International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

AN EFFICIENT MULTI OBJECTIVE TASK SCHEDULING TECHNIQUES IN CLOUD COMPUTING CONTEXT USING SWARM OPTIMIZATION ALGORITHM

Vigneshwaran Thangaraju
CGI Technology and Solutions
Aldie, Virginia, USA
vigsan714@gmail.com

Abstract

Task scheduling is a critical aspect in cloud computing as it manages a number of virtualized resources to provide efficient performance. Manual scheduling is infeasible because clients might require thousands of virtualized assets per task in a cloud environment. The organization of tasks aims to maximize resource utilization and distribute workloads while minimizing programming time and expense. In this research, we present a multi-objective optimization strategy using the Artificial Fish Swarm Algorithm (AFSA) for the use of cloud computing scheduling employment. The proposed methodology proposes AFSA parameters initialization and preying, swarming, following, and random behaviors to iteratively optimize task allocation. The evaluation is done with respect to the execution cost, completion of the job time, and load-balancing volatility. Additionally, the suggested method's performance in comparison is contrasted with that of the Particle's solution Swarm Optimization (PSO) technique. Experimental findings indicate that AFSA outperforms PSO in terms of execution costs, load balancing, as well as task completion time. In particular, AFSA achieves a load balancing value of 1.16 at a task size of 150, while PSO loses its load balancing value with the size of the task decreasing and reaches its lowest value of 0.51 at a size of 250. Additionally, AFSA always achieves optimal execution cost and task completion time as a function of task sizes. From these findings, we derive that AFSA is the outstanding optimization technique for Cloud-based scheduling assignments, and it guarantees high utilization of resources, balanced workload distribution, and improved computational performance against traditional approaches.

Keywords—Cloud computing, task scheduling, Artificial Fish Swarm Algorithm (AFSA), Particle Swarm Optimization (PSO), load balancing, execution cost, optimization algorithms.

I. INTRODUCTION

Clients utilize an approach to cloud computing that is "pay for each use" basis, accessing services without fully understanding hosting details and distribution regulations [1]. This reduces the amount of time needed for businesses to shop and ascertain the logical conclusions by offering worldwide access to a shared resource pool, such as servers for computers, file cabinets, and internet places of confinement, upon application [2]. Customers may gradually utilize these resources without worrying about it or having to get in touch using the establishment supplier [3][4]. The goal of cloud organization is to give dynamic applications a user-friendly workspace. Although a lot of study has been done, there are still issues with freight harmonizing in cloud-

related applications in the rapidly developing field of cloud computing. Static and dynamic mist environments are used to observe load-balancing techniques. In cloud computing, effective job scheduling is crucial, and building an algorithm requires a knowledge of load balancing from this angle. Recently, a lot of study has been done on how to schedule tasks in the internet of things [5].

In many situations, such as computing via the cloud, distributed systems, and parallel computing, task scheduling is essential to maximizing system performance and resource utilization. Traditional task scheduling algorithms face challenges in effectively balancing the workload and minimizing the execution time [6]. In order to overcome these obstacles, scientists have resorted to artificial intelligence. The main foundation of the Swarm Intelligence (SI) approach is the collective behaviors of biological and natural evolutionary phenomena, such as flocks of birds, schools of fish, worms, and bees and ants [7]. The capacity of SI-based algorithms to self-learn, quick convergence, flexibility, simple structure, insensitivity to starting parameters, and adaptation to external variations are the primary factors contributing to their current prominence in addressing optimization issues [8]. Through basic interaction principles, the swarm's self-organizing capacity often achieves the changing behavior towards optimality [9][10]. However, task scheduling issues frequently entail several competing optimization goals, such as lowering system energy consumption and prices, boosting task completion rates, and enhancing dependability, as cloud computing systems get more intricate and varied. Because they only concentrate on one goal and disregard the influence of other goals, traditional single-objective optimization algorithms frequently fail to solve these multi-objective optimization issues [11].

It is essential to research methods of multi-objective optimized performance for cloud computing work scheduling in order to overcome this difficulty. Multi-object optimization algorithms are capable of optimizing a set of all optimum solutions and several objective functions at the same time, given the tradeoff between different objectives[12]. With Cloud computing systems can operate more sustainably, dependably, and effectively with the help of these algorithms. Furthermore, they may be utilized for thorough system performance analysis and assessment, as well as providing a thorough guide on how to maximize cloud computing systems' overall performance[13]. The main objective of this research is to investigate and assess how characteristics and load balancing affect the effectiveness of the algorithm in relation to the environment and task scheduling context, using the multi-objective swarm optimization algorithm to determine task parameters like execution cost and completion time.

A. Motivation and Contributions

Task Scheduling is the key to efficiently utilizing resources and scalability in the face of the complexity of cloud computing environments. Generally, workload distribution in traditional schedulers is difficult, causing the accomplishment cost, task efficiency issues with time frames for completion and balance of loads. The AFSA has been demonstrated to be superior to bio inspired optimization algorithms in the dynamic environments in terms of near optimal scheduling solutions. The objective of this study is to design a scalable and efficient task scheduling framework for the enhancement of the cloud performance and minimization of cost of operations.

- In contrast to more traditional techniques like Particle Swarm Optimization (PSO), an optimization model is created using AFSA to maximize job scheduling efficiency.

- Combining important performance indicators, such as load balancing, job completion time, and execution cost, to guarantee optimal resource allocation.
- Implementation of an adaptive scheduling algorithm that dynamically responds to workload variations, improving scalability and efficiency.
- Extensive evaluation of the proposed AFSA-based approach across different task sizes (50–250) to validate its effectiveness.
- Comparative analysis demonstrating AFSA's superiority over PSO by achieving maximum load balancing, thereby improving workload distribution and computational efficiency.

B. Structure of paper

The research paper is organized as follows: In the section II, bio-inspired optimization for cloud work scheduling is reviewed. Section III describes the suggested framework based on AFSA. In Section IV, the comparison analysis is presented. The study is concluded in Section V, which also suggests further paths of inquiry.

II. LITERATURE REVIEW

The task-scheduling rules that are implemented One essential element of the cloud computing system is a safe cloud, which can additionally be used for a cloud environment. The relevant literature on task-scheduling issues is displayed in Table I. This research demonstrates the efficacy of optimizing task arrangement within cloud computing environments, significantly enhancing load balancing and execution cost reduction.

Archana and Kumar (2023) considered the 100 to 1000 task iteration with a task size of 50. From the simulation results, it is stated that the values of execution time, fitness, the mean and standard deviation of the SMO method are 6 ms, 0.0197, 0.0236, and 0.0011, respectively. In contrast, the values for the PSO method are 57 ms, 0.5675, 0.0567, and 0.5108, respectively. SMO has been found to effectively impact the provisioning of resources by minimizing the execution time, optimizing the fitness value, and lowering the mean and standard deviation values [14].

Saemi et al. (2023) mentioned problem in MCC is addressed by Hybrid Multi-objective Harris Hawks Optimization (HMHHO), a non-dominated, Harris Hawks Optimization (HHO) based multiple-purpose approach. Allocating workloads Public cloud processors along with cloud patches enable information processing from mobile source nodes mobile resource processors were the goals of this study. The suggested approach often completes tasks more quickly and consumes less energy than the supplementary four procedures: the Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Cuckoo Search Algorithm (CSA)[15].

Mishra and Gupta (2023) examined scheduling heuristic techniques based on make span, throughput, and ARUR, including RALBA, DLBA, DRALBA, Min-min, Max-min, and Smoothed Robin, utilizing workflows on artificial workloads and Google's GoCJ datasets, which are genuine workloads. Using both synthetic and GoCJ data, this study shows that the current DRALBA technique performs superior than the earlier methods in terms of performance characteristics [16].

Sharma and Pandey (2022) the key execution parameters, such as It is possible to optimize availability, makes pan time, resource utilization, energy usage, reliability, response time, etc., and resource imbalance can be prevented. A number of algorithms, including hybrid, meta-heuristic, and heuristic, are proposed to assist with the above indicated scheduling. The suggested VTO-QABC is put into practice and contrasted with other approaches on the parameter throughput. When associated to Max-Min (84.51%), MOPSO (37.82%), HABC_LJF (19.85%), Q-Learning (7.72%), VTO-QABC_FCFS (3.89%), and VTOABC_LJF (3.89%) less time than MOCS, a notable improvement is shown[17].

Mahmoud et al. (2022) presented a new method for allocating and completing an application's task called job scheduling-decision trees (TS-DT). The effectiveness of the proposed TS-DT algorithm was evaluated by comparing it with the existing algorithms, namely Heterogeneous Earliest Finish Time (HEFT), The methodology for Order of preference by Similarities to Ideal Solution incorporating the Experimental Weight Method (TOPSIS-EWM), and combining Q-Learning with the Heterogeneously Earliest Finish Time (QL-HEFT). The suggested TS-DT algorithm performs better than the current HEFT, TOPSIS-EWM, and QL-HEFT algorithms by, on average, decreasing make span by 5.21%, 2.54%, and 3.32%, increasing reserve utilization by 4.69%, 6.81%, and 8.27%, and enlightening load complementary by 33.36%, 19.69%, and 59.06%[18].

Devi and Winster (2022) combines Using a blockchain-based key aggregation encryption system in conjunction using attribute-based encryption (ABE) technology to enhance job scheduling and guarantee user data security and privacy. The study contrasts the performance of the ABE-BKAC model and the suggested meta-heuristic with that of other methods, including BCP-ABE-PHAS, CEVP, H3CSA, PPSO, and EDS. Superior performance is demonstrated by the experimental results in terms of reaction time, manufacture duration, energy consumption, key generation time, encrypting time, and time required for decryption, and completion ratio [19].

Faragardi et al. (2020) changed to take into consideration a spending limit, the HEFT algorithm. GRP-HEFT is compared to innovative production methods for scheduling like GA (Genetic Algorithm), PSO (Particle Swarm Optimization), and MOACS (Multi Objective Ant Colony System). In a number of well-known scientific workflow applications on both issue sizes, the trials' results indicate that GRP-HEFT outperforms GA by an average of 13.64 percent, PSO by 19.77 percent, and MOACS by 11.69 percent. Additionally, in rapports of temporal complexity, GRP-HEFT performs better than GA, PSO, and MOACS [20].

The relevant work on job scheduling in cloud computing utilizing different optimization techniques is summarized in Table I

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

TABLE I. OVERVIEW OF RELATED WORK ON TASK SCHEDULING IN CLOUD COMPUTING

Author	Objectives	Methodology	Parameters	Features
Archana and Kumar (2023)	Optimise cloud computing job execution and resource provisioning.	SMO vs. PSO method for task execution	Execution time, fitness, mean, standard deviation	Compared to PSO, SMO optimises task execution and reduces execution time
Saemi et al. (2023)	Mobile Cloud Computing (MCC) multi-objective task scheduling	Harris Hawks Optimisation (HMHHO) against Hybrid Multi-objective vs. GA, ACO, PSO, and CRM	Execution time, energy consumption	HMHHO performs better in job completion and energy efficiency
Mishra and Gupta (2023)	Evaluate scheduling heuristics based on makespan, throughput, and ARUR	Comparison of Google and synthetic datasets using Round Robin, Min-min, Max-min, RALBA, DLBA, and DRALBA	ARUR, throughput, and makespan	DRALBA outperforms other scheduling approaches
Sharma and Pandey (2022)	Optimize resource utilization and execution time in cloud scheduling	Compare VTO-QABC with HABC_LJF, Q-Learning, Max-Min, MOPSO, VTO-QABC_FCFS, VTOABC_LJF, and MOCS	Throughput, execution time	VTO-QABC achieves higher throughput and lower execution time
Mahmoud et al. (2022)	Improve task allocation and load balancing	Decision In comparison to HEFT, TOPSIS-EWM, and QL-HEFT, Tree-Based Task Scheduling (TS-DT)	Makespan, use of resources and balance of load	TS-DT enhances load balancing and resource utilisation while decreasing makespan
Devi and Winster (2022)	Enhance task scheduling security using encryption	Meta-heuristic + ABE-BKAC vs. EDS, CEVP, H3CSA, PPSO, BCP-ABE-PHAS	Completion ratio, keygen time, makespan, reaction time, and energy usage	ABE-BKAC model improves security and scheduling efficiency
Faragardi et al. (2020)	Optimize workflow scheduling within budget constraints	GRP-HEFT vs. MOACS, PSO, GA	Execution time, budget, scientific workflows	GRP-HEFT improves execution efficiency and time complexity

III. METHODS AND MATERIALS

In this work, the AFSA is utilized to optimize task scheduling in relation to cloud facility access, minimizing execution costs, improving load balancing, and ultimately decreasing job completion time. The methodology starts by defining the problem and defining the parameters, e.g., swarm size, step length and visual distance, and set the initial AF positions to random positions. The execution cost, load balancing variance, and task completion time are used to compute the fitness

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

function. Preying, swarming, following, and random behaviors, used iteratively to optimize scheduling solutions are employed by AFSA. First, AF positions are updated in each iteration according to their respective behaviors, and then the fitness function is recomputed. As an alternative, a convergence check ensures termination if a predefined condition is met, such as a minimum level of fitness increase or a limit number of iterations. The fourth option is the task scheduling approach, which distributes jobs across virtual machines based on the best AF locations. Also, the effectiveness of AFSA is assessed by contrasting it with PSO. The comparison evaluates important performance indicators and shows that AFSA continuously performs better than PSO in terms of minimizing implementation costs, load balancing, and finishing the project time. In a cloud computing environment, Figure 1 demonstrates the systematic use of the Swarm Optimization Algorithm for job scheduling.

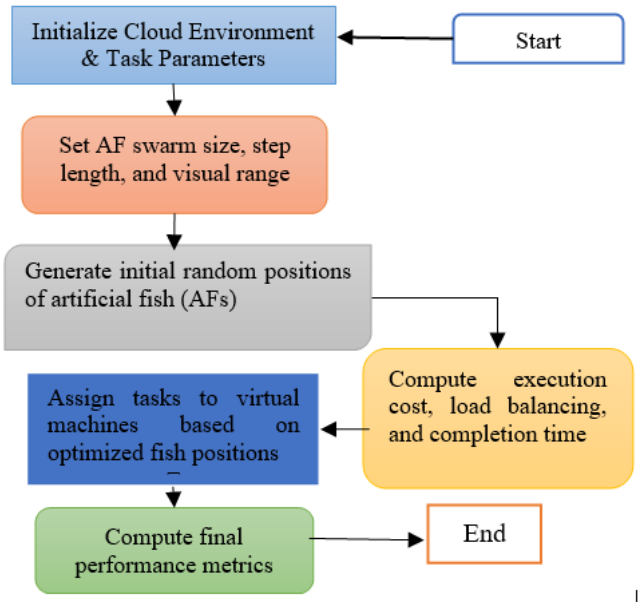

Fig. 1. Flowchart of the Proposed Task Scheduling Methodology in Cloud Computing

A. Proposed Artificial Fish Swarm Algorithm

A location with a high concentration of fish is often nutrient-dense in nature. By engaging in sophisticated behaviors like preying, swarming, following, etc., fish may identify the most nutrient-dense location. The AFSA is an artificially intelligent system that mimics the behavioral patterns of a population of fish. By mimicking the collective movement of artificial fish (AF), our program can approach the global optimum. Good robustness, global search capability, parameter setting tolerance, and insensitivity to initial values are some of the appealing aspects of the AFSA.

Figure 2 is an illustration of the AF's vision idea. According to the graph, the visual distance is denoted by the word Visual and the step length by the term Step. X = $(x_1, x_2, x_3, \ldots, x_n)$ represents the spatial coordinate of the AF, where x_i is a possible solution. The objective function Y = f (X) represents The AF's dietary composition at the moment location. $D_{ij} = \|X_i - X_j\|$ is the expression for the stance between neighbouring AF people (ith and jth), and Δ is the crowd factor.

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

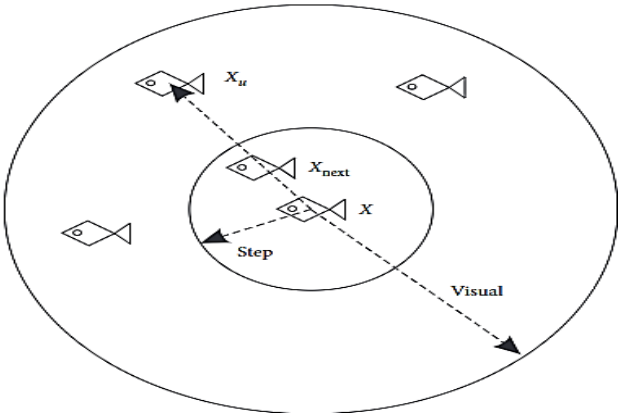

Fig. 2. Vision concept of the artificial fish

The behavior of fish depends on examining the surrounding area until a behavior requirement is satisfied. As a result, the AF advances to X_{next} if it is conditioned to do so; if not, it keeps examining within its field of vision.

The refreshed position can be described as Equation. (1 and 2)

$$x_v = x_i + Visual \times Randie(0, n) \quad (1)$$

$$X_{next} = \frac{X_v - X}{\|X_v - X\|} \times Step \times Rand \quad (2)$$

where X_v is a place inside the view; n is the number of variables, and Rand are randomly generated numbers between 0 and 1. Others are identical to the ones mentioned above.

Four classical activities are included in the AF structure prey, swarm, follower, however, and unpredictable movement. individuals.

1. Preying Behavior

Preying behavior is mostly thought of as a form of food treatment. As seen in the AF visual idea, it is an iterative method of transitioning to a more nutrient-dense location within the framework of an optimization algorithm.

Let the AF's present location is X_i, and a random place within the visible range is X_j.The situation, therefore, becomes an Equation. (3) when we use the maximum use the opposite of the goal function to transform a maximum dilemma into an appropriate problem, for instance

$$X_j = X_i + Visual \times (2 \times rand - 1) \quad (3)$$

where other terms are the same as above.

Therefore, the AF moves ahead in this direction if the objective criterion, $Y_i < Y_j$, is satisfied; if not, choose a new random position X_j and carry out the objective condition. Step randomly after a certain number of times, known as the try numeral, if the criteria is not met. A short try quantity in

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

the predatory behavior indicates that the AF swims at random and deviates within the realm of local extreme values. The revised location is shown in Equation (4).

$$\begin{cases} X_i(t+1) = X_i(t) + \frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|} \times Step \times rand() & Y_i < Y_j, \\ Eqation(4), & Y_i \geq Y_j, \end{cases}$$

(4)

here the rapports are the similar as overhead.

2. **Swarming Behavior**.

The two Reynolds rules may be used to characterized swarming behavior:

- Relocating as close to the center of the closest mates as feasible.
- To minimize overpopulation, so that the artificial fish's capacity to swarm could be essentially realized.

The fish instinctively congregate in bunches when moving in order to prevent danger and ensure the colony's survival. Let X_c serve as the focal point of this meeting space, as Equation.(5):

$$X_c = \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad (5)$$

where n is the whole fish populace.

Let n_f determine how many of AF's friends live nearby (d_{ij} < Visual). If $(Y_c/n_f) > \delta Y_i$, The AF steps to the companion center if there is more food there (greater value of the fitness function) and there is less crowding; if not, the AF engages in preying behavior. The revised location requirements are Equation (6).

$$X_i(t+1) = X_i(t) + \frac{X_c(t) - X_i(t)}{\|X_j(t) - X_i(t)\|} \times Step \times rand() \quad \frac{Y_c}{n_f} > \delta Y_i,$$

(6)

where the rapports are the equivalent as above.

3. Following Behavior.

The following actions can be interpreted as a move in the direction of the best national buddies. Because of its lack of goal, the random behavior does not specify its direction.

Let X_i be the AF current position, and the neighbourhood friend (d_{ij}<Visual) with the greatest food consistency X_j If $(Y_j//n_f) > \delta Y_i$, The AF advances because of its companion's more spacious surroundings and increased food concentration (higher fitness function score); otherwise, it behaves like a predator. /e conditions are Equation (7).

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

$$X_i(t+1) = X_i(t) + \frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|} \times Step \times rand() \frac{Y_j}{n_f} > \delta Y_i,$$

(7)

4. Random Behavior.

In reality, the fish are searching for food or mates in wider areas, which is why they move aimlessly in the water. Predation does this by default. AF is located at Equation (8):

$$X_i(t+1) = X_i(t) + Visual \times (2 \times rand() - 1) \quad (8)$$

where the terms are the same as above.

B. Performance Metrics

The following performance measures are calculated for computing cloud job scheduling.

1. Execution Cost

The amount that the resource node must pay once all tasks have been completed is known as the task execution cost. Each resource node has a distinct cost per unit of time. Therefore, the task's execution cost is Equation (9)

$$Cost = \sum_{i=1}^{n} Time_i * P_i \quad (9)$$

In the formula, P_i symbolizes the cost that resource node j must pay each time unit.

2. Load Balancing

The loads of the virtual machines that are operating on a node may be added to determine its load. examining the decoded sequence, the quantity of tasks that are executing on the node, and the tasks that are executing on virtual machines. Over this $TotalTime$ period, the VM load[21]. The load of VM No. i in $TotalTime$ is V $(i, TotalTime)$f the load of virtual machines is comparatively constant during the period. At the same time, there are K VMs operating on node j.Co sequent, it may be said that the load of node j in $TotalTime$ is.Equation (10).

$$P_j(i, TotalTime) = \sum_{i=1}^{K} V(i, TotalTime) \quad (10)$$

The average load across all nodes during this $TotalTime$ is displayed as follows Equation (11).

$$\overline{P(T)} = \frac{1}{N} \sum_{j=1}^{N} P(j, T) \quad (11)$$

To accurately depict the magnitude of the loads on various nodes, incorporate variance Equation (12).

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

$$\alpha(T) = \sqrt{\sqrt{\frac{1}{N-1}}\sum_{j=1}^{N}(P(j,T) - \overline{P(T)})^2} \quad (12)$$

It is clear that load balancing and sensible work scheduling are increasingly important the smaller the organization. Finding the optimal scheduling that balances load and takes the least amount of time is the aim of this research.

3. Completion Time

The time needed for the reserve node to complete all of the tasks in the task set once they have all been assigned is known as the task completion time. Because the resource nodes execute tasks concurrently, the task completion time refers to the maximum value of the resource node's task execution time rather than the total duration of execution of all resource nodes, specifically. Equation (13 and 14).

$$Time_j = \frac{TotalLenght_j}{MIPS_j} \quad (13)$$

$$T = max(Time_j) \quad (14)$$

The time it takes for virtual node j to do all of its duties is represented by $Time_j$ in the formula, and the overall length of all the jobs that must be completed on virtual node j is represented by $TotalLenght_j$ and the virtual node j's capacity to handle tasks is represented by $MIPS_j$.

IV. RESULT ANALYSIS AND DISCUSSION

In this paper [25], a ubiquitous and adaptable simulation framework, Cloud Sim, is utilized to model the AFSA scheduling experiment on the cloud. An HP operating system, along with an Intel Core i7 CPU and 32 GB of RAM, was used to conduct the trials. It was Java that was utilized. Using execution cost, load balancing, and completion time as performance metrics, Table III displays the experimental outcomes of task scheduling using the suggested AFSA algorithm. Properties of the calculation method are provided in Table II.

TABLE II. PARAMETERS FOR THE AFSA ALGORITHM

Parameter	Value
Number of attempts: Attempt	3
Step length: Step	2.5
Field of vision: View	3.5
Crowding factor: δ	2
Threshold: t	5
Number of iterations: iter	100
Population size: Scale	40

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

The AFSA algorithm is configured with key parameters to optimize task scheduling, shows in Table II. It allows three attempts (Attempt) for movement decisions, with a 3.5-degree field of view and a 2.5-meter step length to direct exploration. A crowding factor (δ) of 2 prevents premature convergence, while a threshold (t) of 5 ensures fitness-based position acceptance. The algorithm runs for 100 iterations (iter) with a population size of 40 (Scale), balancing computational efficiency and solution diversity for improved execution cost, load balancing, and task completion time.

TABLE III. PERFORMANCE OF TASK SIZE FOR AFSA

Task Size	Execution cost	Load balancing	Completion Time
50	1.85	0.81	5.06
100	3.78	0.94	9.18
150	6.03	1.16	13.25
200	8.24	1.07	17.09
250	10.87	1.01	21.32

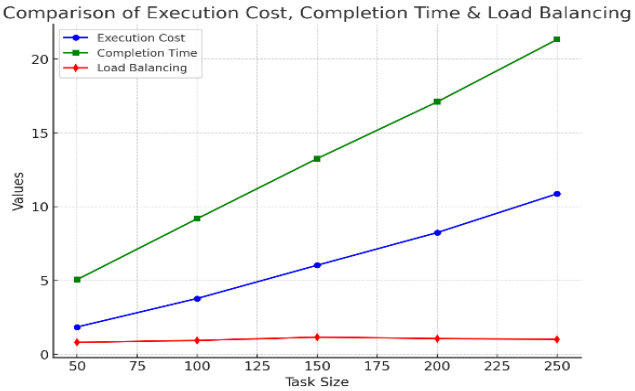

Fig. 3. Comparison of Execution Cost, Completion Time, & Load Balancing in Cloud Task Scheduling

A situation like using cloud computing, load balancing can reduce implementation cost, and time needed for completion are compared with respect to task size (see Table III and Image 3). as task size increases, execution cost and completion time exhibit a linear growth pattern, with execution cost rising from 1.85 to 10.87 and completion time increasing from 5.06 to 21.32, indicating higher resource consumption and prolonged processing durations. However, load-balancing values fluctuate slightly, peaking at 1.16 for a task size of 150 before decreasing to 1.01 at 250, suggesting that optimal load distribution is affected by increasing workloads. This graphic emphasizes the trade-offs associated with multi-objective task scheduling and the necessity of appropriate optimization strategies to reduce execution costs and completion times while preserving load balancing.

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

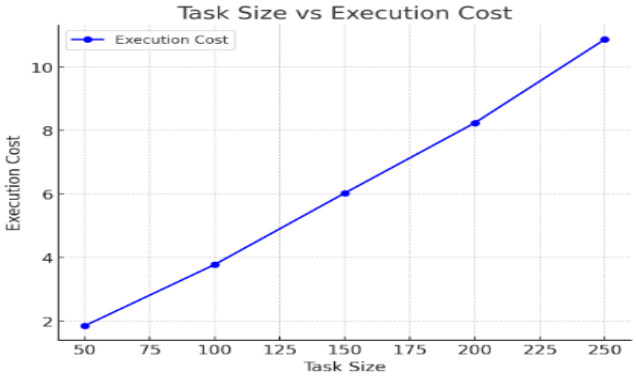

Fig. 4. Task Size vs Execution Cost

The relationship between the task size and the execution cost in a cloud computing environment is shown in Figure 4. Through the graph, it can be seen that an increase in execution cost with task size follows almost a linear trend. For a task size of 50, the execution cost grows from 1.85 to 10.87 for the task size of 250, which means that larger tasks require much more computational resources. In light of this tendency, it is clear that cloud-based systems require careful scheduling and allocation of resources to provide optimal performance at minimal cost.

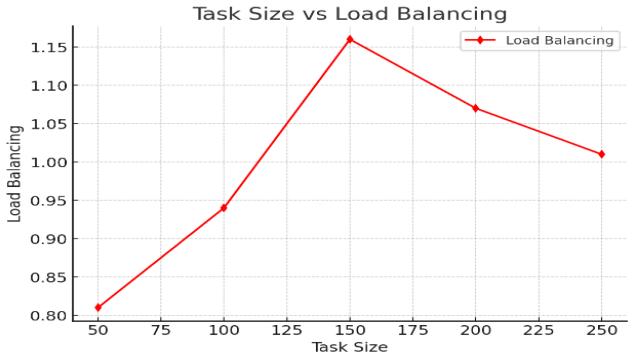

Fig. 5. Task Size vs Load Balancing

In the framework of online computing, Figure 5 illustrates how load balancing varies according on job size. Initially, load balancing improves from 0.81 for a task size of 50 to the peak of 1.16 for a task size of 150 as a better task distribution across computing resources. Beyond this point, load balancing drops slightly to 1.01 at a task size of 250, which may indicate the mass imbalance as workloads increase. This can be considered as a trend that points out the problem with maintaining the optimal load distribution with the growing task size, which is the motivation for developing dynamic scheduling strategies to utilize resources efficiently.

International Journal of Core Engineering & Management
Volume-7, Issue-09, 2024 ISSN No: 2348-9510

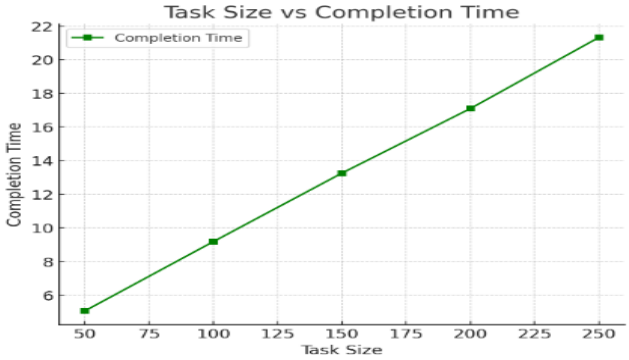

Fig. 6. Task Size vs Completion Time

In Figure 6, the relationship of task size and completion time in a cloud computing environment. Completion time increases linearly in an upward trend, from 5.06 for a task size of 50 to 21.32 for a task size of 250. It means that large tasks need a dramatic increase in processing time, which suggests that it is very important to have an efficient scheduling mechanism to reduce delays. The pattern shown shows the effect of the workload size on system performance, and the conclusion is that they are very significant to use the optimization techniques in command to diminish execution delays while maintaining the high efficiency.

TABLE IV. COMPARATIVE ANALYSIS OF LOAD BALANCING PERFORMANCE BETWEEN AFSA AND PSO

Load balancing		
Task Size	**AFSA**	**PSO[22]**
50	0.81	-
100	0.94	0.68
150	1.16	0.57
200	1.07	0.56
250	1.01	0.51

The load-balancing performance using the AFSA as well as PSO in terms of task size is presented in Table IV. When the sizes of task ranges from 50 to 250, and the corresponding AFSA values are shown through which the algorithm is efficient compared to PSO. In particular, AFSA yields its best performance (1.16) at a task size of 150, whereas the performance of PSO decreases from 0.51 to 0.49 when the task size increases from 250 to 350. The lack of PSO data for a task size of 50 is either infeasibility or lack of experimentation in that configuration. The comparison shows that PSO is not able to handle increasing task sizes as good as AFSA.

A. Discussion

The comparative analysis demonstrates that AFSA is effective in performing the comparative analysis to minimize execution cost, reduce completion time, improve distributing load, and optimize cloud computing time management. There needs to be an effective allocation of resources because the execution cost and completion time grow almost linearly with increasing job sizes. The

load balancing now fluctuates, peaks at 150, and then decreases slightly, indicating that it is difficult to maintain the best distribution under increasing workloads. It is found that AFSA is superior to PSO in load balancing and its task distribution is much better, particularly for large tasks. This highlights the superiority of AFSA in the cloud scheduling of tasks multi-objective optimization, which is regarded as a solution to improve system efficiency.

The advantages of AFSA offers the advantages of lower task execution cost, better load balancing, and smaller completion time, which makes it a robust optimization technique in cloud task scheduling. On the other hand, the high resource utilization possible with it due to its ability to dynamically adapt to different workloads improves system efficiency. This study has implications for coordinating in real time in expansive cloud settings, enabling AFSA to minimize real-time scheduling cost, reduce operational cost as well and increase service reliability. In the context of complicated cloud computing, future research might focus on utilizing hybrid AFSA models that integrate heuristic or deep learning approaches to improve performance, scalability, and flexibility.

V. CONCLUSION AND FUTURE SCOPE

Resource scheduling, which entails allocating accepted tasks to available Virtual Machines, is a crucial impending difficulty in cloud computing settings. For cloud computing, effective task scheduling is essential as it improves patterns of resource utilization, lowers execution costs, and maintains ideal load-balancing behaviors. Using PSO performance analysis, this study created work scheduling techniques using AFSA. For every recent experiment, AFSA shows better outcomes than PSO across a range of job sizes. The load-balancing evaluation reached its peak at 1.16 when AFSA processed 150 tasks, while PSO demonstrated decreasing performance metrics, which resulted in a minimum value of 0.51 at the task size of 250. The AFSA scheduling approach achieved better execution cost reduction and task completion time compared to PSO due to its effectiveness in dynamic cloud environments. The results of these results show that AFSA has greater facility in workload distribution and computational efficiency compared to the existing techniques. In order to improve scheduling performance, future studies will examine other schedules that use hybrid metaheuristic algorithms, ML-driven predictive scheduling, and real-time dynamic workload changes. Moreover, the proposed approach is also extendable to multi-cloud and fog computation environments for increasing the flexibility and scaling for real-life application

REFERENCES

1. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms,” Futur. Gener. Comput. Syst., 2009.
2. N. Tziritas, S. U. Khan, C. Z. Xu, and J. Hong, “An optimal fully distributed algorithm to minimize the resource consumption of cloud applications,” in Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS, 2012. doi: 10.1109/ICPADS.2012.19.
3. J. Li, Q. Li, S. U. Khan, and N. Ghani, “Community-based cloud for emergency management,” in Proceedings of 2011 6th International Conference on System of Systems Engineering: SoSE in Cloud Computing, Smart Grid, and Cyber Security, SoSE 2011, 2011. doi:

10.1109/SYSOSE.2011.5966573.
4. Godavari Modalavalasa, "The Role of DevOps in Streamlining Software Delivery: Key Practices for Seamless CI/CD," Int. J. Adv. Res. Sci. Commun. Technol., vol. 1, no. 12, pp. 258–267, Jan. 2021, doi: 10.48175/IJARSCT-8978C.
5. M. S. Samarth Shah, "Deep Reinforcement Learning For Scalable Task Scheduling In Serverless Computing," Int. Res. J. Mod. Eng. Technol. Sci., vol. 3, no. 12, pp. 1845–1852, 2021, doi: DOI : https://www.doi.org/10.56726/IRJMETS17782.
6. D. K. P. Kumar, M. T. Sree, C. Krithika, P. Swapna, and N. Bhargavi, "Task Scheduling in Cloud Computing Using Particle Swarm Optimization Algorithm," vol. 11, no. 6, pp. 1–5, 2023.
7. V. N. Boddapati et al., "Data migration in the cloud database: A review of vendor solutions and challenges," Int. J. Comput. Artif. Intell., vol. 3, no. 2, pp. 96–101, Jul. 2022, doi: 10.33545/27076571.2022.v3.i2a.110.
8. A. Al-Maamari and F. A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," Int. J. Grid Distrib. Comput., 2015, doi: 10.14257/ijgdc.2015.8.5.24.
9. R. Tarafdar, "Algorithms on Majority Problem," Univ. Missouri-Kansas City, 2017.
10. B. Boddu, "Cloud Dba Strategies For Sql And Nosql Data Management For Business-Critical Applications,"," Int. J. Core Eng. Manag., vol. 7, no. 1, 2022.
11. M. Shah, I. Researcher, A. Gogineni, and I. Researcher, "Distributed Query Optimization for Petabyte-Scale Databases," no. July, pp. 223–231, 2022.
12. S. Srichandan, T. Ashok Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm," Futur. Comput. Informatics J., 2018, doi: 10.1016/j.fcij.2018.03.004.
13. B. Kruekaew and W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning," IEEE Access, 2022, doi: 10.1109/ACCESS.2022.3149955.
14. Archana and N. Kumar, "Spider Monkey Optimization based Resource Provisioning in Cloud Computing Environment," in Proceedings of the 10th International Conference on Signal Processing and Integrated Networks, SPIN 2023, 2023. doi: 10.1109/SPIN57001.2023.10116420.
15. B. Saemi, A. A. R. Hosseinabadi, A. Khodadadi, S. Mirkamali, and A. Abraham, "Solving Task Scheduling Problem in Mobile Cloud Computing Using the Hybrid Multi-Objective Harris Hawks Optimization Algorithm," IEEE Access, 2023, doi: 10.1109/ACCESS.2023.3329069.
16. R. Mishra and M. Gupta, "Cloud Scheduling Heuristic Approaches for Load Balancing in Cloud Computing," in 2023 6th International Conference on Information Systems and Computer Networks, ISCON 2023, 2023. doi: 10.1109/ISCON57294.2023.10112056.
17. S. Sharma and N. K. Pandey, "Improved Task Scheduling Strategy Using Reinforcement Learning in Cloud Environment," in Proceedings - 2022 2nd International Conference on Innovative Sustainable Computational Technologies, CISCT 2022, 2022. doi: 10.1109/CISCT55310.2022.10046618.
18. H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "Multiobjective Task Scheduling in Cloud Environment Using Decision Tree Algorithm," IEEE Access, 2022, doi: 10.1109/ACCESS.2022.3163273.
19. D. Devi and S. G. Winster, "An Efficient Task scheduling and Data security in Heterogeneous Cloud Computing using Hybrid Meta-Heuristic Algorithm and Block Chain based Key Aggregate Cryptosystem," in 3rd International Conference on Power, Energy, Control and

Transmission Systems, ICPECTS 2022 - Proceedings, 2022. doi: 10.1109/ICPECTS56089.2022.10047356.

20. H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A Budget-Constrained Resource Provisioning Scheme for Workflow Scheduling in IaaS Clouds," IEEE Trans. Parallel Distrib. Syst., 2020, doi: 10.1109/TPDS.2019.2961098.
21. T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai, "Load balancing task scheduling based on genetic algorithm in cloud computing," in Proceedings - 2014 World Ubiquitous Science Congress: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, DASC 2014, 2014. doi: 10.1109/DASC.2014.35.
22. D. Yu, Z. Xu, and M. Mei, "Multi-objective Task Scheduling Optimization Based on Improved Bat Algorithm in Cloud Computing Environment," Int. J. Adv. Comput. Sci. Appl., 2023, doi: 10.14569/IJACSA.2023.01406117.