HISA-LB: A Novel Load Balancing Approach for Improvising Harmony Memory in Cloud Environment

Ms. Shruti Tiwari [1] , Dr. Chinmay Bhatt [2]
[1]Research Scholar,Dept. Of Computer Science Engineering,RKDF College, Bhopal (M. P.), India
[1]shruti.tiwari08@gmail.com
[2] Professor, Dept. Of Computer Science Engineering, RKDF College, Bhopal (M. P.), India
[2]chinmay20june@gmail.com

Article Info
Page Number: 9029 - 9055
Publication Issue:
Vol 71 No. 4 (2022)

Abstract
In a cloud context, task scheduling (TS) is an NP-hard combinational optimization issue since the number of tasks rises, and their durations vary rapidly. The mappings between resources and tasks are challenging to develop. As a result, we essential a capable task scheduling strategy that can better handle the work and address the NP-hard problem. Several researchers have concentrated on heuristic, meta-heuristic, and hybrid scheduling algorithms to address this problem. This work provides a unique load balancing strategy based on two optimization algorithms to address the problem of uneven load distribution at virtual machines (VMs) via proper task scheduling. This approach consists of music inspired harmony search algorithm (HSA) and simulated annealing (SA) algorithm, named HISA-LB (load balancing) approach. This approach works on HMCR and PAR values in which probability is checked. Based on the probability threshold, it calculates the best objective function by choosing either Harmony search or simulated annealing to distribute tasks across available cloud resources. This is accomplished by maintaining accurate information among the data center's tasks and resources. The CloudSim simulator is applied to implement the proposed algorithm for load balancing. The simulations have been done by considering two scenarios 1) 3 VMs and 10 to 50 cloudlets, 2) 5 VMs and 10 to 50 cloudlets. Both scenarios have been experimented with and considered only the five best performance results. The simulation outcomes indicate that the proposed HISA-LB strategy outperforms other existing LBMPSO approaches in decreasing Makespan and increasing resource usage. In addition, throughput is another parameter for comparison and achieved the highest throughput than the LBMPSO approach.

Article History
***Article Received**: 15 September 2022*
***Revised**: 25 October 2022*
***Accepted**: 14 November 2022*
***Publication**: 21 December 2022*

Index Terms— Cloud computing, Load balancing, Task scheduling, VM process, Harmony search, Simulated annealing.

I. INTRODUCTION

Cloud computing (CC) is a potential paradigm in contrast to conventional information technology techniques. Businesses and organizations employ elastic cloud services on a pay-per-use basis to cut expenses. Virtualization is a widely used technology in modern data centers (DCs) to increase resource usage, minimize greenhouse gas emissions, and cut costs. VM migration is frequently used inside or between DCs to suit a broad range of virtualized cloud environment demands. [1].

The distribution and management of resources is a hard problem for cloud service providers, even though cloud computing has several advantages. The demand for cloud resources is growing by the day. Because of the increased demand for cloud resources, cloud service providers have difficulties allocating resources [2]. Similarly, the work of cloud resource management develops much more challenging when there is a rapid increase in on-demand resources due to a lack of natural physical resources at cloud premises. Inappropriate cloud resource allocation may result in either on-time resource unavailability to service customers or poor usage of cloud resources, resulting in Service Level Agreement (SLAs) breaches [3].

SLA infractions can occasionally lead to a reduction in return on investment or even the termination of the contract. Overall, the "consistent availability" of the high-performance network has been an essential criterion of clients from cloud service providers (CSPs) since the creation of CC. This has been one of the greatest foundational demands of consumers from CSPs ever since the discovery of CC [4]. Applications running on a set of interconnected instances of VMs in fundamental CC systems may leave a few instances loaded down. At the same time, other VM occurrences may be idle or lightly packed, causing efficiency loss and expenditure of CC resources and power.

The problem of inefficient resource usage in CC settings may be solved by distributing the workload across networked VMs using an LB approach. This will allow for more effective use of the available resources. In cloud computing, control of efficiency and performance has continued to be a pressing issue. In addition, unbalanced workload allocation can potentially lower the efficiency of cloud virtual machines [5]. In contrast to the method of migrating a whole virtual machine, the method proposed in this study consists of moving just the needed tasks from one virtual machine to another inside a cloud environment. The technique contributes to the resolution of the issue of load balancing that affects virtual machines. The load balancing approach offers a potential solution to the issue of inefficient resource allocation and usage in CC. An innovative method for LB in CC environments is proposed in this work for effective usage of available computer resources.

This paper provides the following contributions to this research work:

1) To solve the load balancing issue at VMs because of uneven distribution of resources.
2) To design an approach to HISA load balancing based on two optimization algorithms for improvising the harmony memory.
3) It has been statistically validated using different performance parameters for assessing execution and makespan time. It will improve the resource utilization of resources with proper task allocations.
4) The HISA-LB technique effectively locates high-performance regions of optimal solution promptly.
5) It provides better load balancing at VMs by efficient allocation of tasks.

The remainder of the paper is organized as shown below. Section II section II will provide research on several existing cloud load balancing approaches. Section III presents a novel load balancing technique. The findings of the experiments are described in section IV. Section V includes a discussion of concluding thoughts and ideas for further study.

II. Related Work

Among the many components of cloud task scheduling, task-LB on VMs is highly recognized as important. The present study of [6] created a dynamic LB solution based on the hybrid optimization methodologies used in this investigation. This study used a Mantaray-modified version of the multi-objective Harris hawk optimization (MMHHO). The hybridization procedure, which considers cost, reaction time, and resource consumption, updates the search space of HHO by employing the MRFO (Manta Ray Forging Optimization) algorithm. The hybrid strategy presented in this work increases system efficiency by increasing VM throughput, balancing load across VMs, and maintaining task priority balance by modifying the waiting time of the associated processes. CloudSim is a tool that implements the suggested MMHHO-based LB algorithm. The efficacy of the proposed method was evaluated using different parameters and associated with other previous algorithms. According to the simulation findings, the proposed MMHHO load balancing technique beats alternative algorithms.

In the research article [7], FIMPSO is a suggested new load balancing algorithm that combines the firefly algorithm with the IMPSO (Improved Multi-Objective Particle Swarm Optimization) method. While the IMPSO method is used to locate the improved response, the Firefly (FF) algorithm is applied to compress the search area. IMPSO algorithm chooses the particle with the smallest point-to-line distance as the global best (gbest). The best particle candidates were selected using a mini distance from a point to a line. The suggested FIMPSO algorithm improved key indicators, including appropriate resource utilization and task response time and attained effective average load for making. According to the simulation results, the suggested FIMPSO model outperformed the competing approaches. The simulation results demonstrate that the FIMPSO method is the most successful of the ones tested, with a makespan of 148, a dependability rate of 67%, a throughput rate of 72%, a reaction time of 13.58ms on average, as well as a CPU utilization rate of 98%.

Every DC negatively impacts the environment because of its excessive energy consumption. Therefore, making cloud computing more successful necessitates addressing challenges related to using computer resources effectively and reducing energy use. One of the most important tools for addressing these issues is load balancing. The authors of [8] created an adaptive cat swarm optimization (ACSO) algorithm-based LB software to address optimization issues. Several value indicators and performance comparisons are used to assess the effectiveness of the suggested solution.

Security, as well as LB, are two major concerns in the CC environment. A node's request time and response time are crucial characteristics of loading balancing. LB in the cloud may be optimized using meta-heuristics algorithms[9]. EMAMBO is a technique to ensure equitable distribution of public clouds across all nodes. Several metrics suggest that the proposed system performs better than several existing benchmarks.

Massive data centers have an impact on the environment as well as the economy due to their high power consumption. Micro-Genetic (MG) Algorithms provide a stable combined processes workload allocation approach using CSO (MG-CSO) by fixing pre-convergence issues and finding the best available resources. The resources are constantly clustered and condensed for maximum computational efficiency. Comparing the results of the MG-CSO Algorithm against those of other popular algorithms like the GA and the BA reveals good results for the earlier. The goal of the study [10]was to decrease cost, time, SLA, and energy, as well as provide a high QoS. Our final tally for the scientific findings was 91%.

To efficiently use a VM's resources, load balancing must be performed to distribute the workload evenly across all VMs. [11] developed an innovative approach to dynamically distributing work across VMs by combining a PSO (MPSO) variant with an enhanced version of the Q-learning algorithm, which they called QMPSO. Hybridization is applied to fine-tune MPSO's velocity using gbest and pbest according to the best action given by refined Q-learning. Hybridization aims to enhance the machine's efficiency by distributing workload evenly across VMs, increasing VMs' throughput to their fullest extent, and keeping workload priorities in check by minimizing the waiting time for each operation. The method's reliability was confirmed by comparing the simulated QMPSO outcomes with the current load balancing and scheduling technique. When our suggested algorithm's performance is measured against that of a leading rival, it performs much better on both simulations and a real platform.

The cloud simulator is a set of Java classes that may simulate various aspects of cloud computing. For improved resource distribution through load balancing, researchers suggest and implement an algorithm based on the work of honey bees, with certain modifications. [12] the empirical study of a suggested effective approach for LBof tasks utilizing honey bee-inspired resource allocation in a cloud scenario.

III. Research Methodology

Within this section, we will suggest a technique that, as illustrated in Figure 3, aims to optimize resource allocation while minimizing the impact on execution time. This method illustrates the importance of framework structure, tasks, and resources. The primary objective of the suggested HISA-LB technique is to properly schedule all incoming tasks to accessible VMs, hence reducing Makespan and increasing machine utilization in CC. Every task must be assigned to one VM. It firstly discusses problems in existing literature work. Then provide a detailed proposed methodology based on two nature-inspired optimization techniques in this section. In this model, the mapping is done with a music-inspired harmony search and a simulated annealing load balancing (HISA-LB) approach to concentrate on a resource allocation method in that resources are maintained such that no task is overlooked or overall execution time is minimized. As a result, the overall approach, which incorporates multiple types of task and resource data, may be optimized in terms of time or resource allocation techniques.

A. Problem Statement

Currently, LB in the CC environment is a significant challenge. A distributed solution has always been necessary. Because it is not always practicable or cost-effective to keep one or more idle services to meet demand, it is not always possible to do so [13]. As the cloud's structure is very complicated and its components are dispersed over a large region, it is impossible to allocate tasks to specific servers and clients for effective load balancing. Assigning specific tasks is accompanied by a degree of uncertainty. In a cloud computing environment, mapping all tasks to available VMs and determining the appropriate solution is difficult. In this instance, a productive TS method is required to balance VM load and assign each user's task to an appropriate resource. The existing work used a PSO-based load balancing technique for task scheduling. However, this technique maximizes resource allocation using different buffers to numerous task or resource information types. But it faced some convergence issues and did not efficiently utilize the task allocation in minimized execution time. This technique was not able to utilize proper resources. This study illustrates the direct relevance of framework structure, tasks, and resources.

B. Proposed Methodology: HISA-LB Model

This paper proposes a new LB approach based on harmony inspired and simulated annealing algorithm named HISA-LB to overcome the abovementioned problems and issues of LB and TS. HISA-LB task scheduling technique is founded on the HISA algorithm, which uses fitness function to determine the optimal arrangement of each harmony. The fitness function computes the execution times of each VM or returns the execution time with the highest score as the fitness value (F) of every harmony.

1) Harmony Inspired Search Algorithm

HS is a search heuristic built on jazz improvisation [14]. When playing jazz music, musicians strive to alter their pitches so that the resulting harmonies are the most beautiful possible version of themselves. They start with specific harmonies and then use improvisation to attempt to improve upon those harmonies. This comparison might be used to build search heuristics, strategies that may be utilized to maximize a particular objective function. Harmonies are not the only application for this analogy. In this scenario, musicians are considered decision variables, and harmony is regarded as a solution. HS algorithm, much like jazz musicians, creates new harmonies by improvisation. Therefore it consistently comes up with unique solutions that depend on the outcomes of recent iterations and random changes. Whereas this framework allows for many interpretations, the fundamental HS method is usually explained in the literature as a diagram, as seen in fig. 1.

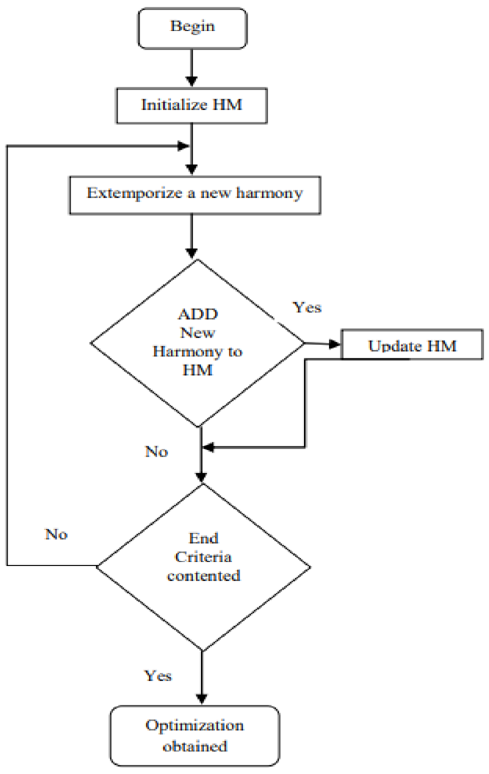


Fig. 1. Flowchart of HSA

HM is initialized with randomly generated solutions using the HS method. HM Size determines no. of explanations saved in HM. Following is how novel solution is developed iteratively. Each decision variable is produced either by considering memory and possibly undergoing further modification or by randomly selecting one of many possible values. Harmony Memory Considering Rate (HMCR) and Pitch Adjusting Rate (PAR) are both terms that refer to factors applied in the procedure of developing novel solutions. Every decision variable is given the value of a particular variable of opportunities in HM with a probability of HMCR, or an additional variation of this value is carried out with a probability of PAR. In addition, this value is subjected to an additional modification with a probability of PAR. Instead, the value of a choice variable is made to be completely arbitrary (with a chance of 1 HMCR). Once a new response has been developed, it is compared against the worst option in HM. If the objective value of this solution is higher than the value of the worst option, then it will replace the worst solution in the HM. This cycle will continue until the conditions for completion have been fulfilled. This algorithm's extensive explanations may be found in [15][16].

The HS algorithm optimization technique is given below in five phases [17].

Step 1: Parameters Initialization

Optimization issue is specifically distinct:

$$Minimize\ f(x)$$

subject to

$$x_j \in X_j = 1,2,\dots,N, \tag{1}$$

here f(x) is an objective function; x set of every decision variable x_j; N no. of decision variables, X_j Set a possible range of values for every decision variable, which is x_j^{min} or x_j^{max} Are lower or upper boundaries of the jth decision parameter correspondingly. HMS, or no. of solution vectors stored in HM, HMCR, PAR, bandwidth distance (BW), no. of improvisations (NI), or halting criteria, are all HS technique parameters specified at this stage.

Step 2: HM Initialization and Evaluation

Random initial population, consisting of elements like, for example:

$$x_{i,j}^0 = x_j^{min} + r_j\,(x_j^{max} - x_j^{min}) \tag{2}$$

In which i= 1,2,…,HMS; j= 1,2,…,N OR $r_j \in [0,1]$ is uniformly distributed random no. produced from scratch for every value of j. In HM, solution vectors are studied, and objective function values are determined.

Step 3: Improvisation

This stage generates a novel harmony vector built on three rules: memory consideration, pitch correction, and random selection. A design variable's value can be chosen with the probability

HMCR from values recorded in HM. It is possible to further adjust it by shifting to a neighbor value of a selected value from HM with a probability of PAR, or it is possible to choose it at random from the set of entirely candidate values without taking into account values that are kept in HM, by the probability of PAR. Either way, adjusting it further (1 - HMCR) is possible.

Step 4: HM Update

An updated harmony vector will replace a previous one if it has a higher goal value and a smaller constraint violation.

Step 5: Termination criterion check

If the stopping requirement (e.g., the largest amount of improvisations) is fulfilled, the HS method is terminated. Steps 3 and 4 are then repeated if necessary.

The HS method is described in pseudocode in Algorithm 1, which gives a general overview of the algorithm.

Algorithm 1: HSA

Procedure:

```
1.  Initialize HM using HMS chosen solutions randomly.
2.  reiteration
3.  Develop novel solutions in a subsequent manner
4.  for entirely decision variables, do
5.  Most probably, HMCR takes a value from one of the possibilities in HM and changes it
    slightly with probability PAR.
6.  Instead, choose a random value for this decision variable (with a probability of 1HMCR).
7.  end for
8.  If the novel approach outperforms the worst solution in HM, it is accepted.
9.  Swap worst resolution via novel one
10. end if
11. until the Termination criterion is satisfied
12. return the best solution in HM
```

2) Simulated Annealing

SA is recognized as an iterative improvement solution to optimization issues, as well as the statistical physics approach for computer simulation of annealing a solid to its energy level, that is, the state with the least amount of energy. In other words, SA is the state with the least amount of potential energy. [18]. The iterative improvement method may be defined as follows, given the collection of configurations, a cost function, and neighborhood architecture. At the start of each iteration, configuration i is provided, and a transition to configuration j ϵ P(i) is created. P (i) is a subset of configurations named neighborhood of I for every configuration i. If C(j) < C(i), next iteration's start configuration is j; or else, it is i. If transitions are produced in an exhaustive enumerative manner, the method will, thru description, terminate at a local minimum. However, the cost of a min may differ significantly from that of a global min. SA may be considered an effort to identify near-optimal local minima by accepting cost-increasing transitions. More precisely, if i as well as j P(i) are two configurations to pick from, the method proceeds with configuration j with such a probability determined by:

$$min\{1, exp(-(C(j) - C(i))/c)\} \quad (3)$$

wherein c is a positive control parameter that will be steadily lowered while the algorithm is executed, and the drop will occur at regular intervals. Therefore, in the process of physical annealing, c is equivalent to the applied temperature. Probability reduces with higher values of C(j) - C(i) or lower values of c, and transitions with decreasing costs are always acceptable. The basic SA algorithm is described in the form of a flowchart, as displayed in fig. 2.

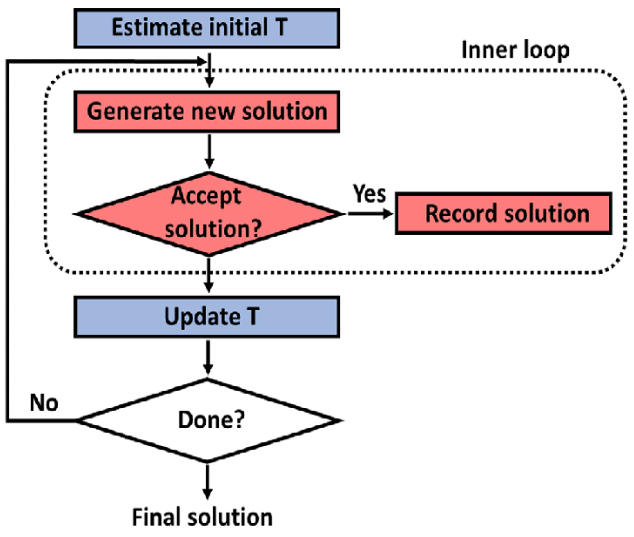


Fig. 2. Flowchart of the simulated annealing algorithm

Simulated annealing attempts to avoid cycling through randomization and simulates an annealing procedure in physics. A neighbor is generated through random choices in any iteration. Whenever produced neighbor has a higher objective function value than the initial solution, the neighbor is always recognized as the new offensive way to solve. Still, a worse neighbor is only approved with a fixed probability. The neighborhood selected and the cooling strategy used are crucial factors in the quality of the outcomes produced by a simulated annealing technique. More information is provided following.

Step 1: Neighborhood

The determination of a suitable neighborhood for an efficient scheduling solution generally has a significant impact on the ultimate solution's quality. The following algorithm is derived from the generation of a neighbor in a certain neighborhood. For permutation issues, the Swap operator may be used to generate a neighbor. The produced neighbor with the highest objective function value among these is then chosen and evaluated to the actual starting solutions using the simulated annealing acceptance criteria. We discovered that the composite neighborhood performed better than each neighborhood.

Swap operator: In this case, two tasks are exchanged at random. Taking Ȣ0 in Example 1, as well as assuming two randomly chosen positions m = 3 as well as n = 5, we get the sequence.

Swap (Ȣ0, m, n) = Swap (Ȣ0, 3, 5) = (3, 1, 5, 2, 4).

Step 2: Cooling scheme

Geometric, exponential, Lundy-Mees, and linear reduction techniques are typical cooling techniques employed in a simulated annealing procedure. The current temperature is lowered using the geometric cooling scheme to the new temperature that would be seen in the next epoch following:

$$Temp_k = \alpha Temp_{k-1}, k = 1,2, \dots \tag{4}$$

where 0 < α < 1. We discovered that the initial temperature must be set such that around 25% of the most undesirable solutions are accepted at the start. At the last step in the algorithm, following the geometric cooling strategy, $Temp_N = \propto^N Temp_0$. So, we have

$$N = \log_{\propto} \frac{Temp_N}{Temp_0} \tag{5}$$

The procedure of SA is given below in algorithm2:

Algorithm2: Simulated Annealing

Procedure:

```
1. Randomly start a group of processing units and set the control parameter to an extremely big positive number.
2. Repeated until the control parameter value approaches the minimal:
    a) Produce a random set of processing units as well as compute the Δcost utilizing:
           $\Delta \text{Cost} = \text{Cost(j)} - \text{Cost(i)}$   (6)
    b) Choose a new set of processing units based on the probability Pij found utilizing:
           $P_{ij}(cost) = min\{1, exp(-(Cost(j) - Cost(i))/cost)\}$   (7)
    c) Repeated until the inner loop break condition is reached.
    d) Reduce temperature at a set rate.
    e) Goto steps two loops.
3. End
```

Algorithm 3 gives an overview of the proposed HISA load balancing algorithm using pseudocode, and the flowchart for this model is given in figure 3.

Algorithm3. HISA-LB Algorithm Pseudocode

Procedure:

```
    1. Start
    2. Generate the harmony memory
    3. Initialization of all parameters
    4. Initialization of Temp
```

```
5. Num = 0, □ = 500
6. Place BestX (the best solution in Harmony Memory (HM)) into Ȣ
7. BstSol = BstSASol = Ȣ
8. For p = (1 to Num) do
9.     If (random (0,1) ≤ HM_CR) Then
10.        Select 2 vectors solution symbolized as υ1 & υ2 at random from Harmony memory
11.        If (random (0,1) ≤ P_A_R) Then
12.            υ = put on a PMX crossover to υ1 and υ2
13.            newX = the best neighbor amongst some of the neighbors created by υ
14.            If (SumCost(newX) < SumCost(worstX)) Then
15.                Swap worstX by newX              //Modifying the HM
16.            End of If
17.        End of If
18.    End of If
19.    Else
20.        Ȣ' = the best neighbor among the generated neighbors of Ȣ
21.        ΔCost = SumCost(Ȣ') – SumCost(Ȣ)
22.        probability = Random (0,1)
23.        If ((ΔCost ≤ 0) or (probability <$e^{-\Delta Cost/Temp}$)) Then
24.            Ȣ = Ȣ', newX = Ȣ'
25.            If (SumCost(newX ) < SumCost(worstX)) Then
26.                Swap the worstX by newX          // Modifying the HM
27.            End of If
28.            If (SumCost(Ȣ) < SumCost(BstSASol)) Then
29.                BstSASol = Ȣ
30.            End of If
31.        End of If
32.        Temp = Modify (Temp)
33.    End of Else
34.    If (SumCost(BestX ) < SumCost(BstSol))
35.        BstSol = BestX, Num = 0
36.    End of If
37.    If (SumCost(BstSASol) - SumCost(BstSol) ≥ □ )
38.        Ȣ = BstSol
39.    End of If
40. End of For
41. Get result BstSol together with its fitness function value
42. Stop.
```

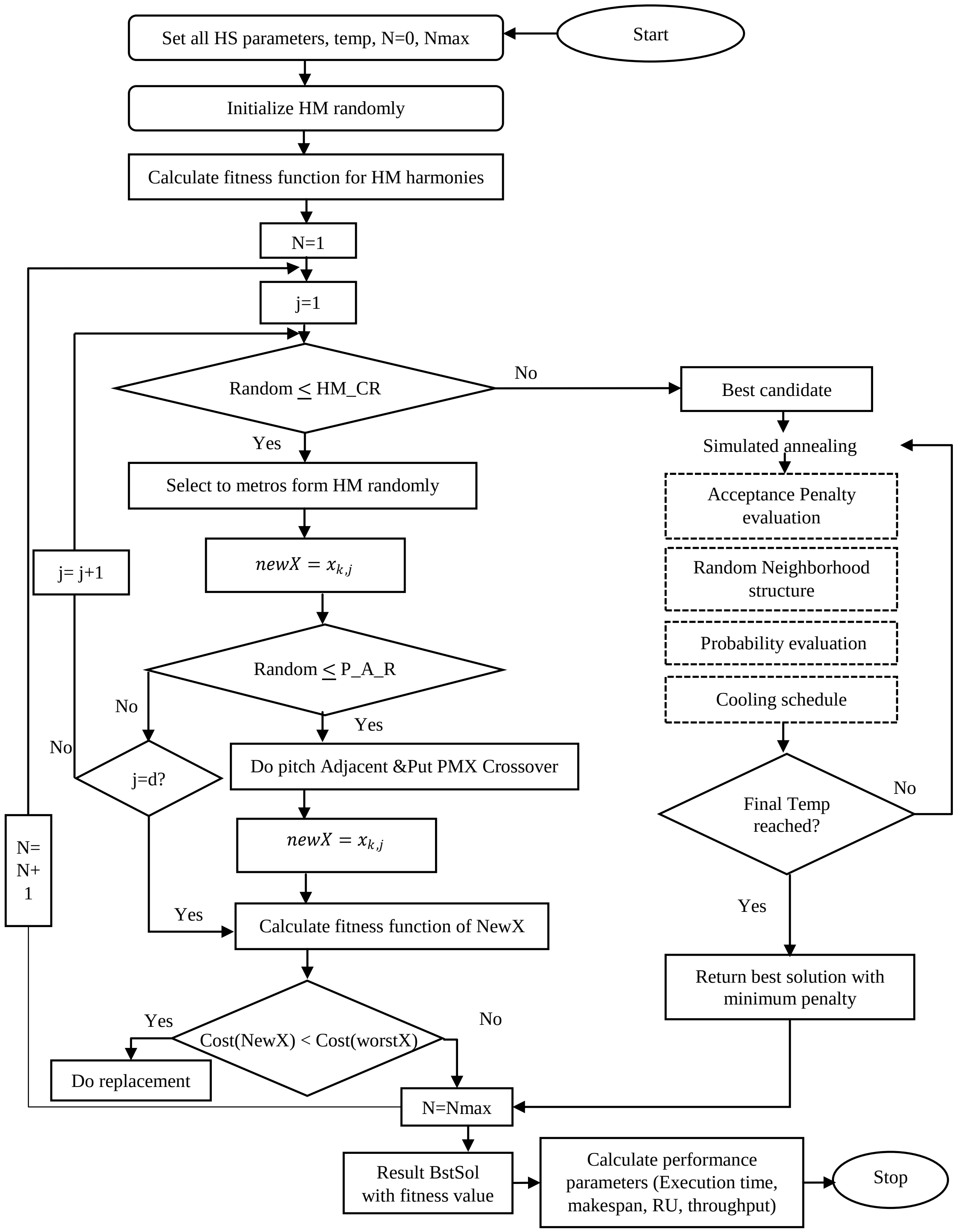

Start
Set all HS parameters, temp, N=0, Nmax
Initialize HM randomly
Calculate fitness function for HM harmonies
N=1
j=1
Random ≤ HM_CR
No
Yes
Best candidate
Simulated annealing
Acceptance Penalty evaluation
Random Neighborhood structure
Probability evaluation
Cooling schedule
Final Temp reached?
No
Yes
Return best solution with minimum penalty
Select to metros form HM randomly
$newX = x_{k,j}$
j= j+1
Random ≤ P_A_R
No
Yes
No
j=d?
Do pitch Adjacent &Put PMX Crossover
$newX = x_{k,j}$
N= N+ 1
Yes
Calculate fitness function of NewX
Yes
Cost(NewX) < Cost(worstX)
No
Do replacement
N=Nmax
Result BstSol with fitness value
Calculate performance parameters (Execution time, makespan, RU, throughput)
Stop

Fig. 3. Flowchart of proposed HISA-LB model in cloud

IV. Results and Discussion

The suggested HISA load balancing approach is analyzed and contrasted to other already used strategies. The suggested HISA load balancing method is implemented using the Eclipse Java Programming Environment and the CloudSim toolset. We utilized a PC with the following specifications: Intel (R) Core (TM) i5-Processor (2.40 GHz), NVIDIA GeForce, and 12GB of RAM. The operating system was Windows 10 64-bit. The parameters used for the simulation are shown below.

A. Experiment Setup

This section discusses different experimental settings with parameters and their values. As shown in Table 1, we examined non-preemptive activities that are free for the test. In a data center, tasks are assigned to several heterogeneous VMs. The HISA load balancing method has many characteristics. We compared the effectiveness of the HISA load balancing algorithm to that of the LBMPSO method in this section. It is necessary to use two separate testing scales: (1) three VMs with ten to fifty tasks each; (2) five VMs with ten to fifty tasks each.

Table 1. Parameters properties.

Parameters	Value
Cloud TaskRange	10-50
CloudLength	1000-6000
Cloud FileSize	300
VMRange	3-5
Memory	256-512
CPUs	1-5
Bandwidths	1000
VMMs	XEN
Processing speed (MIPS)	250-300
HM Size (SIZE_HM)	20
Number of New Harmonies (NUM_IMP)	30
HM Consideration Rate (HM_CR)	0.9f
PAR	0.1f
Fret Width (F_W)	0.02f
Number of iterations	500
cooling rate (ALPHA)	0.9
initial temperature (t)	1000

B. Performance Metrics

The computing model/layer orchestrated greatly influences performance metrics.

Makespan: Makespan, also known as completion time, is overall time essential to procedure a collection of tasks from start to finish. It is described as the moment when the last task exited the system. Makespan may be reduced by allocating a set of Ji tasks to a set of VM 'vm'; the sequence of execution of the jobs in VMs is unimportant.

The scheduling task is to minimize the Makespan since most users want their programs to run as quickly as possible, as indicated in Eq (8):

$$Makespan = CT_n \qquad (8)$$

Where, CT_n is the completion time of the last task.

Average Execution Time: CPU time or execution time of a supplied task is the amount spent with the system performing that task, such as time spent executing run-time or system functions on its behalf. Generally said, it is the amount of time the task needs to accomplish its execution.

Throughput is described as the ratio of arrival tasks to processed tasks during a certain time period. The equation reads as follows:

$$Throughput = (number\ of\ requests)/(total\ time) \quad (9)$$

Average Resource Utilisation Ratio: The created algorithm's objective is to optimize the cloud's resources' most efficient utilization possible. The resource utilisation ratio is determined using Eq. 10[13].

$$ARUR = (meantime/makespan) * 100 \qquad (10)$$

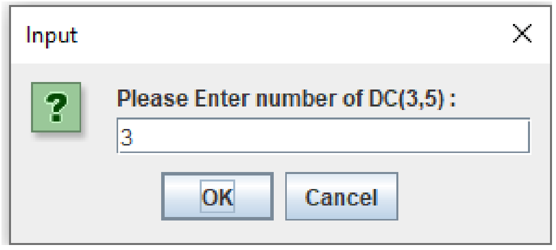
Resource Utilization is defined as the percentage of resources consumed by the incoming workload. In other words, it shoes
When average time = ∑processing time resources VMj to carry out tasks that have been allocated, the value of j may vary from 1 to m, and the range of ARUR can be anything from 0 to 1. The value 1 indicates that a resource is being used to its maximum utilization (100%), whereas the value 0 indicates that the resource is at its optimal condition.

Response time is described as a time that elapses between making a request to the server and completing the task execution.

Average response time: The total amount of time spent replying during the selected time period, multiplied by the number of reactions received during the selected time period.

C. Results

This section discusses the results of the proposed HISA load balancing cloud computing approach for two scenarios, including 3VMs 10 to 50 cloudlets and 5VMs 10 to 50 cloudlets.



Please Enter number of DC(3,5) :
3

Fig. 4. Enter GUI for VMs at the Data center

Figure 4 shows the GUI for inputting the Virtual machine 3 or 5 at the data center. From here, the user can choose one of the scenarios to test the approach.

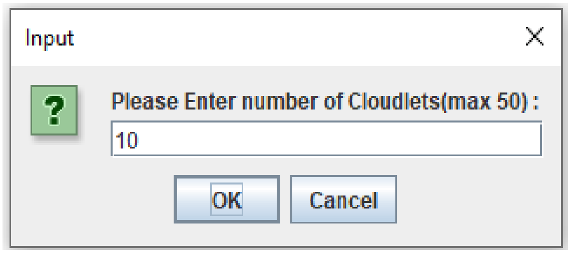
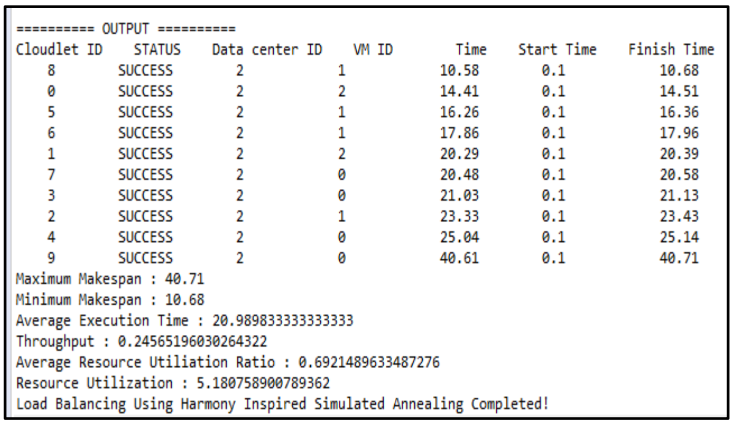


Fig. 5. Enter GUI for Cloudlets

Figure 5 shows the GUI for inputting the cloudlets between 10 to 50 for any scenario after selecting VMs at the data center. From here, the user can choose a maximum of 50 cloudlets for any scenario to test the approach.

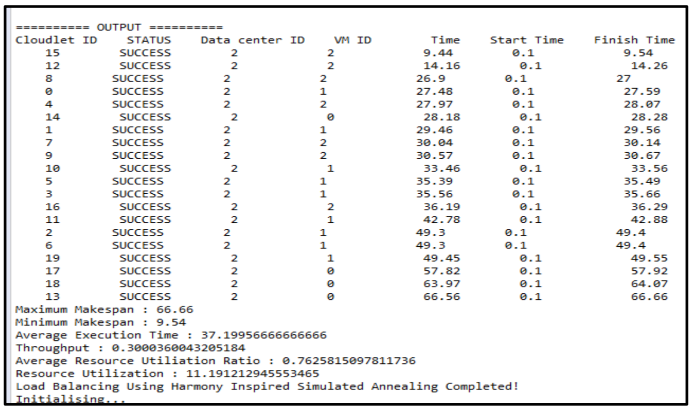
1) *Scenario 1: 3 VMs 10 to 50 cloudlets*



```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time     Start Time    Finish Time
     8         SUCCESS        2              1         10.58       0.1          10.68
     0         SUCCESS        2              2         14.41       0.1          14.51
     5         SUCCESS        2              1         16.26       0.1          16.36
     6         SUCCESS        2              1         17.86       0.1          17.96
     1         SUCCESS        2              2         20.29       0.1          20.39
     7         SUCCESS        2              0         20.48       0.1          20.58
     3         SUCCESS        2              0         21.03       0.1          21.13
     2         SUCCESS        2              1         23.33       0.1          23.43
     4         SUCCESS        2              0         25.04       0.1          25.14
     9         SUCCESS        2              0         40.61       0.1          40.71
Maximum Makespan : 40.71
Minimum Makespan : 10.68
Average Execution Time : 20.989833333333333
Throughput : 0.24565196030264322
Average Resource Utiliation Ratio : 0.6921489633487276
Resource Utilization : 5.180758900789362
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
```

Fig. 6. 3 VMs 10 cloudlets

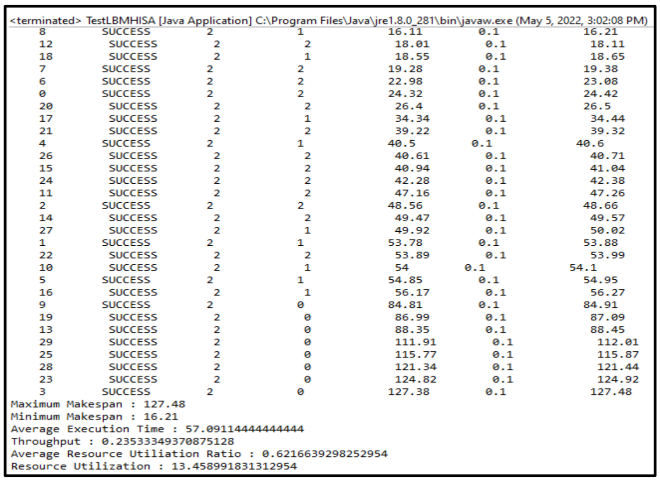
Figure 6 shows the output for scenario-1 with 3 VMs and ten cloudlets. In this cloudlets ID, status, DC ID, VM ID, time, start time and finish time are displayed, where cloudlet ID 8 takes minimized finish time of 10.68 seconds, and the highest finish time is 40.71 seconds at Cloudlets ID 9. By experimenting, five runs at this scenario achieved at least 40.71 seconds maximum makespan. The highest resource utilization is 5.189, and the throughput is 0.246.



```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time     Start Time    Finish Time
    15          SUCCESS        2              2          9.44       0.1          9.54
    12          SUCCESS        2              2          14.16       0.1          14.26
    8          SUCCESS        2              2          26.9       0.1          27
    0          SUCCESS        2              1          27.48       0.1          27.59
    4          SUCCESS        2              2          27.97       0.1          28.07
    14          SUCCESS        2              0          28.18       0.1          28.28
    1          SUCCESS        2              1          29.46       0.1          29.56
    7          SUCCESS        2              2          30.04       0.1          30.14
    9          SUCCESS        2              2          30.57       0.1          30.67
    10          SUCCESS        2              1          33.46       0.1          33.56
    5          SUCCESS        2              1          35.39       0.1          35.49
    3          SUCCESS        2              1          35.56       0.1          35.66
    16          SUCCESS        2              2          36.19       0.1          36.29
    11          SUCCESS        2              1          42.78       0.1          42.88
    2          SUCCESS        2              1          49.3       0.1          49.4
    6          SUCCESS        2              1          49.3       0.1          49.4
    19          SUCCESS        2              1          49.45       0.1          49.55
    17          SUCCESS        2              0          57.82       0.1          57.92
    18          SUCCESS        2              0          63.97       0.1          64.07
    13          SUCCESS        2              0          66.56       0.1          66.66
Maximum Makespan : 66.66
Minimum Makespan : 9.54
Average Execution Time : 37.19956666666666
Throughput : 0.3000360043205184
Average Resource Utiliation Ratio : 0.7625815097811736
Resource Utilization : 11.191212945553465
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
Initialising...
```

Fig. 7. 3 VMs 20 cloudlets

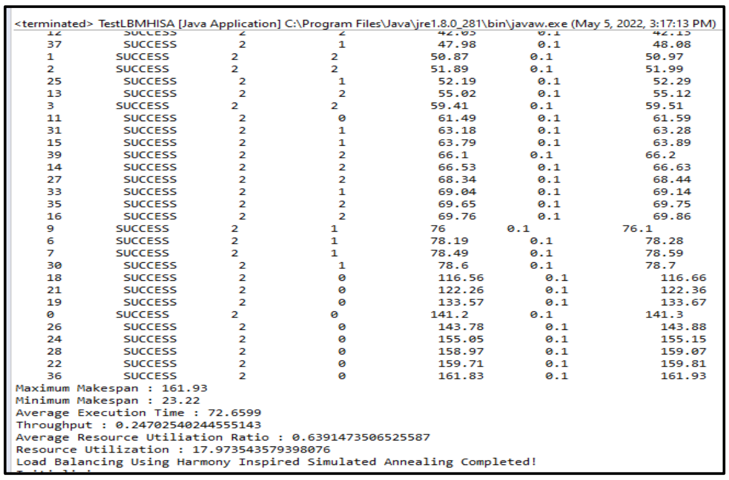
Figure 7 shows the output for scenario-1 with 3 VMs and 20 cloudlets. In this scenario, cloudlet ID 15 takes minimized finish time of 9.54 seconds, and the highest finish time is 66.66 seconds at Cloudlets ID 13. By experimenting, five runs at this scenario achieved the least 66.66 seconds maximum makespan. The highest resource utilization is 11.19, and the throughput is 0.3.



```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 3:02:08 PM)
    8        SUCCESS       2            1            16.11       0.1             16.21
    12        SUCCESS       2            2            18.01       0.1             18.11
    18        SUCCESS       2            1            18.55       0.1             18.65
    7        SUCCESS       2            2            19.28       0.1             19.38
    6        SUCCESS       2            2            22.98       0.1             23.08
    0        SUCCESS       2            2            24.32       0.1             24.42
    20        SUCCESS       2            2            26.4        0.1             26.5
    17        SUCCESS       2            1            34.34       0.1             34.44
    21        SUCCESS       2            2            39.22       0.1             39.32
    4        SUCCESS       2            1            40.5       0.1             40.6
    26        SUCCESS       2            2            40.61       0.1             40.71
    15        SUCCESS       2            2            40.94       0.1             41.04
    24        SUCCESS       2            2            42.28       0.1             42.38
    11        SUCCESS       2            2            47.16       0.1             47.26
    2        SUCCESS       2            2            48.56       0.1             48.66
    14        SUCCESS       2            2            49.47       0.1             49.57
    27        SUCCESS       2            1            49.92       0.1             50.02
    1        SUCCESS       2            1            53.78       0.1             53.88
    22        SUCCESS       2            2            53.89       0.1             53.99
    10        SUCCESS       2            1            54         0.1             54.1
    5        SUCCESS       2            1            54.85       0.1             54.95
    16        SUCCESS       2            1            56.17       0.1             56.27
    9        SUCCESS       2            0            84.81       0.1             84.91
    19        SUCCESS       2            0            86.99       0.1             87.09
    13        SUCCESS       2            0            88.35       0.1             88.45
    29        SUCCESS       2            0            111.91       0.1             112.01
    25        SUCCESS       2            0            115.77       0.1             115.87
    28        SUCCESS       2            0            121.34       0.1             121.44
    23        SUCCESS       2            0            124.82       0.1             124.92
    3        SUCCESS       2            0            127.38       0.1             127.48
Maximum Makespan : 127.48
Minimum Makespan : 16.21
Average Execution Time : 57.09114444444444
Throughput : 0.23533349370875128
Average Resource Utiliation Ratio : 0.6216639298252954
Resource Utilization : 13.458991831312954
```

Fig. 8. 3 VMs 30 cloudlets

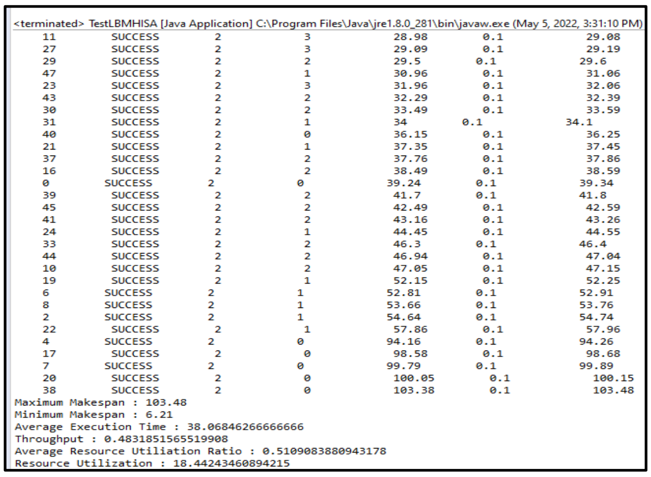
Figure 8 shows the output for scenario-1 with 3 VMs and 30 cloudlets. In this scenario, cloudlet ID 8 takes minimized finish time of 16.21 seconds, and the highest finish time is 127.48 seconds at Cloudlets ID 3. But the start time for each cloudlet is 0.1 seconds. By experimenting, five runs at this scenario achieved least 127.48 seconds maximum makespan. The highest resource utilization is 13.45, and the throughput is 0.235.



```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 3:17:13 PM)
    12        SUCCESS       2            2            42.05       0.1             42.15
    37        SUCCESS       2            1            47.98       0.1             48.08
    1        SUCCESS       2            2            50.87       0.1             50.97
    2        SUCCESS       2            2            51.89       0.1             51.99
    25        SUCCESS       2            1            52.19       0.1             52.29
    13        SUCCESS       2            2            55.02       0.1             55.12
    3        SUCCESS       2            2            59.41       0.1             59.51
    11        SUCCESS       2            0            61.49       0.1             61.59
    31        SUCCESS       2            1            63.18       0.1             63.28
    15        SUCCESS       2            1            63.79       0.1             63.89
    39        SUCCESS       2            2            66.1       0.1             66.2
    14        SUCCESS       2            2            66.53       0.1             66.63
    27        SUCCESS       2            2            68.34       0.1             68.44
    33        SUCCESS       2            1            69.04       0.1             69.14
    35        SUCCESS       2            2            69.65       0.1             69.75
    16        SUCCESS       2            2            69.76       0.1             69.86
    9        SUCCESS       2            1            76         0.1             76.1
    6        SUCCESS       2            1            78.19       0.1             78.28
    7        SUCCESS       2            1            78.49       0.1             78.59
    30        SUCCESS       2            1            78.6        0.1             78.7
    18        SUCCESS       2            0            116.56       0.1             116.66
    21        SUCCESS       2            0            122.26       0.1             122.36
    19        SUCCESS       2            0            133.57       0.1             133.67
    0        SUCCESS       2            0            141.2       0.1             141.3
    26        SUCCESS       2            0            143.78       0.1             143.88
    24        SUCCESS       2            0            155.05       0.1             155.15
    28        SUCCESS       2            0            158.97       0.1             159.07
    22        SUCCESS       2            0            159.71       0.1             159.81
    36        SUCCESS       2            0            161.83       0.1             161.93
Maximum Makespan : 161.93
Minimum Makespan : 23.22
Average Execution Time : 72.6599
Throughput : 0.24702540244555143
Average Resource Utiliation Ratio : 0.6391473506525587
Resource Utilization : 17.973543579398076
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
```

Fig. 9. 3 VMs 40 cloudlets

Figure 9 shows the output for scenario-1 with 3 VMs and 40 cloudlets. In this scenario, cloudlet ID 36 takes the highest finish time, 161.93 seconds, at Cloudlets ID 3. But the start time for each cloudlet is 0.1 seconds. By experimenting, five runs at this scenario achieved 161.93 seconds maximum makespan. The highest resource utilization is 17.97, and the throughput is 0.247.

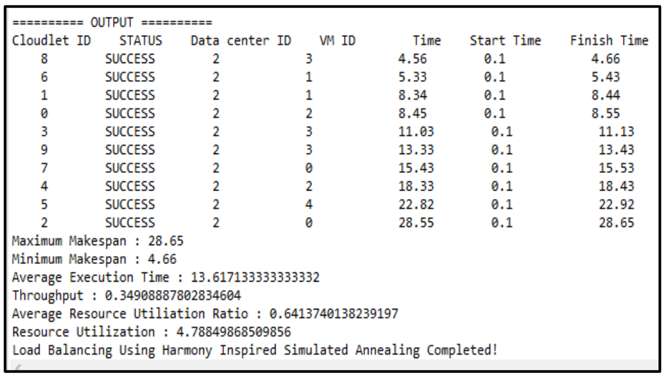


```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 3:31:10 PM)
    11        SUCCESS      2          3          28.98      0.1        29.08
    27        SUCCESS      2          3          29.09      0.1        29.19
    29        SUCCESS      2          2          29.5      0.1        29.6
    47        SUCCESS      2          1          30.96      0.1        31.06
    23        SUCCESS      2          3          31.96      0.1        32.06
    43        SUCCESS      2          2          32.29      0.1        32.39
    30        SUCCESS      2          2          33.49      0.1        33.59
    31        SUCCESS      2          1          34      0.1        34.1
    40        SUCCESS      2          0          36.15      0.1        36.25
    21        SUCCESS      2          1          37.35      0.1        37.45
    37        SUCCESS      2          2          37.76      0.1        37.86
    16        SUCCESS      2          2          38.49      0.1        38.59
    0        SUCCESS      2          0          39.24      0.1        39.34
    39        SUCCESS      2          2          41.7      0.1        41.8
    45        SUCCESS      2          2          42.49      0.1        42.59
    41        SUCCESS      2          2          43.16      0.1        43.26
    24        SUCCESS      2          1          44.45      0.1        44.55
    33        SUCCESS      2          2          46.3      0.1        46.4
    44        SUCCESS      2          2          46.94      0.1        47.04
    10        SUCCESS      2          2          47.05      0.1        47.15
    19        SUCCESS      2          1          52.15      0.1        52.25
    6        SUCCESS      2          1          52.81      0.1        52.91
    8        SUCCESS      2          1          53.66      0.1        53.76
    2        SUCCESS      2          1          54.64      0.1        54.74
    22        SUCCESS      2          1          57.86      0.1        57.96
    4        SUCCESS      2          0          94.16      0.1        94.26
    17        SUCCESS      2          0          98.58      0.1        98.68
    7        SUCCESS      2          0          99.79      0.1        99.89
    20        SUCCESS      2          0          100.05      0.1        100.15
    38        SUCCESS      2          0          103.38      0.1        103.48
Maximum Makespan : 103.48
Minimum Makespan : 6.21
Average Execution Time : 38.06846266666666
Throughput : 0.4831851565519908
Average Resource Utiliation Ratio : 0.5109083880943178
Resource Utilization : 18.44243460894215
```

Fig. 10. 3 VMs 50 cloudlets

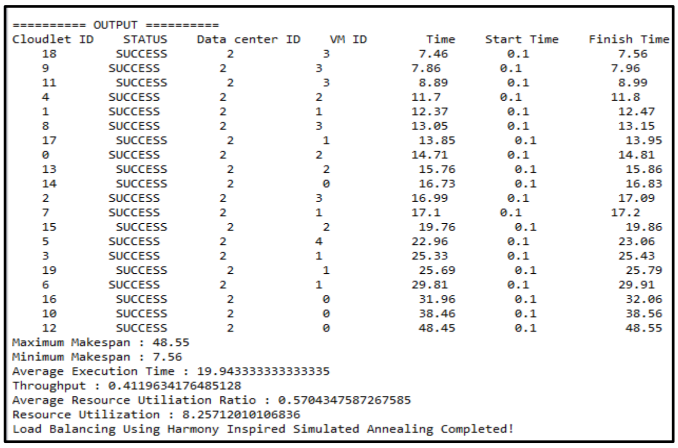
Figure 10 shows the output for scenario-1 with 3 VMs and 50 cloudlets. In this scenario, cloudlet ID 38 is successful at Datacenter ID 2 with no allocation of VM and takes 103.48 seconds highest finish time Cloudlets ID 3 with 0.1 seconds start time. Out of five experiments for 3 VMs 50 cloudlets, it has achieved 103.48 seconds maximum Makespan. The highest resource utilization is 18.44, and the throughput is 0.483.

2) *Scenario 2: 5 VMs 10 cloudlets*



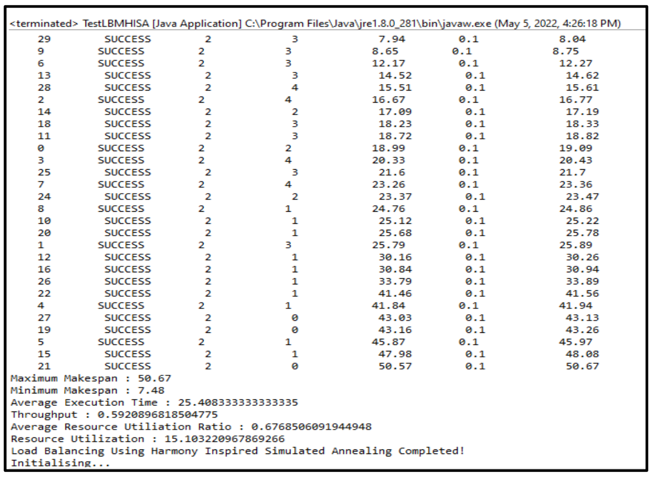
```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time     Start Time     Finish Time
    8        SUCCESS        2           3          4.56       0.1          4.66
    6        SUCCESS        2           1          5.33       0.1          5.43
    1        SUCCESS        2           1          8.34       0.1          8.44
    0        SUCCESS        2           2          8.45       0.1          8.55
    3        SUCCESS        2           3          11.03       0.1          11.13
    9        SUCCESS        2           3          13.33       0.1          13.43
    7        SUCCESS        2           0          15.43       0.1          15.53
    4        SUCCESS        2           2          18.33       0.1          18.43
    5        SUCCESS        2           4          22.82       0.1          22.92
    2        SUCCESS        2           0          28.55       0.1          28.65
Maximum Makespan : 28.65
Minimum Makespan : 4.66
Average Execution Time : 13.617133333333332
Throughput : 0.34908887802834604
Average Resource Utiliation Ratio : 0.6413740138239197
Resource Utilization : 4.78849868509856
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
```

Fig. 11. 5 VMs 10 cloudlets



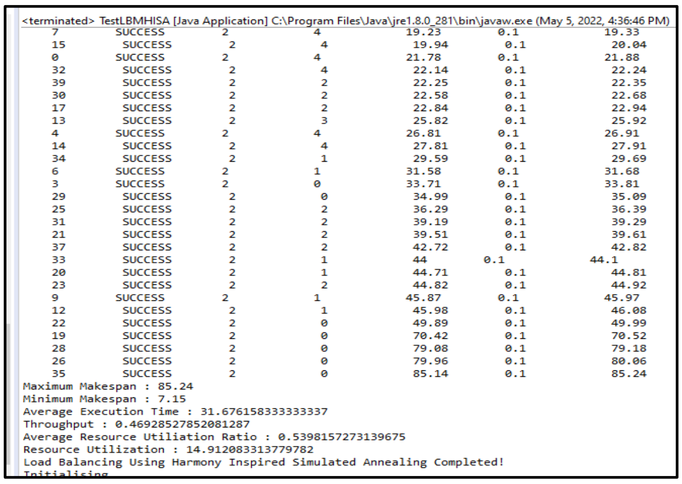
```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time     Start Time     Finish Time
    18        SUCCESS        2           3          7.46       0.1          7.56
    9        SUCCESS        2           3          7.86       0.1          7.96
    11        SUCCESS        2           3          8.89       0.1          8.99
    4        SUCCESS        2           2          11.7       0.1          11.8
    1        SUCCESS        2           1          12.37       0.1          12.47
    8        SUCCESS        2           3          13.05       0.1          13.15
    17        SUCCESS        2           1          13.85       0.1          13.95
    0        SUCCESS        2           2          14.71       0.1          14.81
    13        SUCCESS        2           2          15.76       0.1          15.86
    14        SUCCESS        2           0          16.73       0.1          16.83
    2        SUCCESS        2           3          16.99       0.1          17.09
    7        SUCCESS        2           1          17.1       0.1          17.2
    15        SUCCESS        2           2          19.76       0.1          19.86
    5        SUCCESS        2           4          22.96       0.1          23.06
    3        SUCCESS        2           1          25.33       0.1          25.43
    19        SUCCESS        2           1          25.69       0.1          25.79
    6        SUCCESS        2           1          29.81       0.1          29.91
    16        SUCCESS        2           0          31.96       0.1          32.06
    10        SUCCESS        2           0          38.46       0.1          38.56
    12        SUCCESS        2           0          48.45       0.1          48.55
Maximum Makespan : 48.55
Minimum Makespan : 7.56
Average Execution Time : 19.943333333333335
Throughput : 0.4119634176485128
Average Resource Utiliation Ratio : 0.5704347587267585
Resource Utilization : 8.25712010106836
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
```

Fig. 12. 5 VMs 20 cloudlets



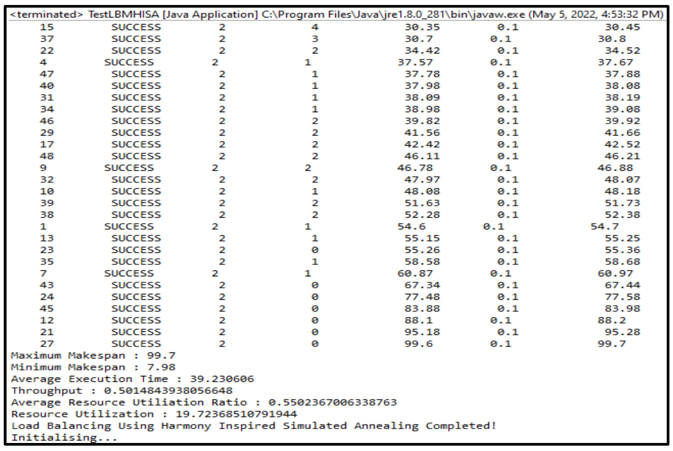
```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 4:26:18 PM)
    29        SUCCESS        2            3            7.94       0.1           8.04
    9        SUCCESS        2            3           8.65       0.1           8.75
    6        SUCCESS        2            3           12.17      0.1           12.27
    13        SUCCESS        2            3            14.52      0.1           14.62
    28        SUCCESS        2            4            15.51      0.1           15.61
    2        SUCCESS        2            4           16.67      0.1           16.77
    14        SUCCESS        2            2            17.09      0.1           17.19
    18        SUCCESS        2            3            18.23      0.1           18.33
    11        SUCCESS        2            3            18.72      0.1           18.82
    0        SUCCESS        2            2           18.99      0.1           19.09
    3        SUCCESS        2            4           20.33      0.1           20.43
    25        SUCCESS        2            3            21.6      0.1           21.7
    7        SUCCESS        2            4           23.26      0.1           23.36
    24        SUCCESS        2            2            23.37      0.1           23.47
    8        SUCCESS        2            1           24.76      0.1           24.86
    10        SUCCESS        2            1            25.12      0.1           25.22
    20        SUCCESS        2            1            25.68      0.1           25.78
    1        SUCCESS        2            3           25.79      0.1           25.89
    12        SUCCESS        2            1            30.16      0.1           30.26
    16        SUCCESS        2            1            30.84      0.1           30.94
    26        SUCCESS        2            1            33.79      0.1           33.89
    22        SUCCESS        2            1            41.46      0.1           41.56
    4        SUCCESS        2            1           41.84      0.1           41.94
    27        SUCCESS        2            0            43.03      0.1           43.13
    19        SUCCESS        2            0            43.16      0.1           43.26
    5        SUCCESS        2            1           45.87      0.1           45.97
    15        SUCCESS        2            1            47.98      0.1           48.08
    21        SUCCESS        2            0            50.57      0.1           50.67
Maximum Makespan : 50.67
Minimum Makespan : 7.48
Average Execution Time : 25.408333333333335
Throughput : 0.5920896818504775
Average Resource Utiliation Ratio : 0.6768506091944948
Resource Utilization : 15.103220967869266
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
Initialising...
```

Fig. 13. 5 VMs 30 cloudlets



```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 4:36:46 PM)
    7        SUCCESS        2            4           19.23      0.1           19.33
    15        SUCCESS        2            4            19.94      0.1           20.04
    0        SUCCESS        2            4           21.78      0.1           21.88
    32        SUCCESS        2            4            22.14      0.1           22.24
    39        SUCCESS        2            2            22.25      0.1           22.35
    30        SUCCESS        2            2            22.58      0.1           22.68
    17        SUCCESS        2            2            22.84      0.1           22.94
    13        SUCCESS        2            3            25.82      0.1           25.92
    4        SUCCESS        2            4           26.81      0.1           26.91
    14        SUCCESS        2            4            27.81      0.1           27.91
    34        SUCCESS        2            1            29.59      0.1           29.69
    6        SUCCESS        2            1           31.58      0.1           31.68
    3        SUCCESS        2            0           33.71      0.1           33.81
    29        SUCCESS        2            0            34.99      0.1           35.09
    25        SUCCESS        2            2            36.29      0.1           36.39
    31        SUCCESS        2            2            39.19      0.1           39.29
    21        SUCCESS        2            2            39.51      0.1           39.61
    37        SUCCESS        2            2            42.72      0.1           42.82
    33        SUCCESS        2            1            44      0.1           44.1
    20        SUCCESS        2            1            44.71      0.1           44.81
    23        SUCCESS        2            2            44.82      0.1           44.92
    9        SUCCESS        2            1           45.87      0.1           45.97
    12        SUCCESS        2            1            45.98      0.1           46.08
    22        SUCCESS        2            0            49.89      0.1           49.99
    19        SUCCESS        2            0            70.42      0.1           70.52
    28        SUCCESS        2            0            79.08      0.1           79.18
    26        SUCCESS        2            0            79.96      0.1           80.06
    35        SUCCESS        2            0            85.14      0.1           85.24
Maximum Makespan : 85.24
Minimum Makespan : 7.15
Average Execution Time : 31.676158333333337
Throughput : 0.46928527852081287
Average Resource Utiliation Ratio : 0.5398157273139675
Resource Utilization : 14.912083313779782
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
Initialising
```

Fig. 14. 5 VMs 40 cloudlets



```
<terminated> TestLBMHISA [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (May 5, 2022, 4:53:32 PM)
    15        SUCCESS        2            4            30.35      0.1           30.45
    37        SUCCESS        2            3            30.7      0.1           30.8
    22        SUCCESS        2            2            34.42      0.1           34.52
    4        SUCCESS        2            1           37.57      0.1           37.67
    47        SUCCESS        2            1            37.78      0.1           37.88
    40        SUCCESS        2            1            37.98      0.1           38.08
    31        SUCCESS        2            1            38.09      0.1           38.19
    34        SUCCESS        2            1            38.98      0.1           39.08
    46        SUCCESS        2            2            39.82      0.1           39.92
    29        SUCCESS        2            2            41.56      0.1           41.66
    17        SUCCESS        2            2            42.42      0.1           42.52
    48        SUCCESS        2            2            46.11      0.1           46.21
    9        SUCCESS        2            2           46.78      0.1           46.88
    32        SUCCESS        2            2            47.97      0.1           48.07
    10        SUCCESS        2            1            48.08      0.1           48.18
    39        SUCCESS        2            2            51.63      0.1           51.73
    38        SUCCESS        2            2            52.28      0.1           52.38
    1        SUCCESS        2            1           54.6      0.1           54.7
    13        SUCCESS        2            1            55.15      0.1           55.25
    23        SUCCESS        2            0            55.26      0.1           55.36
    35        SUCCESS        2            1            58.58      0.1           58.68
    7        SUCCESS        2            1           60.87      0.1           60.97
    43        SUCCESS        2            0            67.34      0.1           67.44
    24        SUCCESS        2            0            77.48      0.1           77.58
    45        SUCCESS        2            0            83.88      0.1           83.98
    12        SUCCESS        2            0            88.1      0.1           88.2
    21        SUCCESS        2            0            95.18      0.1           95.28
    27        SUCCESS        2            0            99.6      0.1           99.7
Maximum Makespan : 99.7
Minimum Makespan : 7.98
Average Execution Time : 39.230606
Throughput : 0.5014843938056648
Average Resource Utiliation Ratio : 0.5502367006338763
Resource Utilization : 19.72368510791944
Load Balancing Using Harmony Inspired Simulated Annealing Completed!
Initialising...
```

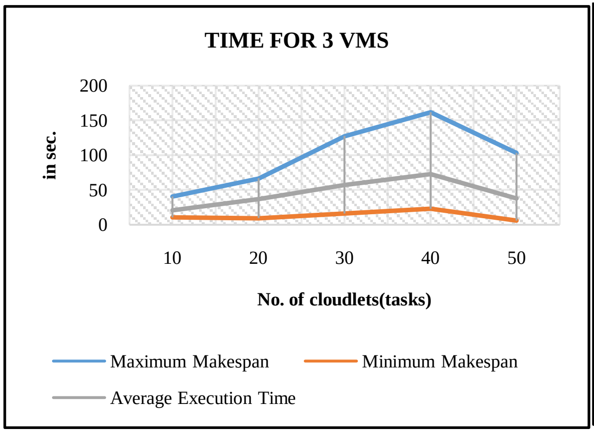
Fig. 15. 5 VMs 50 cloudlets

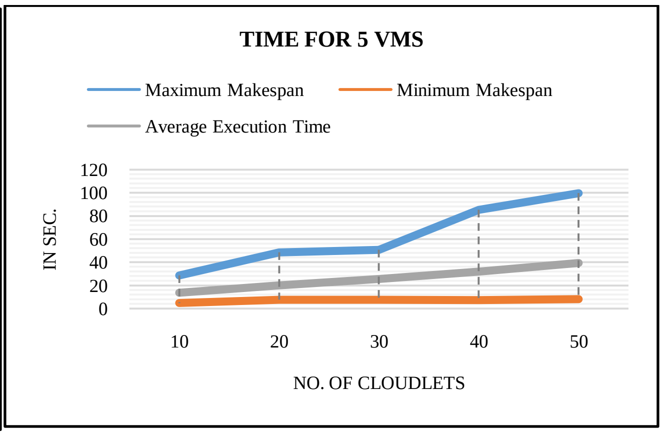
Similar to scenario-1, Figures 11 to 15 show the output for scenario-2 with 5 VMs and 10 to 50 cloudlets. In this, cloudlet IDs are included as per their status and finish time, where cloudlet IDs are arranged as minimized finish time, and the highest finish time is considered the maximum Makespan in seconds. In this case, five different experiments have been done and take only the least maximum makespan results at scenario-2. Due to the proposed HISA-LB approach, some other performance parameters like highest resource utilization, ARUR, average execution time, minimum Makespan and throughput are added.

Table 2. Experiments run for the proposed HISA-LB approach

Scenarios	**Cloudlets**	**Maximum Makespan**	**Minimum Makespan**	**Average Execution Time**	**Throughput**	**ARUR**	**RU**
Scenario-1	10	40.710	10.68	20.99	0.246	0.692	5.18
	20	66.66	9.54	37.2	0.300	0.763	11.19
	30	127.48	16.21	57.09	0.235	0.622	13.46
	40	161.93	23.22	72.66	0.247	0.639	17.97
	50	103.48	6.21	38.07	0.483	0.511	18.44
Scenario-2	10	28.65	4.66	13.62	0.349	0.641	4.79
	20	48.55	7.56	19.94	0.412	0.57	8.26
	30	50.67	7.48	25.41	0.592	0.676	15.10
	40	85.24	7.15	31.68	0.469	0.539	14.91
	50	99.7	7.98	39.23	0.501	0.550	19.72

Table 2 represents two scenarios' different experimental results in the cloud computing running environment. It calculated the results on different cloudlets by including performance parameters like min and max makespan, avg. Execution time, throughput, resource utilization and average resource utilization resources.



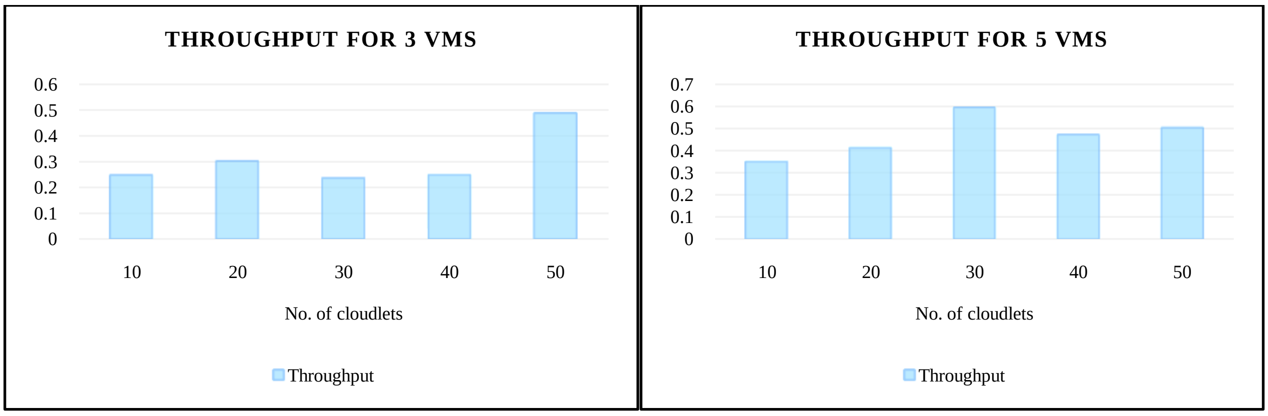


(a)Scenario-1 (b) Scenario-2

Fig. 16. Time taken results using the HISA-LB approach

Figure 16 shows the time results found by the proposed HISA-LB approach. Here, three-timing results are included in which maximum Makespan, minimum Makespan and average execution time are measured in seconds. In this figure, the x-axis specifies no. of cloudlets, or the y-axis specifies time in seconds. Fig. 16 (a) is used to display the Makespan and avg. Execution time results for 3 VMs in the first scenario. From this graph, it can be seen that initially, the Makespan is a minimum of 10 cloudlets. Makespan increases as the number of cloudlets or tasks increases, and the highest

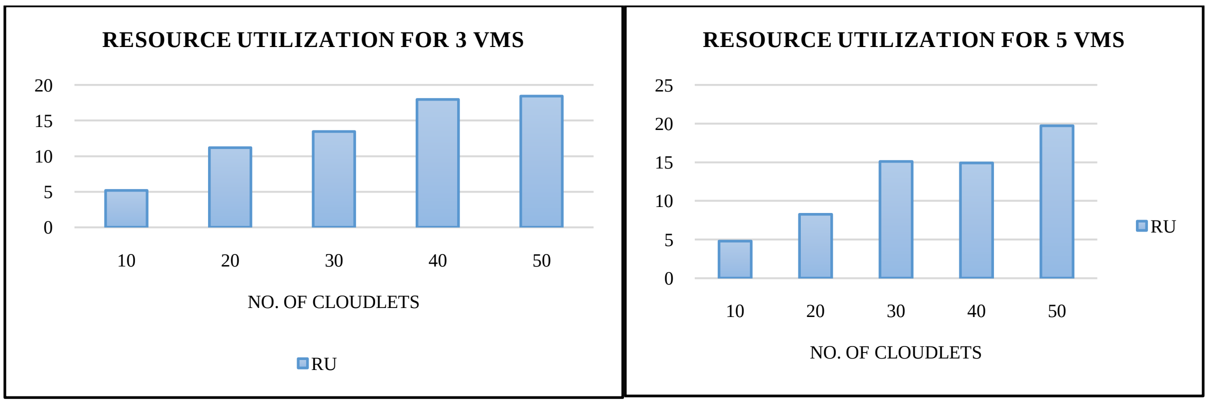
Makespan is at 40 cloudlets. But makespan decreases 50 cloudlets for all maximum and minimum values and reduces the average execution time. Similarly, Fig. 16 (b) displays the Makespan and avg. Execution time results for 5 VMs in the second scenario. This graph shows that initially, the maximum Makespan is less than 30 seconds at ten cloudlets, then it increases slowly and reaches the highest at 50 cloudlets by taking approx. 100 seconds. If we talk about the minimum Makespan, this is initially less than 5 seconds. It is constant with 7 to 8 seconds till the 50th cloudlets, and average execution time increases as the number of cloudlets or tasks increases with 6 seconds time differences at each task size.



(a) Scenario-1 (b) Scenario-2

Fig. 17. Throughput results using the HISA-LB approach

Figure 17 shows the throughput results found by the proposed HISA-LB approach. Here, the x-axis specifies the no. of cloudlets, and the y-axis specifies the throughput value. Fig. 17 (a) displays the throughput results for 3 VMs in the first scenario. This graph shows that the throughput value is highest at 50 cloudlets only among all cloudlets, achieving 48.3% throughput. But when looking for 5 VMs in scenario 2 (in figure 17 (b)), it achieved a higher throughput rate compared to the scenario1, and maximum throughput was achieved at 30 cloudlets size by achieving 59.2% throughput.



(a) Scenario-1 (b) Scenario-2

Fig. 18. Resource utilization results using the HISA-LB approach

Figure 18 shows the Resource utilization results found by the proposed HISA-LB approach. Here, the x-axis indicates several cloudlets of 10-50 size, and the y-axis indicates Resource utilization values. Fig. 18 (a) displays the Resource utilization results for 3 VMs in the first scenario. From this graph, it may be seen that Resource utilization rises as no. of tasks rises. Therefore, the highest resource utilization value is 18.44 at 50 cloudlets size.
Similarly, in fig. 18 (b), Resource utilization is increased per increment in cloudlets size for 5 VMs in the second scenario. But the Resource utilization is higher than in the first scenario because of maximum availability of VMs. So, the number of cloudlets can efficiently utilize their available resources and achieved the highest Resource utilization19.72 at 50 cloudlets.

Table 3. Comparison table for different experiments in Scenario-1

Scenarios	**Runs**	**Maximum Makespan**	**Minimum Makespan**	**Average Execution Time**	**Throughput**	**ARUR**	**RU**
HISA-LB	Experiment 1	40.71	10.68	20.99	0.246	0.692	5.18
	Experiment 2	66.66	9.54	37.2	0.300	0.763	11.19
	Experiment 3	127.48	16.21	57.09	0.235	0.622	13.46
	Experiment 4	161.93	23.22	72.66	0.247	0.639	17.97
	Experiment 5	103.48	6.21	38.07	0.483	0.511	18.44
LBMPSO	Experiment 1	54.1	4.27	24.04	0.185	0.604	4.46
	Experiment 2	107.8	13.38	49.22	0.186	0.565	9.15
	Experiment 3	155.26	13.81	69.16	0.193	0.553	13.38
	Experiment 4	208.63	23.54	92.18	0.192	0.539	17.69
	Experiment 5	246.82	24.47	102.41	0.203	0.541	20.77

Table 3 represents the best five experimental results for both proposed HISA-LB and existing LBMPSO approaches in the cloud computing running environment for the scenario1. It compares the experimented results on different cloudlets sizes in terms of min makespan and max makespan, avg. Execution time, throughput, average resource utilization, and resource utilization for scenario 1.

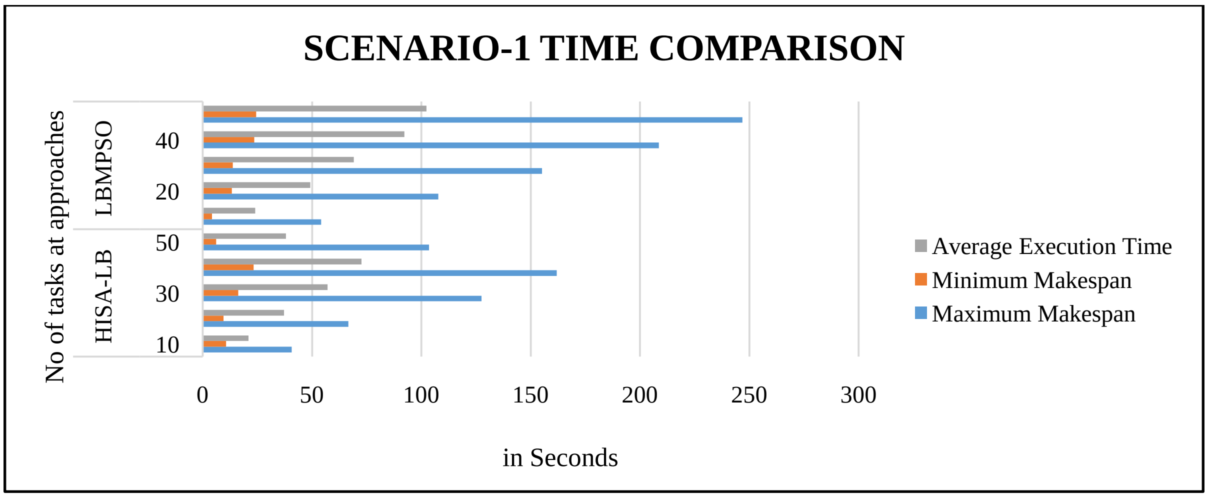


Fig. 19. Comparison graph of Makespan and execution time found for 3 VMs and 10-50 task sets

Fig. 19 represents the bar graph comparison of the HISA-LB approach with the LBMPSO approach for 3 VMs and 10-50 tasks in scenario 1 to show the comparison of Makespan. This bar comparison graph reveals that maximum Makespan is less, with a significant difference as the number of tasks increases. Initially, the maximum Makespan is 40.71 seconds at ten cloudlets, which is minimal to the LBMPSO approach, which takes 54.1 seconds. Still, the highest maximum Makespan was achieved at 40 tasks, which is 161.93 seconds by the HISA-LB approach. At the same time, minimum Makespan is also minimized in the proposed HISA-LB approach than LBMPSO except for ten and 30 tasks in this scenario. Also, a comparison of average execution time is made between the HISA-LB approach and the LBMPSO approach for scenario 1.

Similarly to maximum makespan comparison, HISA-LB takes minimized average execution time for all task sets compared to the LBMPSO approach. Here the considerable difference in execution time of both approaches with an increased number of tasks and least execution time is 20.99 seconds at ten tasks by HISA-LB. However, the highest execution time is 72.66 seconds for 40 tasks by HISA-LB, but LBMPSO takes more than 92.18 seconds for 40 tasks and 102.41 seconds for 50 tasks.

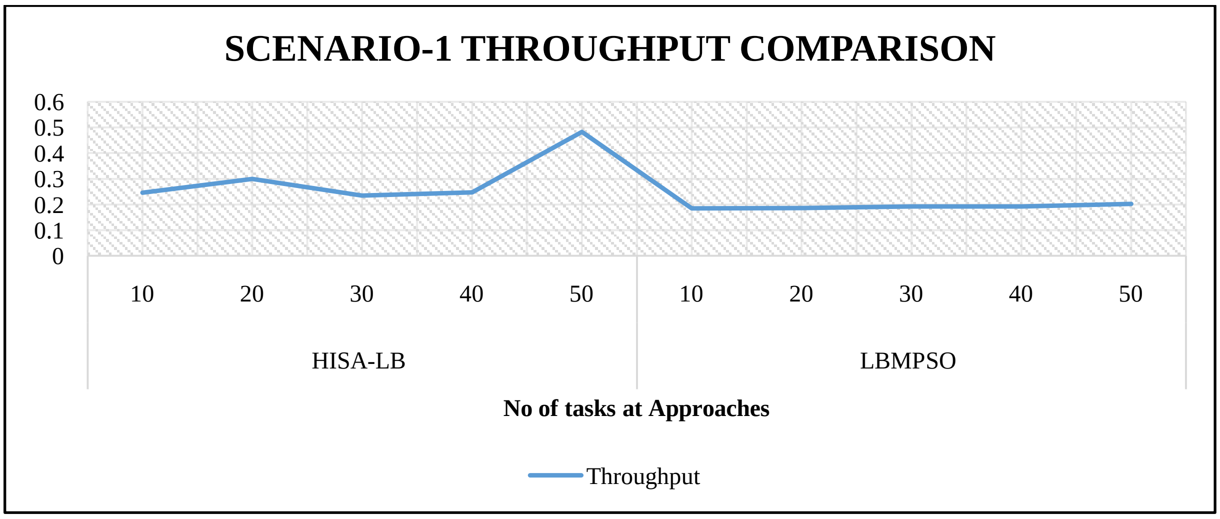


Fig. 20. Comparison graph of throughput found for 3 VMs and 10-50 task sets

Fig. 20 represents the throughput comparison between HISA-LB and LBMPSO approach for 3 VMs and 10-50 task sets in scenario 1. From this line comparison graph, we can see that throughput is variate as the number of tasks increases. Initially, throughput was 0.246 at ten cloudlets, then constant in between 30 to 40 cloudlets, but suddenly throughput increased to 0.483 at 50 cloudlets for the HISA-LB approach. But when we look for the LBMPSO approach, it achieved a constant throughput value from 10 to 20 task sets in a cloud with minor differences and lastly achieved 0.203 throughputs at 50 task sets.

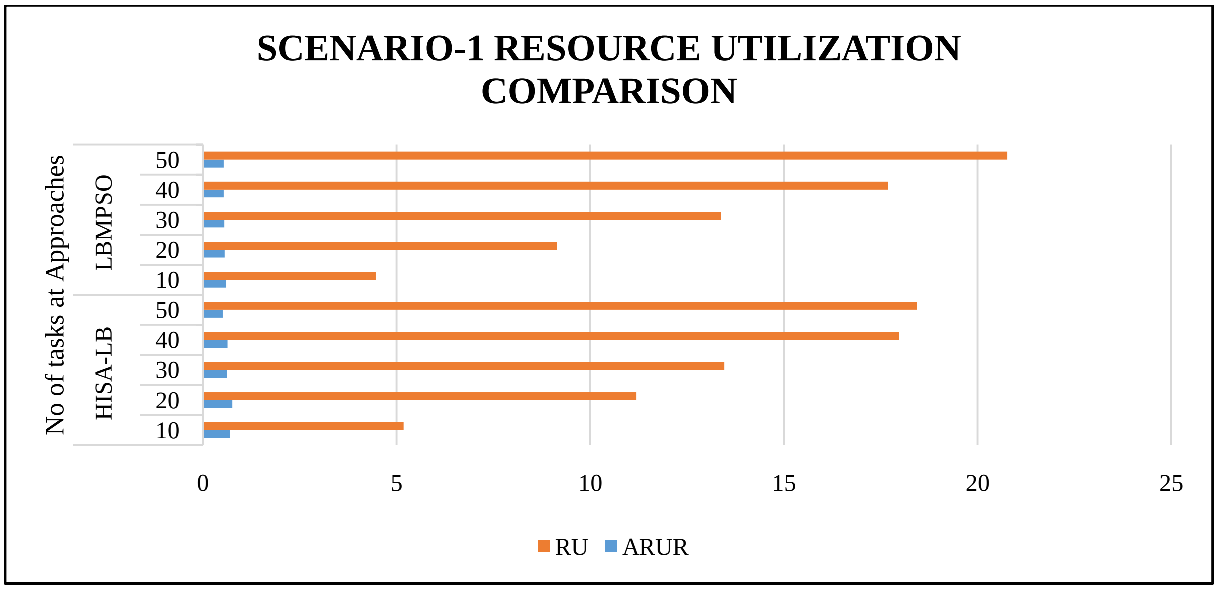


Fig. 21. Comparison graph of resource utilization found for 3 VMs and 10-50 task sets

Figure 21 depicts the resource utilization and average resource utilization in the HISA-LB technique when compared to other LBMPSO-based strategies with task sets ranging from 10-50 run across three virtual machines in scenario 1. Furthermore, regardless of the number of VMs, resource use increases with the tasks. This resource utilization is less at 50 cloudlets/tasks using the proposed HISA-LB approach than LBMPSO, achieving 18.44 and 20.77, respectively. Similarly, if resource utilization of the HISA-LB approach is good than LBMPSO, then ultimately, resource utilization is achieved the same results. ARUR is also good at each cloudlet size except 50 cloudlets.

Table 4. Comparison table for different experiments in Scenario-2

Scenarios	**Runs**	**Maximum Makespan**	**Minimum Makespan**	**Average Execution Time**	**Throughput**	**ARUR**	**RU**
HISA-LB	Experiment 1	28.65	4.66	13.62	0.349	0.641	4.79
	Experiment 2	48.55	7.56	19.94	0.412	0.57	8.26
	Experiment 3	50.67	7.48	25.41	0.592	0.676	15.10

	Experiment 4	85.24	7.15	31.68	0.469	0.539	14.91
	Experiment 5	99.7	7.98	39.23	0.501	0.550	19.72
LBMPSO	Experiment 1	27.4	4.44	14.03	0.365	0.702	5.16
	Experiment 2	100.1	4.93	37.21	0.199	0.328	7.46
	Experiment 3	92.1	4.95	29.58	0.326	0.431	9.67
	Experiment 4	128.83	6.74	40.56	0.310	0.404	12.63
	Experiment 5	130.23	8.83	45.69	0.384	0.456	17.58

Table 4 represents scenario two experimental results, including the best five experiments on 10-50 cloudlets for proposed HISA-LB and existing LBMPSO approaches in the cloud. The proposed HISA-LB is compared with LBMPSO performance parameters and achieves better results than the LBMPSO approach on different cloudlets.

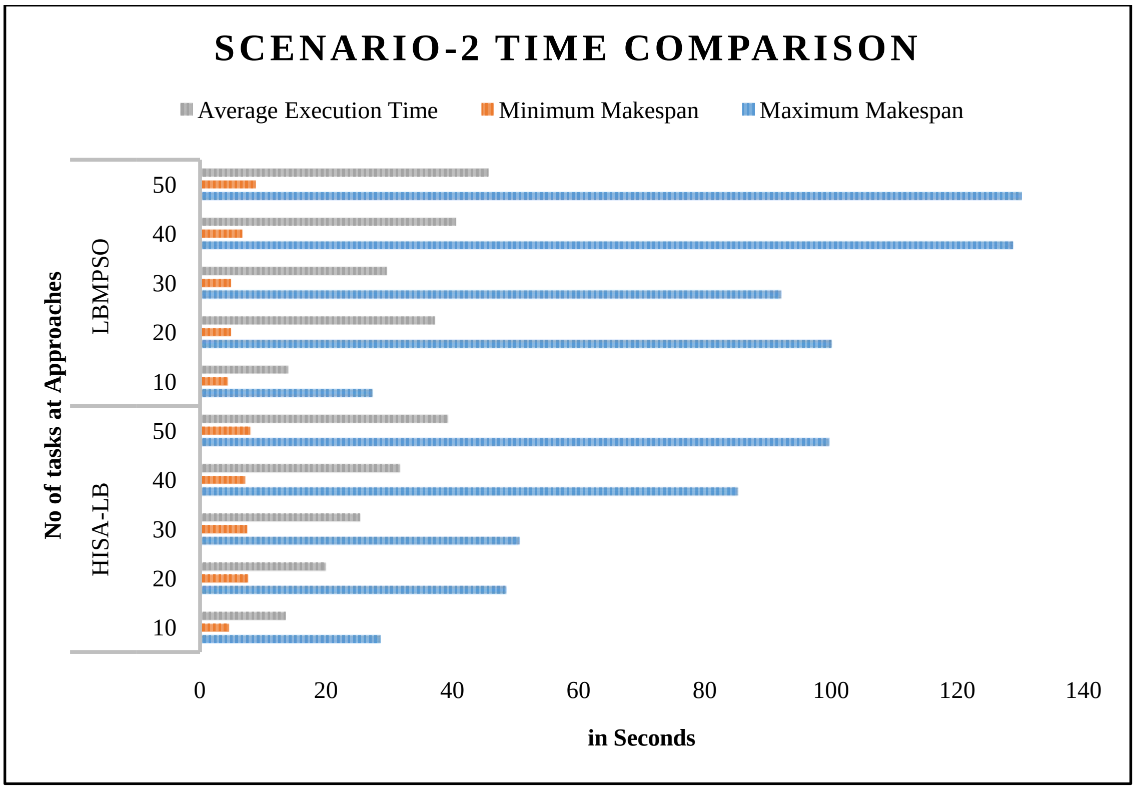


Fig. 22. Comparison graph of Makespan and execution time found for 5 VMs and 10-50 task sets

Figure 22 represents the bar graph comparison between of HISA-LB approach and the LBMPSO approach for 5 VMs and 10-50 tasks in scenario 2 to show the comparison of Makespan. From this comparison graph, we found that the maximum Makespan is very less as the number of tasks increases in HISA-LB. However, initially, the maximum Makespan is high 28.65seconds at ten cloudlets than the LBMPSO approach, which takes 27.4 seconds by exceeding 1.25 seconds. Still, it is very minimum for remaining all task sets from 20 to 50 cloudlets/tasks than LBMPSO maximum makespan, and the difference is just double approximately for maximum Makespan. If seeing at minimum Makespan, then found that minimum Makespan is high in HISA-LB compared to LBMPSO except for 50 tasks set, which takes 7.98 seconds and 8.83seconds, respectively, in scenario 2. Also, a comparison of average execution time is visualized between the HISA-LB approach and LBMPSO approach for scenario 2.

Similarly to the maximum makespan comparison, HISA-LB takes a reduced average execution time for all task sets compared to the LBMPSO approach. The difference in average execution time is significant using both approaches. If the number of tasks increases, then their execution time is also increasing, and the least execution time is 13.62 seconds for ten tasks by HISA-LB than 14.03 seconds by LBMPSO. However, the highest execution time is 39.23 seconds at 50 tasks by HISA-LB, but LBMPSO takes 45.69 seconds which is higher.

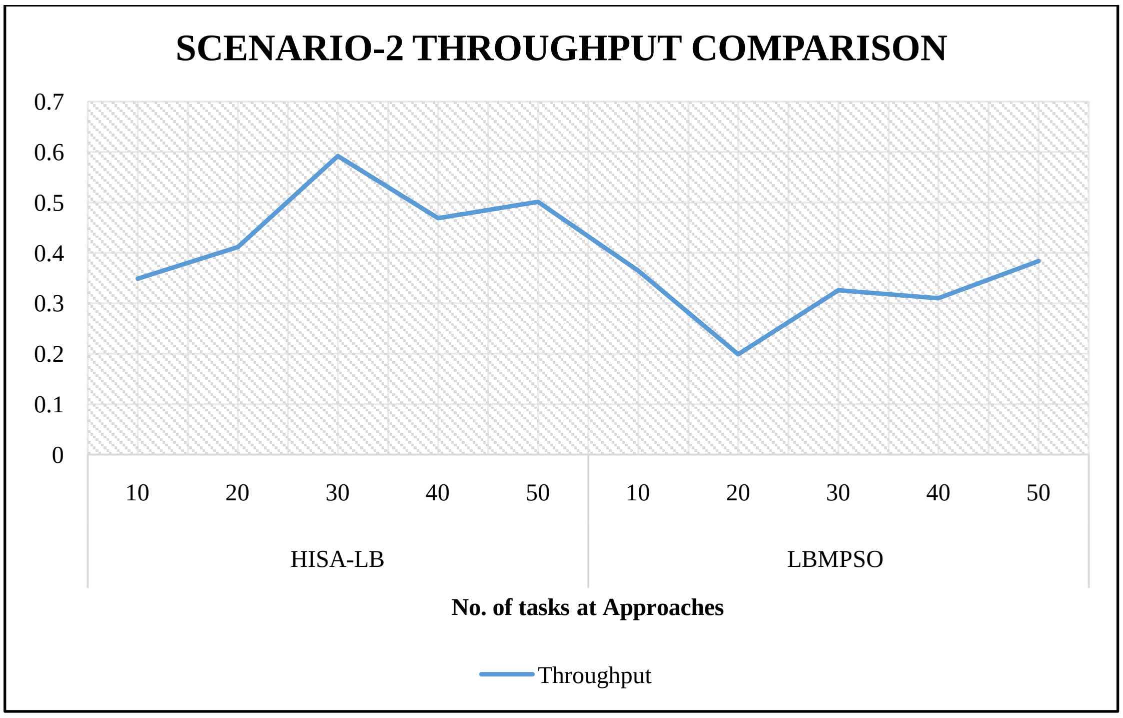


Fig. 23. Comparison graph of throughput found for 5 VMs and 10-50 task sets

Figure 23 represents the line comparison graph of throughput between HISA-LB and LBMPSO approaches for 5 VMs and 10-50 task sets in scenario 2. From this line graph, we can compare and find high variations in throughput values for several tasks. Initially, throughput is 0.349 at ten cloudlets, then it increases till 30 cloudlets with the highest throughput is 0.592, then it decreases at 40 cloudlets and increases at 50 cloudlets for the HISA-LB approach. When we look for the LBMPSO approach, it achieved not such a good throughput value as HISA-LB from 10-50 tasks sets but increased with several tasks minor differences except at 20 cloudlets which achieved 0.199 and lastly achieved 0.384 throughputs at 50 tasks set.

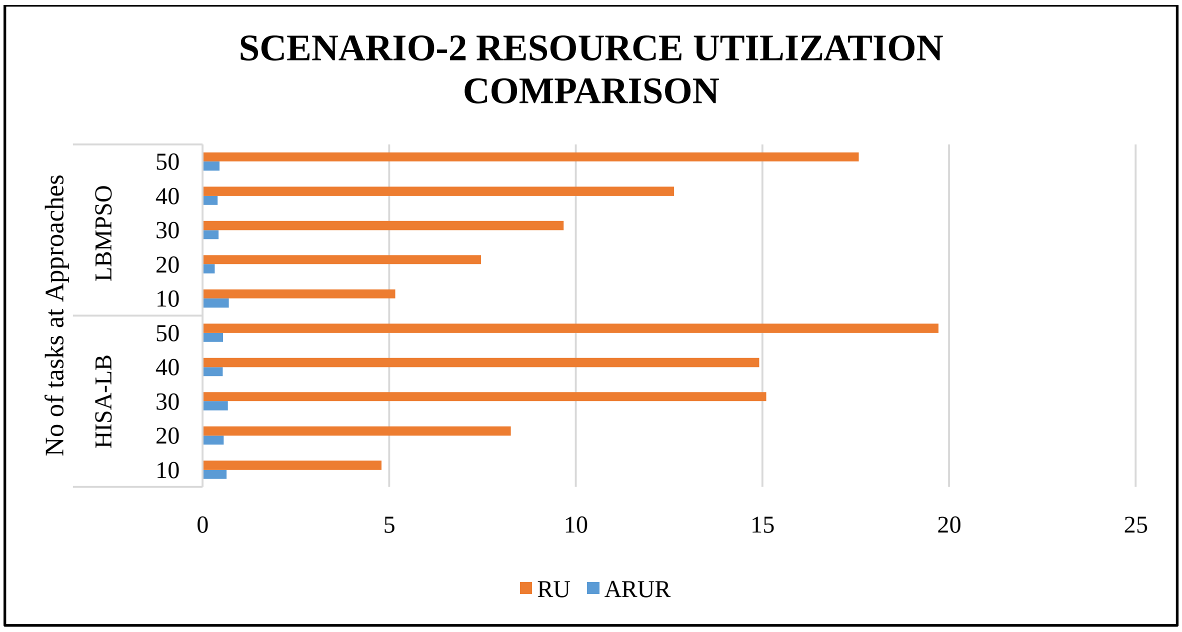


Fig. 24. Comparison graph of resource utilization found for 5 VMs and 10-50 task sets

Figure 24 represents the resource utilization with average resource utilization using HISA-LB and LBMPSO approach for 5 VMs or 10-50 task sets in scenario 2. From this comparison, we can see that as the number of tasks increases, resource utilization also increases, and the highest resource utilization HISA-LB achieves average utilization-LB compared to LBMPSO. However, resource utilization is high at each cloudlet's size, but the highest resource utilization was achieved, i.e., 19.72 at 50 cloudlets by HISA-LB, while LBMPSO achieved 17.58.

VI. Conclusion

The scheduling process in a cloud computing infrastructure involves several difficult challenges, including computation time, cost, load balancing, and others. One of the most significant issues for the Cloud infrastructure is LB. LB is the balancing load to obtain greater throughput or improved resource usage. Because scheduling is an NP-complete issue, heuristic or meta-heuristic techniques are favored solutions. In this research, we used a nature-inspired meta-heuristic technique to tackle task scheduling problems in a cloud context, concentrating on two key goals: decreasing the makespan/ computation time and improving LB. This work introduces HISA-LB, an LB strategy based on harmony inspired by simulated annealing that may reduce total makespan time, boost resource usage, and balance the load for each virtual machine. CloudSim Toolkit was used to run the simulations. The results of the tests show that, under each situation, HISA-LB reduces the makespan time and increases resource consumption compared to the LBMPSO approach. Furthermore, as the number of tasks grows, so does resource usage with throughput in all circumstances.

In preparation for future work, we are putting a lot of effort into developing a novel approach that, in the not-too-distant future, will allow for an improvement in the quality of service characteristics.

References

[1] G. Singh, M. Malhotra, and A. Sharma, “A Comprehensive Study on Virtual Machine Migration Techniques of Cloud Computing,” in *Lecture Notes in Electrical Engineering*, vol. 553, 2019, pp. 591–603. doi: 10.1007/978-981-13-6772-4_51.

[2] N. Joshi, K. Kotecha, D. B. Choksi, and S. Pandya, “Implementation of Novel Load Balancing Technique in Cloud Computing Environment,” in *2018 International Conference on Computer Communication and Informatics, ICCCI 2018*, 2018, pp. 1–5. doi: 10.1109/ICCCI.2018.8441212.

[3] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, “An iterative mathematical decision model for cloud migration: A cost and security risk approach,” *Softw. - Pract. Exp.*, vol. 48, no. 3, pp. 449–485, 2018, doi: 10.1002/spe.2528.

[4] N. A. Joshi, “Performance-Centric Cloud-Based e-Learning,” *IUP J. Inf. Technol.*, vol. 10, no. 2, pp. 7–17, 2014.

[5] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, “A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: Taxonomy and challenges,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 3, pp. 267–286, 2020, doi: https://doi.org/10.1016/j.jksuci.2018.07.001.

[6] M. Haris and S. Zubair, “Mantaray modified multi-objective Harris hawk optimization algorithm expedite optimal load balancing in cloud computing,” *J. King Saud Univ. - Comput. Inf. Sci.*, 2022, doi: 10.1016/j.jksuci.2021.12.003.

[7] A. F. S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. L. Lydia, and K. Shankar, “Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments,” *J. Parallel Distrib. Comput.*, vol. 142, pp. 36–45, 2020, doi: https://doi.org/10.1016/j.jpdc.2020.03.022.

[8] K. Balaji, P. Sai Kiran, and M. Sunil Kumar, “An energy-efficient load balancing on cloud computing using adaptive cat swarm optimization,” *Mater. Today Proc.*, 2021, doi: 10.1016/j.matpr.2020.11.106.

[9] R. Kaviarasan, P. Harikrishna, and A. Arulmurugan, “Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization,” *Adv. Eng. Softw.*, vol. 169, p. 103128, 2022, doi: 10.1016/j.advengsoft.2022.103128.

[10] K. S. Kannan, G. Sunitha, S. N. Deepa, D. Vijendra Babu, and J. Avanija, “A multi-objective load balancing and power minimization in the cloud using bio-inspired algorithms,” *Comput. Electr. Eng.*, vol. 102, p. 108225, 2022, doi: https://doi.org/10.1016/j.compeleceng.2022.108225.

[11] U. K. Jena, P. K. Das, and M. R. Kabat, “Hybridization of a meta-heuristic algorithm for load balancing in a cloud computing environment,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, Part A, pp. 2332–2342, 2022, doi: https://doi.org/10.1016/j.jksuci.2020.01.012.

[12] A. K. Sharma, K. Upreti, and B. Vargis, “Experimental performance analysis of load balancing of tasks using honey bee inspired algorithm for resource allocation in a cloud environment,” *Mater. Today Proc.*, 2020, doi: 10.1016/j.matpr.2020.09.359.

[13] M. Kumar and S. C. Sharma, “Load balancing algorithm to minimize the makespan time in a cloud environment,” *UK World J. Model. Simul.*, vol. 1, no. 4, pp. 276–288, 2018.

[14] Z. W. Geem and J. C. Williams, “Harmony search and ecological optimization,” *Int. J.*

Energy Environ., vol. 1, pp. 150–154, 2007.

[15] M. Mahdavi, M. Fesanghary, and E. Damangir, “An improved harmony search algorithm for solving optimization problems,” *Appl. Math. Comput.*, vol. 188, no. 2, pp. 1567–1579, 2007, doi: https://doi.org/10.1016/j.amc.2006.11.033.

[16] K. S. Lee and Z. W. Geem, “A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice,” *Comput. Methods Appl. Mech. Eng.*, vol. 194, no. 36, pp. 3902–3933, 2005, doi: https://doi.org/10.1016/j.cma.2004.09.007.

[17] O. Abdel-Raouf and M. Metwally, “A Survey of Harmony Search Algorithm,” *Int. J. Comput. Appl.*, vol. 70, pp. 17–26, 2013, doi: 10.5120/12255-8261.

[18] B. Mondal and A. Choudhury, “Simulated Annealing (SA) based Load Balancing Strategy for Cloud Computing,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 4, pp. 3307–3312, 2015.