

Hybrid technique for optimal task scheduling in cloud computing environments

Nihar Ranjan Sabat¹, Rashmi Ranjan Sahoo², Manas Ranjan Pradhan³, Biswaranjan Acharya⁴

¹Faculty of Engineering, Biju Patnaik University of Technology (BPUT), Rourkela, India

²Center of Excellence on Cyber Security and Cloud Computing, Department of Computer Science and Engineering, Parala Maharaja Engineering College (PMEC) affiliated to Biju Patnaik University of Technology (BPUT), Berhampur, India

³School of Computing, Skyline University College, Sharjah, United Arab Emirates

⁴Department of Computer Engineering-AI and Big Data Analytics, Marwadi University, Gujarat, India

Article Info

Article history:

Received Jun 16, 2023

Revised Nov 18, 2023

Accepted Dec 9, 2023

Keywords:

Ant colony optimization

Cloud computing

Job scheduling

Load balancing

Particle swarm optimization

Task scheduling

ABSTRACT

Since cloud computing has an abundance of users, the system has to execute a wide range of tasks. Task scheduler methods that are both robust and efficient while delivering the best outcomes are required. The task volume and runtime in the cloud vary rapidly, making task assessment and resource mapping difficult. Security issues, communication delays, and data loss are substantial barriers to scheduling. Furthermore, optimization techniques can be utilized to reduce load and assign tasks so that the user can finish tasks faster. This paper offers a hybrid job scheduling technique for cloud computing using adaptive particle swarm optimization and ant colony optimization particle swarm optimization-ant colony optimization (adaptive PSO-ACO). After rapidly finding the initial solution via particle swarm optimization, the ant colony optimization approach establishes its first pheromone distribution. The suggested hybrid algorithm is compared to standalone PSO and ACO algorithms. Compared to ACO, the percentage decrease is 7.9%. Hybrid method has the lowest total cost, 55% less compared to PSO. Tasks vary when virtual machines (VMs) are constant and VMs vary when tasks are constant. Parameters like final cost, makespan, fitness value, computation time and weighted time are assessed to evaluate the performance of the hybrid task scheduling algorithm.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Biswaranjan Acharya

Department of Computer Engineering-AI and Big Data Analytics, Marwadi University

Rajkot, Gujarat-360003, India

Email: biswaacharya@ieee.org

1. INTRODUCTION

The emergence of cloud computing has had a profound and transformational effect on the field of computer technology [1]. The services provided supply consumers with accessible resources, allowing them to effectively and adaptively modify their scale [2]. Multiple virtual machines (VMs) can potentially coexist on a single host within the data center [3]. The scheduling of tasks is a crucial component inside the cloud computing environment [4]. There exist three main categorizations metaheuristic like population based algorithms, swarm intelligence algorithms and evolutionary algorithms [5]. The concept of self organized agents in swarm intelligence (SI) exemplifies the operation of the collective entity [6]. Algorithms grounded in imitation have been developed within the domain of social intelligence (SI) [7], [8]. The balanced candidate suffrage value (BCSV) scheduling technique proposed by Chiang *et al.* [9] enables improved job scheduling. Liu [10] has proposed an adaptive task scheduling methodology utilizing the ant colony algorithm (ACO). A novel hybrid algorithm based

on the whale optimisation algorithm (WOA) was presented by Manikandan *et al.* [11]. Guo [12] proposed a fuzzy self defense based multiobjective task scheduling optimization approach. The challenge of work scheduling is addressed by incorporating a hybrid fuzzy C-means clustering system [13] and black widow optimization.

– Motivation

For any activity, there are hundreds of digital instruments available. The user is unable to effectively delegate tasks to virtual resources. The cloud computing mechanism must be improved so that service providers can reduce resource utilization costs while increasing profit from client application support. The scheduling system manages a large number of cloud-based tasks to improve completion times, resource productivity, ultimately, and computing power.

Data loss, increased communication delays, and security are all scheduling challenges. Furthermore, the optimisation algorithm can be used to effectively allocate tasks and reduce load. As a result, the user is able to complete all assigned tasks in less time. Furthermore, in a cloud environment, the number and duration of tasks can change rapidly, making it difficult to measure all tasks and conduct optimal resource mapping. Consequently, it is necessary to develop an optimal scheduling algorithm that can manage tasks effectively so as to reduce excessive loads and increase computing efficiency. Incorporating the ACO and particle swarm optimization (PSO) methodologies, the present study introduces a novel hybrid approach to the challenge of task scheduling in cloud computing systems.

– Contribution

The following is a succinct overview of the significant accomplishments that have been attained through this research.

- i) To design and implement an improved task allocation algorithm that effectively distributes tasks across various computer resources for users.
- ii) To perform efficient task scheduling by incorporating the positive aspects of PSO and ACO for resource utilization.
- iii) To reduce the task's execution time and costs associated with task execution.
- iv) To conduct a test study of the proposed methodology using the cloudsim simulator and compare the performance of the proposed and existing systems in terms of performance metrics such as makespan, final cost, and execution time.

The remainder of the paper is organized as follows: section 2 presents the relevant literature on task scheduling in a cloud environment. Section 3 describes the method. Section 4 presents the simulation results and discussions. Section 5 concludes the paper with an overview of future research.

2. LITERATURE SURVEY

The resource and deadline aware dynamic load-balancer (RADL) task scheduler distributes computationally complex tasks equitably in autonomous runtime jobs [14]. Energy efficient virtual machine mapping algorithm (EViMA) increases task scheduling and resource management to reduce data cost, execution time, and energy consumption [15]. The scheduling algorithm (SOSA) has been designed with the purpose of performing job scheduling in a unique manner [16]. Through the utilisation of end-user weights, a java approach is able to optimise process scheduling while concurrently reducing both the makespan and the execution cost [17]. Genomics and quasi-reflection-based learning improved firefly over meta-heuristic firefly [18]. Movement and pollination, which have the highest exploitation-exploration tenacity, can be improved by modifying the sun flower optimisation algorithm [19]. For better cloud performance and urgent concerns, local pollination-based moth search algorithm (LPMSA) optimizes the cloud-based assignment of tasks using flower pollination and moth search algorithms [20]. Multi-object searching approach spacing multi-objective antlion algorithm (S-MOAL) decreases VM makespan and consumption cost [21]. Binary particle swarm optimization (BPSO) distributes applications among VMs, optimizes QoS, and meets end-user expectations, but its inefficient transfer function makes it a suboptimal option [22].

Black widow optimization (BWO) and adaptive neuro-fuzzy inference system (ANFIS) allocate tasks to the right VM and optimize environmental resource consumption [23]. Non-decomposition large-scale global optimisation (N-LSGO) algorithm reduces population density loss and early convergence in six phases to obtain an optimal solution [24]. Multi-objective workflow optimisation strategy (MOWOS) splits down large tasks into smaller subtasks to decrease makespan, execution, and workflow scheduling [25]. Genetic-based multi-objective scheduling for cloud-based scientific workflow scheduling optimizes cost, makespan, and load balancing to overcome cloud variations in environments, service quality, task dependencies, and user deadlines [26]. Hybrid weighted ant colony optimisation (HWACOA) optimizes job scheduling and minimizes cloud computing costs [27]. Kumar and Tyagi [28] distances assign task clusters to processors to demonstrate fuzzy system performance and optimal response time. Two hybrid genetic algorithms (HGAs) improve GA convergence, system cost, response time, and distributed real-time system reliability via new encoding, population initialization, and genetic operations [29]. Integrating local and global search methods improves the

convergence rate of the proposed PSO-based strategy for solving a distributed computing system assignment problem [30]. Table 1 provides an extensive overview of scheduling approaches from the literature. A comprehensive analysis was carried out to explore different scheduling strategies that have been outlined previously in the literature. In a cloud computing environment, task scheduling is a major concern.

Table 1. Comparison of existing scheduling algorithms in the literature

| Author and year | Scheduling algorithm | Application |
|--|---|---|
| Nabi <i>et al.</i> [14] (2022) | Resource and deadline aware dynamic load-balancer (RADL) | Cloud-based task scheduling utilising cloud computing technology to effectively manage and allocate tasks. |
| Konjang <i>et al.</i> [15] (2022) | Energy efficient virtual machine mapping algorithm (EViMA) | Perform efficient task scheduling while effectively managing the environment's resources. |
| Kumar and Suman [16] (2022) | Sailfish optimization-based scheduling algorithm (SOAS) | Enhance the quality of service for a range of technologies, including mobile computing and the internet of things, to manage the NP-hardness of the task scheduler. |
| Gupta <i>et al.</i> [17] (2021) | Java-based algorithm for workflow scheduling | Reduce the makespan and execution cost and to generate a result based on end-user-specified weights. |
| Bacarin <i>et al.</i> [18] (2022) | Improved firefly algorithm | Effectively addresses the problem of workflow scheduling in edge-based cloud environments by using quasi-reflection-based learning methods and genetic operators. |
| Chandrasekar and Krishnadoss [19] (2022) | Opposition-based sunflower optimization (OSFO) algorithm | Balance exploration and exploitation searches with optimal tenacity to optimise the task scheduler's makespan and execution cost. |
| Gokuldhev and Singaravel [20] (2022) | Local pollination-based moth search algorithm (LPMSA) | Determine the best way to distribute tasks in a cloud environment to maximise efficiency and address pressing issues. |
| Belgacem <i>et al.</i> [21] (2022) | Spacing multi-objective antlion algorithm (S-MOAL) | Respond to user resource requests in a timely and effective manner by reducing VM makespan and consumption cost. |
| Kumar <i>et al.</i> [22] (2020) | Binary particle swarm optimization (BPSO) | Equitable application distribution among VMs, QoS optimisation, and end user requirements fulfillment. |
| Nanjappan <i>et al.</i> [23] (2021) | Adaptive neuro-fuzzy inference system (ANFIS)-black widow optimization (ANFIS-BWO) | Assign each task to the right VM. |
| Shahraki and Zamani [24] (2021) | Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization (DMDE) | Reduce population density loss and premature convergence. |
| Sardaraz and Tahir [25] (2020) | Multi-objective scheduling algorithm for scheduling scientific workflows | Enhance workflow scheduling by reducing makespan and execution costs. |
| Konjaang and Xu [26] (2021) | Multi-objective workflow optimization strategy (MOWOS) | Optimise load balancing, cost, and makespan during scientific workflow scheduling. |
| Chandrasekhar [27] <i>et al.</i> (2023) | Hybrid weighted ant colony optimization algorithm | Lower the makespan and cost of cloud computing while optimizing performance. |
| Kumar and Tyagi [28] (2020) | Hierarchical clustering task allocation model | Maximize the utilization of resources, partition tasks, and allocate them strategically. |
| Kumar and Tyagi [29] (2021) | Hybrid approach for scheduling tasks | Reduce system costs and response time, maximize system reliability. |
| Karishma and Kumar [30] (2023) | PSO-based metaheuristic algorithm | Decrease system costs, response time, and flowtime by boosting performance. |

3. METHOD

VMs can be provisioned using cloudsims on two different levels. Figure 1 depicts the impact of various provisioning policies on task unit execution for VMs and tasks in Figure 1(a); Figure 1(b) space-shared provisioning for VMs and time-shared provisioning for tasks; Figure 1(c) time-shared provisioning for VMs, space-shared provisioning for tasks; and Figure 1(d) time-shared provisioning for VMs and tasks. A request to host two virtual machines (VM_s) with each needing two processing cores and planning to house four task units is made to a VM host with two processing cores. The tasks t_1, t_2, t_3 and t_4 will be hosted by VM_1 , and the tasks t_5, t_6, t_7 and t_8 will be hosted by VM_2 . Figure 1 presents a straightforward virtual machine provisioning scenario. A VM host with two processing cores receives a request to host two VM_s . VM_1 will be used to host the tasks t_1, t_2, t_3 and t_4 , while VM_2 will be used to house the tasks t_5, t_6, t_7 and t_8 . A job controlled by a VM may have its estimated finish time, $eft(p)$ which is supplied by (1).

$$eft(p) = est(p) + \frac{rl}{capacity \times cores(p)} \quad (1)$$

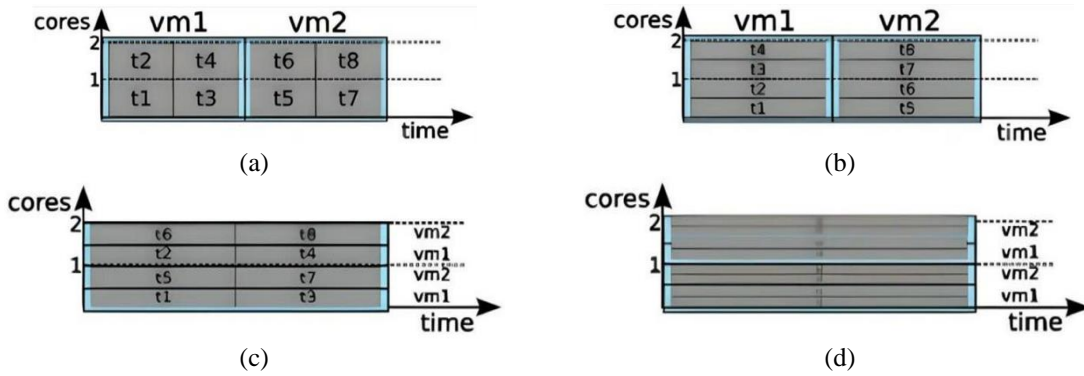


Figure 1. Impact on task unit execution of various provisioning policies in (a) for VMs and tasks, space-shared provisioning; (b) space-shared provisioning for VMs and time-shared provisioning for tasks; (c) time-shared provisioning for VMs, space-shared provisioning for tasks; and (d) time-shared provisioning for VMs and tasks

Where rl represents the total amount of processor instructions needed to run the cloudlet and $est(p)$ is the estimated start time of the cloudlet. This policy uses (2).

$$capacity = \sum_{i=1}^{np} \frac{cap(i)}{np} \tag{2}$$

$cap(i)$ represents each constituent’s processing power.

$$eft(p) = ct + \frac{rl}{capacity \times cores(p)} \tag{3}$$

Where $eft(p)$ is the estimated finish time, ct is the current simulation time, and $cores(p)$ is the number of cores (PEs) required by the cloudlet. In this case, the cloud host’s overall processing power is calculated as (4).

$$capacity = \frac{\sum_{i=1}^{np} cap(i)}{\max(\sum_{j=1}^{np} cores(j), np)} \tag{4}$$

It is proposed to utilise a hybrid adaptive PSO-ACO technique that combines the strengths of both algorithms. Use the system *Fairness* (F) to determine the fitness value of each particle j . This system fairness is defined for k tasks (t_1, t_2, \dots, t_k), their related fairness values (f_1, f_2, \dots, f_k) and represented by (5).

$$F = \sum_{j=1}^k |F_j| \tag{5}$$

In the proposed research, the optimized fitness value of each particle is compared to its previous best position (p_{best}). If the current p_{best} value is better than the previous p_{best} values, it is designated as the particle’s best value. These factors are calculated in order to figure out inertia weight in a swarm S . The swarm’s current average fitness is represented by (6).

$$F_a = \frac{1}{S} \sum_{j=1}^S F_j \tag{6}$$

The group fitness G is provided by (7).

$$G = 1 - \frac{1}{S} \sum_{j=1}^S (F_j - F(a))^2 \tag{7}$$

Each particle position is represented with respect to group fitness $X_i = (x_{i1}, x_{i2}, x_{i3} \dots, x_{in})$ and the corresponding velocity is shown as $V_i = (v_{i1}, v_{i2}, v_{i3} \dots, v_{in})$ that can be obtained by $P_i = (p_{i1}, p_{i2}, p_{i3} \dots, p_{in})$ denotes the best prior position of any particle. The updated rules for both velocity and position in the adaptive PSO are (4) and (5), respectively. Velocity is denoted as (8).

$$V_i = w * V_i + c_1 * rand() * (P_i - X_i) + c_2 * rand() * (P_i - X_i) \tag{8}$$

Group fitness is represented as (9):

$$X_i = X_i + V_i \quad (9)$$

Where C_1 and C_2 are the so-called “self cognitive” and “social learning” acceleration coefficients respectively, and w is the inertia weight. The rand () function can generate a random number in the range[0,1]. The mathematical representation of the cross entropy and logarithmic loss is expressed as (10) and (11):

$$Entropy(t) = \sum_{j=1}^M [y_j \log(p(y_j))] \quad (10)$$

$$Log Loss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (11)$$

The probability of selecting a certain item within a loop is denoted by the notation $p(y_i)$. The variable N represents the total number of items. The variable (y_i) represents the output of the i th item. The variable (p_i) represents the likelihood of an event occurring, where (p_i) is equal to 1. Conversely, the probability of the event not occurring is represented by $(1 - p_i)$, where p_i is equal to 0. The value of the position vector must adhere to the binary constraint of either 0 or 1 [31]. The formula for determining transition probability for the initial n ants within a given time frame t , when said node is not included in the taboo table of ant n , can be expressed as (12).

$$P_{ab}^n = \frac{[R_{ab}(t)]^\varphi [I_{ab}(t)]^\omega}{\sum_{y_{ab} \notin T_n} [R_{ab}(t)]^\varphi [I_{ab}(t)]^\omega} \quad (12)$$

Where n be a variable ranging from 1 to v , where v is a constant. Additionally, let $R_{ab}(t)$ represent the remaining pheromone on y_{ab} at time t . The data that serves as motivation is derived from $I_{ab}(t)$. The data volume and the heuristic data can be quantified by the relative degrees φ and ω , respectively. When an ant selects y_{ab} at time t , a designated set of nodes will be appended to T_n . The amount of information contained in the $path(a, b)$ is determined by (13).

$$\Delta R_{ab}^n(t) = 0 \text{ or } \frac{z}{I_{ab}^n(t)} \quad (13)$$

The pheromone can be updated at a local level by employing the (14).

$$(a, b). R_{ab}(t + 1) = R_{ab}(t)(1 - \xi) + \sum_{n=1}^v \Delta R_{ab}^n(t) \quad (14)$$

Where ξ is the pheromone volatilization coefficient [0, 1]. $1 - \xi$ is the information remaining coefficient.

The algorithm proposed was implemented in cloudsim. Cloudsim can model, simulate, and explore cloud systems. The core of cloudsim are datacenters, hosts, VMs, cloudlets, and brokers. This process maps customer requests to the best provider. Time shared scheduling allows multiple virtual machines to run simultaneously.

| Algorithm | Proposed adaptive PSO-ACO algorithm |
|-----------|---|
| Input: | Tasks $T_p, 1 \leq p \leq n$ and virtual machines $VM_k, 1 \leq k \leq n$ |
| Output: | Scheduled task order |
| 1: | Randomise particle j 's positions and velocities. |
| 2: | Calculate j 's fitness using system fairness. |
| 3: | Evaluate the swarm's estimated mean fitness. |
| 4: | Find the optimal particle and swarm values. |
| 5: | Update the inertia weight. |
| 6: | Refresh the particle positions and velocities. |
| 7: | Assess group fitness, variety, and adjustment function to calculate inertia weight. |
| 8: | Create an undirected network using velocity and distribution relationship matrices. |
| 9: | Calculate y_{ab} transition probability. |
| 10: | Choose y_{ab} at time t by ant n . |
| 11: | Update T_n with the precise nodes and assign the task to a virtual machine. |
| 12: | Use path information to locally update node's pheromone. |

4. RESULTS AND DISCUSSION

In order to conduct the result analysis, the simulation is carried in two different scenarios: (i) by keeping the fixed number of VMs and cloudlets (tasks) and (ii) by varying the both VMs and cloudlets. In scenario (i) best makespan, best fitness value, total cost, total computation time, and total weighted time are considered as performance metrics to assess the proposed methodology in comparison to other algorithms. In scenario (ii) the performance analysis is conducted by using three critical parameters such as final cost, execution time, and final makespan. Furthermore, statistical analysis such as one way analysis of variance (ANOVA), five paired t-test are performed to compare the makespan among various algorithm.

4.1. Comparison of results with fixed number of cloudlets

The cloud execution time chart, datacenters used, scheduling time, and start and finish times are all shown in Table 2. In the result analysis, a fixed number of five virtual machines are used along with ten cloudlets. A chart demonstrating the cloud's execution time is included, along with the scheduling, start and finish times for each of the three data centres. The graphical representation of the cloud execution time chart is shown in Figure 2. The results of the PSO algorithm [11] and the ACO algorithm [12] are compared with the proposed PSO-ACO algorithm. We obtained several significant performance metrics, including the best fitness value, the best makespan value, the total cost value, the total computation time, and the total weighted time, by simulating the aforementioned methods. Table 3 displays a comparison of the simulation results produced by the suggested technique and the existing PSO and ACO approach.

Table 2. Cloud execution time chart

| Cloudlet ID | Status | Data center ID | VM ID | Scheduling time | Start time | Finish time |
|-------------|---------|----------------|-------|-----------------|------------|-------------|
| 0 | Success | 3 | 0 | 61 | 0.1 | 62 |
| 1 | Success | 3 | 0 | 61 | 0.1 | 62 |
| 2 | Success | 3 | 1 | 62 | 0.1 | 63 |
| 3 | Success | 3 | 1 | 62 | 0.1 | 63 |
| 4 | Success | 3 | 2 | 63 | 0.1 | 64 |
| 5 | Success | 3 | 2 | 63 | 0.1 | 64 |
| 6 | Success | 3 | 3 | 64 | 0.1 | 65 |
| 7 | Success | 3 | 3 | 64 | 0.1 | 65 |
| 8 | Success | 3 | 4 | 65 | 0.1 | 66 |
| 9 | Success | 3 | 4 | 65 | 0.1 | 66 |

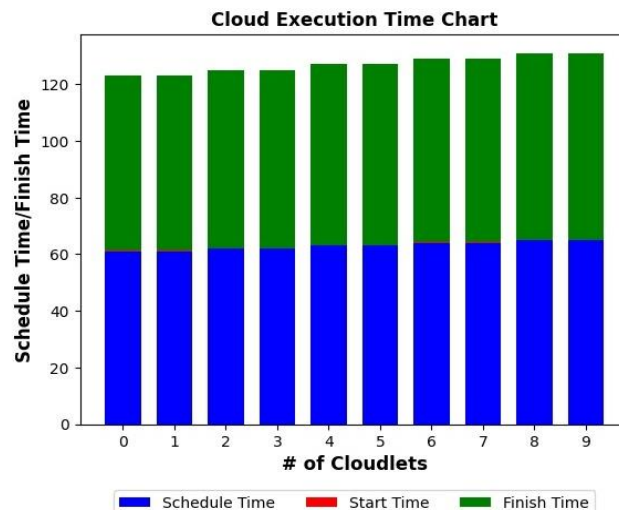


Figure 2. Cloud execution time chart

Table 3. Comparison of results

| Algorithm | Best fitness value | Best makespan | Total cost | Total computation time | Total weighted time |
|------------------|--------------------|---------------|-------------|------------------------|---------------------|
| PSO [11] | 4786.127043 | 2620.772051 | 1574.17712 | 39354.428 | 1.37E+08 |
| ACO [12] | 9146.792102 | 11579.63127 | 3688.222622 | 147528.9049 | 6.29E+07 |
| Adaptive PSO-ACO | 1792 | 918.08 | 704 | 46080 | 9764812.5 |

A comprehensive analysis of several key performance metrics has been provided, encompassing the following: best fitness value, best makespan, total cost, total computation time, and total weighted time. Figure 3 depicts a comparison of the best fitness values attained by the hybrid algorithm proposed in this research, in relation to the PSO and ACO algorithms. The ACO algorithm achieves its best fitness value at a maximum of 9146.79. The proposed hybrid algorithm exhibits the lowest value, which is recorded as 1792. Figure 4 presents a comparison of the best makespan values.

The proposed hybrid algorithm achieves the lowest makespan value of 918.08. Figure 5 presents a comparison of the total computation time. The proposed hybrid algorithm achieves the shortest computation time of 46080 seconds. In Figure 6, the total weighted time is compared. It was established that the proposed method required the shortest total weighted time, which was visually represented in the comparison.

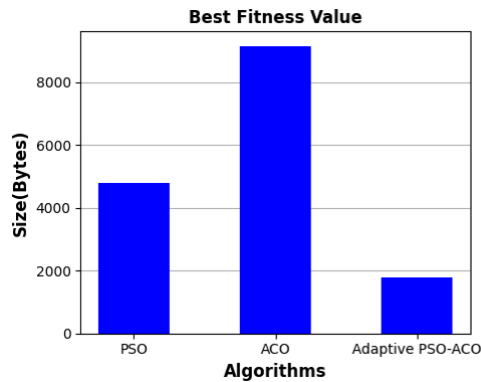


Figure 3. Comparison of best fitness values

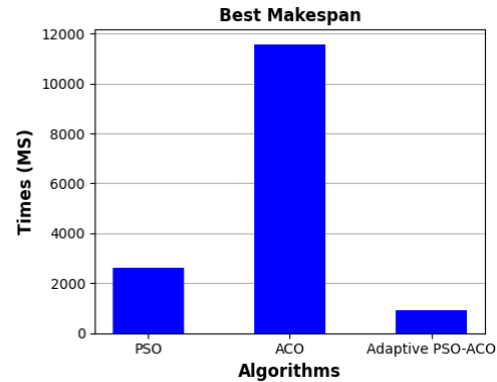


Figure 4. Comparison of best makespan values

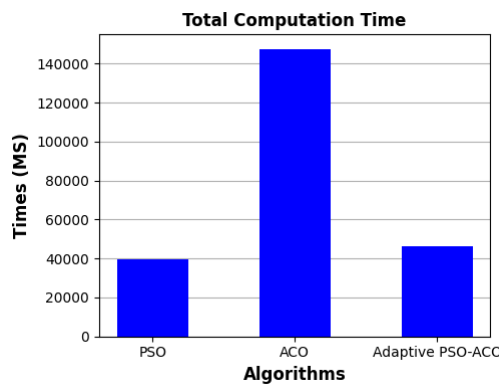


Figure 5. Comparison of total computation time

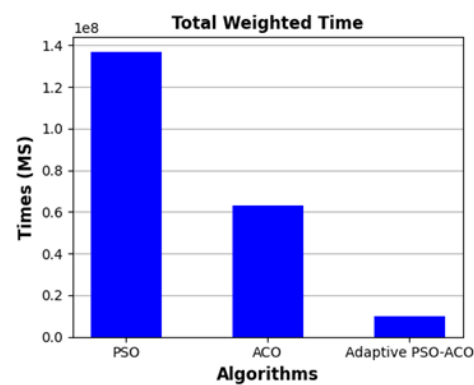


Figure 6. Comparison of total weighted time

4.2. Comparison of results with varying number of cloudlets

In the realm of VMs, the task allocation process has been effectively executed. To facilitate the execution of our study, we have opted to utilize a sample including ten VMs that exhibit different numbers of VMs and tasks. In the context of conducting performance analysis, three key metrics to consider are execution time, final makespan, and final cost. The evaluation of these three important metrics involves the configuration of different numbers of VMs and tasks, such as (5, 5), (5, 10), (10, 10), and so on.

4.2.1. Execution time

The total time required for the completion of a task after its start is commonly known as the execution time. It's vital to analyse a task's structure in order to accurately estimate its execution time. Thus, each phase begins and ends distinctly. The phase would begin or end with waiting for input or delaying output. According to the following notation, the mathematical representation of execution time is denoted.

$$Execution\ Time = \sum_{i=0}^n ET_j(Cloulet_i) \tag{15}$$

$ET_j(Cloudlet_i)$ is the execution time of *cloudlet i* on VM_j . A cloudlet is the term used to describe the process of assigning a single task to a VM. VMs and task loads affect execution time, as seen in Figure 7. The execution time is represented graphically for varying numbers of virtual machines and tasks.

The analysis found a positive correlation between VM performance and execution time. This applies even if tasks and VMs remain unaltered. Variations in execution time associated with existing task loads and VMs have been demonstrated visually. Based on the findings of the research, it has been established that there is a connection between the fluctuation in the performance of VMs and the amount of time that is necessary to finish a task. In point of fact, this principle holds true even in circumstances in which the tasks and the VMs do not undergo any modifications.



Figure 7. Execution time with respect to varying number of VMs and task

4.2.2. Final makespan

The final makespan refers to the accumulated duration of the schedule, encompassing the completion time of every task after processing. It is the entire duration required to complete a given set of tasks, specifically the longest completion time among all the tasks. The mathematical representation of final makespan is denoted as (16).

$$Final\ Makespan = \max_{1 \leq i \leq n} \{Completion\ Time_i\} \quad (16)$$

The variable “*n*” serves as a parameter to represent the number of VM. The temporal metric final makespan is shown graphically in Figure 8. The figure illustrates the changes in the final makespan that are attributed to task loads and VMs. The variations in final makespan that are linked to VMs and task loads are shown. The final makespan exhibits an increasing trend due to the variability of the VMs while the tasks remain constant, or alternatively, due to the variability of the tasks while the VMs remain constant.

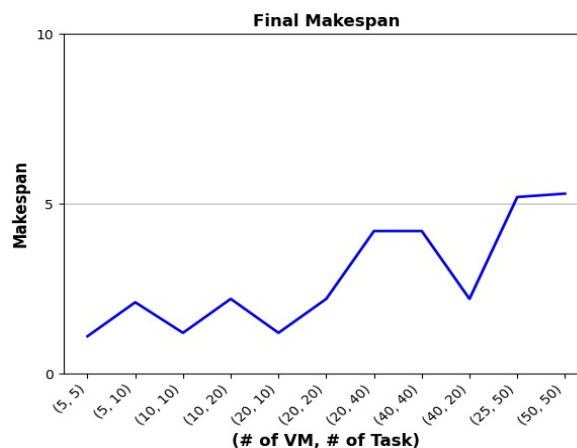


Figure 8. Final makespan representation for varying numbers of VMs and tasks

4.2.3. Final cost

The final cost in cloud task scheduling depends on source usage, instance types, task execution time, data transmission costs, storage pricing, and task priority. Understanding these components is essential for cost optimization and control. Optimizing cloud task scheduling expenses entails managing these variables and understanding the cloud service provider’s price. Reassessing consumption patterns and changing needs is key to cloud computing final cost management.

The incorporation of a simple cost function enables to reduce the complexity of the computation. The virtual machine with higher CPU speed is a relatively costlier alternative while compared with the virtual machine with a lower CPU speed. The costs (C) have been determined using the established definition, as outlined in the suggested approach.

$$C(a, b) = \frac{(W(a,b) \times VM_b)}{\text{minute}} \tag{17}$$

Where $C(a, b)$ is the expense of completing job V_a within the limits of the VM_b . The final cost, which represents the cumulative amount of all expenditures, is denoted as (18).

$$\text{Final Cost} = \sum_{bc \text{ Selected } VM} C(a, b) \tag{18}$$

The final cost representation is shown in Figure 9, which shows the effect of changing the number of VMs in respect to both VMs and tasks. The range of changes seen in the final costs of VMs and the variety seen in the final costs of task loads are shown. It is clear that changing the tasks while maintaining the same number of VMs or changing the VMs while maintaining the same number of tasks results in varying final cost.

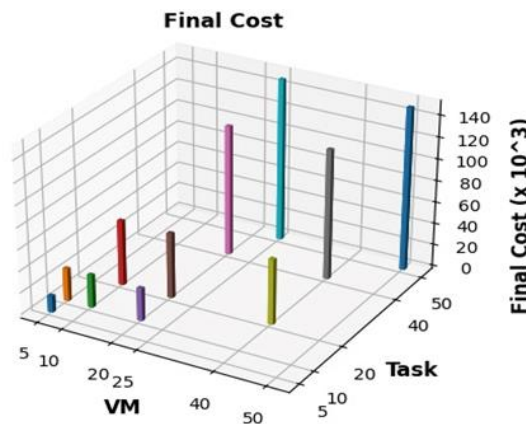


Figure 9. Final cost vs VMs and tasks

4.3. Discussion

The fitness of a schedule is evaluated by considering its performance in different topologies in a period [32]. The ACO algorithm achieves its optimal fitness value at a maximum of 9146.79. The proposed hybrid algorithm exhibits the lowest value, which is recorded as 1792. A comparison of the optimal makespan settings is completed and the suggested hybrid algorithm achieves the lowest makespan value, which amounts to 918.08. Makespan is the completion time of the final task to exit the system. The hybrid algorithm under consideration has an intermediate time of 46080 seconds. A visual representation of the comparison between the total weighted time is represented in the results section. The cumulative weighted time for PSO is 1.38×10^8 . Table 4 presents the data pertaining to the comparison of makespan between our recommended approach and other current approaches of a similar kind [33]. When compared to the makespan values of other techniques, our proposed approach obtained the lowest minimum makespan value of 2620.77. Figure 10 shows a makespan comparison between our proposed approach and other existing approaches of a similar nature.

| Algorithm | MTCT | MAXMIN | ACO | NSGA-II | DCLCA | Proposed work |
|-----------|----------|-----------|-----------|----------|--------|---------------|
| Makespan | 14042.70 | 13,671.90 | 11,057.40 | 10099.50 | 8898.8 | 2620.77 |

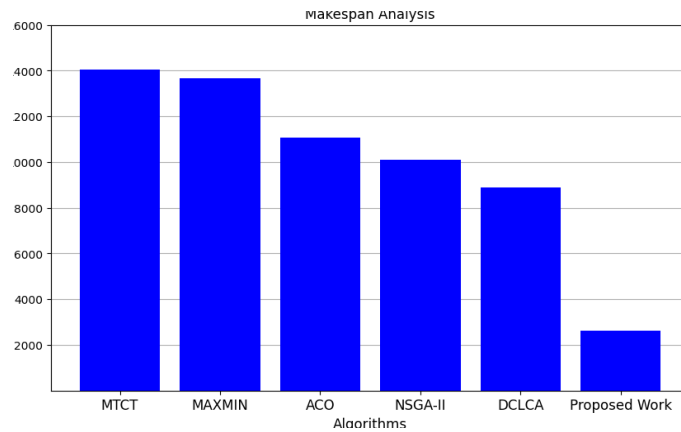


Figure 10. Comparison of makespan

We conducted a makespan analysis that compared our proposed approach and other methods that were of identical character. In terms of attaining the lowest possible makespan value, the graph presents strong reasons that our suggested approach outperformed similar approaches. The primary goal of offline parameter tuning is to obtain parameter values that can be advantageous in solving a significant number of instances, which requires conducting multiple experimental evaluations. In practice, it is common to adjust metaheuristic by modifying one parameter at a time, and the optimal values for these parameters are typically determined through empirical means. In order to optimize the results, multiple iterations of simulations were conducted, wherein the parameters of position and velocity of particles in PSO, as well as the pheromone evaporation rate and pheromone intensity in ACO, were systematically varied.

4.3.1. ANOVA

The proposed method's overall makespan is compared to that of other existing methods, including MTCT, MAXMIN, ACO, NSGA-II, and DCLCA. This comparison is conducted by performing ten iterations for each algorithm. The comparison of makespan between different algorithms is conducted through the utilisation of a one-way ANOVA approach. The null hypothesis for this analysis is denoted as:

- H0: the average Makespan of all six methods, including the proposed method, is equal. The alternative hypothesis.
- H1: the mean Makespan of all the aforementioned methods is not equal.

Table 5 summarises the results obtained from the ANOVA method.

Table 5. One-way ANOVA parametric test results

| Source | DF | Sum of square | Mean square | F statistic | P-value |
|-------------------------|----|---------------|-------------|-------------|----------|
| Groups (between groups) | 5 | 823355035.2 | 164671007 | 3130.5314 | 2.22E-16 |
| Error (within groups) | 54 | 2840487.161 | 52601.6141 | | |
| Total | 59 | 826195522.3 | 14003313.94 | | |

It is clear that the calculated F statistic is far higher than the associated P value. Consequently, the null hypothesis, which states that the mean makespan of all six methods, including the proposed method, are equal, is rejected. Therefore, it can be concluded that the mean makespan of the six methods, including the proposed method, differs from each other.

4.3.2. Statistical t-test

In this study, a total of six methods were examined, one of which was the proposed approach for conducting pairwise parametric t-tests. A total of five paired t-test calculations were conducted to ascertain the approach that yielded the most effective overall makespan. To assess the efficacy of our proposed approach, a comprehensive set of experiments was conducted, wherein we conducted comparative analyses with established methodologies including MTCT, MAXMIN, ACO, NSGA-II, and DCLCA. A one-tailed hypothesis test was performed at a significance level of 0.05. The determination of the degrees of freedom, which is equal to $10-1$ resulting in 9, was a direct outcome of having a sample size of $N = 10$ for each test. A total of ten iterations are conducted for each algorithm included in the analysis to facilitate the calculation of the t-test. The null hypothesis (H0) states that the makespan of the existing algorithm being tested is equal

to that of our proposed work. According to the alternative hypothesis (H1), the makespan of our proposed work is lower than the Makespan of the existing algorithm participating in the test.

Table 6 presented below exhibits the outcomes obtained from the t-test where degree of freedom $df = (10 - 1) = 9$, $\mu = 0$. The results of five tests, each of which demonstrates a p-value below 0.00001, indicating statistical significance at a significance level of $p < 0.05$, have been presented. Hence, it can be inferred that the proposed methodology consistently achieves the lowest possible value for the overall makespan in all conducted experiments.

Table 6. Outcomes obtained from the t-test

| Test | Treatment 1 | Treatment 2 | Difference scores and t-value calculation | | | | | |
|------|-------------|---------------|---|-----------|------------------|-------------------|----------------------------|-----------------------------|
| | | | M | SS | $\frac{S^2}{df}$ | $\frac{S_M^2}{N}$ | $\frac{S_M}{\sqrt{S_M^2}}$ | $t = \frac{(M - \mu)}{S_M}$ |
| 1 | MTCT | Proposed work | -11294.7 | 50030.57 | 5558.95 | 555.9 | 23.58 | -479.05 |
| 2 | MAXMIN | Proposed work | -10634.55 | 715509.69 | 79501.08 | 7950.11 | 89.16 | -119.27 |
| 3 | ACO | Proposed work | -8036.76 | 721415.42 | 80157.27 | 8015.73 | 89.53 | -89.77 |
| 4 | NSGA-II | Proposed work | -7223.34 | 195895.46 | 21766.16 | 2176.62 | 46.65 | -154.83 |
| 5 | DCLCA | Proposed work | -6089.53 | 135279.11 | 15031.01 | 1503.1 | 38.77 | -157.07 |

5. CONCLUSION

This study presents a novel hybrid optimization approach for cloud task-scheduling algorithms, integrating artificial particle swarm optimization and ant colony optimization. The hybrid optimization technique can be visually represented by a graphical illustration of its optimal positions. Furthermore, this study presents the execution cost and time associated with the scheduling process that has been attained through the utilization of optimization techniques. In our future research, we plan to develop a mechanism that facilitates efficient scheduling within a dynamic cloud environment. The primary objective of this mechanism is to minimize system cost and response time, while simultaneously enhancing system reliability.




REFERENCES

- [1] D. Sabella, "Principles of Edge Computing, Fog and Cloud Computing," in *Textbooks in Telecommunications Engineering*, 2021, pp. 3–18. doi: 10.1007/978-3-030-79618-1_1.
- [2] M. Ibrahim *et al.*, "A Comparative Analysis of Task Scheduling Approaches in Cloud Computing," in *Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020*, IEEE, May 2020, pp. 681–684, doi: 10.1109/CCGrid49817.2020.00-23.
- [3] S. Nabi and M. Ahmed, "PSO-RDAL: particle swarm optimization-based resource- and deadline-aware dynamic load balancer for deadline constrained cloud tasks," *Journal of Supercomputing*, vol. 78, no. 4, pp. 4624–4654, 2022, doi: 10.1007/s11227-021-04062-2.
- [4] S. A. Sheik and A. P. Muniyandi, "Secure authentication schemes in cloud computing with glimpse of artificial neural networks: A review," *Cyber Security and Applications*, vol. 1, p. 100002, Dec. 2023, doi: 10.1016/j.csa.2022.100002.
- [5] Q. Z. Yang, X. L. Xie, and Z. T. Li, "Research on cloud computing task scheduling based on evolutionary algorithm," in *Proceedings - 2020 International Conference on Big Data and Artificial Intelligence and Software Engineering, ICBASE 2020*, IEEE, Oct. 2020, pp. 377–380. doi: 10.1109/ICBASE51474.2020.00086.
- [6] K. Dubey and S. C. Sharma, "A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 32, p. 100605, Dec. 2021, doi: 10.1016/j.suscom.2021.100605.
- [7] A. Keivani and J. R. Tapamo, "Task scheduling in cloud computing: A review," in *icABCD 2019 - 2nd International Conference on Advances in Big Data, Computing and Data Communication Systems*, IEEE, Aug. 2019, pp. 1–6, doi: 10.1109/ICABCD.2019.8851045.
- [8] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing," *Sensors*, vol. 22, no. 3, p. 920, Jan. 2022, doi: 10.3390/s22030920.
- [9] M. L. Chiang, H. C. Hsieh, Y. H. Cheng, W. L. Lin, and B. H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," *Expert Systems with Applications*, vol. 212, p. 118714, Feb. 2023, doi: 10.1016/j.eswa.2022.118714.
- [10] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," *Optik*, vol. 258, p. 168677, May 2022, doi: 10.1016/j.ijleo.2022.168677.
- [11] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35–44, Apr. 2022, doi: 10.1016/j.comcom.2022.01.016.
- [12] X. Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5603–5609, Dec. 2021, doi: 10.1016/j.aej.2021.04.051.
- [13] N. Manikandan, P. Divya, and S. Janani, "BWFSO: Hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing," *Materials Today: Proceedings*, vol. 62, pp. 4903–4908, 2022, doi: 10.1016/j.matpr.2022.03.535.
- [14] S. Nabi, M. Aleem, M. Ahmed, M. A. Islam, and M. A. Iqbal, "RADL: a resource and deadline-aware dynamic load-balancer for cloud tasks," *Journal of Supercomputing*, vol. 78, no. 12, pp. 14231–14265, 2022, doi: 10.1007/s11227-022-04426-2.




- [15] J. K. Konjaang, J. Murphy, and L. Murphy, "Energy-efficient virtual-machine mapping algorithm (EViMA) for workflow tasks with deadlines in a cloud environment," *Journal of Network and Computer Applications*, vol. 203, p. 103400, 2022, doi: 10.1016/j.jnca.2022.103400.
- [16] M. Kumar and Suman, "Scheduling in IaaS Cloud Computing Environment using Sailfish Optimization Algorithm," *Trends in Sciences*, vol. 19, no. 10, p. 4204, 2022, doi: 10.48048/tis.2022.4204.
- [17] S. Gupta, R. S. Singh, U. D. Vasant, and V. Saxena, "User defined weight based budget and deadline constrained workflow scheduling in cloud," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 24, p. e6454, 2021, doi: 10.1002/cpe.6454.
- [18] N. Bacanin, M. Zivkovic, T. Bezdan, K. Venkatachalam, and M. Abouhawwash, "Modified firefly algorithm for workflow scheduling in cloud-edge environment," *Neural Computing and Applications*, vol. 34, no. 11, pp. 9043–9068, 2022, doi: 10.1007/s00521-022-06925-y.
- [19] C. Chandrashekar and P. Krishnadoss, "Opposition based sunflower optimization algorithm using cloud computing environments," *Materials Today: Proceedings*, vol. 62, pp. 4896–4902, 2022, doi: 10.1016/j.matpr.2022.03.534.
- [20] M. Gokuldhev and G. Singaravel, "Local Pollination-Based Moth Search Algorithm for Task-Scheduling Heterogeneous Cloud Environment," *Computer Journal*, vol. 65, no. 2, pp. 382–395, 2022, doi: 10.1093/comjnl/bxaa053.
- [21] A. Belgacem, K. Beghdad-Bey, H. Nacer, and S. Bouznad, "Efficient dynamic resource allocation method for cloud computing environment," *Cluster Computing*, vol. 23, no. 4, pp. 2871–2889, 2020, doi: 10.1007/s10586-020-03053-x.
- [22] M. Kumar, S. C. Sharma, S. Goel, S. K. Mishra, and A. Husain, "Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm," *Neural Computing and Applications*, vol. 32, no. 24, pp. 18285–18303, 2020, doi: 10.1007/s00521-020-04955-y.
- [23] M. Nanjappan, G. Natesan, and P. Krishnadoss, "An Adaptive Neuro-Fuzzy Inference System and Black Widow Optimization Approach for Optimal Resource Utilization and Task Scheduling in a Cloud Environment," *Wireless Personal Communications*, vol. 121, no. 3, pp. 1891–1916, 2021, doi: 10.1007/s11277-021-08744-1.
- [24] M. H. N. -Shahraki and H. Zamani, "DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization," *Expert Systems with Applications*, vol. 198, p. 116895, 2022, doi: 10.1016/j.eswa.2022.116895.
- [25] M. Sardaraz and M. Tahir, "A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 8, p. 1550147720949142, 2020, doi: 10.1177/1550147720949142.
- [26] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing," *Journal of Cloud Computing*, vol. 10, no. 1, 2021, doi: 10.1186/s13677-020-00219-1.
- [27] C. Chandrashekar, P. Krishnadoss, V. K. Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing," *Applied Sciences (Switzerland)*, vol. 13, no. 6, p. 3433, 2023, doi: 10.3390/app13063433.
- [28] H. Kumar and I. Tyagi, "Task allocation model based on hierarchical clustering and impact of different distance measures on the performance," *International Journal of Fuzzy System Applications*, vol. 9, no. 4, pp. 105–133, 2020, doi: 10.4018/IJFSA.2020100105.
- [29] H. Kumar and I. Tyagi, "Hybrid model for tasks scheduling in distributed real time system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2881–2903, 2021, doi: 10.1007/s12652-020-02445-6.
- [30] Karishma and H. Kumar, "A new hybrid particle swarm optimization algorithm for optimal tasks scheduling in distributed computing system," *Intelligent Systems with Applications*, vol. 18, p. 200219, 2023, doi: 10.1016/j.iswa.2023.200219.
- [31] L. Qu, W. He, J. Li, H. Zhang, C. Yang, and B. Xie, "Explicit and size-adaptive PSO-based feature selection for classification," *Swarm and Evolutionary Computation*, vol. 77, p. 101249, Mar. 2023, doi: 10.1016/j.swevo.2023.101249.
- [32] A. Anjum and A. Parveen, "Optimized load balancing mechanism in parallel computing for workflow in cloud computing environment," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 12, no. 2, p. 276, Jul. 2023, doi: 10.11591/ijres.v12.i2.pp276-286.
- [33] A. R. Shaheen and S. S. Kumar, "Tasks Scheduling in Cloud Environment Using PSO-BATS with MLRHE," *Intelligent Automation and Soft Computing*, vol. 35, no. 3, pp. 2963–2978, 2023, doi: 10.32604/iasc.2023.025780.

BIOGRAPHIES OF AUTHORS






Nihar Ranjan Sabat    received a Bachelor of Technology degree in Computer Science and Engineering from Biju Patnaik University of Technology (BPUT) located in Odisha, India in 2008. In 2011, he obtained a Master of Technology degree in Computer Science and Engineering from the International Institute of Information Technology (IIIT) located in Bhubaneswar, India. Currently, he is engaged in pursuing his doctoral study within the Faculty of Engineering at Biju Patnaik University of Technology (BPUT) in Rourkela, Odisha, India. The primary focus of his continuing research involves investigating strategies for task scheduling and load balancing in the context of cloud computing environments. He can be contacted at email: n.ranjan9@gmail.com.






Rashmi Ranjan Sahoo    received a Bachelor of Technology degree in Computer Science and Engineering from Biju Patnaik University of Technology, situated in Odisha, India, in the year 2005. He graduated from Jadavpur University in Kolkata, India, in 2012 with a master's degree in Electronics and Telecommunication Engineering with a specialization in Computer Engineering. In 2021, he graduated from Jadavpur University in Kolkata, India, with a Doctor of Philosophy in Engineering with a specialization in Ad-hoc Network Security. He is currently employed as an assistant professor in the Department of Computer Science and Engineering and serves as the head of the Center of Excellence on Cyber Security and Cloud Computing at Parala Maharaja Engineering College, which is affiliated to Biju Patnaik University of Technology, Rourkela, India. With a specific focus on wireless sensor networks, vehicular networks, and machine learning, he published more than 20 scholarly articles in the discipline of computer science and engineering. He can be contacted at email: rashmiranjan.cse@pmec.ac.in.



Manas Ranjan Pradhan    (Member, IEEE) received the M. Tech. and Ph.D. degrees in Computer Science from Utkal University and the University of Mysore, respectively, in India. He has extensive expertise in academic management, research, and teaching both nationally and internationally. He is now working with the United Arab Emirates' Skyline University College in Sharjah. He held two positions as an academic leader: Dean of the Faculty of IT and Science at INTI International University in Malaysia, and Head of the Programme at the University of Petroleum and Energy Studies (UPES), India. Through industry-academic collaboration, internships, placements, and workshops, he has been involved in the IT industry. Under Laureate International Universities, USA, he oversaw the IBM Centre of Education for Cloud Computing and Business Analytics at INTI International University. His numerous research works have been published in journals and presented at conferences. Three Australian and three Indian patents are among his accomplishments. His areas of interest in research include business process modelling, artificial intelligence, machine learning, retail/e-commerce analytics, data mining, data warehouses, and business analytics. The Confederation of Indian Industry (CII) has awarded him the Mentor Award for the i-Talent Project Contest. Three international conferences, including NGCT-2015 (UPES), ICQMOIT-2008 (ICFAI, India), and ICD-2019 (SUC, United Arab Emirates), were all organised in large part thanks to his contributions. He can be contacted at email: manaspradhan@yahoo.com.



Biswaranjan Acharya    (Senior Member, IEEE) received an M.C.A from the Indira Gandhi National Open University (IGNOU) in New Delhi, India, in 2009. In 2012, he received an M. Tech in Computer Science and Engineering from BPUT in Rourkela, Odisha, India. He is pursuing a Ph. D in computer application at VSSUT in Burla, Odisha. He is an assistant professor at Marwadi University's Computer Engineering-Artificial Intelligence and Big Data Analytics Department. His eleven years of experience include software development and academia at several well-known colleges like Ravenshaw University. He edited seven books, co-authored 60 research articles in renowned journals, and reviewed several peer-reviewed journals. In addition, he has 72 co-invented IPRs. Data analytics, optimization, computer vision, machine learning, and IoT are his research interests. He is a member of several scientific and educational organisations, including ISC, IEEE, CSI, IACSIT, and IAENG. He can be contacted at email: biswaacharya@ieee.org.