

# Navigating the Cloud: An In-Depth Exploration of HISA Load Balancing for Dynamic Task Appropriation

Ms. Shruti Tiwari<sup>1</sup>, Dr. Chinmay Bhatt<sup>2</sup>

<sup>1</sup>Research Scholar, Department of Computer Science Engineering, RKDF College, Bhopal (M. P.), India

<sup>2</sup>Professor, Department of Computer Science Engineering, RKDF College, Bhopal (M. P.), India

**Corresponding Author:**

**Ms. Shruti Tiwari**

Department of Computer Science Engineering, RKDF College, Bhopal (M. P.), India

Email: shruti.tiwari08@gmail.com

## Abstract

In a cloud computing (CC) environment, jobs exhibit variations in durations, start times, and execution times when assigned to virtual machines (VMs). Therefore, achieving load balancing (LB) across these VMs becomes crucial to optimize system efficiency and performance. The present research introduces a novel LB method leveraging two optimization algorithms to address VM load balancing challenges. The proposed Dynamic Improved HISA Load Balancing approach integrates an augmented harmony-inspired algorithm with a simulated annealing algorithm for dynamic task allocation. In the harmony-inspired algorithm, an improved strategy for calculating Harmony Memory Consideration Rate (HMCR) is employed through a linear decreasing approach, updating HMCR and Pitch Adjustment Rate (PAR) values dynamically. A threshold probability is then evaluated to determine the fitness suitability of the current Harmony, choosing either the augmented harmony-inspired algorithm or simulated annealing for task allocation across available cloud resources. Simulations are conducted using the CloudSim simulator, considering scenarios with 3 or 5 VMs and 10 to 50 cloudlets. Each scenario is tested five times under operational conditions, and only the best performance outcomes are reported. Experimental results specify that the proposed Dynamic Enhanced HISA-LB approach outperforms the prevalent LBMP approach, demonstrating either minimized makespan or enhanced resource utilization with increased performance.

**Keywords:** Cloud computing, Load balancing, Task scheduling, Dynamic task allocation, Improved Harmony search, Simulated annealing.

## 1. INTRODUCTION

Cloud computing (CC) represents an extensive distributed computing model characterized by abstraction, virtualization, and dynamism, with economic scalability being a critical factor. It encompasses supervised computing ability, devices, platforms, and services provided to unfamiliar customers via the Internet on demand. The overarching aim of CC is to alleviate users from the burdens of supervising hardware, software including data assets along with outsourcing them to cloud service providers [1]. Clouds offer a plethora of resources including high-performance computing systems, data centers, storage, and software applications. Additionally, cloud computing facilitates resource management by enabling seamless access to these resources from anywhere without encountering performance constraints. Cloud resources and services are typically categorized into 3 levels: SaaS, PaaS and IaaS[2].

Effective resource management in cloud computing heavily relies on advanced techniques for instance load balancing. Load balancing involves redistributing task workloads across various nodes within a cloud computing platform. This technique entails identifying overloaded and underutilized cloud machines and migrating workloads accordingly to optimize resource utilization. By preventing instances of virtual machines from becoming overburdened, underloaded, or inactive, load balancing contributes to optimal cloud resource utilization [3]. A wide array of load-balancing algorithms have been proposed in the literature as are commonly deployed in both open and closed cloud computing environments[4].

Task scheduling under a cloud environment poses significant challenges due to the dynamic nature of task volumes and durations, rendering it an NP-hard combinatorial optimization problem. The complexities involved in mapping tasks to resources necessitate the development of efficient task

scheduling (TS) strategies capable of effectively addressing a computational problem H problems [5][6]. Researchers get focused on approximate and not precise, meta-heuristic, and hybrid scheduling algorithms to tackle these challenges, with swarm intelligence techniques gaining prominence. PSO, introduced by Kennedy and Eberhart, stands out as a prominent swarm intelligence optimization technique [7]. Building upon PSO, the reformed PSO algorithm well-known LBMP SO was proposed in [8] to address load balancing and task scheduling challenges. LBMP SO utilizes a fitness function to determine the optimal arrangement of particles, with the fitness function computing execution times for each virtual machine (VM) and returning shoot up implementation count as the particle's fitness code(F). Anyhow, conventional MP SO methods may not be suitable for all scenarios, particularly as the problem size increases.

To address these challenges comprehensively, a novel dynamic load balancing perspective is offered, merging enhanced Harmony Search Algorithm (HSA) and Simulated Annealing Algorithm (SAA) in a cloud computing domain. This approach aims to dynamically allocate tasks for coherent materials usage and minimize makespan.

The primary benefaction of that view are outlined as follows:

1. Addressing task scheduling (TS) and load Optimization (LO) challenges across various virtual machines (VMs) including cloudlets.
2. Introducing a novel Dynamic Enhanced HISA Load Balancing proposal that leverages 2 expansion algorithms to enhance harmony memory.
3. Optimizing the enhanced HISA-LB perspective through dynamic task allocation, updating parameters such as Harmony Memory Consideration Rate (HMCR), Pitch Adjustment Rate (PAR), and Fret width for efficient work scheduling.
4. Implementing dynamic task allocation to achieve load balancing in different scenarios.
5. Comparing the preferred process with the subsisting LBMP SO algorithm applying various executions metrics to reduce makespan and raise assets fulfillment.

The structure of the examination is classified as observe:

- Section I reviews existing cloud load balancing strategies.
- Section II introduces the new dynamic and efficient load balancing approach.
- Section III discusses test results obtained from experiments conducted in a cloud-running nature.
- Finally, Section IV concludes the work and provides recommendations for future work.

## 2. Method

The indicated module present a novel load-balancing method termed Dynamic enhanced HISA-LB, which integrates an enhanced harmony-stimulated algorithm and simulated annealing algorithm for dynamic task allocation to address the task scheduling (TS) and load balancing (LB) challenges discussed below.

**Proposed Solution:** To mitigate these issues, we propose a task scheduling algorithm that considers both task allocation and host load balancing. This approach aims to prevent host overloading to meet user time requirements, thereby adhering to SLAs and enhancing Quality of Service (QoS). Additionally, it identifies underloaded hosts to reduce energy consumption and increase throughput.

### Enhanced Harmony-Inspired Algorithm for Dynamic Task Allocation

In the Improved Harmony-Inspired Algorithm (IHIA), harmony represents a potential solution, where all determination variable correlate with a note. The algorithm employs a Harmonic Memory (HM) to store a predestined quantity of harmonies (N). The objective is to either decrease or increase a abilityfunction (f) influenced by d decision factors, defined as follows:

$$f(x_i) = \sum_{i=1}^d (w_i \times t_i) + U$$

Where:  $f(x_i)$  is the fitness function of the Dynamic Enhanced HISA-LB.

$w_i$  is the waiting time for task assignment to a VM.

$t_i$  is the resource utilization.

U represents the summation of waiting times and resource utilization.

The algorithm progresses through the following stages:

**Step 1: Inception of HM** At the start of the Harmony Search (HS), N harmonies are created in the metric span and kept in HM. Each harmony, denoted as Harmony I, is represented by a vector: Harmony I =  $[x_{i1}x_{i1}, x_{i2}x_{i2}, \dots, x_{id}x_{id}]$ . This initialization process can be defined using the equation:

$$x_{ij} = \text{Rand} \times (\text{Max}_j - \text{Min}_j) + \text{Min}_j$$

Where  $Rand$  is a uniformly distributed random number between 0 and 1, and  $Max_j$  and  $Min_j$  are the maximum and minimum values for decision variable  $j$ , respectively.

**Step 2:** Innovation of HM The next iteration involves improvising a current harmony, denoted as  $x'$ , by considering all existing harmonies in the HM. This unique characteristic of the HS algorithm ensures the exploration of the entire solution space. The pitch of each component is evaluated to determine if adjustments are necessary, controlled by the Pitch Adjustment Rate (PAR) parameter. The pitch-adjustment process can be defined by the equation:

$$x_{ij}' = x_{ij} + PAR \times (Rand - 0.5)$$

Where  $Rand$  is a uniformly distributed random number between 0 and 1.

**HMCR:** The HMCR update process is defined as:

$$HMCR(t) = HMCR_{max} - (T_{max} - t) / T_{max} \times (HMCR_{max} - HMCR_{min})$$

Where:

- $t$  is the update replication number.
- $T_{max}$  represents the greatest number of replications.
- $HMCR_{max}$  and  $HMCR_{min}$  denote the increase and decrease values of HMCR, appropriately.

**Pitch Adjustment Rate:**

In this study, an effective switch method is adopted for PAR, and its correlative mathematical formula is as follows:

$$PAR(t) = PAR_{max} - (T_{max} - t) / T_{max} \times (PAR_{max} - PAR_{min})$$

Where:

- $PAR(t)$  is the pitch adjustment rate for generation  $t$ .
- $PAR_{max}$  is the maximum pitch adjustment rate.
- $PAR_{min}$  is the minimum pitch adjustment rate.
- $T_{max}$  represents the maximum number of generations or iterations.

**Fret Width**

This dynamic variation in  $F\_W$  is dependent on the generation number, as expressed by the following formula:

$$FW(t) = FW_{max} - (t) / (T_{max}) \times (FW_{max} - FW_{min})$$

Where:

$FW(t)$  is the pitch fret width for generation  $t$ .

$FW_{max}$  and  $FW_{min}$  are the increase and decrease harmony memory processing rates, appropriately.

$T_{max}$  represents the maximum number of generations or iterations.

The expansion pace for the refined HS algorithm are brief in following Algorithm 1.

### Algorithm 1: Improved HSA

**Method:**

- Step 1. Initialize HSA parameters.
- Step 2. Initialize maximal no. of iterations  $T_{max}$ ; the harmony memory size (HMS), maximal & minimal HM considering rate,  $HM\_CR_{max}$  &  $HM\_CR_{min}$ ; maximal and minimal pitch adjusting rate,  $P\_A\_R_{max}$  and  $P\_A\_R_{min}$ ; maximal and minimal fret width,  $F\_W_{max}$  or  $F\_W_{min}$ .
- Step 3. Initialize harmony memory (HM).
- Step 4. Establish novel Harmony based on HM. Utilize equations (2), (3), (4), (5), as well as (6) to create a novel harmony.
- Step 5. Increase HM. Employ (1) to assess the suitability of the new Harmony. If novel Harmony is superior to the least favorable harmonic in HM, the least favourable Harmony is removed from HM and novel Harmony is added.
- Step 6. Evaluate the termination condition. Improved HS will end if no. of iterations exceeds maximal no. of iterations  $T_{max}$ . Otherwise, proceed to Step 4.
- Step 7. **Return** The best-optimized solution in the harmony memory for dynamic task allocation

1) **Simulated Annealing**

In this work, SA is incorporated into the Harmony Search (HS) algorithm by utilizing in the other ways provision when expectations are subordinate or equal to the Harmony Memory Consideration Rate (HMCR), thereby augmenting the remainder extent of Harmony-Inspired (HI) algorithms. The SA way is outlined as follows:

Step 1: Initialize the initial value.

Step 2: Generate a new feasible solution  $x'x'$  under temperature  $TT$ , where  $x'x'$  is in the neighborhood solution of the current solution  $xx$ .

Step 3: Evaluate the change in fitness function  $\Delta f\Delta f$  between the current solution  $xx$  and the new solution  $x'x'$ .

Step 4: Accept the new solution with a certain probability  $pp$ , where  $pp$  is a random number between 0 and 1. If the solution reaches a temperature balance status, proceed to Step 5; otherwise, return to Step 2.

Step 5: Decrease the temperature using a specific function. The temperature drop function may vary, typically denoted as  $T_{new}=\alpha \times T$ , where  $\alpha$  is a cooling factor such as 0.9.

The suggested dynamic enhanced HISA load balancing method is described in narrate via pidgincode in Algorithm 3, including the schema for the present design can be seen in Figure 3.

**Algorithm 3. Dynamic Improved HISA Load Balancing Approach**

**Pseudocode:**

1. Begin
2. Generate the harmony memory
3. Initialization of all parameters,  $tmp$  (temperature),  $Num(t) = 0$ ,  $T_{max} = 5000$
4. Calculate all parameters values
5. Evaluate the fitness function by the formula
 
$$Fit = T_{wait} + Makspan + RU$$
6. Place BestX (the best solution in Harmony Memory (HM)) into  $\delta$
7.  $BstSol = BstSASol = \delta$
8. For  $p = (1 \text{ to } t)$  do
9. Update the HMCR by the given formula
 
$$HM\_CR(t) = HM\_CR_{max} - \frac{(HM\_CR_{max} - HM\_CR_{min}) * t}{T_{max}}$$
10. If  $(\text{random}(0,1) \leq HM\_CR)$  Then
11. Select 2 vectors solution symbolized as  $v1$  &  $v2$  at random from Harmony memory
12. Update PAR by dynamic change strategy by the given formula
 
$$P\_A\_R(t) = \frac{(P\_A\_R_{max} - P\_A\_R_{min})}{\frac{\pi}{2}} * \arctan t + P\_A\_R_{min}$$
13.  $F\_W$  changes dynamically with the number of iterations by the given formula
 
$$F\_W(t) = F\_W_{max} - \frac{(F\_W_{max} - F\_W_{min}) * t}{T_{max}}$$
14. If  $(\text{random}(0,1) \leq P\_A\_R)$  Then
15.  $v =$  put on a PMX crossover to  $v1$  and  $v2$
16.  $newX =$  the best neighbor amongst some of the neighbors created by  $v$
17. If  $(\text{SumFit}(newX) < \text{SumFit}(worstX))$  Then
18. Swap  $worstX$  by  $newX$  //Modifying the HM
19. End of If
20. End of If
21. End of If
22. Else
23.  $\delta'$  = the best neighbor among the generated neighbors of  $\delta$
24.  $\Delta Fit = \text{SumFit}(\delta') - \text{SumFit}(\delta)$
25.  $\text{probability} = \text{Random}(0,1)$
26. If  $((\Delta Fit \leq 0) \text{ or } (\text{probability} < e^{-\Delta Fit/tmp}))$  Then

```

27.   $\delta = \delta'$ , newX =  $\delta'$ 
28.  If (SumFit(newX ) < SumFit(worstX)) Then
29.  Swap the worstX by newX      // Modifying the HM
30.  End of If
31.  If (SumFit( $\delta$ ) < SumFit(BstSASol)) Then
32.  BstSASol =  $\delta$ 
33.  End of If
34.  End of If
35.  tmp = Modify (tmp)
36.  End of Else
37.  If (SumFit(BestX ) < SumFit(BstSol))
38.  BstSol = BestX, t = 0
39.  End of If
40.  If (SumFit(BstSASol) - SumFit(BstSol)  $\geq T_{max}$ )
41.   $\delta =$  BstSol
42.  End of If
43.  End of For
44.  Get the result BstSol together with its fitness function value
45.  Stop.
    
```

**Output:** Dynamic task allocation

### 3. RESULTS AND DISCUSSION

The suggested Dynamic enhanced HISA load balancing algorithm exist initiated applying the CloudSim tool via implemented in the Eclipse Java framing surroundings. Various specifications were evaluated to assess the performance of the suggested algorithm and facilitate comparison with existing techniques. One crucial parameter evaluated is Resource

Utilization, which represents the prospects concerning assets absorbed at the approaching task. It provides insights into how efficiently the cloud resources are utilized to handle the workload. In this section, the results obtained from experiments conducted on two cloud cases using the proposed effective enhanced HISA load balancing method are presented. The referred outcome are depicted through screencast, tables, including pictorial representations for scenarios involving 3VMs and 5VMs with 10 to 50 cloudlets.

1) For Case 1, where 3 virtual machines are used with cloudlets ranging from 10 to 50:

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
8	SUCCESS	2	2	6.7	0.1	6.8
3	SUCCESS	2	0	11.75	0.1	11.85
1	SUCCESS	2	2	16.68	0.1	16.78
2	SUCCESS	2	1	18.66	0.1	18.76
4	SUCCESS	2	1	20.34	0.1	20.44
9	SUCCESS	2	2	20.55	0.1	20.65
7	SUCCESS	2	2	23.72	0.1	23.82
0	SUCCESS	2	2	25.83	0.1	25.93
5	SUCCESS	2	1	28.59	0.1	28.69
6	SUCCESS	2	1	28.77	0.1	28.87

Maximum Makespan : 28.87  
 Minimum Makespan : 6.8  
 Average Execution Time : 20.158533333333333  
 Throughput : 0.34636432989229453  
 Average Resource Utilization Ratio : 0.7694906904965326  
 Resource Utilization : 7.0168333063938855  
 Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!

Fig. 3. 10 cloudlets at 3 VMs

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
17	SUCCESS	2	2	18.45	0.1	18.55
16	SUCCESS	2	1	18.68	0.1	18.78
18	SUCCESS	2	1	21.74	0.1	21.84
11	SUCCESS	2	1	25.11	0.1	25.21
3	SUCCESS	2	0	25.22	0.1	25.32
10	SUCCESS	2	1	30.82	0.1	30.92
13	SUCCESS	2	0	32.83	0.1	32.93
8	SUCCESS	2	0	35.38	0.1	35.48
14	SUCCESS	2	2	35.38	0.1	35.48
0	SUCCESS	2	2	39.1	0.1	39.2
4	SUCCESS	2	1	39.55	0.1	39.65
5	SUCCESS	2	1	39.67	0.1	39.77
6	SUCCESS	2	1	39.92	0.1	40.02
15	SUCCESS	2	2	45.2	0.1	45.3
2	SUCCESS	2	2	46.98	0.1	47.08
9	SUCCESS	2	2	48.08	0.1	48.18
12	SUCCESS	2	2	49.07	0.1	49.17
4	SUCCESS	2	0	49.18	0.1	49.28
19	SUCCESS	2	2	49.29	0.1	49.39
1	SUCCESS	2	0	50.18	0.1	50.28

Maximum Makespan : 50.28  
 Minimum Makespan : 18.45  
 Average Execution Time : 37.092266666666674  
 Throughput : 0.397408320738206  
 Average Resource Utilization Ratio : 0.9260891568778227  
 Resource Utilization : 16.79288308071422  
 Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!

Fig. 4. 20 cloudlets at 3 VMs

Figure 3: Case-1 along Three Virtual Machines including Ten Cloudlets

- Exhibit a graphical representation of the effect concerning Case-1 along 3- VMs including ten cloudlets.
- Cloudlet ID 6 has the extended conclude measures of 28.87 seconds, while Cloudlet ID 8 has the low end measures of 6.8 seconds.
- 5 tests conducted over that instance derived in a maximal makespan of 28.87 seconds.

- The mean assets utilization proportion is 0.769, and the utmost throughput is 0.346.

Figure 4: Case-1 along 3 Virtual Machines including 20 Cloudlets

- Represents the effect as Case-1 along 3 virtual machines including 20 cloudlets.

- Similar to Figure 3, it displays the details of each cloudlet including its ID, status, data center ID, virtual machine ID, start time, end time, and total execution time.
- Cloudlet ID 1 find the prolonged conclude count of 50.28 seconds, while Cloudlet ID 16 find the small conclude count of 18.55 seconds.
- 5 trial conducted throughout that case followed in a maximal makespan of 50.28 seconds, of the kind that is the minimal in all 5 trials.

```

===== Test (BMD) HISA (Java Application) C:\Program Files\Java\jre1.8.0_191\bin\java.exe (Oct 3, 2022, 4:05:04 PM)
***** OUTPUT *****
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
0 S SUCCESS 2 0 0 0.00 0.0 0.00
1 S SUCCESS 2 0 0 0.00 0.0 0.00
2 S SUCCESS 2 0 0 0.00 0.0 0.00
3 S SUCCESS 2 0 0 0.00 0.0 0.00
4 S SUCCESS 2 0 0 0.00 0.0 0.00
5 S SUCCESS 2 0 0 0.00 0.0 0.00
6 S SUCCESS 2 0 0 0.00 0.0 0.00
7 S SUCCESS 2 0 0 0.00 0.0 0.00
8 S SUCCESS 2 0 0 0.00 0.0 0.00
9 S SUCCESS 2 0 0 0.00 0.0 0.00
10 S SUCCESS 2 0 0 0.00 0.0 0.00
11 S SUCCESS 2 0 0 0.00 0.0 0.00
12 S SUCCESS 2 0 0 0.00 0.0 0.00
13 S SUCCESS 2 0 0 0.00 0.0 0.00
14 S SUCCESS 2 0 0 0.00 0.0 0.00
15 S SUCCESS 2 0 0 0.00 0.0 0.00
16 S SUCCESS 2 0 0 0.00 0.0 0.00
17 S SUCCESS 2 0 0 0.00 0.0 0.00
18 S SUCCESS 2 0 0 0.00 0.0 0.00
19 S SUCCESS 2 0 0 0.00 0.0 0.00
20 S SUCCESS 2 0 0 0.00 0.0 0.00
21 S SUCCESS 2 0 0 0.00 0.0 0.00
22 S SUCCESS 2 0 0 0.00 0.0 0.00
23 S SUCCESS 2 0 0 0.00 0.0 0.00
24 S SUCCESS 2 0 0 0.00 0.0 0.00
25 S SUCCESS 2 0 0 0.00 0.0 0.00
26 S SUCCESS 2 0 0 0.00 0.0 0.00
27 S SUCCESS 2 0 0 0.00 0.0 0.00
28 S SUCCESS 2 0 0 0.00 0.0 0.00
29 S SUCCESS 2 0 0 0.00 0.0 0.00
30 S SUCCESS 2 0 0 0.00 0.0 0.00
31 S SUCCESS 2 0 0 0.00 0.0 0.00
32 S SUCCESS 2 0 0 0.00 0.0 0.00
33 S SUCCESS 2 0 0 0.00 0.0 0.00
34 S SUCCESS 2 0 0 0.00 0.0 0.00
35 S SUCCESS 2 0 0 0.00 0.0 0.00
36 S SUCCESS 2 0 0 0.00 0.0 0.00
37 S SUCCESS 2 0 0 0.00 0.0 0.00
38 S SUCCESS 2 0 0 0.00 0.0 0.00
39 S SUCCESS 2 0 0 0.00 0.0 0.00
40 S SUCCESS 2 0 0 0.00 0.0 0.00
41 S SUCCESS 2 0 0 0.00 0.0 0.00
42 S SUCCESS 2 0 0 0.00 0.0 0.00
43 S SUCCESS 2 0 0 0.00 0.0 0.00
44 S SUCCESS 2 0 0 0.00 0.0 0.00
45 S SUCCESS 2 0 0 0.00 0.0 0.00
46 S SUCCESS 2 0 0 0.00 0.0 0.00
47 S SUCCESS 2 0 0 0.00 0.0 0.00
48 S SUCCESS 2 0 0 0.00 0.0 0.00
49 S SUCCESS 2 0 0 0.00 0.0 0.00
50 S SUCCESS 2 0 0 0.00 0.0 0.00
Maximum Makespan : 50.28
Minimum Makespan : 18.55
Average Execution Time : 0.01609777777777778
Throughput : 0.27782947763344813
Average Resource Utilization Ratio : 0.834423034674034
Resource Utilization : 26.39766828183478
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
  
```

Fig. 5. 30 cloudlets at 3 VMs

Based on the provided descriptions, Figures 5 and 6 illustrate the outcomes for Case-1 along 3 virtual machines including varying numbers of cloudlets. Here's a summary of the information presented in each figure:

Figure 5: Case-1 along 3 Virtual Machines including 30 Cloudlets

- Cloudlet ID 3 find the prolonged completion count, clocking in at 79.4 seconds.
- 5 trials conducted throughout that case concluded in a maximal makespan of 79.4 seconds, of the kind that is the minimal in all 5 trials. The minimal makespan observed was 18.63 seconds.
- The greatest assets fulfillment recorded is 20.748, with an average resource utilization ratio (ARUR) of 0.83.
- The maximal throughput achieved is 0.377.

```

===== Test (BMD) HISA (Java Application) C:\Program Files\Java\jre1.8.0_191\bin\java.exe (Oct 3, 2022, 4:29:45 PM)
***** OUTPUT *****
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
7 S SUCCESS 2 3 5.66 0.1 5.76
8 S SUCCESS 2 3 12.13 0.1 12.23
9 S SUCCESS 2 4 12.9 0.1 13
10 S SUCCESS 2 3 15.04 0.1 15.14
11 S SUCCESS 2 3 15.15 0.1 15.25
12 S SUCCESS 2 3 17.92 0.1 18.02
13 S SUCCESS 2 4 20.66 0.1 20.76
14 S SUCCESS 2 2 21.12 0.1 21.22
15 S SUCCESS 2 2 22.02 0.1 22.12
16 S SUCCESS 2 4 22.14 0.1 22.24
Maximum Makespan : 22.24
Minimum Makespan : 5.76
Average Execution Time : 16.471966666666667
Throughput : 0.4496537665997183
Average Resource Utilization Ratio : 0.5609340807578165
Resource Utilization : 7.451647231631644
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
  
```

Fig. 7. 50 cloudlets at 3 VMs

Based on the description provided, Figure 7 illustrates the outcomes for Case-1 along 3 virtual machines including 50 cloudlets. Here's a summary of the information presented in the figure:

- Cloudlet ID 38 is successfully executed at Datacenter ID 2 as no VM allocation including took 155.1 seconds to concluded.

- The greatest assets fulfillment is 14.79, the mean assets fulfillment proportion is 0.926, including the maximal throughput is 0.397.

These figures provide insights into the performance of the Dynamic enhanced HISA load balancing method under different workload scenarios, highlighting metrics such as makespan, resource utilization, and throughput.

```

===== Test (BMD) HISA (Java Application) C:\Program Files\Java\jre1.8.0_191\bin\java.exe (Oct 3, 2022, 4:12:41 PM)
***** OUTPUT *****
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
0 S SUCCESS 2 0 0.00 0.0 0.00
1 S SUCCESS 2 0 0.00 0.0 0.00
2 S SUCCESS 2 0 0.00 0.0 0.00
3 S SUCCESS 2 0 0.00 0.0 0.00
4 S SUCCESS 2 0 0.00 0.0 0.00
5 S SUCCESS 2 0 0.00 0.0 0.00
6 S SUCCESS 2 0 0.00 0.0 0.00
7 S SUCCESS 2 0 0.00 0.0 0.00
8 S SUCCESS 2 0 0.00 0.0 0.00
9 S SUCCESS 2 0 0.00 0.0 0.00
10 S SUCCESS 2 0 0.00 0.0 0.00
11 S SUCCESS 2 0 0.00 0.0 0.00
12 S SUCCESS 2 0 0.00 0.0 0.00
13 S SUCCESS 2 0 0.00 0.0 0.00
14 S SUCCESS 2 0 0.00 0.0 0.00
15 S SUCCESS 2 0 0.00 0.0 0.00
16 S SUCCESS 2 0 0.00 0.0 0.00
17 S SUCCESS 2 0 0.00 0.0 0.00
18 S SUCCESS 2 0 0.00 0.0 0.00
19 S SUCCESS 2 0 0.00 0.0 0.00
20 S SUCCESS 2 0 0.00 0.0 0.00
21 S SUCCESS 2 0 0.00 0.0 0.00
22 S SUCCESS 2 0 0.00 0.0 0.00
23 S SUCCESS 2 0 0.00 0.0 0.00
24 S SUCCESS 2 0 0.00 0.0 0.00
25 S SUCCESS 2 0 0.00 0.0 0.00
26 S SUCCESS 2 0 0.00 0.0 0.00
27 S SUCCESS 2 0 0.00 0.0 0.00
28 S SUCCESS 2 0 0.00 0.0 0.00
29 S SUCCESS 2 0 0.00 0.0 0.00
30 S SUCCESS 2 0 0.00 0.0 0.00
31 S SUCCESS 2 0 0.00 0.0 0.00
32 S SUCCESS 2 0 0.00 0.0 0.00
33 S SUCCESS 2 0 0.00 0.0 0.00
34 S SUCCESS 2 0 0.00 0.0 0.00
35 S SUCCESS 2 0 0.00 0.0 0.00
36 S SUCCESS 2 0 0.00 0.0 0.00
37 S SUCCESS 2 0 0.00 0.0 0.00
38 S SUCCESS 2 0 155.1 0.1 155.1
39 S SUCCESS 2 0 0.00 0.0 0.00
40 S SUCCESS 2 0 0.00 0.0 0.00
Maximum Makespan : 155.1
Minimum Makespan : 0.00
Average Execution Time : 74.31879166666667
Throughput : 0.8238422222222222
Average Resource Utilization Ratio : 0.8238775982787662
Resource Utilization : 27.36138224881014
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
  
```

Fig. 6. 40 cloudlets at 3 VMs

Figure 6: Case-1 along 3 Virtual Machines including 40 Cloudlets

- Depicts the outcomes for the scenario with three virtual machines and forty cloudlets.
- Five tests conducted during that case concluded in a maximal makespan of 108.86 seconds as a minimal makespan of 19.63 seconds.
- The maximal assets fulfillment observed is 27.34, including the maximal throughput achieved is 0.367.

These figures provide detailed insights into the performance of the Dynamic Enhanced HISA load balancing method under different workload scenarios, highlighting metrics such as makespan, resource utilization, and throughput.

- Cloudlet ID 3 has a start time of 0.1 seconds.
- The maximum resource usage observed is 26.357, with an average resource utilization ratio (ARUR) of 0.74.
- The maximal performance achieved is 0.322.

2) For Case 2, where 5 virtual machines are used with cloudlets ranging from 10 to 50:

```

===== OUTPUT =====
Cloudlet ID  STATUS  Data center ID  VM ID  Time  Start Time  Finish Time
15  SUCCESS  2  2  5.49  0.1  5.59
13  SUCCESS  2  3  6.86  0.1  6.96
19  SUCCESS  2  2  7.29  0.1  7.39
18  SUCCESS  2  4  7.47  0.1  7.57
9  SUCCESS  2  2  9.1  0.1  9.2
4  SUCCESS  2  3  10.11  0.1  10.21
5  SUCCESS  2  4  10.22  0.1  10.32
11  SUCCESS  2  3  11.15  0.1  11.25
6  SUCCESS  2  4  14.11  0.1  14.21
12  SUCCESS  2  4  15.34  0.1  15.44
1  SUCCESS  2  3  17.73  0.1  17.83
0  SUCCESS  2  3  19.65  0.1  19.75
7  SUCCESS  2  3  20.3  0.1  20.4
14  SUCCESS  2  3  22.1  0.1  22.2
10  SUCCESS  2  3  22.98  0.1  23.08
2  SUCCESS  2  3  23.37  0.1  23.47
17  SUCCESS  2  4  24.6  0.1  24.7
16  SUCCESS  2  4  24.96  0.1  25.06
8  SUCCESS  2  4  26.48  0.1  26.58
3  SUCCESS  2  4  26.94  0.1  27.04
Maximum Makespan : 27.04
Minimum Makespan : 5.59
Average Execution Time : 16.308441666666667
Throughput : 0.739655912669705
Average Resource Utilization Ratio : 0.5442851718713788
Resource Utilization : 12.13660886600885
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
    
```

Fig. 9. 20 cloudlets at 5 VMs

```

===== OUTPUT =====
Cloudlet ID  STATUS  Data center ID  VM ID  Time  Start Time  Finish Time
28  SUCCESS  2  2  14.69  0.1  14.79
10  SUCCESS  2  2  17.18  0.1  17.28
23  SUCCESS  2  4  19.23  0.1  19.33
24  SUCCESS  2  3  20.26  0.1  20.36
3  SUCCESS  2  4  23.33  0.1  23.43
8  SUCCESS  2  4  24.26  0.1  24.36
4  SUCCESS  2  4  24.61  0.1  24.71
25  SUCCESS  2  4  25.13  0.1  25.23
16  SUCCESS  2  3  25.96  0.1  26.06
2  SUCCESS  2  4  26.74  0.1  26.84
27  SUCCESS  2  3  26.93  0.1  27.03
21  SUCCESS  2  2  27.63  0.1  27.73
26  SUCCESS  2  2  28.77  0.1  28.87
7  SUCCESS  2  3  28.88  0.1  28.98
12  SUCCESS  2  4  31.31  0.1  31.41
5  SUCCESS  2  4  31.99  0.1  32.09
22  SUCCESS  2  3  32.54  0.1  32.64
14  SUCCESS  2  3  33.06  0.1  33.16
4  SUCCESS  2  3  33.21  0.1  33.31
17  SUCCESS  2  3  33.32  0.1  33.42
33  SUCCESS  2  4  33.69  0.1  33.79
20  SUCCESS  2  4  35.06  0.1  35.16
9  SUCCESS  2  4  35.33  0.1  35.43
25  SUCCESS  2  3  35.85  0.1  35.95
19  SUCCESS  2  3  37.1  0.1  37.2
Maximum Makespan : 37.38
Minimum Makespan : 14.12
Average Execution Time : 27.408222222222222
Throughput : 0.3024823453884812
Average Resource Utilization Ratio : 0.588526287558466
Resource Utilization : 22.14708609743019
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
    
```

Fig. 10. 30 cloudlets at 5 VMs

```

===== OUTPUT =====
Cloudlet ID  STATUS  Data center ID  VM ID  Time  Start Time  Finish Time
31  SUCCESS  2  2  28.48  0.1  28.58
22  SUCCESS  2  4  29.34  0.1  29.44
11  SUCCESS  2  4  30.51  0.1  30.61
13  SUCCESS  2  4  32.37  0.1  32.47
0  SUCCESS  2  4  33.04  0.1  33.14
25  SUCCESS  2  2  35.95  0.1  36.05
9  SUCCESS  2  3  35.92  0.1  36.02
4  SUCCESS  2  2  36.16  0.1  36.26
7  SUCCESS  2  3  38.41  0.1  38.51
12  SUCCESS  2  3  44.04  0.1  44.14
21  SUCCESS  2  3  44.15  0.1  44.25
15  SUCCESS  2  2  44.3  0.1  44.4
38  SUCCESS  2  4  44.41  0.1  44.51
30  SUCCESS  2  2  46.34  0.1  46.44
20  SUCCESS  2  2  47.02  0.1  47.12
14  SUCCESS  2  3  47.73  0.1  47.83
24  SUCCESS  2  2  48.40  0.1  48.50
5  SUCCESS  2  3  48.07  0.1  48.17
29  SUCCESS  2  3  48.70  0.1  48.80
18  SUCCESS  2  2  49.04  0.1  49.14
33  SUCCESS  2  2  50.29  0.1  50.39
34  SUCCESS  2  3  53.73  0.1  53.83
17  SUCCESS  2  3  54.02  0.1  54.12
1  SUCCESS  2  3  55.92  0.1  56.02
32  SUCCESS  2  3  56.73  0.1  56.83
2  SUCCESS  2  3  57.73  0.1  57.83
35  SUCCESS  2  3  58.01  0.1  58.11
0  SUCCESS  2  3  60.2  0.1  60.3
Maximum Makespan : 60.3
Minimum Makespan : 11.97
Average Execution Time : 38.505641666666666
Throughput : 0.634040920124427
Average Resource Utilization Ratio : 0.4800644606671072
Resource Utilization : 25.6111034320346
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
    
```

Fig. 11. 40 cloudlets at 5 VMs

```

===== OUTPUT =====
Cloudlet ID  STATUS  Data center ID  VM ID  Time  Start Time  Finish Time
14  SUCCESS  2  4  49.28  0.1  49.38
0  SUCCESS  2  4  50.62  0.1  50.72
5  SUCCESS  2  4  52.72  0.1  52.82
9  SUCCESS  2  4  53.25  0.1  53.35
7  SUCCESS  2  3  53.36  0.1  53.46
35  SUCCESS  2  3  53.47  0.1  53.57
42  SUCCESS  2  4  53.71  0.1  53.81
21  SUCCESS  2  4  54.2  0.1  54.3
29  SUCCESS  2  2  54.31  0.1  54.41
17  SUCCESS  2  4  54.31  0.1  54.41
45  SUCCESS  2  3  54.42  0.1  54.52
17  SUCCESS  2  4  54.72  0.1  54.82
4  SUCCESS  2  3  55.85  0.1  55.95
40  SUCCESS  2  4  56.72  0.1  56.82
40  SUCCESS  2  4  57.29  0.1  57.39
33  SUCCESS  2  2  59.4  0.1  59.5
46  SUCCESS  2  2  60.82  0.1  60.92
34  SUCCESS  2  2  61.14  0.1  61.24
31  SUCCESS  2  2  65.77  0.1  65.87
3  SUCCESS  2  2  71.08  0.1  71.18
43  SUCCESS  2  2  77.56  0.1  77.66
22  SUCCESS  2  2  77.77  0.1  77.87
19  SUCCESS  2  2  81.53  0.1  81.63
39  SUCCESS  2  2  83.49  0.1  83.59
28  SUCCESS  2  2  83.66  0.1  83.76
23  SUCCESS  2  2  84.66  0.1  84.76
8  SUCCESS  2  2  84.77  0.1  84.87
Maximum Makespan : 84.87
Minimum Makespan : 11.54
Average Execution Time : 31.307120666666667
Throughput : 0.5891455818089903
Average Resource Utilization Ratio : 0.4946552712810023
Resource Utilization : 30.321026748780467
Load Balancing Using Dynamic Harmony Inspired Simulated Annealing Completed!
    
```

Fig. 12. 50 cloudlets at 5 VMs

Based on the description provided, Figures 8 to 12 represents the outcomes for Case 2 along 5 virtual machines including up to 50 cloudlets, same to Case 1. Here's a summary of the information presented in each figure:

Figures 8 to 12: Case 2 along 5 Virtual Machines and Up to 50 Cloudlets

- Depict the results for the scenario involving five virtual machines and varying numbers of cloudlets, similar to Case 1.
- Each figure displays the cloudlet IDs along with their states and finish times, sorted based on minimal conclude count including maximal makespan in seconds.
- 5 free trials were managed for each case, along only the smallest maximal makespan detecting reported.

- The performance indicators included in the figures are maximum resource utilization, average resource utilization ratio (ARUR), minimum makespan, and throughput.
- The outcomes demonstrate the effectiveness of the introduced Dynamic Enhanced HISA load balancing method in optimizing resource utilization and minimizing makespan across different workload scenarios along 5 virtual machines and up to 50 cloudlets.

These figures provide detailed insights into the performance of the Dynamic Enhanced HISA load balancing method under varying workload scenarios, highlighting key performance indicators and showcasing the algorithm's effectiveness in improving resource allocation and minimizing execution time.

Table 2. Trials run for the suggested Dynamic Enhanced HISA-LB Method

Cases	Cloudlets size	Maximal Makespan	Minimal Makespan	Average Execution Count	Throughput	RU	ARUR
Case-1	10	28.87	6.8	20.16	0.35	7.02	0.77
	20	50.28	18.55	37.09	0.40	14.79	0.93
	30	79.4	18.63	54.82	0.38	20.75	0.83

	40	108.86	19.63	74.31	0.37	27.34	0.87
	50	155.1	29.43	81.66	0.32	26.36	0.74
Case-2	10	22.24	5.76	16.47	0.45	7.45	0.56
	20	27.04	5.59	16.31	0.74	12.14	0.54
	30	37.38	12.12	27.49	0.80	22.15	0.59
	40	60.3	11.97	38.51	0.66	25.61	0.48
	50	84.87	11.54	51.37	0.59	30.32	0.49

This table provides a comparative analysis of the test outcomes for both cases across various cloudlet sizes, highlighting metrics such as makespan, throughput, resource utilization, and ARUR. It demonstrates the performance of the Dynamic Enhanced HISA load balancing method in different scenarios with varying numbers of virtual machines and cloudlets. Adjust the table

format and content as needed to reflect the specific details and findings of your experiment.

It presents results for metrics such as maximum and minimum makespan, average execution count, performance, average resource use, and assets fulfillments, demonstrating the effectiveness of the proposed approach.

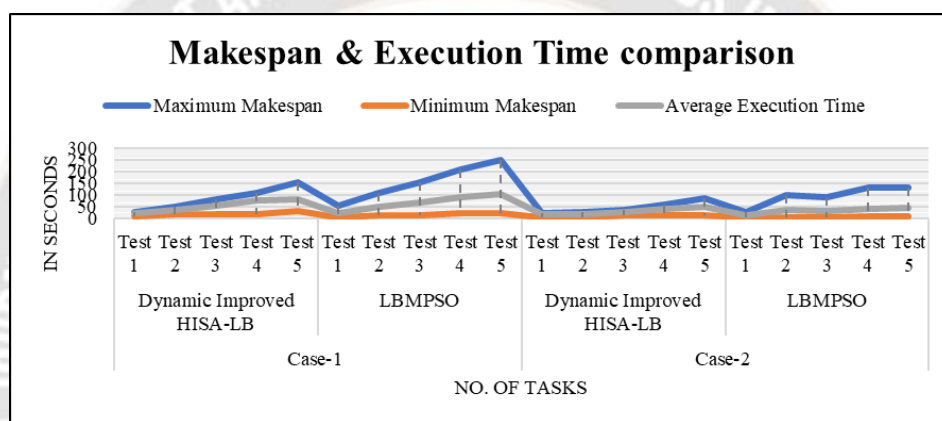


Fig. 13. Comparison line graph of Makespan and execution count obtained for both cases

The comparison focuses on minimal and maximal makespan codes uniformed in seconds. In Case 1, it is observed that the maximal makespan decreases as the count of tasks expands when using the proposed method, indicating improved efficiency compared to the existing approach. However, the difference in maximum makespan is less significant in Case 2.

Similarly, the comparison reveals that the minimum makespan is minimized using the proposed approach, particularly evident in Case 1. Notably, the makespan is considerably higher at trial 5 along 50 cloudlets still smallest at trial 1 along 10 cloudlets in both cases, indicating variations in performance across different task loads.

Additionally, the graph depicts variations in average execution time. Initially, the proposed LB approach minimizes average execution time. However, as the quantity of tasks expand (or tests), the mean execution time tends to increase, particularly evident in test 5 in both cases. This suggests potential scalability challenges with increasing task loads when using the proposed approach.

Overall, Figure 13 provides insights into the performance comparison between the suggested dynamic enhanced HISA-LB method including the active LBMPSO method, highlighting trends in makespan and execution time across different task loads and virtual machine configurations in both cases.



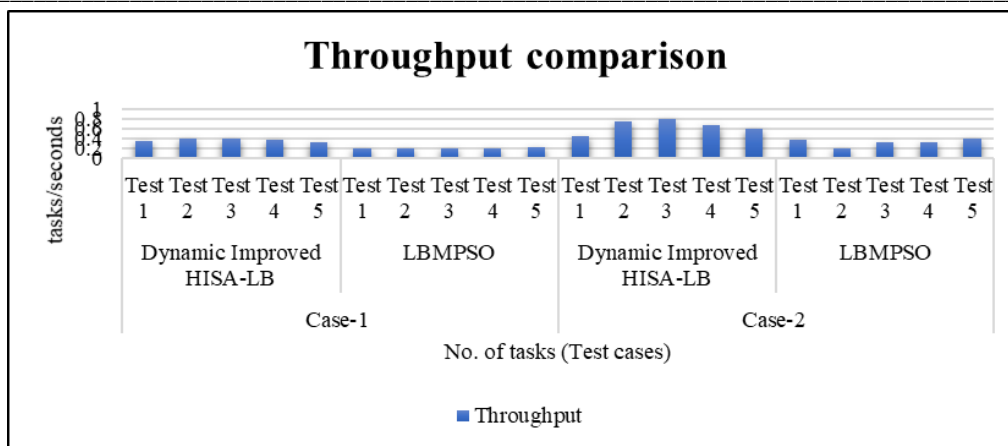


Fig. 14. Comparison bar graph of throughput obtained for both cases

Figure 14 provides a clear comparison of throughput between the suggested Dynamic enhanced HISA-LB and active LBMPSO methods across different test scenarios. It underscores

the enhanced throughput capabilities of the proposed approach, particularly evident in both cases compared to the existing approach.

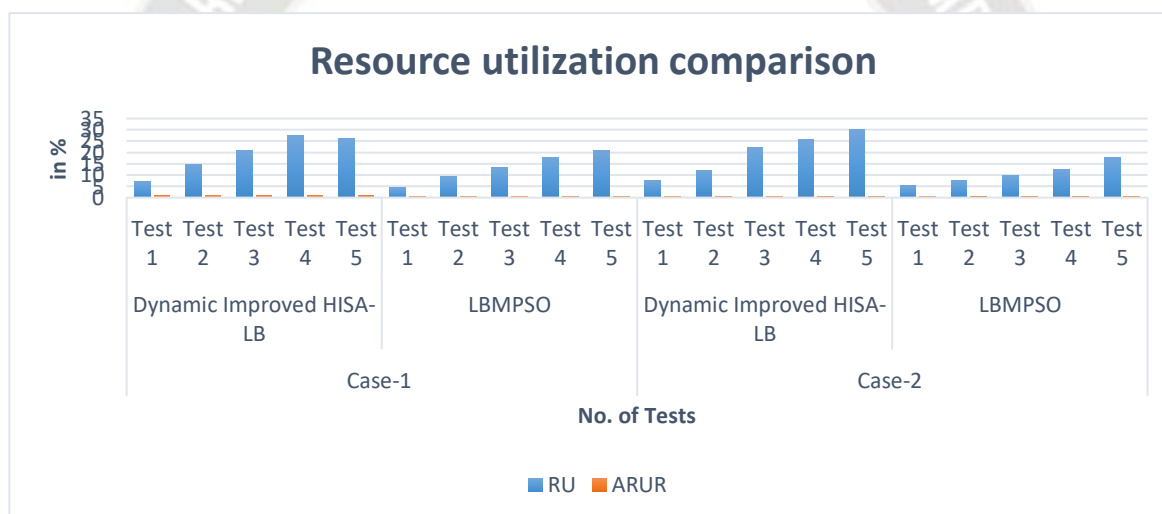


Fig. 15. Comparison bar graph of assets fulfillment and ARUR obtained for both cases

Figure 15 provides valuable insights into the resource utilization performance of the suggested Dynamic enhanced HISA-LB method compared to the active LBMPSO method. It

highlights the effectiveness of the proposed approach in efficiently utilizing resources, particularly under varying task loads and virtual machine configurations in both cases.

**CONCLUSION**

The present research introduces a novel load balancing methodology termed dynamic enhanced HISA-LB, built upon enhanced Harmony-Inspired Algorithm (HIA) and Simulated Annealing (SA) optimization strategy within a dynamic including nature-inspired cloud environment. Modifications to parameters such as HMCR, PAR, and fret width improve the improvisation of the Harmony Memory phase in the harmony-inspired algorithm. Extensive testing by the CloudSim simulator as varying fulfillment metrics has been managed.

The methodology findings demonstrate that the dynamic enhanced HISA-LB strategy effectively reduces both maximum and minimum makespan times, resulting in reduced average execution count similar to the LBMPSO method. Moreover, the proposed technique achieves higher resource utilization and throughput compared to the LBMPSO approach in both cases. This dynamic task allocation at 3 or 5 virtual machines contributes to balanced load distribution.

Furthermore, the algorithm shows potential for adaptation to supercomputers in the future. Online scheduling with preemption could further enhance utilization and revenue generation. Additionally, incorporating variable pricing structures corresponding to different working hours, such as









"peak time," could enhance the proposed model's versatility and effectiveness.

#### REFERENCES

- [1] B. Hayes, "Cloud Computing," *Commun. ACM*, vol. 51, no. 7, pp. 9–11, Jul. 2008, doi: 10.1145/1364782.1364786.
- [2] G. Gan, T. Huang, and S. Gao, "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment," in *International Conference on Intelligent Computing and Integrated Systems*, pp. 60–63. doi: 10.1109/ICISS.2010.5655013 2010.
- [3] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, doi: <https://doi.org/10.1016/j.jksuci.2018.01.003>, 2020.
- [4] N. A. Joshi, "Technique for Balanced Load Balancing in Cloud Computing Environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 3, pp. 110–118, 2022, doi: 10.14569/IJACSA.2022.0130316.
- [5] T. Prem Jacob and K. Pradeep, "A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization," *Wirel. Pers. Commun.*, vol. 109, no. 1, pp. 315–331, 2019, doi: 10.1007/s11277-019-06566-w.
- [6] M. Agarwal and G. M. S. Srivastava, "A PSO Algorithm-Based Task Scheduling in Cloud Computing," in *Soft Computing: Theories and Applications*, 2019, pp. 295–301.
- [7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.
- [8] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *J. King Saud Univ. - Comput. Inf. Sci.*, 2020, doi: 10.1016/j.jksuci.2020.10.016.
- [9] N. Joshi, K. Kotecha, D. B. Choksi, and S. Pandya, "Implementation of Novel Load Balancing Technique in Cloud Computing Environmen," in *2018 International Conference on Computer Communication and Informatics, ICCCI 2018*, 2018, pp. 1–5. doi: 10.1109/ICCCI.2018.8441212.
- [10] M. Haris and S. Zubair, "Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing," *J. King Saud Univ. - Comput. Inf. Sci.*, 2022, doi: 10.1016/j.jksuci.2021.12.003.
- [11] S. G. Domanal, R. M. R. Guddeti, and R. Buyya, "A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 3–15, 2020, doi: 10.1109/TSC.2017.2679738.
- [12] M. Junaid *et al.*, "Modeling an Optimized Approach for Load Balancing in Cloud," *IEEE Access*, vol. 8, pp. 173208–173226, 2020, doi: 10.1109/ACCESS.2020.3024113.
- [13] 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>.
- [14] T. Alfakih, M. M. Hassan, and M. Al-Razgan, "Multi-Objective Accelerated Particle Swarm Optimization With Dynamic Programing Technique for Resource Allocation in Mobile Edge Computing," *IEEE Access*, vol. 9, pp. 167503–167520, 2021, doi: 10.1109/ACCESS.2021.3134941.
- [15] A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, "Intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in Cloud Environment," *IEEE Access*, vol. 10, pp. 76939–76952, 2022, doi: 10.1109/ACCESS.2022.3192628.
- [16] W. Sun and X. Chang, "An Improved Harmony Search Algorithm for Power Distribution Network Planning," *J. Electr. Comput. Eng.*, vol. 2015, p. 753712, 2015, doi: 10.1155/2015/753712.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] N. A. Joshi, "Performance-Centric Cloud-Based e-Learning," *IUP J. Inf. Technol.*, vol. 10, no. 2, pp. 7–17, 2014.
- [21] 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>.
- [22] 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.
- [23] 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>.
- [24] 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.
- [25] 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.

#### BIOGRAPHIES OF AUTHORS

	<p><b>Shruti Tiwari</b>    is Ph.D. student in computer science engineering from RKDF Institute of Science &amp; Technology Sarvapalli Radhakrishnan University, Bhopal M.P. India. Her research includes area of cloud computing. She can be contacted at email: <a href="mailto:shruti.tiwari08@gmail.com">shruti.tiwari08@gmail.com</a></p>
	<p><b>Dr. Chinmay Bhatt</b>    received Ph.D. degree in computer science engineering from RKDF Institute of Science &amp; Technology Sarvapalli Radhakrishnan University, Bhopal M.P. India. He is currently a Associate Professor with the Dept. of Computer Science &amp; Engineering RKDF Institute of Science &amp; Technology Sarvapalli Radhakrishnan University, Bhopal M.P. India. He has supervised and co-supervised Ph.D. students. His research interests includes area of computer science engineering side. He can be contacted at email: <a href="mailto:chinmay20june@gmail.com">chinmay20june@gmail.com</a></p>