay-per-use concept. Due to the flexibility of cloud, several organizations are setting up more data centers and switching their businesses to the cloud technology. These industries need a proper load balancing to ensure the efficient resources utilization, which reduces resource wastage and helps to optimize costs. Optimal resource allocation can be achieved through efficient task scheduling and load-balancing. An efficient scheduling with load-balancing allocates resources in a balanced way and optimizes the quality of service (QoS) parameters. Task migration is the best way to balance the load. This paper hybridizes the Salp Swarm Algorithm (SSA) with the Firefly Algorithm (FFA), named as Hybrid Firefly Salp Swarm Algorithm (HFFSSA). This approach utilizes FFA's operators to enhance the exploitation capability of SSA by functioning as a local search. Further, a load balancing (LB) heuristic is proposed and incorporated with HFFSSA, named as Load Balancing Salp Swarm Algorithm (LBFFSSA). For verification, the presented work is evaluated by two experimental series. First HFFSSA is tested on global benchmark functions, where it shows its superiority over other existing metaheuristic approaches such as Firefly Algorithm (FFA), Grey Wolf Algorithm (GWO), Particle Swarm Optimization (PSO), and Salp Swarm Algorithm (SSA). In the second series, the LB-FFSSA is evaluated on real datasets (Planet Lab and NASA) using CloudSim Simulator; again, results outperform similar metaheuristics. The simulation results show that LB-FFSSA significantly reduces makespan and improves resource utilization. Furthermore, the proposed algorithm minimizes the Load imbalance Factor (LIF) by migrating the task from an over utilized virtual machine to an underutilized one. It also shows improvement in waiting time and throughput. Simulation results prove that proposed model improves by an average up to 32.3%, LIF by 50.4%, throughput by 42.1%, resource utilization by 40%, and waiting time by 50%.

Index Terms – Cloud Computing, Hybrid Task Scheduling, Firefly (FFA), Salp Swarm Algorithm (SSA), Task Migration, Load Balancing.

1. INTRODUCTION

Cloud Computing (CC) is becoming a reliable and trusted computing technology that enhances the utilization of virtualized resources and services for end users [1, 2]. It provides software and hardware as computing resources. To manage resource sharing in a heterogeneous environment, Task Scheduling (TS) with efficient load balancing plays a key role [3, 4]. TS is an NP-hard problem [5] that needs to be optimized in CC. The primary goal of TS is to allocate the resources to the user's task while optimizing at least one Quality of Service (QoS) parameter [6], such as cost, makespan, etc. This distribution of tasks on heterogeneous resources must be done in a balanced way to improve resource utilization. The assignment of tasks on a resource is denoted load. The load balancing mechanism transfers the excess load from an overloaded to an unloaded resource. Zhou et al. [7] showed importance of load balancing in task scheduling and presented a survey of various literature that used metaheuristic for load balancing.

Several heuristic such as HEFT [8], MAX-MIN [9], Round Robin [10] and metaheuristic approaches such as PSO[11], SSA[12], GWO[13], of TS and load balancing exist in literature [14-16]. Metaheuristics algorithms improve the efficiency of heuristic algorithms [17].

In 2017, Mirjalili et al. [12] introduced a nature-inspired meta-heuristic optimization approach named Salp Swarm Algorithm (SSA). Author [18] also proved the efficacy of SSA by optimizing the Extreme Learning Machine. Further, Jain et al. [19, 20, 21] proved the efficiency of SSA in their literature. Later, Abualigah et al. [22] reviewed SSA thoroughly and showed its strengths and weaknesses. SSA has only one controlling parameter and excellent potential to explore the search space, leading to determining the feasible region with the optimal solution. These are the reasons which

RESEARCH ARTICLE

make SSA suitable for task scheduling. Additionally, FFA [23] has the extraordinary ability to exploit the feasible region to determine the optimal solution. Therefore, hybridizing SSA and FFA is more beneficial in finding the optimal solution. Therefore, this work hybridizes FFA and SSA, called as Hybrid Firefly Salp Swarm Algorithm, abbreviated as HFFSSA. HFFSSA allocates tasks to Virtual Machines (VMs) to optimize QoS parameters. Further, a heuristic technique (LB) is proposed that reschedules the best-obtained solution from HFFSSA to achieve load balancing called LB- FFSSA.

1.1. Problem Definition

In task scheduling, assigning tasks to the appropriate VMs while maintaining promised QoS and SLA is always a big challenge. On the other hand, an imbalanced distribution of tasks on VMs may lead the SLA violation and performance degradation. After task assignment, if some VMs are overloaded and some are underloaded, a load-balancing technique is needed to balance the load to achieve optimal resource utilization. Due to the heterogeneous nature of tasks and resources, efficient task scheduling with a balanced distribution of tasks is still a crucial issue in cloud computing.

So, the objective of the paper is to propose a task scheduling algorithm that optimizes makespan, resource utilization, throughput, and waiting time and a load balancing heuristic that minimizes load imbalance factor.

The significant contribution of the proposed work:

- This work proposes a hybrid metaheuristic HFFSSA for task scheduling that optimizes makespan, resource utilization, throughput, and waiting time.
- This proposed HFFSSA is combined with the proposed load balancing (LB) heuristic, i.e., LB-FFSSA that improves the load imbalance factor.
- The novelty of HFFSSA is tested on 13 benchmark functions. Results are compared with existing metaheuristics FFA, SSA, PSO, and GWO.
- LB-FFSSA is simulated on CloudSim, where workloads are provided through real-world datasets from Planet Lab [24] and NASA [25].
- LB-FFSSA is compared with other existing algorithms, this work schedules tasks on a similar VM configuration, which is used by Amazon EC2 [26].

The remaining of the paper is categorized as follows: Section 2 gives a literature review of existing approaches. Section 3 briefly describes the Salp Swarm Algorithm and Firefly Algorithm. Section 4 represents QoS Metrics, and Section 5 proposes a Hybrid Firefly Salp Swarm Algorithm (HFFSSA) and LB-FFSSA. In section 6, Performance Evaluation is done, where the proposed work is evaluated with 13 benchmark functions, and then the comparative analysis is done using CloudSim and two real datasets. Finally, section 7 concludes the work.

2. REVIEW OF PREVIOUS STUDIES

In cloud computing, task scheduling algorithms assign client tasks to concerned resources under various scheduling constraints such as deadline, cost, or profit. It causes scheduling a complicated problem; thus, it falls in an NP-Hard category [27]; therefore, heuristic or metaheuristic algorithms are more suitable than traditional algorithms. However, the solution produced by heuristic algorithms often engages with the local optima, far from the global optimal [28]. However, metaheuristic algorithms are the best approach to overcome local optima [29, 30]. Furthermore, when the scheduler assigns the task on Virtual Machines (VM), it must consider the Quality of Service (QoS) parameters. QoS parameters include makespan, response time, throughput, cost, load imbalance, and deadline [31].Load balancing is also an essential aspect of task scheduling, ensuring the even distribution of tasks. The main objective of load balancing is to decrease the imbalance in the system, which is achieved by migrating tasks from the overutilized VM to the underutilized VM. Scheduling on an overloaded VM may decrease the overall performance of the CC [32].

Thanka, M. R. et al.[33] presented an improved Artificial Bee Colony-based algorithm for QoS-aware scheduling and security. They focused on security and QoS-aware parameters such as makespan, load imbalance, task migration, and cost. The tasks may be dependent or independent of each other and are scheduled over VM. For evaluation purposes, they used CloudSim but compared their work only with ABC. D. Ramesh et al.[34] introduced a nature-inspired heuristic VM load balancing (HFQ-LB) technique in which load balancing has been achieved by fair queuing and VM migration. Initially, tasks are assigned to VM, and the load is continuously examined; VM is migrated to another underloaded Host if needed. They use the CloudSim Simulator tool for evaluation and further validate their work regarding makespan and resource utilization. They compared their work with various algorithms. In [35], the Author presented a binary PSO algorithm for load-balanced tasks for scheduling that minimizes cost and time complexity. The objective function of this proposed work is to maximize the difference in completion time among various VMs to identify the underloaded and overloaded VMs. They optimize waiting time, makespan, load imbalance, and resource utilization. Further, the algorithm is evaluated using the CloudSim simulator, and results show the superiority of proposed algorithm.

Adhikari, M. et al.[36] proposed an algorithm of load balancing for long-term processes called Load Balancing Resource Clustering (LB-RC). In this approach, they

RESEARCH ARTICLE

identified the optimal clusters of resources and cluster centers for fast convergence using a metaheuristic BAT algorithm. They also proposed a dynamic task assignment policy to attain minimum execution time and makespan within limitations. Finally, they evaluate and compare their work with the help of the CloudSim simulator. Alguliyev, R. M. et al.[37] proposed an algorithm named αPSO-TBLB, which is a task-based algorithm. This algorithm selects tasks from an over utilized VM to an underutilized VM. This proposed work minimizes the task transfer time and execution time. They tested their work using jswarm and CloudSim. The result shows that they equally distributed tasks among VM and achieved optimal task scheduling.

Hanine, M., et al.[38] presented an improved simulated annealing (SA) based algorithm for load balancing. This approach is divided into two steps; in step 1, the approach identifies the load threshold value for each to identify an overloaded VM, and in step 2, the task is allocated to VM using improved SA. The acceptance probability of SA is modified In this work. The results produced by the CloudSim simulation tool show that they could distribute tasks among VM in fewer time intervals. Still, they compare their results only with static approaches. Kruekaew B. et al. [39] proposed a multi-objective scheduling algorithm (MOABCQ) that is based on Artificial Bee Colony and Q-learning. This work focused on optimizing load balancing and task scheduling. They use multi-objective fitness functions consisting of cost, makespan, and resource utilization. They also use First Come First Serve (FCFS) and Longest Job First (LJF). Performance evaluation of proposed work is done using CloudSim and compared with existing balancing and scheduling techniques such as FCFS, Max-Min, Q-Learning, MOCS, and MOPSO. For this analysis, they use three datasets named Google Cloud Jobs (GoCJ), Random, and Synthetic workload. The result shows they improve makespan, cost, resource utilization, and throughput.

Zhou, Z. et al. [40] proposed an MGGS algorithm in which the Modified Genetic Algorithm is blended with Greedy Strategy (GS). MGGS optimizes the task scheduling procedure. This approach is capable of finding optimized solutions in less iteration. For evaluation, MGGS was compared with existing algorithms based on average response time, total completion time, and total cost, and they found it better. For this purpose, they used CloudSim Toolkit but did not use any real dataset. Neelima P. et al. [41] presented a load-balancing aware task scheduling algorithm using the Adaptive Dragonfly Algorithm (ADA), a blend of dragonfly and firefly algorithms. In this approach, they use a multi-objective function to decide on scheduling. Finally, the performance of the presented algorithm is assessed based on distinct metrics, i.e., execution time and cost, using the CloudSim simulator tool and compared with existing algorithms. Still, they needed to evaluate their approach to any benchmark function.

George et al.[42] proposed a model fractional IWSOA for load balancing. Initially, they allocated tasks using round robin algorithm. Then proposed work migrates tasks from overloaded VMs to underloaded VMs. The work can be improved by adopting other powerful optimization technique, i.e., machine learning algorithms. Ramya et al. [43] proposed HDWOA-LBM that is a load balancing mechanism that hybridized dingo and whale optimization algorithm. It effectively balanced the load and in additional improved throughput, resource utilization, and reliability.

After analyzing the above literature, the work focuses on improving the performance of LB-FFSSA with efficient resource utilization and load balancing in task scheduling. Therefore, the key objectives are: decreasing makespan, load imbalance factor, and waiting time while at the same time increasing resource utilization and throughput.

3. BACKGROUND

3.1. Salp Swarm Algorithm

As mentioned above, in 2017, Mirjalili proposed a nature-inspired swarm-based algorithm called Salp Swarm Algorithm (SSA). This algorithm is inspired by the natural swarming behavior of Salp(s). The main aim of Salp(s) is to obtain better and more effective locomotion; therefore, these salps create a salp chain and relocate by applying synchronized updates and foraging. Figure 1 represents the Salp chain which contains two types of Salp, one Leader Salp, which leads all the remaining Salp, and the second Follower Salp, which follows the leader Salp. Leader Salp updates its position to approach a food source. Meanwhile, the food source also updates its position by obtaining a solution, so the salp chain automatically moves toward the optimum solution. Here a food source is the best-obtained solution. The leader salp updates its position using equation 1

$$pos_1^j = \begin{cases} FS_j + r_1\left(\left(up_j - low_j\right) r_2 + low_j\right) r_3 \geq 0.5 \\ FS_j - r_1\left(\left(up_j - low_j\right) r_2 + low_j\right) r_3 < 0.5 \end{cases} \quad (1)$$

Here,

FS_j = Food source, up_j and low_j are Upper bound and Lower bound, respectively, in the j^{th} dimension

pos_1^j = Position of Leader Salp in j^{th} dimension

r_2 and r_3 are Random Numbers from 0 to 1 and r_1 = Controlling parameter used to balance exploitation and exploration.

It is computed using equation 2:

RESEARCH ARTICLE

$$r_1 = 2e^{-\left(\frac{4m}{maxiteration}\right)^2} \quad (2)$$

Here, m is current iteration and maxiteration is total number of iterations.

The position of follower salp can be updated as given in equation 3:

$$pos_i^j = \frac{1}{2}\left(pos_i^j + pos_{i-1}^j\right) \quad (3)$$

Where pos_i^j is the follower's position in the jth dimension for i >1

Algorithm 1 shows the standard SSA Algorithm.

1. Initialize the number of iterations = maxiteration, upper, Population size
   np, lower, it=1.
2. Initialize salp population $SP_x \forall x = 1: np$
3. while (it < maxiteration)
4. Calculate the fitness function for each salp.
5. Assign best-fitted salp as a leader and remaining as follower salp.
6. Assign the best-obtained solution to FS
7. Update r1 using equation 2.
8. for every salp SP_x
9. if(x=1)
10. leader salp position updated using equation 1
11. else
12. follower salp position updated using equation 3
13. end of loop
14. Update the salp population using upper and lower
15. End while
16. Return FS

Algorithm 1 Salp Swarm Algorithm

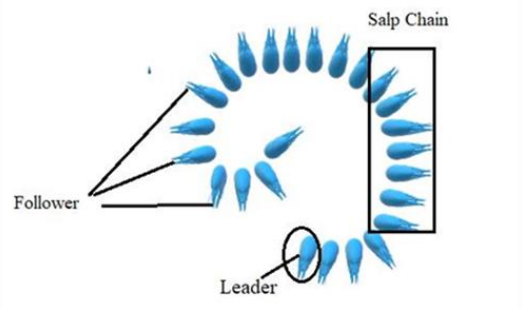

Figure 1 The Salp Chain [21]

3.2. Firefly Algorithm (FFA)

FFA is a nature-inspired meta-heuristic algorithm proposed by Xin-She Yang in 2009 that simulates the flashing manners of fireflies. The FFA is motivated by flashing patterns and behavior at night.

From introductory physics, it is apparent that intensity of light is reciprocal proportion to square of distance, therefore variation in attractiveness with the distance from a source a can be defined by using equation 4:

$$\beta(r) = \beta_0 e^{-(\gamma r^2)} \quad (4)$$

Where β0 represents attractiveness at r = 0, γ represents light absorption coefficient for a given medium.

If a firefly X_j is brighter than any other firefly X_i in the search space, then the firefly X_i will move towards X_j using given equation 5:

$$X_i^{t+1} = X_i^t + \beta_0 e^{-\left(\gamma r_{ij}^2\right)}\left(X_j^t - X_i^t\right) + \alpha_t E_i^t \quad (5)$$

Where r_{ij} denotes the distance between i^{th} and j^{th} fireflies

α_t is a randomized parameter with $0 \leq \alpha_t \leq 1$, which controls the randomness and, E_i^t represents the vector of random numerals drawn by the Gaussian function or uniform function, or any other distribution function, and the 't' Iteration number. α_t is computed using equation 6:

$$\alpha_t = \alpha_0\, \delta^{-t} \quad (6)$$

Where α_0 denotes the initial randomness scaling factor or range, and δ is the essential cooling factor; $0 \leq \delta \leq 1$

Algorithm 2 shows the FFA Algorithm.

1. Initialize lower, upper, number of iterations = maxiteration, population size np,it=1.
2. Initialize fireflies population SP_x (x=1 to np).
3. Calculate the fitness value for every solution SP_x called using fit(SP_x)
4. Determine the light absorption coefficient
5. while it<maxiteration
6. Calculate the fitness function of each firefly
7. for each $SP_i \forall i = 1: np$
8. for each $SP_i \forall j = 1: np$
9. if fit(SP_i)>fit(SP_j)
10. Move the i^{th} firefly towards the j^{th} firefly using equation 5
11. end if
12. Update attractiveness using equation 4

RESEARCH ARTICLE

13. End for
14. End for
15. Determine the best solution.
16. End while
17. return the best solution

Algorithm 2 Firefly Algorithm (FFA)

4. QUALITY OF SERVICE (QOS) METRICS

Let T_i denote a set of tasks, where i ∈ {1: n}, and a set of the virtual machines as VM_j, where j ∈ {1: m}. Here' n' and 'm' are no. of tasks and no. of VMs, respectively. These tasks and VMs are independent and heterogeneous.

4.1. Makespan (MS)

If the same task is scheduled on a different VM, execution time (ET) may vary. The sum of the start time (ST) and ET taken for any VM to run all the tasks is known as Finishing Time (FT), and so as Maximum FT is referred to as makespan. It is better to have a minimal makespan for better load balancing. Further, if the execution time of i^{th} task (T_i)on a j^{th} virtual machine (VM_j) is ET_{ij}, then execution time can be characterized by equation 7 as:

$$ET_{ij} = \frac{len_i}{cap_{VMj}} + \frac{leni_{infile}}{BW_{VMj}} \quad (7)$$

len_i represents the length of $task_i$, $leni_{infile}$ denotes the input file's length of the i^{th} task, and cap_{VMj} and BW_{VMj} denote the capacity and Bandwidth of VM_j, respectively.

The finishing time of i^{th} task (T_i) on j^{th} virtual machine VM_j can be calculated using equation 8 as:

$$FT_{ij} = ST_i + ET_{ij} \quad (8)$$

The makespan can be defined as given in equation 9

$$MS = \max(FT_{ij}) \quad (9)$$

4.2. Resource Utilization (RU)

In scheduling, maximum resource utilization and minimum makespan are two conflicting QoS parameters. They shared an inverse relationship [44, 45]. Makespan is a consumer-driven QoS parameter; however, resource utilization service provider-driven parameter [46]. RU can be calculated by the ratio of the total Execution Time and capacity of the VM. In continuation of the above, RU of j^{th} VM can be formulated using equation 10 as below.

$$RU_J = \frac{\sum_{i=1}^{n} ET_{ij}}{MS} \quad (10)$$

Average RU can be calculated using equation 11

$$RU_{AVG} = \frac{\sum_{j=1}^{m} RU_j}{m} \quad (11)$$

4.3. Waiting Time

It can represent by the difference between finishing time and execution time as given in equation 12.

$$WT_i = FT_i - ET_i \quad (12)$$

Where i ∈ {1, 2, 3….n}

4.4. Throughput

This parameter indicates the number of tasks finished per unit of time. It is calculated using equation 13

$$\text{Throughput} = \frac{\text{No of the task finished Successfully}}{\text{makespan}} \quad (13)$$

4.5. Fitness function

In this work, Fitness Function is defined as given in equation 14:

fit = Minimize (MS) (14)

Here MS has been calculated by equation 9.

5. PROPOSED LB-FFSSA MODEL

Figure 2 represents the proposed model for task scheduling and load balancing. First, a cloud user submits the task to Cloud Service Provider (CSP).CSP allocates the cloud resources, i.e., VMs to these tasks. This model schedules the tasks on VMs using proposed HFFSSA and balances the load in VMs using proposed Load balancing (LB) heuristics. Description of each component of the proposed model is given below:

5.1. User's Task

In simulation-based assessment, it is vital to commit an investigation using actual workload traces. Here, two real datasets are used, Planet Lab and NASA Ames iPSC/860. Planet Lab traces is furnished as a part of the CoMon project, which is accessible from Beloglazov's GitHub repository (https://github.com/beloglazov/planetlab-workload-traces). In this, CPU utilization data is fetched from 1000+ VMs from various heterogeneous servers located at 500 different places around the globe.

It takes five minutes intervals for utilization measurements. Based on CPU utilization, this data is divided into ten categories. Each category represents a single-day workload, and each trace file has 288 readings. Workload traces were measured from March 2011 to April 2011. For the experiments, one-day workload (20110303) traces are used.

For NASA Ames iPSC/860 hypercube, workload traces were fetched from 128 iPSC/860 nodes between October 1995 and December 1995. This workload consists of a mixture of interactive and batch jobs. This setup is located at NASA Ames Research Center.

RESEARCH ARTICLE

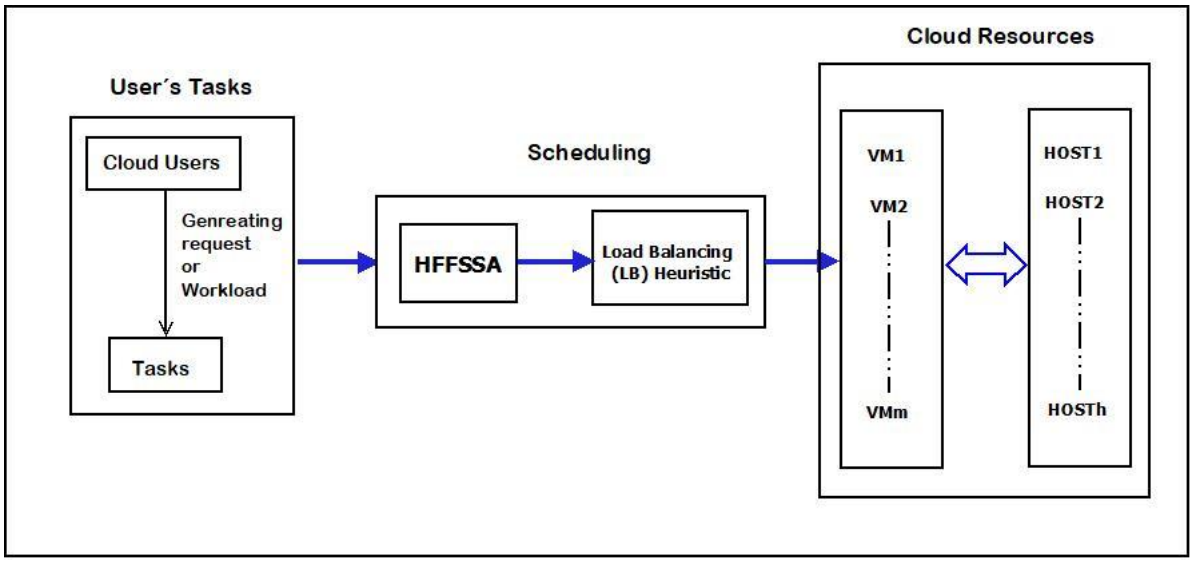

Figure 2 Proposed LB-FFSSA Model

5.2. Cloud Resources

In Cloud Computing, computing resources provide from a virtual infinity pool of resources. These computing resources are known as Virtual Machines (VM), which consist of CPU cores, RAM, network bandwidth, cost, and other essential components. This work considers a single cloud data center model similar to the one offered by Amazon EC2. This datacenter consists of six heterogeneous VM types, listed in table 1, and two heterogeneous hosts, with enough capacity to serve all the VMs. These VM types belong to the general purpose instance group of the US East region. Proposed work assumes a set of virtual machines as VM_j, where $j \in \{1: m\}$.

Table 1 VM Instance type based on Amazon EC2 [26]

VM TYPE	Bandwidth (Mbps)	VCPU	RAM(Gib)	Price ($)
a1.xlarge	3,500	4	4	0.102
a1.metal	3,500	16	16	0.408
t4g.small	2,085	2	2	0.0168
t4g.large	2,780	2	8	0.16
t3.mediu	2085	2	4	0.03
t3.xlarge	2780	4	32	0.05

5.3. Hybrid Firefly Salp Swarm Algorithm (HFFSSA)

This province shows the hybridization of two metaheuristic optimization approaches, SSA and FFA, and forms a new optimization algorithm, HFFSSA. This new approach utilizes the benefits of both SSA and FFA to map tasks with VMs in efficient way. Algorithm 3 shows the steps of HFFSSA. Firstly parameters are initialized in line 2, followed by defining the fitness function using equation 14. Line 3 defines fitness function using equation 14. This fitness function is selects the solution having minimum makespan. In line 4, the salp population is randomly initialized. After that, line 5 calculates fitness of each population. In line 6, the best solution and best fitness are assigned to gbestsolution and gbestfitness, respectively. From lines 7 -13, the solution is updated using FFA (equation 5), but if condition in line 11 is not satisfied, it is updated using SSA using equation 3(line 17). After updating the solution, amend it by lower and upper (line 19). Then line 20 calculates the fitness value for the solutions, and lines 21-23 update gbestfitness and gbestsolution. This complete process is repeated till the maxiteration. After the maxiteration, line 27 calls load_balance function for the final solution. This load_balance function equally distributes the load among VMs. Equal distribution of load improves resource utilization and minimize load imbalance factor.

Global best solution is the best position among all the salp. It is assigned using line 22 in the propose HFFSSA. Local best solution is the salp's best position among all the iteration. It is assigned using lines 11 &12.

Input: Tasks T_i, where $i \in \{1: n\}$

Output: Allocation of T_i on VM_j, where $j \in \{1: m\}$.

1. Start
2. Initialize lower, upper, number of iterations maxiteration, Population size np
3. Define fitness function (fit) using Equation 14
4. Initialize the salp population $SP_x \forall x = 1: np$ randomly

RESEARCH ARTICLE

5. Calculate the fitness of each solution //fit(SP_x)
6. Initialize gbestsolution ← solution of minimum fitness, gbestfitness ← minimum fitness value
7. while (it<maxiteration)
8. for each solution SP_x(x=1 to np)
9. flag=0
10. for solutions SP_i (i=1 to x)
11. if (fit(SP_x)> fit(SP_i))
12. Update SP_x using equation 5
13. flag=1
14. end if
15. end for
16. if flag is zero
17. Update SP_x using equation 3
18. end if
19. Amends the solution based on lower and upper
20. Calculate fit(SP_x) using equation 14
21. if fit(SP_x) < gbestfitness
22. Assign gbestsolution ← SP_x
23. Assign gbestfitness ← fit(SP_x)
24. end if
25. end for
26. end while
27. Call load_balance(gbestsolution) //Algorithm 4
28. return gbestsolution
29. End

Algorithm 3 HFFSSA

5.4. Load Balancing HFFSSA (LB-FFSSA)

In task scheduling, load balancing is a vital aspect affecting the entire system's performance. It is a technique in which a client's task is distributed among multiple servers to increase resource utilization and decrease makespan and execution time. A proper load balancing technique can (a) stop overloaded and underloaded situations, (b) improve the VM's efficiency (c) reduce makespan.

To identify load imbalances in Cloud Environment, compare VM's Load ($Load_{VM}$) with the average load on all VMs (Avg_{LoadVM}). It is determined by calculating the difference from the average load. The present load of i^{th} VM is calculated using equation 15, where LEN_{task_j} is the length of the j^{th} task allotted to i^{th} VM, PC_{VM_i} is the total processing capacity of i^{th} VM, and allot refers to the total allocated task to i^{th} VM.

$$Load_{VM_i} = \frac{\sum_{j=1}^{Allot} LEN_{task_j}}{PC_{VM_i}} \quad (15)$$

Here, processing capacity of i^{th} VM can be determined by equation 16, where PE_{num} is the number of the processing element allotted to the VM, PE_{len} is the length of the processing element in a million instructions per second (MIPS), $VM_{Bandwidth}$ is allocated Bandwidth of i^{th} VM, and VM_{RAM} is allotted RAM of i^{th} VM.

$$PC_{VM} = PE_{num} * PE_{len} * VM_{Bandwidth} * VM_{RAM} \quad (16)$$

The average load of VM can be calculated using equation 17. 'm' is the total no. of VM.

$$AvgLoad_{VM} = \frac{\sum_{i=1}^{m} Load_{VM_I}}{m} \quad (17)$$

Further Load imbalance factor (LIF) can be calculated by using equation 18

$$LIF_{VM} = \frac{\sum_{i=1}^{m} \left| AvgLoad_{VM} - Load_{VM_I} \right|}{m} \quad (18)$$

Based on load, VMs are categorized into three parts : (a) Overloaded – where VM has more load compared to Average Load, (b) Balanced - where VM has equal load compared to Average Load; and (c) Underloaded - where VM has less load compare to Average Load. When unbalanced, the tasks are migrated from the overloaded VM to the underloaded VM. Algorithm 4 shows the load balancing heuristic.

load_balance (gbestsolution)

1. Start
2. for each VM_j ∀ j =1: m
3. Determine the present load of VM ($Load_{VM_i}$) using equation 15
4. end for
5. Determine the average load ($AvgLoad_{VM}$) using equation 17
6. for each VM_j ∀ j =1: m
7. if ($Load_{VM_j} < AvgLoad_{VM}$)
8. VM_j is grouped into Underloaded
9. else if ($Load_{VM_j} > AvgLoad_{VM}$)
10. VM_j is grouped into Overloaded
11. else VM is grouped as Balanced

RESEARCH ARTICLE

12. end if
13. end if
14. end for
15. Arrange the Overloaded group in decreasing order and the Underloaded in non-decreasing order
16. for each task in each overloaded VM, search for the best suitable Underloaded VM according to capacity.
17. Update the underloaded and overloaded group
18. End

Algorithm 4 Load_balance Heuristic

6. PERFORMANCE EVALUATION

This work used two experimental series to evaluate the presented work's quality in this part. In the first series, well-established benchmark functions, and in the next series, proposed work is tested using two real dataset workload traces, Planet Lab and NASA Ames iPSC/860. Both the series are compared with four metaheuristic techniques, namely SSA [12], FFA [23], GWO [13], and PSO [47].

6.1. Experimental Setup

The proposed work is simulated on a laptop operating on Intel® Core™ i5-7200U CPU @ 2.50GHz × 4 with 8 GB of memory using the CloudSim [48] simulator tool. CloudSim is a simulator that provides a virtualized environment for the user to model, simulate and experiment with cloud applications. It also supports on-demand provisioning.

6.2. Parameters Setting

The simulation environment consists of a data center with six types of VM instances, as shown in table 1. For both experimental series, population size and maximum iteration consider as 50 and 1000, respectively. In both experiment series, the total no of VMs are taken 25, 50, 75,100, 125, and 150. However, the number of tasks scheduled on VMs is 1052 and 5000 for Planet Lab and NASA, respectively. Each reading is calculated by running the experiment 10 independent times, and their mean value is noted as final result. Table 2 represents the parameters used in algorithms.

Table 2 Parameters Used in Algorithms

Algorithms	Parameters
HFFSSA	r1 =$2e^{-\left(\frac{4*m}{maxiteration}\right)^2}$, r2 and r3 - Random Numbers from 0 to 1
	β_0 = 1; and γ is a light absorption coefficient for a given medium; α_t is a randomized parameter with $0 \leq \alpha_t \leq 1$ α_0 is the initial randomness, δ is the essential cooling factor(from 0.95 to 0.97).
SSA	r2 and r3 are Random Numbers from 0 to 1
FFA	β_0 = 1; and γ is a light absorption coefficient for a given medium; α_t is a randomized parameter with $0 \leq \alpha_t \leq 1$ α_0 is the initial randomness . δ is the essential cooling factor(from 0.95 to 0.97).
GWO	a=Random Numbers from 0 to 1
PSO	Acceleration coefficients C1, C2 = 1.5

RESEARCH ARTICLE

6.3. Experiment Test Series 1: Benchmark function and Evaluation

HFFSSA is analyzed and examined on 13 well-known benchmark functions in table 3. Furthermore, HFFSSA is compared with other famous metaheuristics, SSA, FFA, PSO, and GWO. The results after comparison are listed in table 4, which contains the worst, mean, and best deviation of fitness value. The results show that HFFSSA obtains a better fitness value than SSA, FA, GWO, and PSO for all functions except f7. For f2,f3,f4,f5,f6, and f12 HFFSSA gives optimum results.

Table 3 Benchmark Functions

Function Name	Function No	Function Description	Range	Dimension	Global Minimum
Ackley	f1	$-aexp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-exp\left(\frac{1}{d}\sum_{i=1}^{d}cos\,(cx_i)\right)+a+exp(1)$	$x_i \in \begin{bmatrix}-32.768,\\ 32.768\end{bmatrix}$, $\forall i=1,.,d$	n	$f(x^*)=0, at\ x^*=(0,..,0)$
Gold Stien	f2	$f(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]*[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1*x_2+27x_2^2)]$	$x_i \in [-2,2]$	2	$f(x^*)=3at\ x^*=(0,-1)$
BUKIN N.6	f3	$f(x)=100\sqrt{\lvert x_2-0.01x_1^2\rvert}+0.01\lvert x_1+10\rvert$	$x_1 \in [-15,-5]$, $x_2 \in [-3,3]$	2	$f(x^*)=0\ at\ x^*=(-10,1)$
Sphere	f4	$f(x)=\sum_{i=1}^{d}x_i^2$	$x_i \in [-5.12,5.12]$ $\forall i=1,..,d$	d	$f(x^*)=0\ at\ x^*=(0,\ldots,0)$
Bohachevsky	f5	$f_1(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1)-0.4cos(4\pi x_2)+0.7f_2(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1)cos(4\pi x_2)+0.3f_3(x)=x_1^2+2x_2^2-0.3cos$	$x_i \in [-100,100]$ $\forall i=1,2$	2	$f_j(x^*)=0, at\ x^*=(0,0), \forall j=1,2,3$
Drop-Wave Function	f6	$f(x)=\frac{-1+cos\left(12\sqrt{x_1^2+x_2^2}\right)}{.05(x_1^2+x_2^2)+2}$	$x_i \in [-5.12,5.12]$ $\forall i=1,2$	2	$f(x^*)=0\ at\ x^*=(0,0)$
Booth	f7	$f(x)=(x_1+2x_2+7)^2+(2x_1+x_2-5)^2$	$x_i \in [-10,10]\forall i=1,2$	2	$f(x^*)=0\ at\ x^*=(1,3)$
Beale	f8	$f(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$	$x_i \in [-4.5,4.5]\forall i=1,2$	2	$f(x^*)=0, x^*=(3,0.5)$

RESEARCH ARTICLE

Function Name	Function No	Function Description	Range	Dimension	Global Minimum
Inverted Cosine Wave	f9	$f(x) = \sum_{i=1}^{n-1} e^{\frac{-\left(x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}\right)}{8}} * cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}}\right)$	$x_i \in [-5,5] \forall i = 1,..,n$	n	$f(x^*) = -n + 1$
Banana shape	f10	$f(x) = \frac{-100}{10[(x_1+1)^2 - (x_2+1)^2] + x_1^2 + 4}$	$x_i \in [-1.5,1.5]$, $x_i \in [-2.5,.5]$	2	$f(x^*) = -25, x^* = (0)$
Rosenbrock Function	f11	$f(x) = \sum_{i=1}^{d-1} [100(x_i + 1 + 1 - x_i^2)^2 + (x_i - 1)^2]$	$x_i \in [-5,10] \forall i = 1,..,dx_i \in \begin{bmatrix} -2.048, \\ 2.048 \end{bmatrix} \forall i = 1,..,d$	n	$f(x^*) = 0, x^* = (1,..,1)$
Three hump camel	f12	$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6 + x_1} x_2 + x_2^2$	$x_i \in [-5,5] \forall i = 1,2$	2	$f(x^*) = 0\ at\ x^* = (0,0)$
Easom	f13	$f(x) = -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$x_i \in [-100,100] \forall i = 1,2$	2	$f(x^*) = 0\ at\ x^* = (\pi, \pi)$

Table 4 Comparison Results of Benchmark Functions

Function	Metrics	HFFSSA	FFA	SSA	GWO	PSO
f1	Best	4.40E-16	19.52678095	6.2148682	20.9087011	1.21E+01
	Mean	4.09E-01	1.96E+01	1.12E+01	21.185129	1.26E+01
	Worst	1.22737352	19.57565031	13.476681	21.3306324	1.32E+01
f2	Best	3	3	3	3	3.08
	Mean	3.00E+00	3.00E+00	3.03E+00	3.00000013	3.3
	Worst	2.99999	3	3.0390328	3.00000039	3.67
f3	Best	0	0.005321134	0.0061328	0.26533978	7.78E-02
	Mean	0	1.77E-02	6.90E-03	0.30530532	1.20E-01
	Worst	0	0.042311452	0.007642	0.32353482	1.87E-01

RESEARCH ARTICLE

f4	Best	0	2.02E-24	2.5084002	2832.89282	2.62E+01
	Mean	0	7.85E-01	4.18E+00	3661.56312	2.90E+01
	Worst	0	2.355692017	5.8455072	4266.45283	3.13E+01
f5	Best	0	0.00E+00	7.33E-15	0	2.74E-01
	Mean	0	0.00E+00	3.84E-14	0	3.17E-01
	Worst	0	0.00E+00	3.12E-14	0	4.02E-01
f6	Best	-1	-0.93624533	-0.99796	-1	-9.36E-01
	Mean	-1	-9.36E-01	-9.57E-01	-1	-9.67E-01
	Worst	-1	-0.93624533	-0.936179	-1	-9.97E-01
f7	Best	0	1.52E-28	3.09E-13	1.43E-09	9.91E-04
	Mean	1.67E-01	2.46E-25	1.02E-10	4.02E-09	6.10E-03
	Worst	0.5011	6.68E-25	3.01E-10	7.18E-09	1.01E-02
f8	Best	2.95E+00	3.63E-27	0.0033244	0.49189747	5.61E-03
	Mean	2.95E+00	1.09E-26	1.00E-02	4.96E-01	3.23E-01
	Worst	2.95E+00	2.09E-26	0.0158038	0.49803542	9.46E-01
f9	Best	-19.5389692	-11.9898383	-12.64006	-22.657141	-9.60E+00
	Mean	-1.26E+01	-6.95E+00	-1.18E+01	-22.328024	-7.61E+00
	Worst	-11.309857	-2.93E-07	-10.77779	-21.78783	-6.51E+00
f10	Best	-2.50E+01	23.42221271	-3.25E+08	-92642.501	-2.95E+05
	Mean	-26.857968	1.99E+01	-4.61E+09	-10895286	-1.34E+05
	Worst	-30.573905	16.0716999	-1.31E+10	-2.55E+07	-8.95E+04
f11	Best	3.90E+01	37.41	76.36	38.4486491	6.89E+02
	Mean	3.90E+01	3.78E+01	9.29E+01	37.8766618	8.57E+02
	Worst	3.90E+01	38.33	108.98	37.1852367	9.66E+02
f12	Best	0	2.75E-26	2.84E-14	0	5.61E-04
	Mean	0	3.01E-26	6.14E-13	0	1.33E-03
	Worst	0	3.54E-26	1.40E-12	0	1.97E-03

RESEARCH ARTICLE

f13	Best	-1	-1	-1	-1	-1.24E-01
	Mean	-.980521	-6.67E-01	-9.50E-01	-1	-4.96E-02
	Worst	-9.42E-01	0.00E+00	-0.892552	-1	-6.59E-05

6.4. Experiment Test Series 2 and Evaluation

The performance of the proposed work is also analyzed using Planet Lab and NASA datasets by comparing it with the existing metaheuristic algorithm concerning for QoS metrics given in section 4. For comparison, all the algorithms are implemented in the same environment.

6.4.1. Results and Discussion

As discussed above, the experiment runs on two real-world datasets, one from Planet Lab and another from NASA. Tables 5 and 6 show the obtained result. After analysis of table 5, it is observed that LB-FFSSA performs better than others. Initially, when the number of VMs was less, it showed superiority over others. However, as the number of VM increased, the scheduling of tasks became effortless; thus, the difference between proposed work performance and others became lesser. Comparative graphs for discussed QoS are shown in figure 3 to figure 7 using the Planet Lab dataset, and figure 8 to figure 12 contain results using NASA.

- Makespan

It can be observed by figures 3 and 8 that LB-FFSSA minimizes makespan as compared to competitive algorithms. It is calculated using equation (9). For Planet Lab workload, LB-FFSSA minimizes makespan by an average of 20.7%, 22.3%, 15.1%, and 24.7% compared to PSO, GWO, SSA, and FFA, respectively. For NASA workload, LB-FFSSA minimizes makespan by an average of 11.8%, 32.3%, 16%, and 15.2% compared to PSO, GWO, SSA, and FFA, respectively.

- Load Imbalance Factor (LIF)

Figures 4 and 9 show that proposed work minimizes LIF as compared to other algorithms. It is calculated using equation 18. For Planet Lab workload, LB-FFSSA minimizes LIF by an average of 41%, 35%, 31.6%, and 65.8% compared to PSO, GWO, SSA, and FFA, respectively. For NASA workload, LB-FFSSA minimizes LIF by an average of 29.9%, 22%, 34%, and 50.4% compared to PSO, GWO, SSA, and FFA, respectively.

- Throughput

Figures 5 and 10 show that LB-FFSSA improves throughput as compared to other algorithms. It is calculated using equation 13. LB-FFSSA improves throughput by an average of 18.6%, 13%, 19.1%, and 42.1% for Planet Lab workload and 8.5%, 10.6%, 10.6%, and 23.4% for NASA workload compared to PSO, GWO, SSA, and FFA, respectively.

- Resource Utilization (RU)

It can be seen by figures 6 and 11 that proposed model efficiently improves RU using LB heuristic. RU is calculated using equation 11. LB-FFSSA improves RU by an average of 20%, and 40% for Planet Lab workload and 33.3%, and 33.3% for NASA workload compared to GWO and FFA, respectively.

- Waiting Time

Figures 7 and 12 show that LB-FFSSA minimizes waiting time compared to other competitive algorithms. It is calculated using equation 12. LB-FFSSA improves WT by an average of 26.5%, 13.8%, 16.7%, and 50% for Planet Lab workload and 12.8%, 6%, 19.3%, and 36.3% for NASA workload compared to PSO, GWO, SSA, and FFA, respectively.

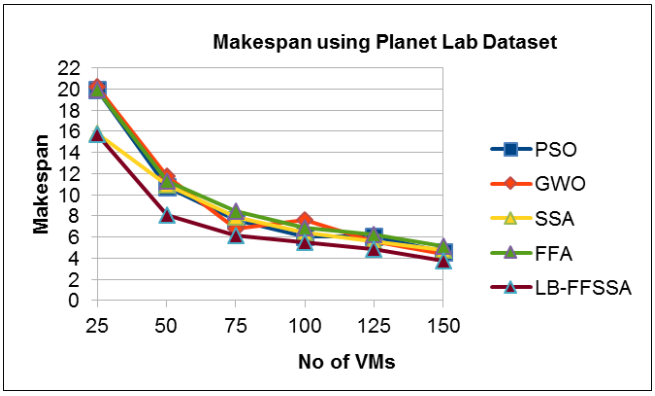

Figure 3 Makespan of Proposed and Existing Algorithm Using Planet Lab Dataset

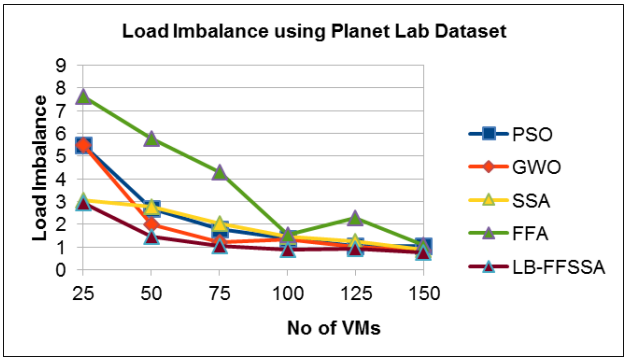

Figure 4 Load Imbalance of Proposed and Existing Algorithm Using Planet Lab Dataset

RESEARCH ARTICLE

Table 5 Comparative Results of the Existing and the Proposed Algorithm for QoS Metrics Using Planet Lab Dataset

QoS Metrics	No. of VM	Algorithms				
		PSO	GWO	SSA	FFA	LB-FFSSA
Makespan (Sec)	25	19.915	20.191	15.79	19.899	15.713
	50	10.775	11.717	10.902	11.316	8.082
	75	7.62	6.804	7.896	8.484	6.141
	100	6.09	7.635	6.466	6.891	5.549
	125	6.091	5.563	5.639	6.256	4.848
	150	4.59	4.397	4.817	5.125	3.764
	Average	9.2	9.4	8.6	9.7	7.3
Load Imbalance	25	5.487	5.477	3.072	7.643	2.942
	50	2.696	1.999	2.766	5.766	1.459
	75	1.777	1.204	2.022	4.293	1.057
	100	1.394	1.352	1.459	1.557	0.879
	125	1.044	1.006	1.259	2.285	0.933
	150	1.034	0.825	0.888	1.068	0.768
	Average	2.2	2	1.9	3.8	1.3
Resource Utilization (in %)	25	0.5827	0.5731	0.566	0.4081	0.5835
	50	0.5248	0.4285	0.5122	0.3499	0.591
	75	0.4828	0.4926	0.4781	0.2958	0.5157
	100	0.4515	0.3565	0.4311	0.4149	0.5005
	125	0.3744	0.3911	0.3942	0.2741	0.465
	150	0.3912	0.4129	0.3812	0.3349	0.4581
	Average	0.5	0.4	0.5	0.3	0.5

RESEARCH ARTICLE

Throughput	25	44.68	44.1	56.34	31.27	56.69
	50	81.89	81.31	80.87	51.88	108.27
	75	114.88	132.04	110.83	62.46	141.63
	100	143.65	149.04	134.13	106.93	177.2
	125	157.69	165.83	153.28	99.52	181.92
	150	186.95	208.28	189.61	166.25	230.84
	Average	121.6	130.1	120.8	86.4	149.4
Waiting Time (in Sec)	25	7.821	7.733	5.333	9.836	5.374
	50	4.085	2.712	4.114	7.433	3
	75	2.761	2.184	2.894	6.088	2.077
	100	2.184	1.996	2.226	2.414	1.723
	125	1.695	1.625	1.862	2.874	1.502
	150	1.605	1.409	1.445	1.583	1.332
	Average	3.4	2.9	3	5	2.5

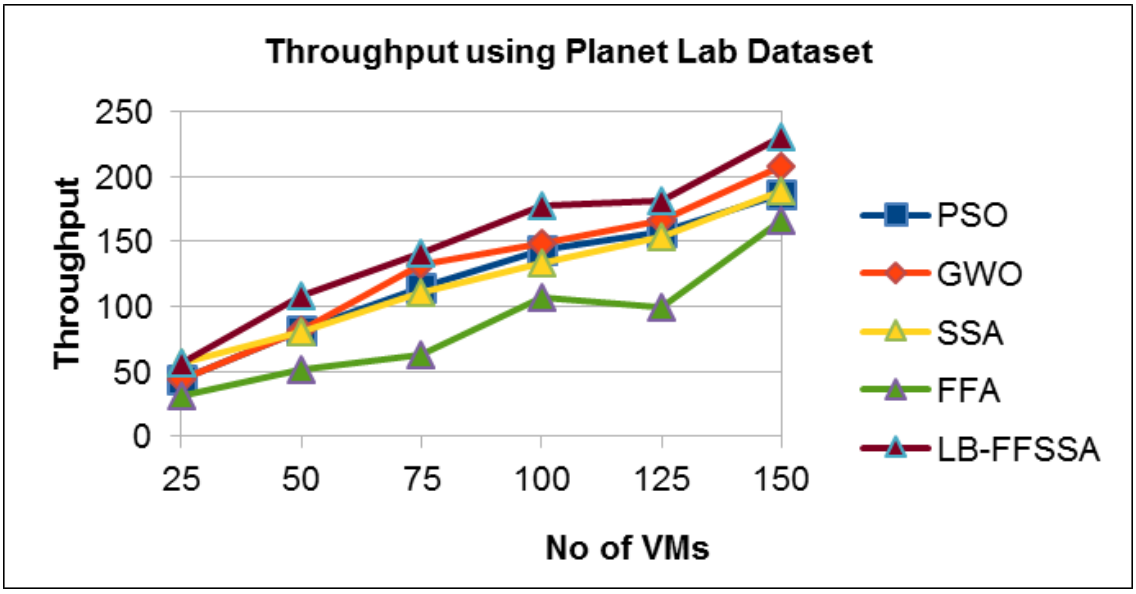

Figure 5 Throughput of Proposed and Existing Algorithm Using Planet Lab Dataset

RESEARCH ARTICLE

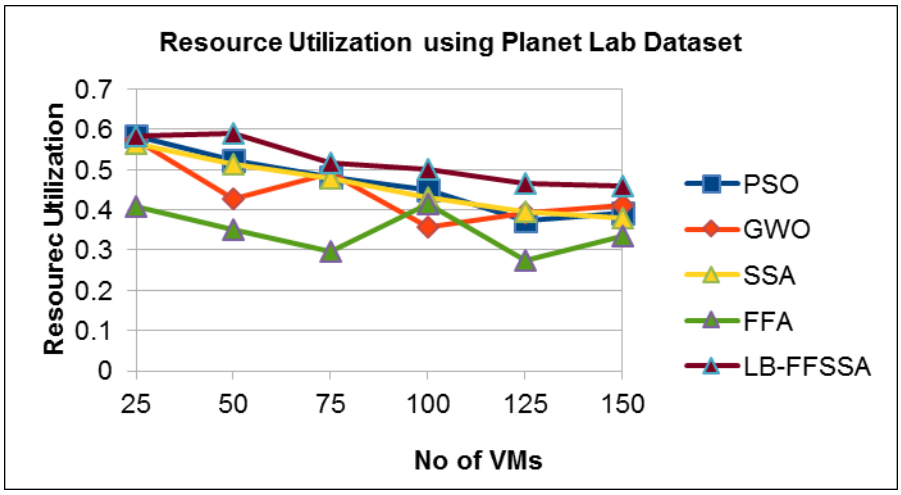

Figure 6 Resource Utilization of Proposed and Existing Algorithm Using the Planet Lab Dataset

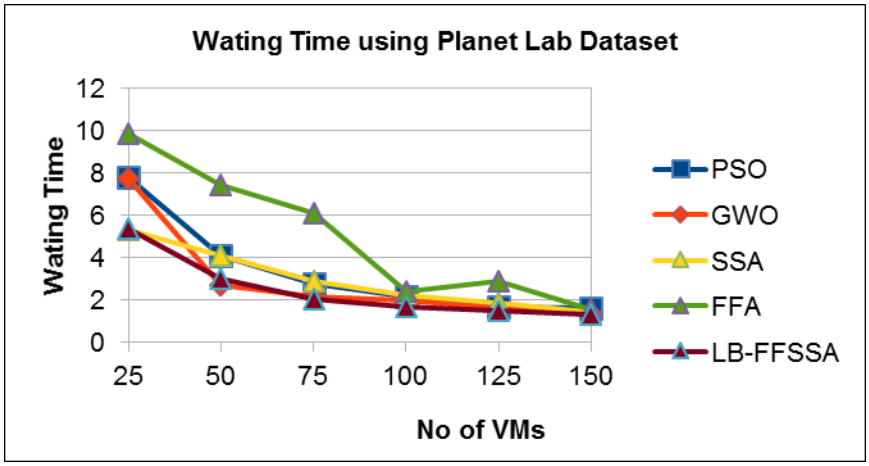

Figure 7 Waiting Time of Proposed and Existing Algorithm Using the Planet Lab Dataset

Table 6 Comparative Results of the Proposed and Existing Algorithm for QoS Metrics Using NASA Ames iPSC/860

QoS Metrics	No. of VM	Algorithms				
		PSO	GWO	SSA	FFA	LB-FFSSA
Makespan (in Sec)	25	1044.5	938.7	984.8	824.1	660.3
	50	622.6	910.3	704	677.8	589.4
	75	563.1	896.3	607.1	631.3	554.7
	100	542	888.5	588.3	650.7	539
	125	545.3	686.8	579.8	618.3	531
	150	509.4	691.7	550.1	566.4	500.9
		637.8	835.4	669	661.4	562.6

RESEARCH ARTICLE

Load Imbalance	25	277.31	160.86	269.33	328.71	150.7
	50	143.06	145.59	156.04	229.76	105.49
	75	106.64	132.47	118.81	188.94	70.46
	100	89.92	101.88	108.95	118.09	76.85
	125	81.05	70.68	86.9	107.3	68.67
	150	70.63	80.47	76.7	113.7	66.94
		128.1	115.3	136.1	181.1	89.9
Resource Utilization (in %)	25	0.453	0.44	0.48	0.246	0.546
	50	0.353	0.264	0.337	0.238	0.386
	75	0.262	0.153	0.243	0.151	0.267
	100	0.213	0.168	0.201	0.136	0.219
	125	0.162	0.149	0.159	0.124	0.164
	150	0.177	0.127	0.139	0.106	0.179
		0.3	0.2	0.3	0.2	0.3
Throughput	25	3.62	4.14	3.69	2.68	4.53
	50	4.33	4.11	4.13	2.9	4.68
	75	4.54	3.82	4.36	3.4	4.63
	100	4.44	4.16	4.34	4.43	4.49
	125	4.51	4.55	4.19	4.21	4.77
	150	4.52	4.3	4.26	4.14	4.8
		4.3	4.2	4.2	3.6	4.7
Waiting Time (in Sec)	25	23.88	18.24	22.81	31.54	17.93
	50	13.58	12.45	14.06	19.41	11.77
	75	10.91	11.38	12.6	10.94	9.9
	100	9.65	9.89	11.34	18.03	9.11
	125	8.79	8.76	10.07	10.76	8.38
	150	8.24	8.72	9.94	12.19	8.05
		12.5	11.6	13.5	17.1	10.9

RESEARCH ARTICLE

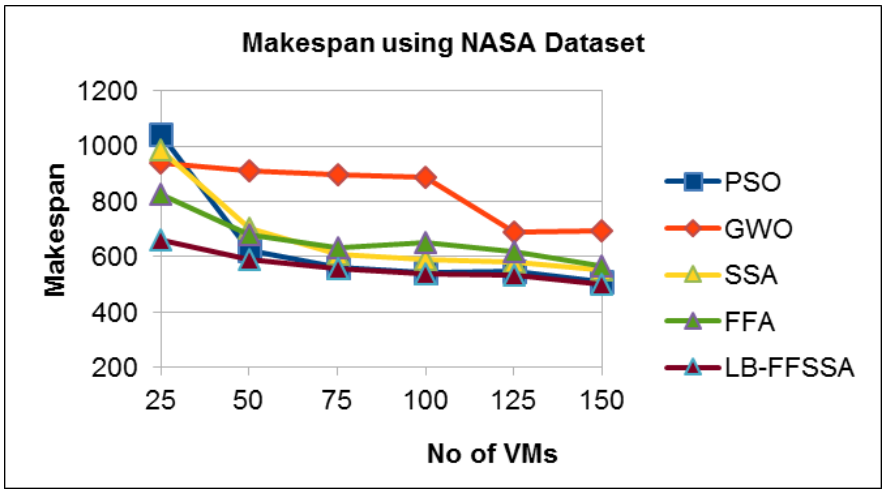

Figure 8 Makespan of Proposed and Existing Algorithms Using NASA Dataset

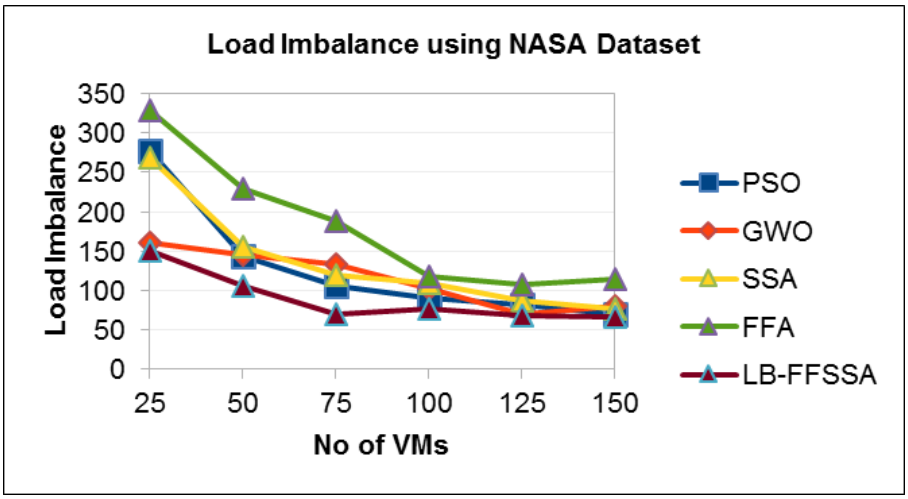

Figure 9 Load Imbalance of Proposed and Existing Algorithms Using NASA Dataset

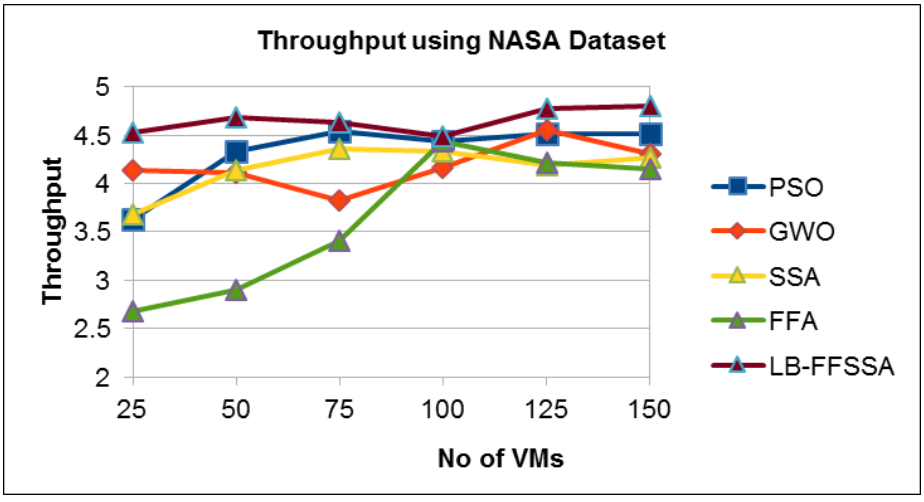

Figure 10 Throughput of the Proposed and Existing Algorithm Using the NASA Dataset

RESEARCH ARTICLE

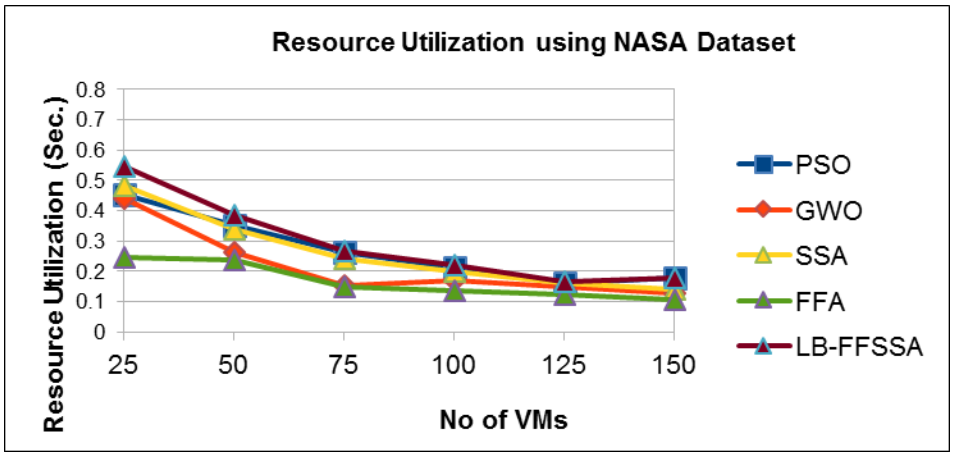

Figure 11 Resource Utilization of Proposed and Existing Algorithms Using the NASA Dataset

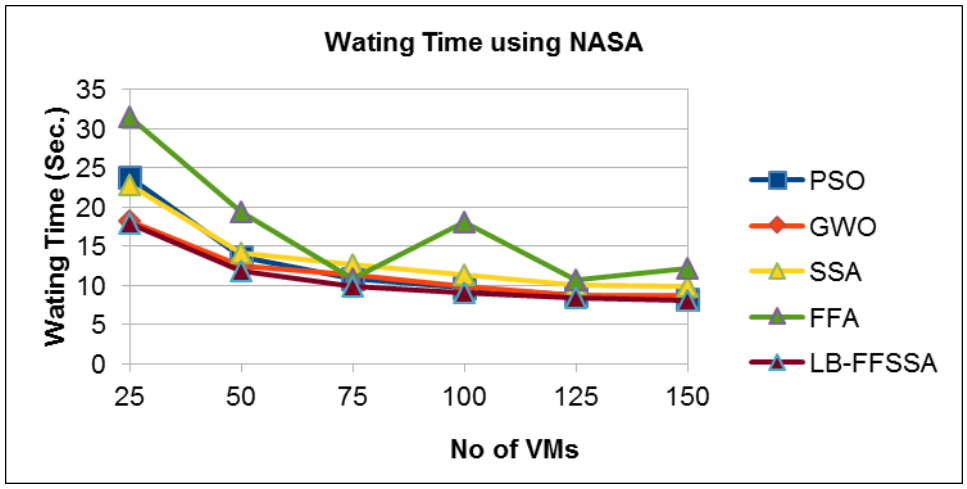

Figure 12 Waiting Time of Proposed and Existing Algorithms Using the NASA Dataset

7. CONCLUSION AND FUTURE WORK

This research work proposes an efficient LB-FFSSA algorithm in a cloud computing environment to achieve optimal load balanced task scheduling. This algorithm optimizes makespan, resource utilization, waiting time, throughput, and load imbalance factor. LB-FFSSA works in two phases 1) Task allocation is accomplished using proposed HFFSSA, i.e., hybridization of FFA and SSA. 2) Then, the best result is rescheduled using the proposed load-balancing (LB) heuristic. This load balancing is required in industries to meet the efficient resources utilization, which reduces resource wastage and helps to optimize costs. The HFFSSA is tested on 13 global benchmark functions, and it is proved that HFFSSA outperforms other metaheuristics. These benchmark functions are used to evaluate the performance of metaheuristic. If an algorithm performs well on these benchmark test then it can be used as effective approach to solve real- world problems. Further, the proposed LB-FFSSA is evaluated using CloudSim on two real workloads (Planet lab and NASA) and compared with other metaheuristics, i.e., SSA, GWO, FFA, and PSO. Simulation results proves that proposed model improves by an average up to 32.3%, LIF by 50.4%, throughput by 42.1%, resource utilization by 40%, and waiting time by 50%. In the future, the work can be extended by optimizing energy to reduce carbon footprint. Other QoS parameters such as, cost, reliability, and availability can be considered in future work.

REFERENCES

[1] Mell, Peter, and Timothy Grance. "Cloud computing: recommendations of the national institute of standards and technology." NIST, Spec. Pub (2011): 800-145.

[2] Dillon, Tharam, Chen Wu, and Elizabeth Chang. "Cloud computing: issues and challenges." 2010 24th IEEE international conference on advanced information networking and applications. Ieee, 2010.

[3] Moharana, Shanti Swaroop, Rajadeepan D. Ramesh, and Digamber Powar. "Analysis of load balancers in cloud computing." International Journal of Computer Science and Engineering 2.2 (2013): 101-108.

[4] Mahmud, Shahid, Rahat Iqbal, and Faiyaz Doctor. "Cloud enabled data analytics and visualization framework for health-shocks prediction." Future Generation Computer Systems 65 (2016): 169-181.

RESEARCH ARTICLE

[5] Masdari, Mohammad, et al. "A survey of PSO-based scheduling algorithms in cloud computing." Journal of Network and Systems Management 25.1 (2017): 122-158.
[6] Kalra, Mala, and Sarbjeet Singh. "A review of metaheuristic scheduling techniques in cloud computing." Egyptian informatics journal 16.3 (2015): 275-295.
[7] Zhou, Jincheng, et al. "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing." Journal of Cloud Computing 12.1 (2023): 1-21.
[8] Sakellariou, Rizos, and Henan Zhao. "A hybrid heuristic for DAG scheduling on heterogeneous systems." 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.. IEEE, 2004.
[9] Aissi, Hassene, Cristina Bazgan, and Daniel Vanderpooten. "Complexity of the min–max and min–max regret assignment problems." Operations research letters 33.6 (2005): 634-640.
[10] Pradhan, Pandaba, Prafulla Ku Behera, and B. N. B. Ray. "Modified round robin algorithm for resource allocation in cloud computing." Procedia Computer Science 85 (2016): 878-890.
[11] Eberhart, Russell, and James Kennedy. "A new optimizer using particle swarm theory." MHS'95. Proceedings of the sixth international symposium on micro machine and human science. Ieee, 1995.
[12] Mirjalili, Seyedali, et al. "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems." Advances in engineering software 114 (2017): 163-191.
[13] 13Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.
[14] Shishira, S. R., A. Kandasamy, and K. Chandrasekaran. "Survey on meta heuristic optimization techniques in cloud computing." 2016 international conference on advances in computing, communications and informatics (ICACCI). IEEE, 2016.
[15] Thakur, Avnish, and Major Singh Goraya. "A taxonomic survey on load balancing in cloud." Journal of Network and Computer Applications 98 (2017): 43-57.
[16] Ghomi, Einollah Jafarnejad, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Load-balancing algorithms in cloud computing: A survey." Journal of Network and Computer Applications 88 (2017): 50-71.
[17] Madni, Syed Hamid Hussain, et al. "An appraisal of meta-heuristic resource allocation techniques for IaaS cloud." (2016).
[18] Faris, Hossam, et al. "Salp swarm algorithm: theory, literature review, and application in extreme learning machines." Nature-inspired optimizers: theories, literature reviews and applications (2020): 185-199.
[19] Jain, Richa, and Neelam Sharma. "A QoS aware binary salp swarm algorithm for effective task scheduling in cloud computing." Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2019, Volume 2. Springer Singapore, 2021.
[20] Jain, Richa, and Neelam Sharma. "A deadline-constrained time-cost-effective salp swarm algorithm for resource optimization in cloud computing." International Journal of Applied Metaheuristic Computing (IJAMC) 13.1 (2022): 1-21.
[21] Jain, Richa, and Neelam Sharma. "A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing." Cluster Computing (2022): 1-24.
[22] Abualigah, Laith, et al. "Salp swarm algorithm: a comprehensive survey." Neural Computing and Applications 32 (2020): 11195-11215.
[23] Yang, Xin-She. "Firefly algorithms for multimodal optimization." Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, October 26-28, 2009. Proceedings 5. Springer Berlin Heidelberg, 2009.
[24] Park, KyoungSoo, and Vivek S. Pai. "CoMon: a mostly-scalable monitoring system for PlanetLab." ACM SIGOPS Operating Systems Review 40.1 (2006): 65-74.
[25] Feitelson, Dror G., and Bill Nitzberg. "Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860." Job Scheduling Strategies for Parallel Processing: IPPS'95 Workshop Santa Barbara, CA, USA, April 25, 1995 Proceedings 1. Springer Berlin Heidelberg, 1995.
[26] https://aws.amazon.com/ec2/instance-types/processing. Springer, Berlin, Heidelberg, 1995. (Accessed on 24 September, 2022)
[27] Ullman, J. D., NP-complete scheduling problems. Journal of Computer and System sciences, 10(3), 384-393. (1975).
[28] Singh, P., Dutta, M., & Aggarwal, N., A review of task scheduling based on meta-heuristics approach in cloud computing. Knowledge and Information Systems, 52, 1-51. (2017).
[29] Tsai, C. W., & Rodrigues, J. J., Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal, 8(1), 279-291. (2013).
[30] Kalra, M., & Singh, S. A review of metaheuristic scheduling techniques in cloud computing. Egyptian informatics journal, 16(3), 275-295. (2015).
[31] Jain, P., & Sharma, S. K. A systematic review of nature inspired load balancing algorithm in heterogeneous cloud computing environment. In 2017 conference on information and communication technology (CICT) (pp. 1-7). IEEE. (2017).
[32] Ghomi, E. J., Rahmani, A. M., & Qader, N. N., Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications, 88, 50-71. (2017).
[33] Thanka, M. R., Uma Maheswari, P., & Edwin, E. B, An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment. Cluster Computing, 22, 10905-10913.(2019).
[34] Ramesh, D., Dey, S., & Bhukya, R, Heuristic and fair-queuing based VM load balancing strategy for cloud data centers: A hybrid approach. Multiagent and Grid Systems, 15(1), 19-38. (2019)
[35] Mapetu, J. P. B., Chen, Z., & Kong, L. , Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. Applied Intelligence, 49, 3308-3330. (2019)
[36] Adhikari, M., Nandy, S., & Amgoth, T. , Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. Journal of Network and Computer Applications, 128, 64-77. (2019)
[37] Alguliyev, R. M., Imamverdiyev, Y. N., & Abdullayeva, F. J. , PSO-based load balancing method in cloud computing. Automatic Control and Computer Sciences, 53, 45-55. (2019)
[38] Hanine, M., & Benlahmar, E. H, A load-balancing approach using an improved simulated annealing algorithm. Journal of Information Processing Systems, 16(1), 132-144. (2020)
[39] Kruekaew, B., & Kimpan, W, Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. International Journal of Computational Intelligence Systems, 13(1), 496-510.. (2020)
[40] Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., & Chowdhury, M. U. , An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Computing and Applications, 32, 1531-1541. (2020)
[41] Neelima, P., & Reddy, A. R. M, An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. Cluster Computing, 23, 2891-2899. (2020)
[42] George, Shelly Shiju, and R. Suji Pramila. "Fractional IWSOA-LB: Fractional Improved Whale Social Optimization Based VM Migration Strategy for Load Balancing in Cloud Computing." International Journal of Wireless Information Networks 30.1 (2023): 58-74.
[43] Ramya, K., and Senthilselvi Ayothi. "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment." Transactions on Emerging Telecommunications Technologies 34.5 (2023): e4760.
[44] Abdelmaboud, Abdelzahir, et al. "Quality of service approaches in cloud computing: A systematic mapping study." Journal of Systems and Software 101 (2015): 159-179.
[45] Manupati, Vijaya Kumar, et al. "Near optimal process plan selection for multiple jobs in networked based manufacturing using multi-objective evolutionary algorithms." Computers & Industrial Engineering 66.1 (2013): 63-76.

RESEARCH ARTICLE

[46] Jain, Richa, Neelam Sharma, and Pankaj Jain. "A systematic analysis of nature inspired workflow scheduling algorithm in heterogeneous cloud environment." 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT). IEEE, 2017.
[47] Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization-An overview. Swarm Intelligence. Volume 1, Issue 1." (2007): 33-57.
[48] Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." 2009 international conference on high performance computing & simulation. IEEE, 2009.

Authors

Pankaj Jain received his B.E. in Computer science and Engineering from Rajasthan University, India in 2007 and M.Tech. in Computer Science and Engineering from Rajasthan Technical University, India in 2014. He is currently pursuing his Ph.D. in Computer Science and Engineering in Banasthali Vidyapith, India. His research interests include cloud computing, Soft computing, Algorithms, etc.

Sanjay Kumar Sharma, PhD is a Professor at Banasthali Vidyapith (NACC A++ Grade University), Rajasthan, India. He obtained PG degree from HBTI Kanpur, and Ph. D. in Computer Science from Banasthali University. He has published many research articles in reputed international journals and conferences. 5 students completed Ph. D. degree under his supervision. He has teaching experience of 15 years. His research interests include Parallel Computing, Cloud Computing, Algorithms Design and Machine Learning. He is a senior member of educational BOS, Academic Council etc.

How to cite this article:

Pankaj Jain, Sanjay Kumar Sharma, "A Load Balancing Aware Task Scheduling using Hybrid Firefly Salp Swarm Algorithm in Cloud Computing", International Journal of Computer Networks and Applications (IJCNA), 10(6), PP: 914-933, 2023, DOI: 10.22247/ijcna/2023/223686.