ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

A Hybrid Approach of Load Balancing in Cloud Computing by Optimization of Metaheuristic Techniques: An Execution Assessment

[1] Athokpam Bikramjit Singh*, [2] Rio D'Souza

[1][2] St. Joseph Engineering College, Mangaluru, India
Email: abikramjit@gmail.com

Abstract*— In recent days, cloud computing has become one of the most encouraging area for technical development. Cloud computing is reflected as the paramount in information technologies and it serves the delivery of services to the user via web/Internet depending on the request from the user and also based on the instant payments method. Main challenges and important aspect for research in this area is load balancing. Load balancing is the most important factor for good system performance as well as for the stability and reliability of the system. Therefore, it is very essential to have an effective load balancing techniques for user request scheduling based on services request parameter. Here we focus on a hybrid method of load balancing through metaheuristic techniques (IPSO & Firefly) so that available resources can be effectively used thereby reducing response time, waiting time, at the same time keeping the system stable and reliable.

***Index Terms*— cloud computing, Load Balancing, Metaheuristic techniques, Job scheduling, Response & Waiting Time and Reliable**

I. INTRODUCTION

Cloud Computing environment are commonly derived from grid computational techniques, distributed computational techniques and virtualization techniques. It provides major benefit in storing media with widespread entree platform, minimum requirement of hardware with the client. Many issues are there in cloud computing, one of the major issue is load balancing scalability, performance, greater accessibility and amount of carbon emission. Among all this, load balancing has been the major challenges in cloud computing environment [1, 2, 3]. Hence, in our research a new hybrid approach of load balancing for cloud by optimizing hybrid method technique has been propose. This research focus on evaluation the performance of the system using hybrid approach with different parameters.

1.1 Problem statement

Many types of load balancing are there in computer field such as network load, memory load and others. It is the process to enhance the uses of resource thereby diminishing the response time by balancing the load at every node in cloud. Relocation of load to some specific node in the system is required for better uses of resource available and enhancing reply for task thereby minimizing the condition where some node are overloaded and vice-versa. One of the major challenges is such application is to go for better performance or retain the same where there is an outburst in data accessing request. In some application, one area of search can become hot search topic area due to some incident taking place, and the heterogeneous node with different search or computing are becoming imbalanced. Many important aspects are to be considered such as load estimation, stability of the system, load comparison, performance of the system, node to node interaction, node selection and many more to consider while developing diverse algorithm.

In cloud computing, the load balancer is an important part in order to make the resources are available with better performance. In complex and huge system, there is always necessity of load balancing to maintain balance and make simpler in the cloud system. Load balancer implement definite algorithm for making load balancing decision. Is decision specifying the remote server on allocating forwarding new task? Based on environment, Load balancing algorithms is categorized into two kinds that is static and dynamic [1]. The classification is sown in the fig. 1.

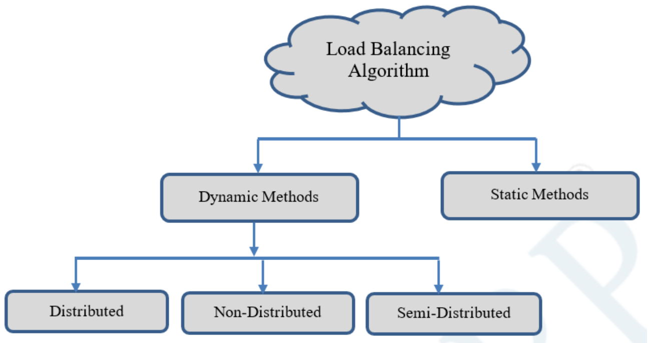


Fig 1. Classification of Load Balancing Algorithm

Static approach is suitable for those environment in which the variation of load is minimum. In such approach the load is equally distributed among the server or node. Previous

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

knowledge of the system resource are required by static algorithm. Here performance of the processor is already decided during process execution, therefore transferring load to other is not determine on the basis of present state of the system. Some disadvantages is that all the task are allocated to the node/server only after it is created and once it is allocated than it cannot be reallocate and shift to other node/server during the execution in order to mention load balancing in the system[1].

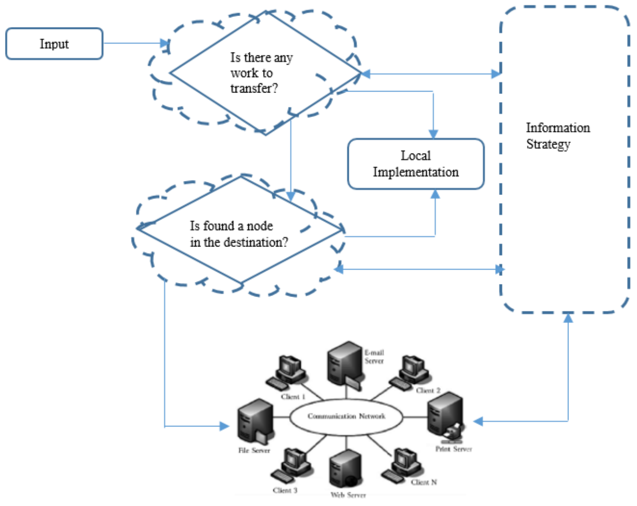


Fig. 2 Dynamic algorithm of load balancing

Dynamic approach keeps on monitoring system workload to check whether it is required to reallocate the work for better service or to keep the load balanced within the system. Such approach consists of three strategies: location strategy, information Strategies and the transfer strategy. Location strategy determines that a remote note in order to handle the transfer ask. Information strategy keeps on monitoring the load at each node/server and pass the information to the load balancing algorithm that is to the load balancer. Transfer policy decide which tasks are qualified to transfer to other node for further processing or completion of the task at the earliest [4]. Fig. 2 demonstrate the strategies.

In figure 1. Presented that dynamic load balancing achieved in three diverse ways: distributed, non-distributed and semi-distributed method. In distributed, all note are share with the distribution of the request. While in non-distributed, one centralized node received all the incoming request and distributed them to the server. In case of semi-distributed approach the nodes are grouped into many cluster where each cluster will work like a significant nodule to allocate the request and every cluster will be answerable for load sharing for maintaining even load across the system [4].

Cloud environment, plenty of algorithm are there for maintaining balance to the load. A number of them are listed, tab. 1.

Tab 1. Various Load Balancing algorithm with their advantages and disadvantages

Algorithm	Type	Benefits	Drawbacks
Round Robin(RR) [6]	Static	Resource utilization in a uniform order Ensure fairness.	Virtual machine load is considered Load distribution to all the nodes are not uniform
Opportunistic load balancing (OLB) [6]	Static	Enriched routine.	Time consumption
Min-Min [20]	Static	Enhances work efficiency. Less time Consumption. Suitable for small task	Starvation.
Max-Min [20]	Static	Best if requirement information is known earlier	Time consuming
Weighted Round Robin [5]	Static	easy to implement using circular queue techniques Enhanced in response and processing with respect to Round Robin algorithm.	No uniformity in the state of the nodes
The Randomize algo. [5]	Static	Suitable when uniform load is distributed	Performance is not up to the mark
The Cenrl. Mngr. Algo [5]	Static	Less price for message	Single point disaster.
Hash IP Algo. [5]	Static	Appropriate for specific Task Simple execution	No uniformity in Internet Protocol Address Load distribution is not uniform
ACCLB [6]	Dynamic	Eliminates diversity and rely to active situations.	Better output

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

Honeybee Foraging Algo [30]	Dynamic	Fault tolerance. scalability Quick response time	Time consuming for lesser importance task
BRSLBA [6]	Dynamic	Increases throughput. Maintain balance by random selection.	Less Performance
Active Clustering Algo. [6]	Dynamic	High speed Throughput Increased	Performance reduces in diverse system
DRR [5]	Dynamic	Minimum Reply period	additional traffic flow
TLB algorithm [5]	Dynamic	Efficient in term of cost cutting	Requirements of software to capture process state Woks well only when hardware configuration is similar for all VMs
The Greedy algorithm [5]	Dynamic	Best suited for heterogeneous environment Can use greedy heuristic task allocations.	Lack of Optimal Solution
Ant Colony Optimization [20]	Dynamic	Throughput Minimizes Makespan.	Time consuming search techniques Very Complex

Generally, the preceding schema of given node is not considered during dispensing the load in static load balancing algorithm [2]. So it is simple to execute load balancing but it is not done effectively [2]. Opposite to this, dynamic approach check for the previous state of node as well as the current schema of the node during allocation of load, i.e. processor, memory, CPU, network and so on [3]. Dynamic algorithm is always difficult in implementing however it maintain balance in the system effectively [5].

Dynamic load balancer keeps on monitoring the load on each and every node/server and whenever there is imbalance of load in the system and raised to certain predefined level there, than redeployment of the node take place. The redeployment of load may take extra overhead during the exhibition time [2]. In order to eliminate such problem in the dynamic load balancing algorithm, Meta-heuristic algorithm could be used. So here we identify the hybrid approach algorithm for node balancing by optimizing hybrid and Meta-heuristic techniques to distribute the task and request between cluster for better scheduling and optimization and to have better performance in the system.

1.2 Influence

Considering significance of emerging innovative technique to minimize the given problematic scenario, we extant a novel hybrid technique through the subsequent study problem.

1. What elements effect load balancing?
2. What are the load balancing characteristics in quality of service?
3. How optimization of load balancing can rise throughput of a system.

As discussed earlier, suitable and ideal resource uses, such as processor, memory, time are the main challenges of load balancing. It is the scheme of allotting the load among different node in a distributed environment for improving the usage of resources and job scheduling. Appropriate arrangement for load balancing significantly reduced cost using tasks division between computers and better supply of resource. This state of the system in cloud environment will maximize user satisfaction, increase resource utilization, minimizes response time and increase in system performance [1].

Meta-heuristic algorithm is the most effective algorithm among the load balancer in reducing limitation of dynamic approaches. IPSO is the most suitable algorithm that works excellent in solving the optimization problem. This particular procedure is motivated with mutual movement of birds when they try to touch their endpoint. It worked with set of solution call group and every solution has a bird as answer called as particle. Every particle is associated with appropriate value that is derived from fitness function[6, 7]. Algorithm inspired by meta-heuristic is a Firefly algo, motivate by the characters of Fireflies. Fireflies' shows unique activity by which it passes communication, attract participation and gives warning for predator. This algorithm assure that all fireflies are unisexual. Firefly invite to each other and their effectiveness is openly correlation with the illumination of each fireflies. Brighter firefly will appeal to the lesser brightness one, more towards them, if no firefly brighter than other then they moves randomly [8].

In tab. 1 shows different algorithm of meta-heuristic in order to solve the challenges in cloud environment.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

Tab 2. Different methods of load balancing

Algorithms	Methods	Outcomes
ASDA [13]	Display a modern arrangement of distinctive calculations i.e. Hadoop mapReduce stack adjusting category, Characteristic Occurrences-created stack, network-conscious group, and workflow-precise grouping.	Loadbalancing strategies are centring on dual basic measurements i.e. energy sparing, lessening carbon production.
ASDA [13]	Different load balancing procedure for cloud situations are examined Carefully and methodically agreeing to a unique categorisation.	Prevailing procedure consume different confinements. Thus there's an amazing scope for improvement.
PEFT Heuristic, ACO Meta-heuristic, HEFT heuristic [14]	A cross breed tactic-built on resources and load adjusting system for workflow implementation to improve the uses of VMs for equal contribution.	Hybrid PEFT-ACO method provides enhanced outcomes.
LB-RC	Recommend modern load adjusting approach to discover ideal group of system for assignment of jobs to adjust the load to maintain reliability.	Able to create a job scheduling arrangement with lesser execution fetched without compromising QoS.
MRS	The strategy builds a Fuzzy-based Multidimensional resource Planning demonstration to get resource planning effectiveness in infrastructure services.	It accomplished improved execution in rapports of normal victory proportion, Resource scheduling effectiveness and reaction time

In scheduling. Our study, represent a hybrid approach of load balancing and task scheduling by optimizing meta-heuristic techniques. Following are the main contribution of our studies.

1. Minimizing the search of optimal response to Firefly algorithm applying IPS0 or algorithm for selecting best optimal response.
2. Consider highest number of parameter for evaluation in our propose hybrid method
3. This hybrid approach achieved in finding the best ideal period in finding the outcome.

II. LITERATURE REVIEW

2.1 Different methods of load balancing

The Rapid growth of internet technology and its uses such as cloud computing has drastically improved and expanded. Cloud computer center is considered as an instrument to give response to the clients need or request [9, 10]. User are able to use service through internet, based on the location dependency, also in those areas wherever the dynamic facility of resource plus application of virtual technologies are significant [11]. Many problem has not been resolved in cloud computing so there is scope of improvement. One problem is load balancing. In certain network, some tasks related cloud computing and load balancing method have been used. Research done so far in this area is given in tab. 2.

In [12], task scheduling and load balancing algorithm have been studied before presenting a novel classification of those algorithm. Such as natural phenomena algorithm, hadoop-mapreduce algorithm, general load balancing categories, agent-based balancing categories, and application based categories, workflow specific categories and network aware categories. The research tells that these techniques classify two critical metrics i.e. reducing the carbon dioxide emission and saving energy.

In [13], propose a framework based on hybridization of meta-heuristic technique to achieve the optimal performance related with makespan and cost by optimizing the misuse of VMs in the system having the similar load allocation. Suggested dual hybrid approach for HDD-PLB structure which predict earliest finish time (PEFT). Heuristic with ant colony optimization (ACO), meta-heuristic (HPA) and hybrid heterogeneous earliest finish time (HEFT) heuristic with ACO. This load balancing approach has been compare and analyzed to define which is the best for propose HDD-PLB framework.

In [14], innovative load balancing routine for discovery of best set of servers for Job management and balancing the load in long run. Proposed algorithm is LB-RC. This procedure describe in four parts that is clustering, merger of cluster, optimization of response and job allocation principle. Further suggest one more dynamic job allocation rule for obtaining smaller makespan and implementation charge with available condition. Simulation results of their propose LB-RC algo is superior to prevailing state-of-art-algo. The offered algo is capable of producing job assignment distribution strategy with low performance charge with quality of service.

In [15], the author propose a method to construct a fuzzy-logic which work with may dimension of scheduling process for achieving better effectiveness in resource scheduling in IAAS environment. Efficiency in VMs utilization increases over active and reasonable load balancing algorithm. This can be accomplished by randomly

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

choosing a demand from the user using best option among other through load optimization techniques. The kind of algorithm is applied to escape over utilization and vice-versa thereby increasing execution period of every demand. Virtual reality show that the offered technique gives better routine while comparing with avg. success ratio, response period and resources efficiency.

In [16], an organized investigation of state-of-the-art LB is suggested in cloud computing. Investigation follows an innovative nomenclature of LB algorithm in cloud network. Comparison of many LB algorithm are accessible in the survey. Result shows that there are plenty of limitation in the existing load balancing algorithm such as static threshold, resource and energy waste and insufficient observing frequency. So there are scope of improvement in load balancing algorithm. There are efficient and adaptive load balancing algorithm to be implemented to make best use of resource consumption, performance, energy saving to deliver eminence of service to user at least price.

2.2 Load balancing method

For any LB algo. balancing system load and dispense the load similarly, therefore LB is considered as a vital factor that need to be address in the cloud computing environment in order to give best service from the service provider point of view [2]. Cloud policy gives load balancing to user with low price and unrestricted resources. Load balancing distribution of request and response is done through the distribution of software at datacenter. Tab. 3 represent outline of LB techniques.

Tab. 3 Overview of load balancing techniques

Technique	Outcomes	Algorithms
Load balancing using job gathering magnitude modification[2]	Excellent convergence ratio, enhanced parallel execution	FIFO, Genetic algorithm, Hill climbing
Precision determined by number of requests and current jobs. [7]	Better-quality LB, inexpensive response period and improved working ability	LABA
Suitable mechanism to works on requirements, keeping resources on physical devices. [11]	Improvement in runtime Less price in work execution	K-P, Bin packing and management algorithms
Even sharing of load among VMs. [3]	Better LB and response	Round Robin , AVLB,TVLB
Discovery of finest position for VMs allowing to the necessities and ecological situations. [12]	Parallel Execution Best convergence promptness	Genetic Algorithm
Elasticity in solutions. [5]	Better load balancing Best response	CELBT, ACO
Dynamic procedure to generate different versions of VMs and discovery of ideal response and reduce energy. [13]	Efficient in energy saving	Floyd-Warshall
Estimate the load of each VMs and avoid the extra load per node. [6]	Decrease in communications costs confirm system consistency	LB-SD
Novel technique for handling cloud resources and migrating VMs creation to decide on load standards norms. [15]	Decrease in space intake Less energy Consumption	AC algorithm and holistic cloud resource management procedure
Created on ant algo. with smallest distance [10]	Discovery of best route in less time	ACO
Supplying unrestricted services to end user. [9]	improved operational capability	Round Robin, min-min and max- min

Genetic algorithms experimental result shows enhancement in the procedure when linked with FIFO and Hill Climbing procedure. This was very slow operation in lesser space and advantages is extraordinary convergence rate and very efficient in multi-processing.

In [17], even workload sharing between VMs focused on hybrid approach of TVLB & AVLB is propose. Virtual Machine remain distributed into two kinds that is busy and unemployed. Comparative result with rotational and ECSE algorithm shows progress in load balancing and response time. In [18], discuss a result oriented on Bee colony algorithm. Result shows better in maximization of operating power, better virtual machine load balancing, and run time and response time minimization. In [19], provide techniques to evaluate the workflow time, response time, load per Virtual Machine using CELBT algorithm. This work in 3 phase i.e. Virtual Machine creation, deployment of VM and randomly selecting the VM if uncertainty in job executing. Result shows improvement while comparing with other methods. Main advantages is in finding optimal result in less time and advantage is sensitiveness against other parameter.

III. OUR METHOD

Our strategy is created by hybridization of IPSO and firefly so that we can get the best initial population and best execution rate. In general, firefly is constructed on three fundamental speculations [17];

1. Firefly are single-sex. In any case, they will attract to each other.
2. Firefly shows interest with respect to brightness, less

illumination fly towards more illuminating one which results into the movement towards it.

3. Objective function is used to define the illumination rate of firefly.

The pseudo code of algorithm is given in algorithm 1.

```
Algorithm 1.
1. Begin
2. Initialization of parameter
3. Assign maximum generation to MG
4. Define f(x) where x=x1,x2,x3,…,xn
5. Create initial inhabitants of firefly i.e xi=1
6. Define intensity of light w.r.t f(x)
        while target lesser than maximum generation
            Assign 1 to i till the last
                Assign 1 to j till the last
                    If luminosity of j greater than i
                    Move j to the target
                    attraction depend on distance and value of –yr2
                    Calculate and inform intensity:
            Select best result
7. Display result;
8. End
```

A firefly's illumination is taken into account while determining its allure. The brightness and attractiveness may be defined after consideration of their illumination and lure (Fig.3). The intensity I0 and attraction β0 are determined through a series of calculations, which can be done in several ways.

$I = I_0 e^{-\gamma r^2}$

$\beta = \beta_0 e^{-\gamma r^2}$

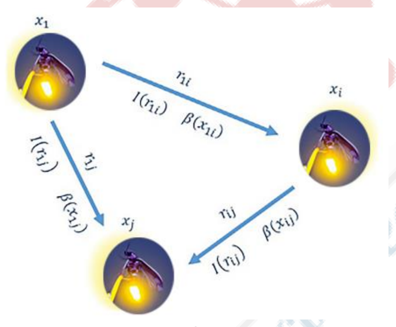


Fig. 3. Firefly attraction

The space amongst two fireflies is presented by Euclidean equation as follows:

$$r_{ij} = \|x_i - x_j\| \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

The movement of the firefly i towards j is given by

$$x_i = x_i + \beta_0 e^{-\gamma r^2}(x_j + x_i) + \alpha(rand - 0.5)$$

In this situation the fly can fly based on three conditions:

- Present firefly situation
- Moving towards high light intensity by absorbing values
- Randomly can move through random motion value α in a uniform interval.

Choosing initial population is one of the most important factor for our research for achieving better response or performance. For this we proposed the technique for initialization of finest populace using firefly algorithm which is given in fig. 4.

Evaluation function is given below:

$$1\text{-} \left(\alpha_{ti} \times \frac{t_i - t_{min}}{t_{max} - t_{min}} + \alpha_{ci} \times \frac{C_i - C_{min}}{C_{max} - C_{min}}\right)$$

t_{min}: Min runtime

t_{max}: Max runtime

c_{min}: Min input period

c_{max}: Max input period

Here, ideal execution order was determined by evaluating each executable process. Thus, the IPSO algorithm has been utilized to schedule various jobs. The evaluation function is given below:

$$1\text{-} \left(\alpha_{ti} \times \frac{t_i - t_{min}}{t_{max} - t_{min}} + \alpha_{ci} \times \frac{C_i - C_{min}}{C_{max} - C_{min}}\right)$$

Proposed IPSO inward process operation is through algo. 2.

O(2(t*n)) is time complexity of our suggested method; where t is iteration, n is best starting population.

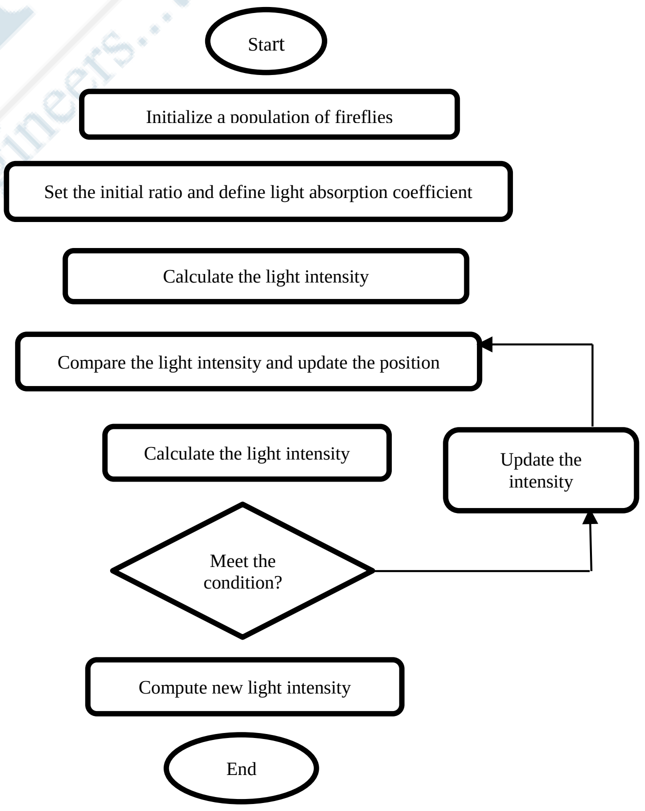

Start
Initialize a population of fireflies
Set the initial ratio and define light absorption coefficient
Calculate the light intensity
Compare the light intensity and update the position
Calculate the light intensity
Update the intensity
Meet the condition?
Compute new light intensity
End

Fig. 4. Firefly algorithm

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

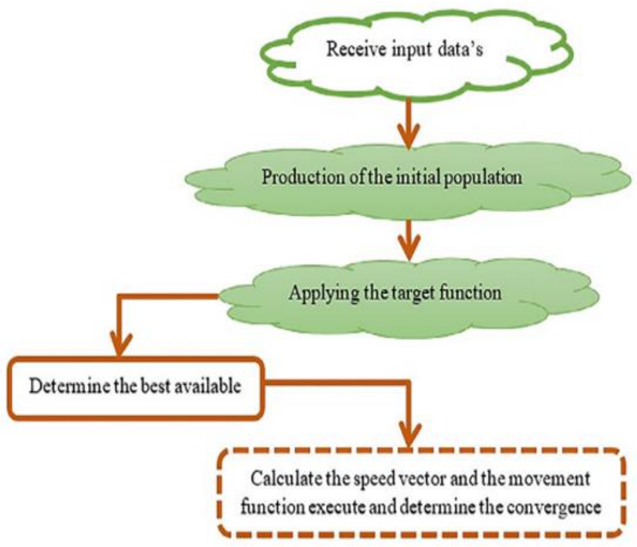

Receive input data's
Production of the initial population
Applying the target function
Determine the best available
Calculate the speed vector and the movement function execute and determine the convergence

Fig. 5 IPSO algorithm for selecting best initial population

Algorithm 2. IPSO pseudo code

```
1. Begin
2. Enter tasks
3. Call algo. 1 to identify finest populace
4. Assign first to I and repeat till the last
      set initial populace to Xi through algo. 1
      assign Xi to Xpbi
      assign Xi to Vi or Zero to Vi
      assign f(Xi) to Fi and Fi to pbi
      if Fi is lesser than Gb
         assign Xi to Xgb
         assign Fi to Fb
         if Fiis greater than Gb and Fi is lesser than from all Xf
            assign Xi to Xsb
         end of if
      end of if
repeat
   Initialize i to 2 till n
   Calculate vi
   If Vi is not equals to Va
      correct Vi;
   end of if
   increment Xi by adding Vi
   assign the value of f(Xi) to Fi
   if Fi I slesser than pbi
      assign Xi to Xpbi and Fi to pbi
   end of if
   if Fi is leaser than gb
      assign Xi to Xgbi and Fi to gb
   end of if
5. Until last
6. Obtain finest location
```

$$T_{resp} = \frac{\sum_{j=1}^{n} r_j}{n}$$

IV. VALIDATION APPROACH

Different mechanism and metrics are required in order to decide the system load for managing the relocation of task to maintain balance in cloud such as Process speed, throughput, response time, workflow, waiting time and performance etc. Quality of service also plays very important factor, so this parameter also we consider in our study.

Following are the metric considered in order to describe our parameter in details:

I	:	Index for task
N	:	Total no. of task
R	:	Required no. Of resources
rt	:	First response time of task I
rwt	:	Waiting time if I
lI	:	size (length) of I
(tRT)K	:	Total time of K resource to complete I
i	:	VM index
n	:	VM number
vi	:	ith VM load
tm	:	Time cycle [(t1-t0),(t2-t1), … , (tn-tn-1)]
Cvi	:	Processor (CPU) capacity of ith VM
uRt	:	Resource utilization rate
Ur	:	User Request (demand)
λ(t)	:	Request time interval
Mbot	:	Makespan bag of task
CTj	:	jth time taken in bot
Fdu	:	File Transfer Protocol download and upload in kb in one session
Tdu	:	Duration for download or upload of a task(I/O)

4.1. Parameters of load balancing:

The following are some of the most significant parameter which we have considered in our research.

1. Response Time: Time in use by system in order to give first response to client/user. The average response time is given by

$$T_{resp} = \frac{\sum_{j=1}^{n} r_j}{n}$$

2. Turnaround Time: Here metric is given as

$$T_{Turn} = \sum_{j=1}^{n} \left(\frac{rwt_j + l_j}{\sum_{k=1}^{m} (+Rj)^k} \right)$$

3. Server Load: average load at the server is define as

$$T_{load} = \sum_{i=1}^{m} v_i (t_i - t_{i-1})$$

4.2. Quality of Service parameters:

There are many parameter or metrics to be considered for defining the QoS. Some of them are given below:

- Execution Time: total time taken by a task I for its

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

complete execution. It is given below when we consider the virtual machine v_i [23].

$$T_{execute} = \frac{l_j}{cp_{vi}}$$

- Resource Utilization: It is the time taken by the tasks for every resource from the cloud server in order to complete its task. Sometimes rescheduling require in order to utilize the resource effectively. Better utilization of resources in cloud environments may execute many tasks in parallel. Resource utilization is given as

$$U_{res}(t)=D_{res} * \lambda\ (t)$$

- Reliability: One of the important factor for quality of service. If there is no reliability in the cloud service then user or client will not take up any service anymore. The reliability R is define as

  T_r=Period * Brustness * Loss * Corruption where Period = r+ , Brustness=r+, Loss=n0, Corruption = {r ε r | 0<= r<= 100}[20]

- Throughput: Number of jobs executed per unit time. Throughput is determine as follows

$$T_{through} = \frac{\sum_{no.\,of\,demand} dt}{\sum_{no.\,of\,demand} sd}$$

- Makespan: Total completing period for processing set of task in a system. Makespan of the bags of tasks is given as

MBoT=max{CTj}

- Objective:
  Min fresp = Tresp
  Min fta time = Tta
  Min fl = Tl
  Min fms = MBoT
  Min fexe time = Texe
  Max fru = ru(t)
  Max frel = Trel
  Max ftp = Ttp

Constraints: ur(t) >= 0
t0 <= rwti <= tn-1
ri < CTi
rwti <CTi

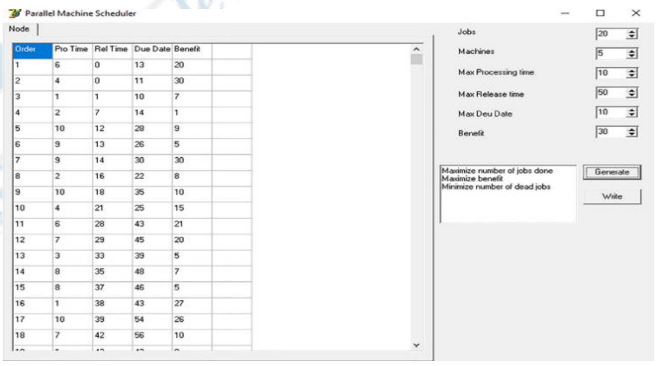
4.3 Validation procedure

The main aim of our algorithm is to bring out an appropriate scheduler for input task at the same time minimizing the server load with IPSO and firefly algorithm and provide solution to such problem by applying optimization in algorithm. Different inputs have been considered and implanted using MATLAB with each individual task login time and run time by randomly assigning values using shop job software (fig. 6) Initial parameter values for implementation is given in tab 5.

In fig. 7 gives ideal initial population at 16th iteration which is best scenario for executing IPSO algorithm and it gives ideal solution for best load balancing in the system. Initial values for implementation is given in tab. 6. With this algorithm the best particle for our problem is achieved at 70th iteration. In tab. 7 records of 10 individual run time is shown.

We have considered four scheduling method with our approach namely RR, FCFS, SJF, GA. In recent years meta-heuristic task scheduling has been widely use therefore we use and simulate it again using the parameter presented in it. Tab. 8 shows the parameters we considered.

Results are presented in tab. 9. From the table we clearly came to know that Firefly with IPSO have better load balancing than rest of the methods. Better load balancing result is achieved using hybrid approach of IPSO and firefly and it's shown in tab. 9. The most important part of our research is the initialization process. Evolutionary algorithm are very sensitive to the initial population consideration especially with IPSO algorithm. Such algorithm are suitable to implement when we have the best initial population initialization. Due to this reason we get better result while comparing with some of the other methods which are already exist till date. Also we compared our method with different quantity of virtual machine. In order to do so we have run our algorithm min 15 times and the average of it has been considered and given in tab. 10.



Order	Pro Time	Rel Time	Due Date	Benefit
1	6	0	13	20
2	4	0	11	30
3	1	1	10	7
4	2	7	14	1
5	10	12	28	9
6	9	13	26	5
7	9	14	30	30
8	2	16	22	8
9	10	18	35	10
10	4	21	25	15
11	6	28	43	21
12	7	29	45	20
13	3	33	39	5
14	8	35	48	7
15	8	37	46	5
16	1	38	43	27
17	10	39	54	26
18	7	42	56	10

Jobs	20
Machines	5
Max Processing time	10
Max Release time	50
Max Deu Date	10
Benefit	30

Maximize number of jobs done
Maximize benefit
Minimize number of dead jobs

Fig. 6. Interface of Job Shop software

Tab 5: Firefly Algorithm parameter setting

Parameter	Value
Population Count	60
Iteration	120
˥	0.10
β0	.99
α	Random

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

Tab. 6: IPSO Algorithm parameter setting

Parameter	Value
Population Count	60
No. of iteration	120
w	.99
Coefficient decrease in %	.98 %
c1,c2,c3	1

Tab. 7: Result from our proposed method

Runtimes	Firefly avg. initial load population	Average load using IPSO algorithm	No. of entries
1	0.43	0.206	55
2	0.45	0.253	55
3	0.49	0.272	102
4	0.56	0.229	102
5	0.42	0.223	205
6	0.33	0.23	205
7	0.53	0.213	510
8	0.56	0.262	510
9	0.42	0.264	1020
10	0.45	0.234	1020

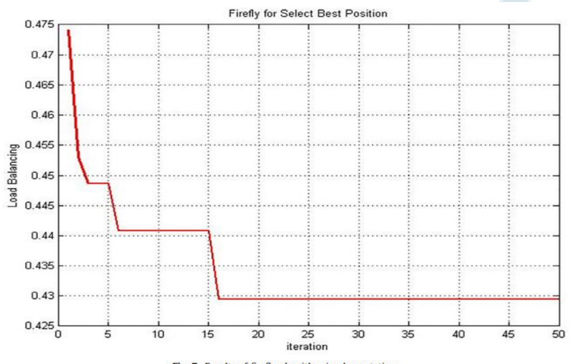


Fig. 7. Results of firefly algorithm implementation.

Tab 8: Genetic Algorithm Parameter Settings [21]

Parameter	**Parameter settings**
Replacement	1: child replace parent 2: Elitist Method
Size of population	30,40,50
Selection of parent method	1: Random 2: Roulette-Wheel
Crossover rate	0.71,0.79,0.91
Crossover type	1 & 2
Mutation rate	0.05 & 0.99
Mutation type	1, 2 & 3

Tab 9: Comparisons with other scheduling methods

Method	Average Load (ms)	Average TA time (ms)	Average Response Time (ms)
Round Robin	0.44	42.05	29.99
First come first Serve	0.46	41.92	30.95
Shortest Job First	0.499	41.58	30.5
Genetic Algorithm	0.315	26.59	20.32
IPSO	0.451	58.72	50.51
Firefly	0.472	55.55	50.03
IPSO-Firefly Algorithm (Our Paper)	0.281	23.19	16.20

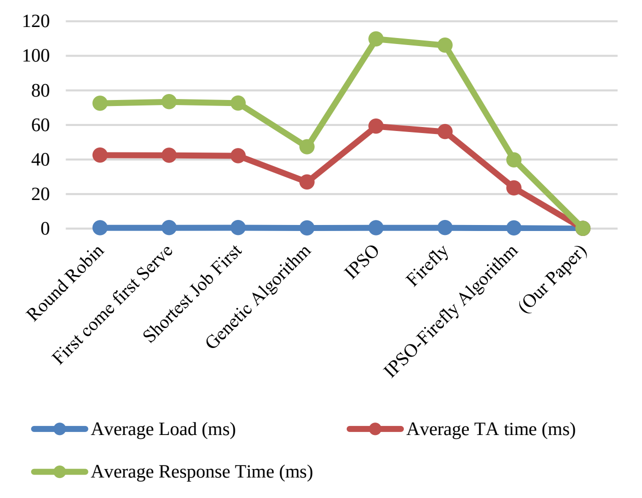


Fig. 8. Comparisons of Scheduling methods

By offering QoS, cloud computing service companies can draw clients and increase their revenue [22]. Given the significance of sustaining QoS, we also contrasted our suggested approach while considering [23] QoS constraints. This reference has been used for assessment since it offered sufficient QoS data for assessment, and since these constraints are of importance for our study. The datasets used in that paper are also used in this paper. In [23]. the dimensions of the various synthetic datasets, given in numbers of related tasks are displayed in Tab. 11. The experiment uses run-time random generation to determine the job sizes. And Millions of Instructions are used to indicate its size (MI). Also, the 80 servers, each with a different load and resource capacity.

Each server runs a variety of virtual machine instances with varying CPU, memory, and cost specifications, as shown in Tab. 12 [23]. This study compares the findings of the suggested method's simulation to those from [23]. The effectiveness of the suggested approach is assessed using a number of QoS metrics, including execution, resource consumption, Dependability, Makespan, Throughput etc. The assessment's findings are listed below.

Tab 10: Comparisons with different number of virtual machines

Method	No. of Virtual Machines				
	50	100	300	600	1000
IPSO-Firefly	57	101	120	321	560
IPSO	61	112	125	335	581
PSO	68	122	155	385	605
Firefly	70	123	160	390	621

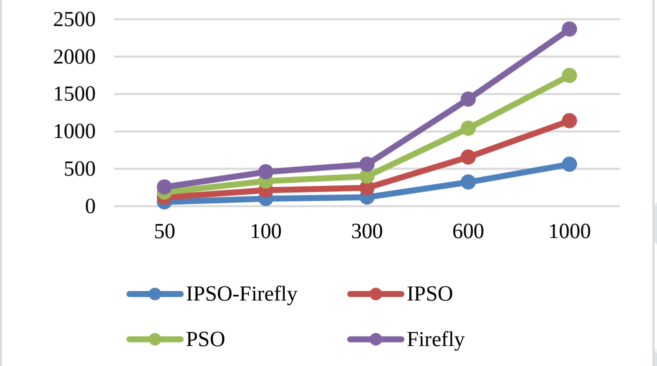


Fig. 9. Camparisons of scheduling methods with Variable VMs

Tab 11: Synthetic database size [23]

Job kind	No. of jobs	Size (MI)
Smaller	100-350	25000-45000
Average	351-500	45001-75000
Larger	501-800	75001-100000
Huge	801-1000	100001-20000

Tab 12: VM instances types [23]

Name	Capacity of CPU in MIPS	Memory Capacity in Gigabytes
Smaller	15000	6
Average	20000	12
Larger	30000	15
Huge	40000	18

Tab 13: Avg. waiting time- dataset 1.

Method	Job types			
	Smaller	Average	Larger	Huge
RD	32	55	87	112
WRR	29	47	80	103
DLB	26	41	75	95
LB-BC	23	37	64	89
LB-RC	17	31	62	81
IPSO-Firefly	17	30	60	78

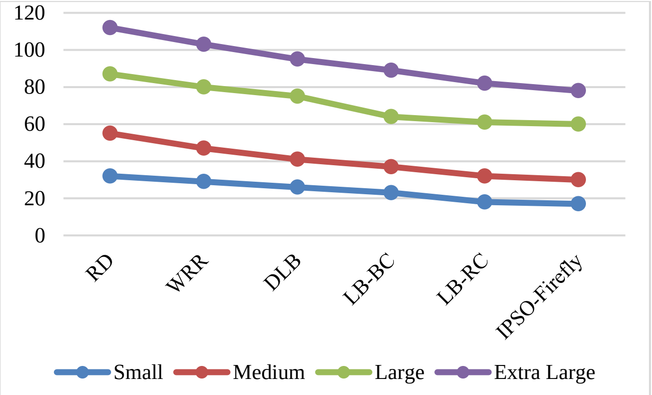


Fig. 10. Average waiting time based on dataset-1

Tab 14: Avg. Execution time based on dataset-2

Method	Job types			
	Smaller	Average	Larger	Huge
RD	35	60	90	120
WRR	30	50	84	110
DLB	27	42	78	101
LB-BC	23	36	65	96
LB-RC	19	34	62	85
IPSO-Firefly	18	31	60	82

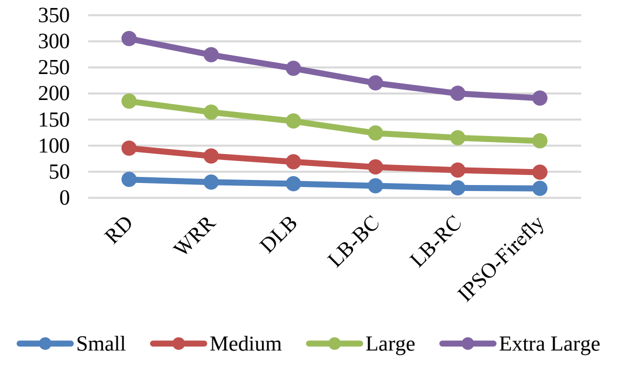


Fig. 11. Avg Execution time based on dataset-2

Execution time: Tab.13 and Tab.14 display the experimental outcomes of all methods across the two distinct datasets. Tab 13 and 14 generally display execution time calculated beside various task categories. The chosen VM instances affect how long the tasks take to complete. When tasks are given to best dominant VM instances, which have a greater resource volume than fewer influential VM instances, task execution time is reduced [23].

• Resource consumption: Using CPU and memory usage as our benchmarks, we assessed resource utilization. Tab 15 and 16 display the findings of the analysis of CPU consumption for all approaches.

Tab 17 and 18 also display the outcomes of the evaluation of memory utilization for all approaches.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

Reliability: Tab 19 and 20 compare the suggested method's reliability to those of the alternative alternatives.

Makespan: Tab 21 and 22 display the experimental findings for the various methods.

• Throughput: Tab 23 and 24 reflect the experimental findings of the methods.

Tab 15: CPU usage dataset-1

Method	Job types			
	Smaller	Average	Larger	Huge
RD	44	51	63	75
WRR	48	57	67	78
DLB	51	62	72	84
LB-BC	56	68	79	89
LB-RC	66	75	85	93
IPSO-Firefly	68	76	87	94

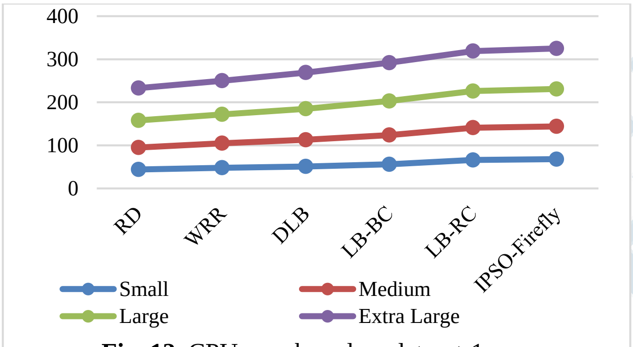


Fig. 12. CPU uses based on dataset-1

Tab 16: CPU usage dataset-2

Method	Job types			
	Smaller	Average	Larger	Huge
RD	46	53	65	76
WRR	50	59	69	80
DLB	53	64	74	85
LB-BC	58	70	79	90
LB-RC	68	77	86	95
IPSO-Firefly	70	78	89	96

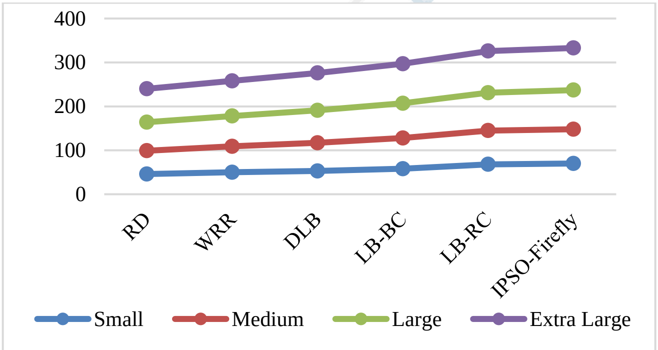


Fig. 13. CPU uses for dataset-2

Tab 17: Memory usage dataset-1

Method	Job types			
	Smaller	Average	Larger	Huge
RD	41	51	62	73
WRR	44	55	65	76
DLB	50	60	71	82
LB-BC	54	64	73	85
LB-RC	58	67	80	90
IPSO-Firefly	59	70	84	92

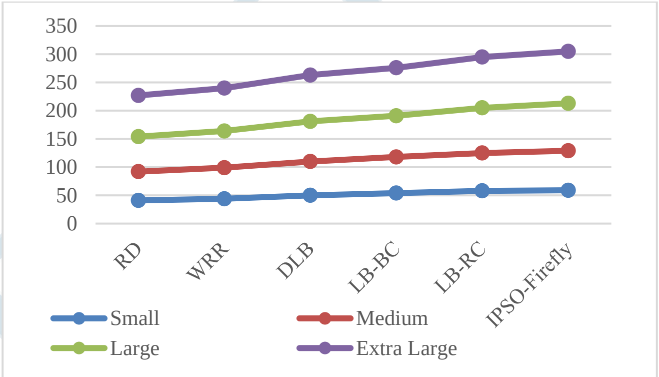


Fig. 14. Memory uses for dataset-1

Tab 18: Memory usage dataset-2

Method	Job types			
	Smaller	Average	Larger	Huge
RD	39	87	59	69
WRR	42	52	62	74
DLB	46	56	65	78
LB-BC	51	62	70	83
LB-RC	56	68	76	88
IPSO-Firefly	59	70	79	90

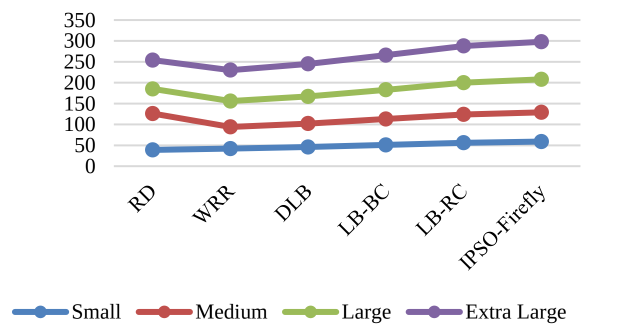


Fig. 15. Memory uses for dataset-2

Tab 19: Reliability dataset-1

Method	Job types			
	Smaller	Average	Larger	Huge
RD	77	62	49	36
WRR	82	68	54	40
DLB	88	75	59	46
LB-BC	93	81	69	55
LB-RC	98	88	78	64
IPSO-Firefly	100	90	79	64

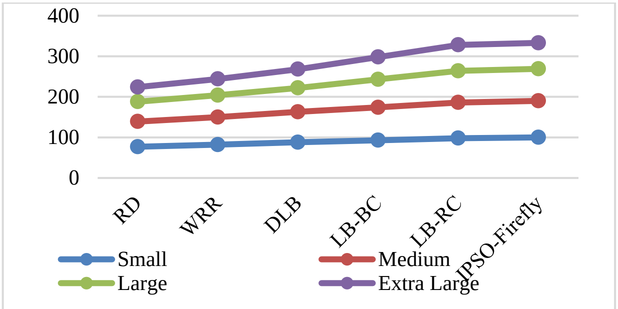


Fig. 16. Reliability of dataset-1

Tab 20: Reliability dataset-2

Method	Job type			
	Smaller	Average	Larger	Huge
RD	79	63	51	38
WRR	83	68	57	42
DLB	88	75	62	48
LB-BC	94	82	68	57
LB-RC	99	89	79	67
IPSO-Firefly	100	90	79	68

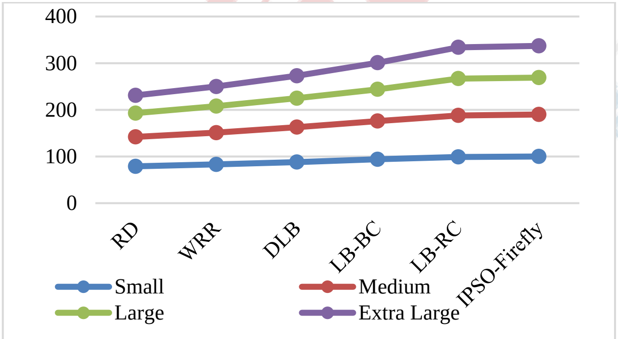


Fig. 17. Reliability of dataset-1

Tab 21: Makespan Test dataset-1

Method	Job types			
	Smaller	Average	Larger	Huge
RD	66	121	202	289
WRR	61	112	196	281
DLB	56	104	189	274
LB-BC	53	99	179	262
LB-RC	49	93	176	154
IPSO-Firefly	48	90	173	150

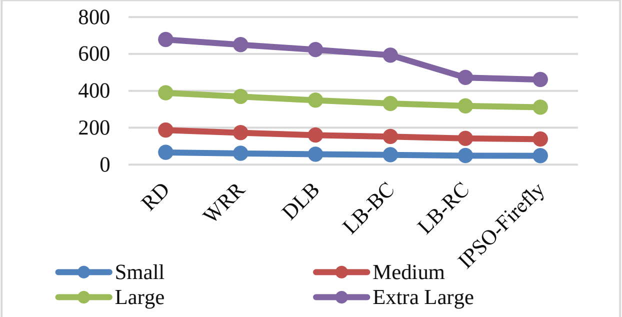


Fig. 18. Kakespan of dataset-1

Tab 22: Makespan Test dataset-2

Method	Job types			
	Smaller	Average	Larger	Huge
RD	74	130	219	301
WRR	69	119	211	291
DLB	65	109	203	287
LB-BC	58	99	197	279
LB-RC	51	93	188	169
IPSO-Firefly	50	90	186	164

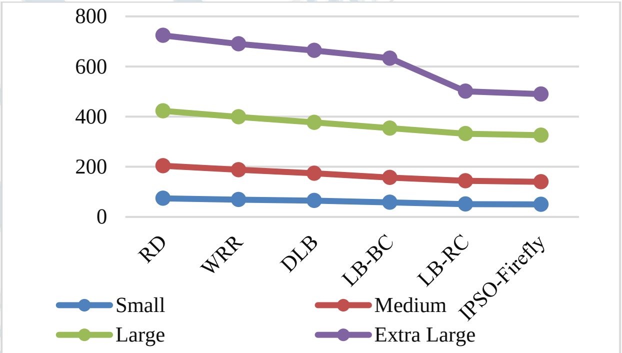


Fig. 19. Makespan of Dataset-2

Tab 23: Throughput avg. Test dataset-1

Method	Job types			
	Smaller	Average	Larger	Huge
RD	66	55	43	31
WRR	73	64	48	37
DLB	82	71	55	45
LB-BC	91	77	65	54
LB-RC	97	85	73	66
IPSO-Firefly	99	85	70	69

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

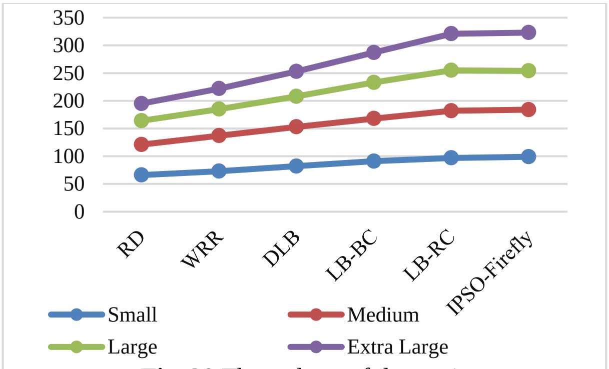


Fig. 20 Throughput of dataset1

Tab 24: Throughput avg. Test dataset-2

Method	Job types			
	Smaller	Average	Larger	Huge
RD	67	57	44	32
WRR	74	67	49	38
DLB	83	72	57	46
LB-BC	92	78	66	55
LB-RC	98	87	74	67
IPSO-Firefly	99	89	75	67

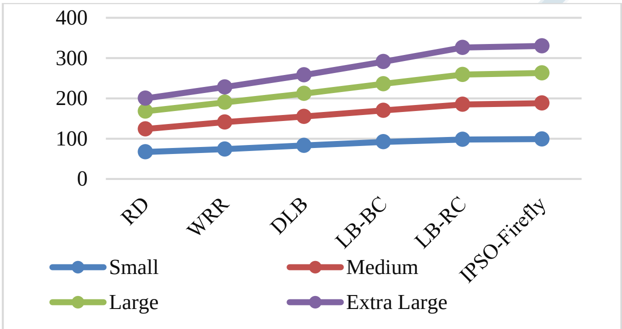


Fig. 21. Throughput of dataset-2

The results demonstrate that, while comparing with other approaches, the projected process has consistently produced the best outcomes. This is a result of how well IPSO and Firefly algo. work together.

Our study used Firefly to identify the primary inhabitants with best avg load, yielding best beginning population in the problem space. The strong attraction between the pairs is the basis for adopting this algorithm for this phase. The suggested technique is able to discover the populace with best avg load rapidity by using our algorithm since attraction changes are particularly effective in defining convergence rate.

This study employed IPSO technique to discover optimum solution with maximum load balancing after determining best beginning population. This algorithm is chosen to identify the optimal solution because of the IPSO algorithm's higher convergence speed and more intelligent behavior when matched to other particle swarm optimization algorithms.

As you can see in Tab. 1, several dynamic algorithms have drawbacks that make them unsuitable for particular circumstances, and LB was not successfully accomplished. As per our findings, our work was able to display how the suggested process is preferable to some of the drawbacks of earlier algo. Throughput of the ACCLB is modest. However, it is demonstrated in this research that the suggested hybrid method offers respectable throughput even with a complicated task load. With more servers, the Biased Random Sampling method performs less effectively. Nevertheless, the results from our suggested process has proved that the validation part to retain performance with an increase in the no. of servers.

Our approach is able to achieved superior outcomes. Compared to other approaches in various parameters. This demonstrates how the proposed strategy is better than previous approaches of a similar nature.

V. CONCLUSION

This research presents a hybrid load balancing optimization solution for cloud environments using the firefly and IPSO algorithms. The IPSO is typically quite sensitive to the starting situations, and if the initial population is not carefully selected, this particular process may not yield into a good result. In order to define an optimal initial settings for the suggested technique, this study used the firefly algorithm. There is significant improvement in reaction time. Additionally, the adaptability of the proposed strategy in average load minimization through various targets is superior to prior approaches.

REFERENCES

6.1. Journal Article

[1] Singh, Athokpam Bikramjit, et al. "A comparative study of various scheduling algorithms in cloud computing." *American Journal of Intelligent Systems* 7.3 (2017): 68-72.

[2] Singh, Athokpam Bikramjit, et al. "Survey on various load balancing techniques in cloud computing." *Adv. Comput* 7.2 (2017): 28-34.

[3] Singh, Athokpam Bikramjit, et al. "A Comprehensive Investigation of Network Virtualization" *International Journal of Latest Trends in Engineering and Technology, Special Issue SACAIM (2017), pp. 622-628*

[4] Singh, Athokpam Bikramjit, et al. "An Efficient and Dynamic Load Balancing Algorithm Through Horizontal Virtual Machine Scaling Techniques" *GIS Science Journal, Volume 9, Issue 6, June (2022)*

[5] Hamadah, Siham. "A survey: a comprehensive study of static, dynamic and hybrid load balancing algorithms." *International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN* (2017): 2249-9555.

[6] Aditya, Abhijit, Uddalak Chatterjee, and Snehasis Gupta. "A comparative study of different static and dynamic load balancing algorithm in cloud computing with special emphasis on time factor." *International Journal of Current Engineering and Technology* 5.3 (2015): 1898-1907.

[7] Golchi, Mahya Mohammadi, and Homayun Motameni. "Evaluation of the improved particle swarm optimization algorithm efficiency inward peer to peer video streaming." *Computer Networks* 142 (2018): 64-75.

[8] Amiri, Farbod, Babak Shirazi, and Ali Tajdin. "Multi-objective simulation optimization for uncertain resource assignment and job

sequence in automated flexible job shop." *Applied Soft Computing* 75 (2019): 190-202.

[9] Eren, Yavuz, İbrahim B. Küçükdemiral, and İlker Üstoğlu. "Introduction to optimization." *Optimization in renewable energy systems*. Butterworth-Heinemann, 2017. 27-74.

[10] Arianyan, Ehsan, Hassan Taheri, and Saeed Sharifian. "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers." *Computers & Electrical Engineering* 47 (2015): 222-240.

[11] Yagoubi, Belabbas, and Yahya Slimani. "Task load balancing strategy for grid computing." *Journal of Computer Science* 3.3 (2007): 186-194.

[12] LD, Dhinesh Babu, and P. Venkata Krishna. "Honey bee behavior inspired load balancing of tasks in cloud computing environments." *Applied soft computing* 13.5 (2013): 2292-2303.

[13] 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.

[14] Kaur, Amanpreet, and Bikrampal Kaur. "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment." *Journal of King Saud University-Computer and Information Sciences* (2019).

[15] Adhikari, Mainak, Sudarshan Nandy, and Tarachand Amgoth. "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud." *Journal of Network and Computer Applications* 128 (2019): 64-77.

[16] Priya, V., C. Sathiya Kumar, and Ramani Kannan. "Resource scheduling algorithm with load balancing for cloud service provisioning." *Applied Soft Computing* 76 (2019): 416-424.

[17] Thakur, Avnish, and Major Singh Goraya. "A taxonomic survey on load balancing in cloud." *Journal of Network and Computer Applications* 98 (2017): 43-57.

[18] Khanchi, Mamta, and Sanjay Tyagi. "An efficient algorithm for load balancing in cloud computing." *International Journal of Engineering Sciences & Research Technology* 5.6 (2016): 468-475.

[19] Mahmood, Amjad, Salman A. Khan, and Rashed A. Bahlool. "Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm." *Computers* 6.2 (2017): 15.

[20] Deepa, T., and Dhanaraj Cheelu. "A comparative study of static and dynamic load balancing algorithms in cloud computing." *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE, 2017.