IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623

ADAPTIVE, FEEDBACK-DRIVEN TASK SCHEDULING AND LOAD BALANCING: A SUPERIOR APPROACH FOR MODERN CLOUD COMPUTING SYSTEMS

Deepak Joshi[1], Kumar Bibhuti Bhushan Singh[2]

Student, Dept. of CSE, Goel Institute of Technology & Management, Lucknow, India[1]

Assistant Professor, Dept. of CSE, Goel Institute of Technology & Management, Lucknow, India[2]

Abstract: Cloud computing has emerged as the foundation of modern IT services, providing scalable, on-demand resources via virtualization. Effective resource management in these settings needs not only skilled load balancing but also smart job scheduling algorithms to guarantee fairness, reduce delays, and improve utilization.

Static algorithms that are commonly used for the process of task scheduling and load balancing are simple and easy to implement, but they cannot adapt to any kind of changes in conditions. As a result, there is an inefficient use of valuable resources, increased response time, and reduced throughput in a large-scale cloud setting. This research aims to compare and test traditional algorithms against bio-inspired algorithms. Cloud Sim is used to simulate an extensive cloud environment, including five VMs and fifty cloudlets. Four algorithms, such as Round Robin, Weighted Round Robin, Honeybee Foraging, and Ant Colony Optimization, were implemented under similar conditions. The performance was measured by calculating the average response time, throughput, and utilization.

It was found that bio-inspired algorithms greatly surpassed traditional algorithms. Ant Colony Optimization was able to achieve the minimum response time, maximum throughput, and optimal utilization. Honeybee algorithm displayed remarkable adaptation ability, whereas classic algorithms appeared to be insufficient for changing circumstances. Therefore, adaptive algorithms are superior to static algorithms in contemporary cloud computing settings.

Keywords: Cloud Computing, Static Algorithm, Dynamic Algorithm, Load Balancing, Task Scheduling, Virtual Machine.

I. INTRODUCTION

Cloud computing refers to offering services on demand. One does not have to buy costly machines or construct vast data centers since the cloud computing provider takes care of these aspects. Some companies that provide cloud services include Amazon Web Services, Microsoft, and Google [1]. Cloud computing enables firms to increase their activities as per their requirements. Cloud computing is common in education, healthcare, financial institutions, entertainment, and many other industries [2]. For instance, virtual classrooms, video streaming software, and online payment services require cloud computing extensively [3].

II. CLOUD COMPUTING

Cloud computing is a pay-per-use service that gives you access to software, hardware, and infrastructure [4]. It lets users get scalable resources like power as needed, so they don't have to be watched over by people. Cloud computing utilizes globally distributed data centers to bypass numerous financial and physical obstacles to IT services [5]. It has quickly emerged as a catalyst for innovation in multiple sectors by guaranteeing adherence to Service Level Agreements (SLA) and upholding Quality of Service (QoS) [6].

III. LOAD BALANCING

Load balancing within cloud infrastructures involves evenly allocating tasks among multiple computing resources, such as servers, VMs, or hosts—ensuring efficiency, reliability, and balanced utilization [7]. It is a very important method that prevents any single resource from being overloaded and surrounded with too many requests while others remain

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623

underutilized or used much less than expected [8]. By smartly managing the allocation or distributing of tasks, load balancing improves system performance, reduces response times, and improves throughput, making sure that users experience consistent and reliable services [9]. Basically, load balancing acts as the "traffic controller" of cloud environments, directing workloads to the good, efficient resources, depending on availability and capacity to handle load [10]. The functioning of load balancing comprises supervising incoming requests and dynamically changing tasks, assigning them to resources [11].

IV. OBJECTIVES OF LOAD BALANCING

The most important goal of load balancing in cloud computing is to make sure that workloads are distributed efficiently across available resources, to maximize performance and minimize delays in service.

V. MOTIVATION

With the increasing growth of data, applications, and services hosted in the cloud, efficient resource management has become a critical challenge. Many traditional static algorithms are not flexible enough to accommodate changes in demand fluctuations, resulting in inefficiencies, delayed responses, and misuse of both VMs and their underlying physical host machines. This constraint makes it essential to have flexibility in order to achieve enhanced scheduling and load balancing techniques that can dynamically adapt to changing workload profiles.

VI. EXISTING LOAD BALANCING AND SCHEDULING ALGORITHMS

Load balancing in cloud computing has been widely explored, with numerous algorithms developed to tackle the challenges of efficiently distributing workloads across available resources [12]. These algorithms can be broadly categorized into static and dynamic methods; each method has its own strengths and limitations [13]. There several algorithms which are designed for different purposes some examples are Round Robin algorithm, Weighted Round Robin Algorithm, Least Connections Algorithm, Shortest Response Time Algorithm, Throttled Load Balancing, Honeybee Foraging, Ant Colony Optimization, Genetic Algorithms and Particle Swarm Optimization [14].

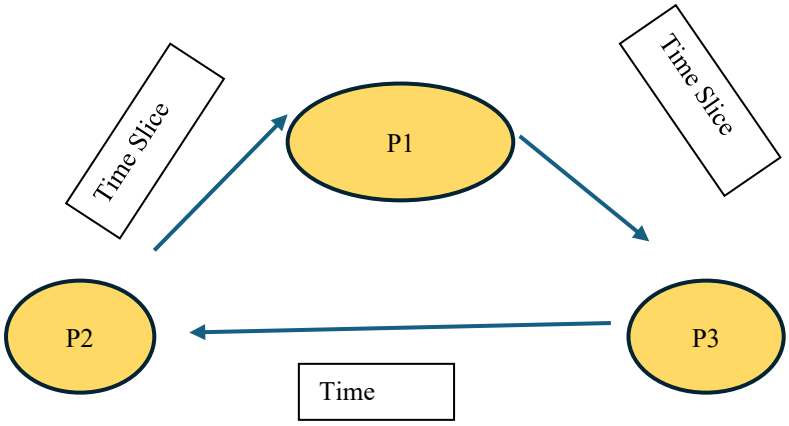


Figure 1: Round Robin Algorithm (Static Algorithm)

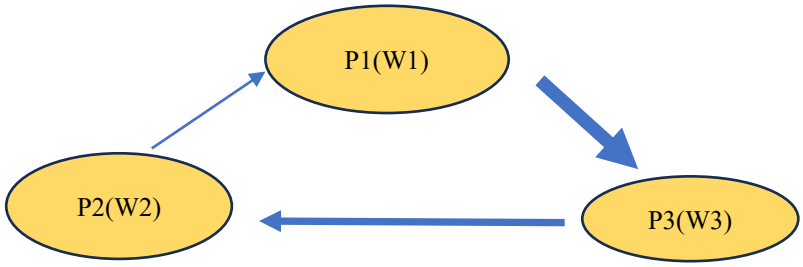


Figure 2: Weighted Round Robin Algorithm (Static Algorithm)

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623

VII. STATIC ALGORITHMS

Static algorithms operate on predefined rules and do not change according to real time changes in workload or resource conditions [15]. One of the simplest static methods is the Round Robin algorithm, which assigns tasks sequentially, one after the other to available servers in a circular order [16]. This makes sure of fairness but does not take care of current load or capacity of each server. A different version of this is Weighted Round Robin algorithm, where servers are assigned, weights based on their capacity, ability, and tasks are distributed accordingly in the same way [17]. Another static approach is the Randomized algorithm, which assigns tasks randomly to servers, reducing the chance of bottlenecks but missing to predict a possible future event and its ability [18]. While static algorithms are easy to use and require very little overhead, they are less effective in dynamic changing environments where workloads go up and down significantly [19].

VIII. DYNAMIC ALGORITHMS

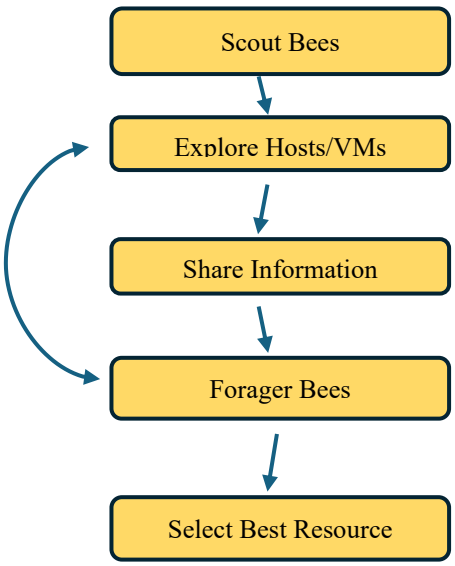


Figure 3: Honey Bee Foraging algorithm (Bio Inspired Algorithm)

Dynamic algorithms continuously monitor system performance and change to make better fit for new conditions taking care job are distributed according to current situation [20]. These are hard to implement but provide better ability to change and efficiency. One widely used dynamic, changing method is Least Connection algorithm, which move tasks to the server with have fewest active connections, making sure of balanced resources usage [21]. In almost the same way, the Shortest Response Time algorithm assigns tasks to servers that respond fastest, improving user experience [22]. Another dynamic, changing approach is the Throttled algorithm, controlled the speed of a request given, where each server is assigned a limit, and tasks are given out based on availability within those limits [23]. Dynamic algorithms also include bio inspired ways of doing operation, which copy natural processes to accomplish smart load distribution [24]. The Honey Bee Foraging algorithm models the behaviour of bees searching for food, where tasks are given out based on profit values, directing workloads toward servers that demonstrate better performance [25].

IX. METHODOLOGY

The experiment begins with the initialization of the CloudSim environment, where a datacenter is created to represent the physical infrastructure, basic conditions for experiments. This datacenter consists of a host prepared with multiple processing elements (PEs), RAM, bandwidth, and valuable storage resources. On top of this infrastructure, five heterogeneous virtual machines (VMs) of different capacities are deployed, each configured with different MIPS values to test servers of different capacities. The workload is modeled using fifty cloudlets, representing user tasks with irregular job sizes. After execution, the simulation, it collects key performance metrics which measures like average response time, throughput, and utilization. As part of our CloudSim experiments, we conducted task scheduling and load balancing evaluations at two distinct levels, namely the VM level and the Host level, to capture performance variations across both layers.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623

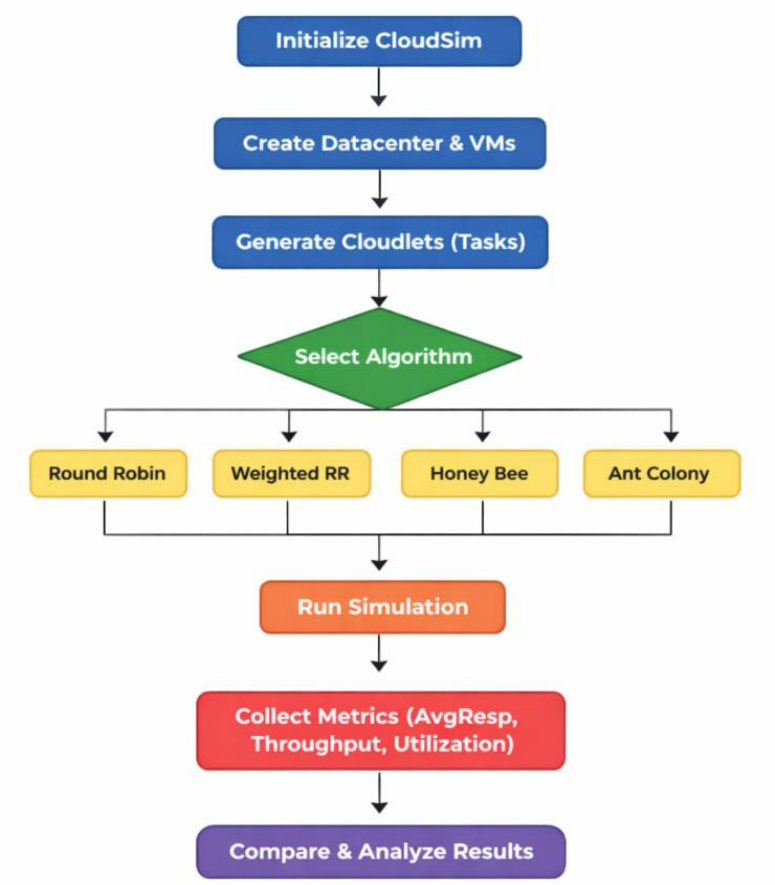


Figure 4: Execution Flow Block Diagram

X. PSEUDOCODE

```
BEGIN
Initialize CloudSim environment

  Create Datacenter with:
    - Host (CPU, RAM, Bandwidth, Storage)
    - Multiple Virtual Machines (VMs) of different capacities

  Create Broker to manage tasks and VMs

  Generate Cloudlets (tasks) with mixed workloads:
    - Heavy jobs
    - Light jobs

  FOR each algorithm in [Round Robin, Weighted RR, Honey Bee, Ant Colony] DO
     Assign cloudlets to VMs using algorithm rules
     Run simulation
     Collect performance metrics:
        - Average Response Time
        - Throughput
        - Utilization
     Store results
  END FOR
Print a comparison table of all algorithms
END
```


IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623

XI. RESULT AND DISCUSSION

We systematically analyze the data obtained from both scenarios and interpret the results to identify key trends, variations, and performance improvements.

Algorithm	Average Response Time(seconds)	Throughput (cloudlets/sec)	VM Utilization Ratio (0–1)
Round Robin Algorithm	141.7754	0.0800	0.1046
Weighted Round Robin Algorithm	57.3581	0.1648	0.2044
Honey Bee Foraging Algorithm (bio-inspired)	43.1352	0.2189	0.2655
Ant Colony Optimization Algorithm (Bio-Inspired)	36.7168	0.3822	0.4057

Table 1: Performance Metrics (Simulation Case 1)

Algorithm	Average Response (seconds)	Throughput(cloudlets/second)	VM Utilization Ratio (0-1)	Host Utilization Ratio (0-1)
Round Robin Algorithm	151.1064	0.0778	0.1121	0.0249
Weighted Round Robin Algorithm	52.0253	0.1723	0.2060	0.0458
Honey Bee Foraging Algorithm	44.4310	0.2671	0.2952	0.0656
Ant Colony Optimization Algorithm	19.5147	0.2454	0.3535	0.0786

Table 2: Performance Metrics (Simulation Case 2)

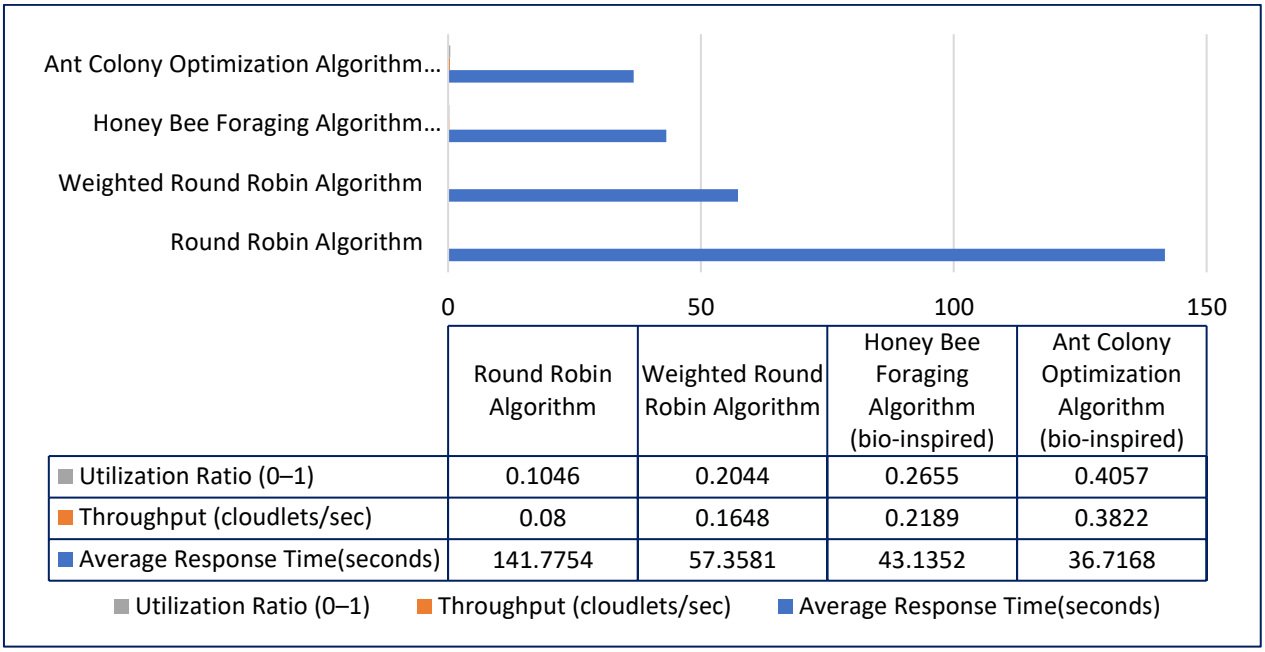
When we examine the performance metrics table for VM level load balancing and scheduling, the comparative and main differences between the four algorithms become very clear. Round Robin scheduling, which is the simplest and most traditional approach, shows the weakest performance with an average response time of 141.77 seconds, throughput of only 0.08 cloudlets per second, and a utilization ratio of resources 0.1046. These numbers highlight that while Round Robin makes sure of fairness by distributing tasks sequentially, one after the other across VMs, it fails to properly use the mixed-up nature of VM capacities. Heavy tasks are often assigned to weaker VMs, which increases waiting times and lowers overall system productivity. In contrast, Weighted Round Robin improves very much by incorporating static weights based on VM capacity. Its average response time drops to 57.35 seconds, throughput doubles to 0.1648 cloudlets per second, and utilization rises to 0.2044. This demonstrates that thinking about VM strength in scheduling decisions leads to better efficiency, though the static nature of weights means the algorithm cannot adapt dynamically, unable to change quickly as needed.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

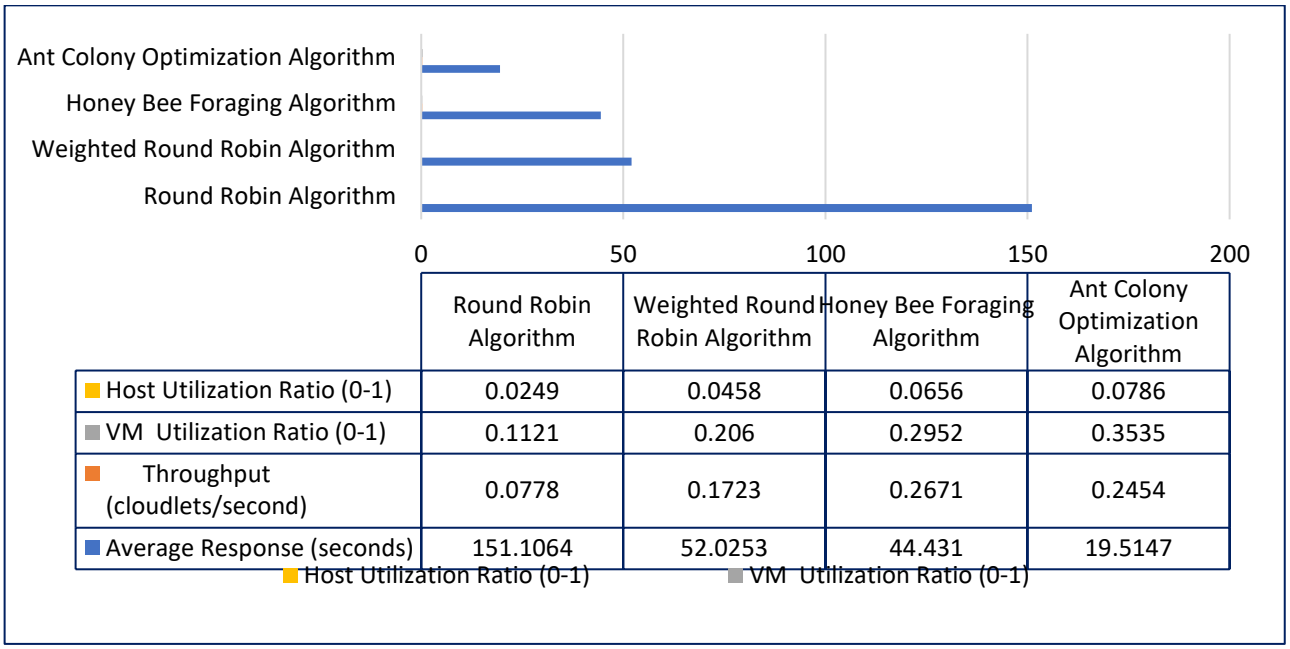
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15623



	Round Robin Algorithm	Weighted Round Robin Algorithm	Honey Bee Foraging Algorithm (bio-inspired)	Ant Colony Optimization Algorithm (bio-inspired)
Utilization Ratio (0–1)	0.1046	0.2044	0.2655	0.4057
Throughput (cloudlets/sec)	0.08	0.1648	0.2189	0.3822
Average Response Time(seconds)	141.7754	57.3581	43.1352	36.7168

Figure 5: Performance of Algorithms (Simulation Case 1)



	Round Robin Algorithm	Weighted Round Robin Algorithm	Honey Bee Foraging Algorithm	Ant Colony Optimization Algorithm
Host Utilization Ratio (0-1)	0.0249	0.0458	0.0656	0.0786
VM Utilization Ratio (0-1)	0.1121	0.206	0.2952	0.3535
Throughput (cloudlets/second)	0.0778	0.1723	0.2671	0.2454
Average Response (seconds)	151.1064	52.0253	44.431	19.5147

Figure 6: Performance of Algorithms (Simulation Case 2)

The bio inspired algorithms show even stronger results. The Honey Bee Foraging algorithm reduces the average response time further to 43.13 seconds, increases throughput to 0.2189 cloudlets per second, and accomplishes a utilization ratio of 0.2655. The progress is because of its adaptive, able to change and get better nature, where profit values are updated based on VM utilization, imitating the foraging behavior of bees. By continuously adjusting task allocation distribution, Honey Bee makes sure that workloads are directed to VMs that can handle them most efficiently, by that way improving responsiveness and valuable resource exploitation. The Ant Colony algorithm outperforms all others, accomplishing the lowest average response time of 36.71 seconds, the highest throughput of 0.3822 cloudlets per second, and the best utilization ratio of 0.4057. Its probabilistic scheduling methods, guided by pheromone trails with evaporation and reinforcement, allows it to balance exploration and exploitation effectively. This ability to change makes sure that workloads are distributed dynamically as needed, making the most of system efficiency and reducing delays.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15623

When examining the algorithms side by side, the progression is obvious: Round Robin provides fairness but suffers from inefficiency, Weighted Round Robin improves efficiency by considering VM capacity but remains static, Honey Bee introduces ability to change and accomplishes better balance, and Ant Colony Optimization combines ability to change with probabilistic decision making to deliver the best overall performance.

XII. CONCLUSION

The comparison of the simulation output from both the cases of simulation shows that the variation lies solely because of the adaptability feature. While the static algorithm acts as the basis, it is essentially limited when it comes to dealing with situations where the workloads are inconsistent and there is diversity in the nature of resources. The bio-inspired algorithms fit in well within such a scenario as they are specifically designed for such a scenario and continue to grow along with it. They do not follow any strict principles, but rather continue to adapt according to their real time conditions.

XII. FUTURE SCOPE

The development of hybrid algorithms is another field that should be considered for further investigation. Although bio-inspired algorithms exhibit great flexibility, their integration with prediction tools like machine learning algorithms would result in even greater efficiency.

REFERENCES

[1]. R. Buyya et al., Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, 2009.
[2]. P. Mell and T. Grance, NIST Special Publication 800-145, 2011.
[3]. M. Armbrust et al., Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[4]. L. Vaquero et al., ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2009.
[5]. E. Caron et al., in Proc. IEEE Cloud Computing, 2010, pp. 456–463.
[6]. J. Broberg et al., Journal of Grid Computing, vol. 6, no. 3, pp. 255–276, 2008.
[7]. M. Randles et al., in Proc. IEEE AINA Workshops, 2010, pp. 551–556.
A. Khiyaita et al., in Proc. IEEE Cloud Computing Technologies and Applications, 2012, pp. 106–111.
[8]. Y. Fang et al., Journal of Computers, vol. 6, no. 1, pp. 132–139, 2011.
A. Beloglazov and R. Buyya, Concurrency and Computation: Practice and Experience, vol. 24, no. 13, pp. 1397–1420, 2012.
[9]. J. Xu et al., in Proc. IEEE MASCOTS, 2007, pp. 327–334.
[10]. H. Liu et al., in Proc. IEEE ICPADS, 2011, pp. 9–16.
[11]. T. Kokilavani and D. Amalarethinam, International Journal of Computer Applications, vol. 37, no. 3, pp. 11–16, 2012.
[12]. S. Wang et al., Journal of Computers, vol. 8, no. 5, pp. 1315–1322, 2013.
[13]. M. Dorigo and T. Stützle, Ant Colony Optimization. MIT Press, 2004.
[14]. D. Karaboga and B. Basturk, Journal of Global Optimization, vol. 39, no. 3, pp. 459–471, 2007.
[15]. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
[16]. J. Kennedy and R. Eberhart, in Proc. IEEE Int. Conf. Neural Networks, vol. 4, 1995, pp. 1942–1948.
[17]. S. Chhabra and R. Singh, International Journal of Computer Applications, vol. 117, no. 18, pp. 5–8, 2015.
[18]. S. D. Kamal et al., International Journal of Cloud Computing and Services Science, vol. 2, no. 2, pp. 118–127, 2013.
[19]. R. N. Calheiros et al., "CloudSim: A toolkit for modeling and simulation of cloud computing environments," Software: Practice and Experience, vol. 41, no. 1, pp. 23–50, 2011.
[20]. K. Li et al., "Efficient resource management for cloud computing," Future Generation Computer Systems, vol. 29, no. 1, pp. 1–12, 2013.
[21]. H. Zhang and G. Guo, "Load balancing in cloud computing: A survey," Journal of Computer Science and Technology, vol. 27, no. 2, pp. 356–371, 2012.
[22]. N. S. Gill and S. Singh, "A comparative study of load balancing algorithms in cloud computing," International Journal of Computer Applications, vol. 84, no. 12, pp. 1–4, 2013.
[23]. R. Kaur and P. Kaur, "Bio-inspired load balancing techniques in cloud computing," International Journal of Computer Applications, vol. 975, no. 8887, pp. 12–18, 2015.

