IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 October 2023, accepted 27 October 2023, date of publication 1 November 2023, date of current version 7 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3329052

RESEARCH ARTICLE

Enhancing Cloud Task Scheduling With a Robust Security Approach and Optimized Hybrid POA

S. V. ASWIN KUMER[1], N. PRABAKARAN[1], E. MOHAN[2], BALAJI NATARAJAN[3], G. SAMBASIVAM[4], (Member, IEEE), AND VAIBHAV BHUSHAN TYAGI[5]

[1]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Andhra Pradesh 522302, India
[2]Department of ECE, Saveetha School of Engineering, SIMATS, Chennai, Tamil Nadu 602105, India
[3]Department of Computer Science and Engineering, Sri Venkateshwaraa College of Engineering and Technology, Ariyur, Puducherry 605102, India
[4]School of Computing and Data Science, Xiamen University Malaysia, Sepang, Selangor Darul Ehsan 43900, Malaysia
[5]Faculty of Engineering, ISBAT University, Kampala, Uganda

Corresponding author: Vaibhav Bhushan Tyagi (tyagi.fict@isbatuniversity.com)

ABSTRACT Dynamic and flexible computing resources are offered by cloud computing (CC), which has gained popularity as a computing technology. Efficient task scheduling (TS) plays a critical role in CC by optimizing the distribution of tasks across available resources to achieve maximum performance. The allocation of computational tasks in a cloud environment is a complicated process that is affected by multiple factors, such as available network bandwidth, make span, and cost considerations. Therefore, it is crucial to optimize available bandwidth for efficient TS in CC. In the present research, a novel pelican-based approach is introduced to optimize TS in the CC environment. The newly developed method also utilizes a security approach called Polymorphic Advanced Encryption Standard (P-AES) to encode cloud information during scheduling. The study evaluates the proposed algorithm's performance in terms of the make span, resource utilization, cost, response time, throughput, latency, execution time, speed, and bandwidth utilization. The simulation is carried out using the Python tool, and it effectively handles a wide range of tasks from 1000 to 5000. The proposed algorithm offers a new perspective on utilizing pelican algorithms to optimize task scheduling in CC. The hybrid optimization enables the proposed algorithm to provide efficient task scheduling by exploiting the strengths of entire algorithms. The proposed approach offers an innovative solution to the challenges of scheduling tasks in cloud environments and provides a more effective and secure way of optimizing cloud services. Overall, this study provides valuable insights into task scheduling optimization in CC and offers an effective approach for enhancing the performance of CC services.

INDEX TERMS Advanced encryption standard, chameleon swarm algorithm, cloud computing, hybrid model, security, moth swarm algorithm, task scheduling.

I. INTRODUCTION

In the domain of cloud computing (CC), a recent revolution has brought forth numerous advantages over traditional distributed computing. This multifunctional and highly efficient computing system leverages large-scale resources to ensure the effectiveness of cloud services [1], [2]. A fundamental aspect of CC is the process of scheduling and distributing tasks across various computing resources, which is vital for providing customized computing, information, and storage services to users [3], [4]. Efficient task scheduling is paramount for successful task execution within cloud environments, as it impacts performance and resource utilization. Therefore, optimizing task scheduling mechanisms can significantly enhance the overall performance of CC services [5], [6]. Task scheduling involves allocating CC resources based on specific optimization objectives to achieve timely task completion within a cost-effective framework while

The associate editor coordinating the review of this manuscript and approving it for publication was Yogendra Kumar Prajapati.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

meeting Quality of Service (QoS) requirements [7], [8], [9]. Virtual machines (VMs) play a crucial role in task processing, and the selection of the appropriate VM is a central task of scheduling algorithms. Researchers primarily focus on reducing make span and task execution time when designing such algorithms [10], [11]. However, the dynamic nature of cloud environments requires dynamic workload distribution among VMs, posing challenges and trade-offs for service providers. While dynamic scheduling schemes enhance efficiency, they also introduce communication overhead and complexity [12], [13], [14], [15]. Various techniques have been employed for bandwidth optimization in task scheduling to balance competing demands such as data size, processing requirements, security needs, and network resource availability [16], [17]. Despite these efforts, effectively integrating bandwidth optimization with task scheduling remains a complex issue. In recent years, cloud computing has transformed modern business operations, offering scalability, cost-efficiency, and agility [18], [19], [20]. Task scheduling stands at the core of this revolution, orchestrating the allocation of computational tasks across diverse cloud resources. The success of cloud services depends on efficient and secure task scheduling, which is challenging due to the dynamic nature of cloud environments and evolving security threats [21], [22]. To address these challenges, this research introduces a novel approach that combines the power of a hybrid Pelican Optimization Algorithm (POA) with the security of the Polymorphic Advanced Encryption Standard (P-AES). This integrated approach aims to optimize task scheduling while ensuring data security. The research evaluates the proposed method using various performance metrics and extensive simulations, demonstrating its potential to enhance both efficiency and security in cloud computing [23], [24], [25]. This work aims to contribute to the ongoing evolution of cloud computing, making it more efficient, secure, and responsive to the demands of the digital age. Sections of this paper provide an in-depth exploration of the proposed approach, methodology, evaluation, and implications, offering a comprehensive perspective on the intersection of efficiency and security in cloud task scheduling.

The subsequent sections of this research paper delve into a comprehensive exploration of the proposed approach, its methodology, evaluation, and implications. Section II conducts a thorough review of relevant works in the field of cloud task scheduling, providing the context for our research. Section III articulates the problem formulation and significance, laying the foundation for our innovative approach. In Section IV, we present the proposed methodology, unveiling the fusion of the hybrid POA model and P-AES security. The outcomes of our simulations and a detailed performance assessment are elucidated in Section V. Finally, Section VI offers concluding remarks, summarizing the key contributions of this research and outlining avenues for further investigation in the dynamic realm of cloud computing.

Intrigued by the potential of this research, we invite readers to embark on a journey through the intricate landscape of cloud task scheduling, where efficiency and security converge to shape the future of cloud computing services.

A. MOTIVATION AND CONTRIBUTION

As cloud computing becomes increasingly prevalent, there is a growing necessity for assigning tasks. The optimization of TS in large-scale is driven by the objective of enhancing task scheduling efficiency, security, and reliability. The optimization of bandwidth and security in the scheduling process ensures that the resources are used effectively, and that the system is protected from potential security threats. Summarizing the key highlights of this paper:

- A novel adaptive approach for optimal task transfer in CC is proposed to minimize transferring of tasks time across available resources.
- A hybrid model-based scheduling framework is designed to achieve efficient task scheduling in CC while ensuring data security.
- Our work introduces a novel scheduling model to enhance the efficiency of task scheduling while ensuring data security in the CC model. The framework is based on hybrid POA model.
- The scheduling of tasks in the cloud incorporates P-AES for ensuring data security.
- A scheduling algorithm that uses a hybrid meta-heuristic technique with low complexity has been developed and shows significant improvements in make span, cost, degree of imbalance, resource utilization, average waiting time, response time, throughput, latency, execution time, speed, bandwidth utilization.

The next portions of this research report are organized as follows: A review of relevant works on task scheduling in CC is provided in Section II. The issue formulation and statement are provided in Section III, and the suggested method for work scheduling optimization is introduced in Section IV. This method integrates the P-AES security technique with a hybrid POA algorithm. Section V discusses the simulation outcomes and performance assessment of the suggested strategy. Section VI concludes by summarizing the article and outlining potential directions for further investigation.

II. RELATED WORK

Efficient TS is a significant challenge in the realm of CC since it greatly affects system performance. Improving the task scheduling process is necessary by developing an efficient algorithm. In the following section, we delve into some of the most recent task-scheduling methods that are currently available.

Nabi et al. [26] presented a task scheduling approach, which enhances PSO model performance concerning throughput, ARUR, and make span. By integrating global and local search more effectively, the LDAIW strategy offers improved performance. However, the presented approach

may require substantial computational resources, limiting its feasibility in resource-constrained environments.

Zubair et al. [27] introduced a modified method related to symbiotic organisms' search (SOS) for mapping heterogeneous tasks onto cloud resources with varying capacities. The new approach streamlines the mutualism process of the algorithm by using equity as an indicator of the species' relationship effectiveness in the ecosystem, enabling them to progress to the subsequent generation. The approach utilizes a geometric mean instead of the initial mutual vector to improve the persistence benefits of two separate species. However, the presented approach facilitates the effective allocation of heterogeneous tasks to utilize cloud resources with varying capacities.

Gupta et al. [28] presented modifications to the HEFT algorithm for rank generation and processor selection phases. These modifications enhance the accuracy of rank calculation and optimize the selection of available processor slots for task scheduling. These enhanced versions of the HEFT algorithm were presented to reduce the makespan of a given task submission on VMs while adhering to user-specified financial constraints. However, the presented model may increase the complexity of the algorithm and require significant computational resources for implementation, which may not be feasible for use in resource-constrained environments or for large-scale workflows.

Amer et al. [29] developed an ELHHO, a modified version of the HHO, to tackle the problem of multi-objective scheduling. This replaces the use of a random solution for achieving a determined initial solution. However, the presented approach is designed specifically for multi-objective scheduling problems, and it may not be optimal for single-objective problems or scenarios where other metrics such as response time or throughput are more crucial.

A research investigation was performed by Alboaneen et al. [30] to address the co-optimization problem of VM placement and TS. MOA was utilized in this work for scheduling independent tasks for VMs and placing VMs on physical hosts (PHs). The study measured various parameters such as resource utilization, DOI, makespan, and execution cost. Even though the integration co-optimization approach may lead to better results, it could also increase the computational complexity of the problem, limiting its feasibility for use in resource-constrained environments.

Albert et al. [31] present a cloud task scheduling solution using a hybrid Whale Harmony optimization algorithm. A small control parameter is used to stimulate the implementation of the WOA, which is inspired by the bubble-net feeding method. To ensure a high convergence rate, the fitness function is formulated by combining load imbalance, cost, energy consumption, resource utilization, and computation time. However, the use of a hybrid Whale Harmony optimization algorithm may increase the algorithm's complexity and result in longer processing times.

Alsadie et al. [32] presented the metaheuristic framework called MDVMA utilizing the NSGA-II algorithm to address the problem of dynamic VM allocation with optimal scheduling of tasks in the cloud. This framework enhances multiple objectives, such as make span, to choose from as per their requirements. However, using these algorithms for task scheduling may pose limitations, as it could require significant computational resources to find optimal or near-optimal solutions.

To prevent premature convergence and increase the performance of the PSO algorithm, Agarwal et al. [33] presented a task scheduling approach employing PSO. The proposed method was contrasted with well-known task scheduling techniques such as mPSO, GA, minimum completion time, minimum execution time, and max-min. However, the PSO algorithm may encounter difficulties when dealing with larger and more intricate task sets, leading to potential inefficiencies. When the task count increases, the scalability of the PSO algorithm may become an issue, making it less suitable for large-scale task scheduling.

Wei et al. [34] developed an improved ACO to avoid local optimization in scheduling tasks. To optimize the overall performance, the algorithm utilizes reward and punishment coefficients for optimizing pheromone updating rules. Additionally, it uses dynamic updates of the volatility coefficient to enhance the optimization process. Additionally, the VM load weight coefficient is introduced to ensure load balance. Moreover, the effectiveness of the algorithm may be impacted by task diversity, resource requirements, and cloud environment complexity.

A task scheduling algorithm and conceptual model for dynamic resource allocation via task migration in VMs are presented by Ramasamy et al. [35]. Feature extraction is performed on the user's task demands, followed by feature reduction using the Modified PCA algorithm, to decrease processing time. Hybrid PSO and Modified GA are utilized for resource allocation after combining both the user task and cloud server features. Finally, the task that has been optimized is scheduled to a specific VM to allocate the resources. The presented algorithm may not be flexible enough to handle dynamic changes in resource demands and server failures.

Bebortta et al. [36] developed a dynamic integer linear programming approach for optimum task offloading to solve the challenges of power consumption and latency in IoT-Fog applications. To improve fog resource energy usage and service delay, it represents the offloading challenge as an integer linear programming problem. This paper is important since it focuses on job offloading in IoT-Fog systems, which is analogous to cloud task scheduling. The optimization approaches investigated can be used to develop strategies for effective task allocation.

Mohapatra et al. [37] used Genetic Algorithms and the Eagle Strategy algorithm to balance exploration and exploitation in order to suggest cloud services with the best Quality of Service (QoS). It covers the issue of choosing cloud

services that provide the appropriate QoS. The study's emphasis on QoS optimization corresponds to the relevance of performance measures in cloud job scheduling and resource allocation, making it applicable to the current framework.

Kruekaew and Kimpan [38] applied a Multi-Objective task scheduling optimization technique based on the Artificial Bee Colony Algorithm with Q-learning to concentrate an independent task scheduling strategy in cloud computing. It was designed to improve scheduling, resource usage, and load balancing. Load balancing is an important part of cloud job scheduling; hence this research is directly applicable. The reinforcement learning component may reveal information about dynamic resource allocation.

Saif et al. [39] developed the Multi-Objectives Grey Wolf Optimizer (MGWO) algorithm for minimizing latency and energy usage in cloud-fog computing. It solves the problem of distributing jobs efficiently across fog and cloud resources. The emphasis on task scheduling in cloud-fog computing settings is relevant to the framework, where effective resource allocation and job offloading are crucial.

Mangalampalli et al. [40] addressed task planning in datacenters using the Cat Swarm Optimization method, taking into account variables such as make span, migration time, energy usage, and overall power cost. The study's emphasis on task scheduling efficiency and cost optimization corresponds with the framework's aims of effective cloud computing job scheduling.

Kumar et al. [41] provide a system for evaluating cloud services based on subjective and objective assessments, solving the problem of optimum cloud service selection. Because cloud service selection is a major part of cloud resource allocation, this study is extremely relevant to the framework's aims.

Khan and Santhosh [2] presented a task scheduling technique based on a hybrid optimization algorithm that prioritizes waiting time and other performance parameters. Because task scheduling is important to cloud resource allocation, this research is directly relevant to the framework's goal of efficient task scheduling.

Thus, the examined research gives insights into many areas of cloud resource allocation, task scheduling, and optimization, making them relevant references for the current framework's aims of improving task scheduling efficiency in cloud computing.

III. PROBLEM STATEMENT AND FORMULATION

Cloud computing infrastructure comprises physical servers that are used for hosting VMs, which in turn are used for executing user tasks. The scheduling types in CC are the selection of server and task distribution. Optimizing task scheduling aims to balance various performance metrics, including bandwidth utilization, latency, make span, execution time, and cost. The objective function of optimization problems depends on the application and organization goals, such as maximizing data transfer, minimizing task completion time, make span, or total service cost. Constraints include available bandwidth, storage, processing capacity, budget, data security, SLAs, and other performance metrics. Mathematical expressions for constraints and the objective function vary by the cloud computing system and organization goals. Task scheduling algorithms aim to allocate VMs according to user service requirements and VM status. However, existing algorithms often suffer from premature convergence, resulting in suboptimal solutions that undermine QoS guarantees. Therefore, there is a need for efficient and adaptable algorithms that can compute the most optimal global solution for scheduling tasks in CC environments.

Eqn. 1 depicts the cloud system (CS) with each machine comprising VMs.

$$CS = [PM_1, PM_2, \ldots, PM_i, \ldots, PM_{Npm}] \tag{1}$$

The cloud's PMs performance is expressed as,

$$PM = [VM_1, VM_2, \ldots, VM_k, \ldots, VM_{Nvm}] \tag{2}$$

The VM_k characteristic is calculated as follows:

$$VM = [SIDV_k, mips_k] \tag{3}$$

$$T = [Task_1, Task_1, \ldots, Task_i, \ldots, Task_{tsk}] \tag{4}$$

$$Task_i = [SIDT_i, Task - length_i, ECT_i, LI_i] \tag{5}$$

Matrix (6) defines the Expected Complete Time (ECT) metric with a size of $N_{tsk} \times N_{vm}$, which represents the length of time needed to complete a job on each computing resource (VM).

$$ECT = \begin{bmatrix} ECT_{1,1} & ECT_{1,2} & ECT_{1,3} \cdots & ECT_{1,N_{vm}} \\ ECT_{2,1} & ECT_{2,2} & ECT_{2,3} \cdots & ECT_{2,N_{vm}} \\ \cdots & \cdots & \cdots \cdots & \cdots \\ \cdots & \cdots & \cdots \cdots & \cdots \\ ECT_{N_{tsk}}, 1 & ECT_{N_{tsk},}2 & \cdots \cdots & ECT_{N_{tsk},N_{vm}} \end{bmatrix} \tag{6}$$

An effective algorithm is required to schedule tasks in CC systems, which can optimize several objectives while ensuring security. Based on the following considerations, the mathematical expression for the efficient task scheduling problem formulation is derived:

A. OBJECTIVE FUNCTION

Minimize the maximum execution time: The fundamental objective is to minimize the longest execution time of each activity. Therefore, the objective function is given by

$$\text{minimize}(\max(\text{ET}_i)) \tag{7}$$

Maximize the throughput: The aim is to optimize the task completion rate within a specified time frame, by maximizing the number of tasks that can be finished. Therefore, the objective function is expressed as:

$$\text{maximize}\left(\frac{n}{\max(\text{ET}_i)}\right) \tag{8}$$

Minimize the average execution time: The main goal is to reduce the meantime required for all tasks to finish their execution. Therefore, the objective function is expressed as:

$$\text{minimize}\left(\frac{\text{sum}(\text{ET}_i)}{n}\right) \tag{9}$$

Maximize the total bandwidth usage: The aim is to optimize the total data transfer between tasks, to maximize it. Therefore, the objective function is given by:

$$\text{maximize}(\text{sum}(\text{BW}_i)) \tag{10}$$

where BW_i is the bandwidth requirement of task i.

Minimize the total cost: Aiming to reduce the overall cost of resources utilized in accomplishing the tasks. Therefore, the objective function is given by:

$$\text{minimize}(\text{sum}(\text{Cost})) \tag{11}$$

where Cost is the cost of resources used to complete the tasks.

Minimize the total execution time: The major objective is to shorten the total amount of time needed to complete all the jobs. Therefore, the objective function is given by:

$$\text{minimize}(\text{sum}(\text{ET}_i)) \tag{12}$$

B. SECURITY CONSTRAINTS

Confidentiality constraint: All data must be encrypted during transmission and storage.

Integrity constraint: Data must not be modified during transmission or storage.

Availability constraint: The cloud system must be always available for task scheduling.

C. RESOURCE AVAILABILITY CONSTRAINTS

1) CPU CONSTRAINTS

The CPU constraint ensures that the total CPU usage does not exceed the available CPU capacity. The sum of CPU usage of all tasks, calculated as the product of the start and end time of each task and its CPU requirement, should be less than or equal to the total CPU capacity available.

$$\text{sum}(\text{ST}_i\text{CPU}_i) <= \text{T}_{\text{cpu}} \tag{13}$$

2) MEMORY CONSTRAINT

The Memory constraint ensures that the total memory usage does not exceed the available memory capacity. The sum of memory usage of all tasks, calculated as the product of the start and end time of each task and its memory requirement, should be less than or equal to the total memory capacity available.

$$\text{sum}(\text{ST}_i Mem_i) <= \text{T}_{\text{mem}} \tag{14}$$

3) STORAGE CONSTRAINT

The Storage constraint ensures that the total storage usage does not exceed the available storage capacity. The sum of storage usage of all tasks, calculated as the product of the start and end time of each task and its storage requirement, should be less than or equal to the total storage capacity available.

$$\text{sum}(\text{ST}_i * \text{Storage}_i) <= \text{T}_{\text{storage}} \tag{15}$$

4) NETWORK BANDWIDTH CONSTRAINT

The Network bandwidth constraint ensures that the total bandwidth usage does not exceed the available bandwidth capacity. The sum of the bandwidth requirement of all tasks, divided by the execution time to the total bandwidth capacity available.

$$\text{sum}\left(\frac{\text{BW}_i}{\text{ET}_i}\right) <= T_{bw} \tag{16}$$

5) DEADLINE CONSTRAINT

The Deadline constraint ensures that all tasks are completed before their respective deadlines. This constraint is formulated as ET_i being less than or equal to DD_i.

$$\text{ET}_i <= \text{DD}_i - \text{ET}_i - \text{DD}_i <= 0 \text{ for all i.} \tag{17}$$

6) MAKE SPAN CONSTRAINT

The Make span constraint ensures that the maximum time taken by any task to complete execution does not exceed T_{max}. This constraint is formulated as $Max\ (ET_i)$ being less than or equal to T_{max}.

$$\text{Max}(\text{ET}_i) <= \text{T}_{\text{max}} \tag{18}$$

7) THROUGHPUT CONSTRAINT

Ensuring that the amount of tasks completed within a specific duration is equal to or greater than a predefined value Tp is the purpose of the throughput constraint. This constraint is formulated as n divided by the maximum execution time of all tasks being greater than or equal to T_p.

$$\frac{n}{T_{\text{max}}} >= T_p \tag{19}$$

8) LATENCY CONSTRAINT

The constraint on Latency guarantees that the mean duration for all task completions is not greater than a specific value Tl. This constraint is formulated as the total execution times of all tasks divided by the number of tasks being less than or equal to T_l.

$$\frac{sum(ET_i)}{n} <= T_l \tag{20}$$

9) BANDWIDTH CONSTRAINT

The Bandwidth constraint ensures that the total amount of data transferred between tasks is greater than or equal to a certain value T_b. This constraint is formulated as the sum of the bandwidth requirements of all tasks being greater than or equal to T_b.

$$\text{sum}(\text{BW}_\text{i}) >= \text{T}_\text{b} \tag{21}$$

10) COST CONSTRAINT

The Cost constraint ensures that the total cost of resources used to complete the tasks is less than or equal to a certain value T_c. This constraint is formulated as the total cost of resources used by all tasks being less than or equal to Tc.

$$\text{sum(Cost)} <= T_c \tag{22}$$

where T_{cpu}, T_{mem}, $T_{storage}$, T_{bw}, T_{max}, T_p, T_l, T_b, and T_c are the available resources, maximum execution time, minimum throughput, maximum latency, minimum bandwidth, and maximum cost, respectively. All of the above constraints are represented as inequalities that must be satisfied by the variables.

In this approach, the weighted sum method is utilized, where each metric is given a weight, and the objective function can then be represented as:

$$\begin{aligned}\text{minimize} & w_1 \text{Max}(ET_i) + w_2 n/T_{max} + w_3 \text{sum}(ET_i)/n \\ & + w_4 \text{sum}(BW_i) + w_5 \text{sum(Cost)} + w_6 \text{sum}(ET_i)\end{aligned} \tag{23}$$

where w, indicates the weights assigned to each metric.

In the above objective function, the first term represents the make span, the second term represents the throughput, the third term represents the latency, the fourth term represents the bandwidth, the fifth term represents the cost, and the sixth term represents the execution time. The optimization problem seeks to determine the start and end times that optimize each task's execution, meeting all constraints and minimizing the overall execution time. The optimization problem is solved using the Hybrid Moth Swarm and Chameleon Swarm algorithm.

IV. PROPOSED SCHEDULING OF TASKS IN THE CLOUDB2N/

In this study, we focus on the optimized transfer of tasks to the cloud and optimal scheduling of the task to VMs with security using P-AES [36]. Cloud task scheduling aims to allocate computing resources efficiently to execute user tasks. Cloud data centers use several physical servers, each running multiple virtual machines, to provide computing services to various applications. Task scheduling allocates and executes user tasks on VMs based on processing capability and computing resource cost. Multiple objectives can be taken into account to enhance task scheduling performance in the cloud data center, including resource utilization, execution time, degree of imbalance, bandwidth utilization, speed, latency, make span, and throughput. In this regard, we propose a hybrid POA to optimize the CC environment's task scheduling process. The algorithm can allocate virtual machines (VMs) to user-submitted tasks according to their needs, while also achieving multiple objectives of the cloud data center. These objectives may include maximizing bandwidth utilization and throughput, and minimizing make span, execution time, bandwidth, and cost. The planned work's architecture is seen in Figure 1.

According to our findings, P-AES adds a significant layer of protection by dynamically changing encryption keys and

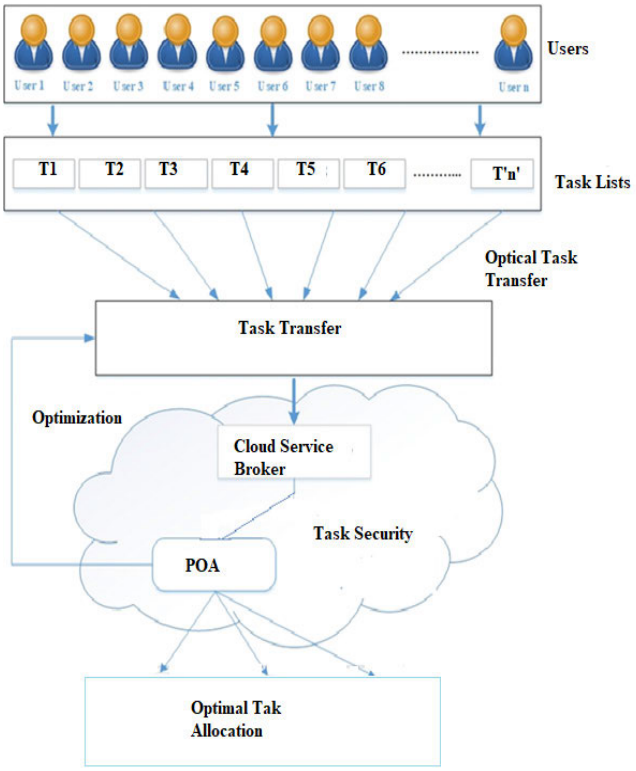

FIGURE 1. The architecture of the proposed methodology in task scheduling.

algorithms, making it extremely difficult for potential attackers to compromise scheduling data. This powerful security feature improves the confidentiality and integrity of cloud task scheduling, maintaining the privacy of sensitive data and protecting against increasing cyber threats, which exactly aligns with the aims of our research.

We implement this algorithm in AWS cloud, which is one of the leading cloud service providers. To ensure the security of the task data during transfer and processing, we use Polymorphic AES, which is a highly secure encryption algorithm. The proposed algorithm will take into account various factors including bandwidth usage, latency, make span, execution time, cost, and more.

The combination of the Pelican algorithm and P-AES in the present research is controlled via a well-defined method that assures both fast job scheduling and strong security. To begin with, the Pelican method is used to improve work scheduling in a cloud computing context. To optimally distribute computing workloads to available resources, it considers numerous performance parameters such as network bandwidth, make span, and cost concerns. Following the improvement of the scheduling, P-AES is added as a critical security layer. It encrypts the scheduling information dynamically, using a polymorphic technique that changes encryption keys and algorithms on a regular basis. This dynamic encryption technique maintains the scheduling data's confidentiality and integrity, making it highly resistant to illegal access and

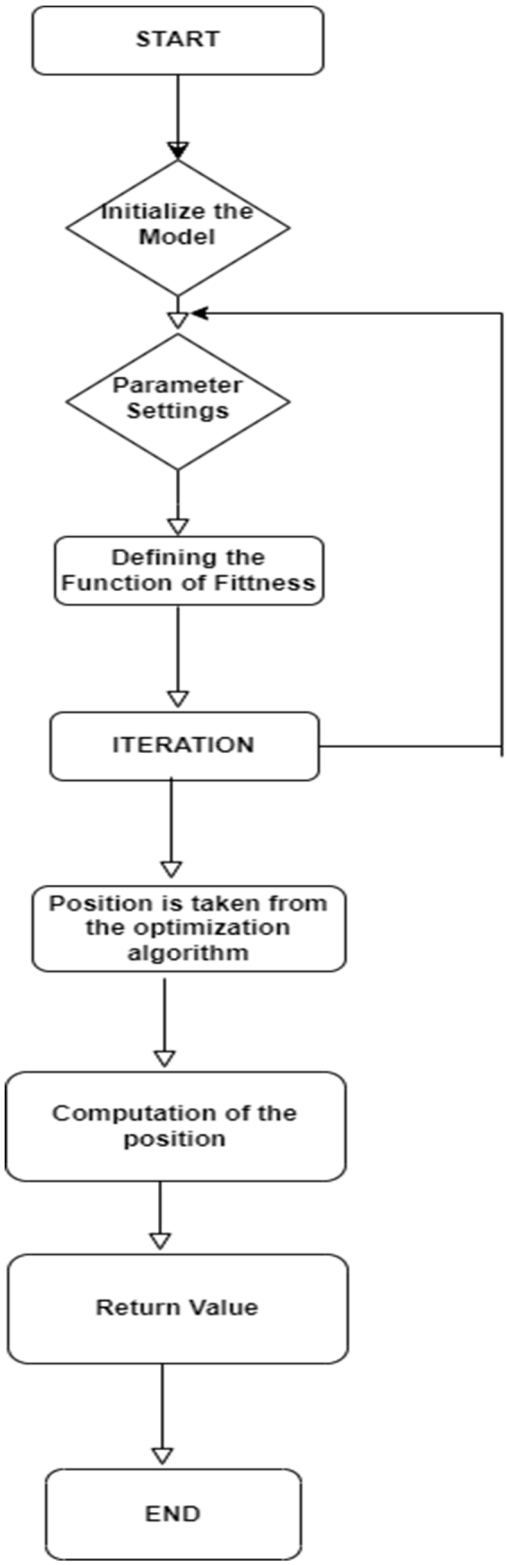

FIGURE 2. Flow chart of the proposed model.

cyber-attacks. We may strike a delicate balance between job scheduling efficiency and data security by integrating these strategies sequentially. It improves resource allocation while simultaneously protecting important scheduling data, providing a comprehensive solution geared to the particular problems of cloud computing settings.

A. SECURITY STRATEGY

Ensuring security is a critical aspect when dealing with cloud mechanisms and scheduling procedures. Attacks such as data tampering and information interception can occur during scheduling, making security a fundamental concern. To address this issue, the P-AES technique is employed to provide security to workflow scheduling. P-AES operates in 128 distinct ways, which makes it challenging for attackers to obtain the encryption key and decipher the cipher's precise structure. P-AES determines the operation specifics during runtime for communicating parties, employing a subset of the key bits.

1) POLYMORPHIC ADVANCES ENCRYPTION STANDARD (P-AES)

The proposed cipher differs from the conventional AES in terms of its operation. The proposed cipher processes a single 16-byte block at a time, unlike AES which divides the input into 16-byte blocks. AES utilizes a padding scheme like PKCS 7 when the input size is less than 16 bytes. However, the proposed cipher does not require such padding. A specific method is employed to copy the input block into the state matrix. To simplify, the state content, that will be presented in hexadecimal format.

The P-AES is capable of accommodating keys that are 16, 24, or 32 bytes long. However, to maintain consistency throughout the discussion, we assume that the key length is 32 bytes.

$$key = [byte_0\,|byte_1|\,byte_2|\ldots byte_{28}\,|byte_{29}|\,byte_{30}\,|byte_{31}|] \tag{24}$$

The sender and receiver sides calculate the following values after generating the key:

$$column\_index = (< int > byte_{29}) \bmod 4 \tag{25}$$

$$row\_shifting\_index = (< int > byte_{30}) \bmod 4 \tag{26}$$

$$bytes\_substitution\_index = (< int > byte_{31}) \bmod 8 \tag{27}$$

2) SUB BYTES STAGE OF P-AES MODIFICATION

A 7-byte substitution index moves each byte's bits in a circular motion to the left in the state entries of the P-AES Modified Sub Bytes stage. Next, the S-Box is used. The state values are taken out of the inverted S-Box during decryption and then relocated to the right by the 7-byte substitution index.

3) STAGE OF P-AES MODIFIED SHIFT ROWS

Equation (26), which extracts the row_shifting_index value of 3 from the key, may be used to ascertain the P-AES Modified Shift Rows stage. In this scenario, the Modified Shift Rows process will perform the following:

- Row 3 remains unchanged.
- Row 2 is shifted leftward by three bytes.
- Row 1 is shifted leftward by two bytes.
- Row 0 is shifted leftward by one byte.

4) AES MODIFIED MIX COLUMNS STAGE

According to the integer of the column_mixing_index, the rows of the Modified Mix Columns matrix in the proposed P-AES are dynamically rearranged while the data is being encrypted. The order of the rows in the Modified Mix Columns matrix is determined at runtime decrypt, the Modified Mix Columns matrix will be reversed in the same way.

5) P-AES FULL OPERATION

Polymorphic cipher is designed to maintain the robustness of AES. The number of rounds assumed in the algorithm is 14, corresponding to a key length of 32 bytes. Depending on how long the key is that is being used, there may be different numbers of rounds.

B. HYBRID PELICAN OPTIMIZATION ALGORITHM (POA)

The hybrid Pelican Optimization Algorithm (POA) is a variant of the original Pelican Optimization Algorithm that incorporates elements from other optimization techniques or algorithms. It combines the benefits and abilities of many algorithms to improve its efficiency and performance while dealing with challenging optimisation issues, particularly those pertaining to job scheduling in a cloud computing environment. The hybridization of the POA entails combining it with other optimisation techniques, such simulated annealing, genetic algorithms, and particle swarm optimisation. By doing this, the hybrid POA overcomes the drawbacks of individual techniques and produces superior optimisation outcomes by combining the exploration and exploitation capabilities of many algorithms. The hybrid POA takes advantage of the diverse search strategies and characteristics of the combined algorithms. It leverages their complementary nature to balance the trade-off between exploration and exploitation. This hybridization allows the algorithm to converge to high-quality solutions efficiently while maintaining a good level of exploration to avoid getting stuck in local optima. In the context of cloud task scheduling, the hybrid POA can leverage its enhanced optimization capabilities to distribute tasks optimally across available cloud resources. By considering factors like resource utilization, make span, and cost considerations, the hybrid POA aims to achieve better task allocation and scheduling decisions. Thus, the hybrid POA represents an innovative approach that harnesses the power of combining different optimization algorithms.

1) INTEGRATION OF POA AND P-AES

The Pelican method is largely utilized in cloud computing settings to optimize work scheduling. It is intended to distribute computing jobs to available resources efficiently by taking into account parameters such as network bandwidth, make span, and cost. P-AES, on the other hand, is used to boost the security of the scheduling process. It encrypts the cloud data involved in the schedule, ensuring that sensitive data is kept private and secure from unwanted access.

The combination of these two techniques occurs sequentially in our suggested strategy. First, the Pelican algorithm optimizes work scheduling depending on the performance requirements supplied. After determining the schedule, P-AES is used to encrypt the scheduling information before it is transferred or stored in the cloud environment. This step guarantees that the scheduling information is kept private, preventing any potential security breaches.

2) BENEFITS OF P-AES IN CLOUD INFORMATION ENCRYPTION

P-AES provides numerous distinct advantages for cloud information encryption while scheduling. P-AES utilizes a polymorphic encryption approach, which means that the encryption keys and algorithms utilized are continually changed. This dynamic encryption technology dramatically improves security by making deciphering encrypted data exceedingly challenging for adversaries. We use P-AES to ensure that even if a breach occurs, the scheduling information is unreadable to unauthorized parties. This protection is critical in cloud computing environments for securing sensitive data. Many companies and associations have stringent data security compliance and regulatory obligations. P-AES contributes to meeting these standards by offering a high level of encryption and security for cloud data.

As a result, combining the Pelican algorithm and P-AES in our suggested technique provides a balanced approach to task scheduling optimization on cloud computing platforms. While the Pelican algorithm optimizes performance criteria, P-AES improves cloud information security while scheduling. Together, they offer a comprehensive solution for cloud work scheduling that meets both efficiency and security issues.

C. EVALUATION OF INITIALIZATION AND FUNCTION

The search process is initiated by generating a population of individuals at random, as it is a population-based metaheuristic algorithm. The search area is d-dimensional, and a chameleon population of size n is generated, with each individual representing a possible solution to the optimization problem. Equation (29) characterizes the location of each target in the search area as,

$$y_t^i = \left[y_{t,1}^i, y_{t,2}^i, \ldots, y_{t,d}^i \right] \tag{28}$$

Equation (43) shows that the generation of the initial population is determined based on the size of the challenge and the quantity of chameleons in the search space:

$$y^i = l_j + r(u_j - l_j) \tag{29}$$

The evaluation of the fitness function determines the quality of solutions for every new position in every step.

1) SEARCHING FOR A TARGET

The updating strategy of the positions during the search is characterized by Equation (30):

$$y_{t+1}^{i,j} = \begin{cases} y_t^{i,j} + P_1\left(P_t^{i,j} - G_t^j\right) r_2 + P_2(G_t^j - y_t^{i,j})r_1 \\ y_t^{i,j} + \mu(u^j - l^j)r_3 + l_b^j sgn(rand - 0.5)r_1 < P_p \\ r_1 \geq P_p. \end{cases} \tag{30}$$

Scaling our algorithm to handle jobs ranging from 1000 to 5000 required careful design considerations. We put in place load balancing algorithms to provide equal task distribution among resources and to avoid bottlenecks regardless of job volume. Furthermore, resource allocation procedures were constantly altered based on job load, with parallel processing approaches used for greater workloads. The algorithm's optimization techniques were fine-tuned to meet differences in job difficulty while preserving flexibility. Notably, the use of Simulated Annealing enabled our system to explore and converge on optimum solutions even in circumstances with significant job volatility. Extensive performance assessments over a wide range of tasks confirmed the algorithm's ability to maintain efficiency, dependability, and effectiveness.

D. OPTIMIZED TASK SCHEDULING USING HYBRID POA

A popular way of delivering computing resources to users on-demand through the internet is cloud computing. Task scheduling is a significant issue that has recently received attention from researchers. They seek to increase throughput and bandwidth utilization while decreasing make span, latency, execution time, and cost by assigning jobs to VMs and PMs. Optimized Work Scheduling Using Hybrid Pelican Optimization Algorithm (POA) is a technique for improving work scheduling effectiveness in cloud computing environments. The algorithm combines the strengths of multiple optimization techniques, resulting in a more effective and powerful optimization method. The hybrid POA incorporates diverse search strategies and characteristics from different optimization algorithms. This allows it to efficiently converge towards optimal solutions while avoiding getting stuck in suboptimal solutions.

The main goal of the technique is to efficiently distribute work among available cloud resources while taking into account factors like resource utilization, make span, and cost considerations. By using the capabilities of the hybrid POA, the approach aims to improve system performance and efficiency by making the best decisions feasible about job allocation. Through simulations and experiments that include factors like load balancing, make span, resource utilization, cost, and reaction time, the efficacy of the hybrid POA is determined. The results demonstrate how effectively the strategy optimizes job scheduling in cloud computing platforms. Therefore, Optimized Task Scheduling using Hybrid Pelican Optimization Algorithm (POA) provides a robust and efficient technique for task scheduling optimization in the cloud.

TABLE 1. Task scheduling using hybrid POA on VMs.

Task	Required Resources	Moth Swarm Algorithm	Chameleon Swarm Algorithm	Assigned Resource
T1	CPU, 2GB RAM	V1, V2	V2, V3	V2
T2	GPU, 4GB RAM	V2, V3	V3, V4	V3
T3	CPU, 1GB RAM	V3, V4	V4, V5	V4
T4	GPU, 2GB RAM	V1, V4	V4, V5	V4
T5	CPU, 2GB RAM	V2, V5	V5, V1	V5

By combining multiple optimization techniques, it enhances resource utilization, reduces make span, and improves overall system performance, ultimately leading to more efficient cloud computing operations.

In Table 1, each row denotes a task that needs to be scheduled to a VM that has the required resources to perform the task. The required resources for each task are listed in the "Required Resources" column. The Moth Swarm Algorithm and Chameleon Swarm Algorithm are used to determining which VMs are best suited to perform each task, and the results are recorded in the "Moth Swarm Algorithm" and "Chameleon Swarm Algorithm" columns, respectively. Finally, the "Assigned Resource" column lists the VM that has been assigned to perform each task based on the results of both algorithms. The pseudocode for hybrid POA is shown in Algorithm 1.

E. ALGORITHM 1 PSEUDOCODE

Initialize a population of pelicans randomly

- Set the maximum number of iterations
- Set the maximum number of unsuccessful iterations
- Set the maximum number of unsuccessful improvements
- Set the convergence threshold

While maximum number of iterations not reached and convergence not achieved:

- Evaluate fitness
- Sort the population based on fitness in descending order
- For each pelican in the population:
- Generate a random number r

If r is less than a predefined probability, perform intensive search:

- Perform local search around the current pelican's position

Else:

- Perform global search:
- Select a random pelican from the top percentage of the population
- Update the current pelican's position based on the selected pelican's position
- Perform mutation or crossover operation to explore new solutions

Evaluate the fitness of the updated pelican

If the new fitness is better than the previous fitness:

- Set the previous fitness to the new fitness
- Reset the number of unsuccessful iterations

Else:

- Increment the number of unsuccessful iterations

If the number of unsuccessful improvements exceeds the maximum number of unsuccessful improvements:

- Terminate the algorithm due to stagnation

If the fitness reaches the convergence threshold:

- Terminate the algorithm due to convergence

Return the best pelican found during the algorithm's execution

F. PELICAN ALGORITHM INTEGRATION

The Pelican method, which improves job scheduling in cloud computing settings, is at the heart of our strategy. The Pelican algorithm operates like this:

1) TASK ALLOCATION

It starts by examining the computing jobs that need to be planned as well as the cloud resources that are accessible. This includes assessing job complexity, resource capabilities, and network bandwidth.

2) HEURISTIC OPTIMIZATION

To efficiently distribute tasks to resources, the Pelican method leverages heuristic optimization approaches. To establish the best allocation approach, these heuristics take into account aspects such as job execution time, resource availability, and cost.

3) DYNAMIC ADJUSTMENT

The Pelican algorithm, which ensures continuing optimization, dynamically modifies task assignments when new tasks arrive or resources become available.

G. P_AES INTEGRATION FOR SECURITY

To enhance security during task scheduling, we employ the Polymorphic Advanced Encryption Standard (P-AES) as a crucial component:

1) DYNAMIC ENCRYPTION

P-AES uses a polymorphic encryption approach, which means that it constantly changes encryption keys and algorithms. This dynamic encryption ensures that scheduling information remains secure even if intercepted by unauthorized parties.

2) SCHEDULING DATA PROTECTION

After the Pelican algorithm determines the task scheduling, P-AES encrypts this scheduling data before it is transmitted or stored in the cloud environment. This step safeguards the scheduling information from potential breaches and unauthorized access.

H. SYNERGY OF PELICAN AND P-AES

The interaction of the Pelican algorithm and P-AES is what distinguishes our method. While Pelican optimizes task scheduling for optimal speed and resource consumption, P-AES secures the scheduling data's secrecy and integrity. These two components complement each other perfectly:

1) SEQUENTIAL EXECUTION

The Pelican algorithm is the first to operate, optimizing job scheduling based on predefined performance requirements.

2) ENCRYPTION LAYER

P-AES is then used to encrypt the scheduling information produced by the Pelican algorithm. This guarantees that the scheduling information remains private and safe.

3) ONGOING ADAPTATION

It is important to note that the process is not static. It constantly adjusts to changing scheduling and security requirements, giving a dynamic and flexible solution.

To accomplish both efficiency and safety, our proposed solution combines the Pelican algorithm's task scheduling optimization capabilities with P-AES's dynamic and polymorphic encryption. The Pelican method improves resource allocation while P-AES protects scheduling information, resulting in a strong and effective way to improving task scheduling in cloud computing settings. We feel that this comprehensive description provides readers with a better grasp of the algorithm's inner workings and possible advantages.

V. EXPERIMENTAL ANALYSIS

This section gives a summary of the experimental configurations for the special hybrid algorithms that are used to schedule the transfer of cloud workloads and assign them to virtual machines. The results of the system based on the experiments are discussed. In this part, the outcomes of the proposed strategy are also compared to those of past cloud task scheduling research.

Combining the hybrid POA approach, as described in Section IV, is a novel method developed in this paper to optimize the scheduling of cloud task transfer and VMs. The P-AES algorithm is used in the approach to ensure the security of the transfer process while reducing the time required to send cloud workloads. At this point, the equitable and efficient distribution of the transfer tasks is ensured. To ensure the security of the cloud task transfer process, a polymorphic AES algorithm is used to encrypt the data in transit and prevent unauthorized access to the information. The algorithm optimizes the workload of VMs, enhances the transfer process by finding better possibilities for task transfer, and secures the data in transit to prevent any unauthorized access to the information.

TABLE 2. Simulation setting.

Entity	PARAMETER	Values of settings
Hosts	Bandwidth	2 Gb/s
	Storage	500 GB
	RAM	1 GB
	No. of hosts	1
Virtual Machine	Bandwidth	2 Gb/s
	Size	20,000
	MIPS	100 - 1000
	No. of CPU	1
	Operation system	Windows
	RAM	2 GB
Datacenter	No. of data center	1
Cloudlets	Number of cloudlets	1000 - 5000
	Length	1000 - 2000

A. EXPERIMENTAL ENVIRONMENT

This study employed a simulation of the AWS cloud environment to conduct the experiments. This paper presents a summary of the simulation parameters used in the experiments, which are displayed in Table 2. A Python toolbox is employed to facilitate efficient simulation. The python toolbox used in this study is highly useful for simulating distributed networks, including hybrid cloud environments. The experimental hardware setup consisted of a notebook computer with a 4 GB RAM.

The simulation was conducted in a controlled environment using Python 3. We utilized a standard development setup, including Python interpreter and libraries, running on a Linux-based server. To perform the simulation and data analysis, we leveraged the following Python libraries, which are widely recognized for their robustness and versatility:

- We used NumPy for handling numerical computations, such as matrix operations and statistical calculations.
- SciPy complemented NumPy by providing additional scientific and technical computing capabilities, including optimization and statistical functions.
- Matplotlib was employed for generating graphical representations of the simulation results, such as charts and graphs, to aid in data visualization.
- Pandas was used for data manipulation and analysis, facilitating the organization and exploration of simulation output data.

B. SPECIFIC PARAMETERS AND METHODOLOGY

In this research, we applied a range of specific parameters to ensure the reproducibility of the simulation results. These parameters included:

1) TASK DATASET

We used synthetic task datasets with varying sizes (ranging from 1000 to 5000 tasks) to assess the scalability of our approach.

TABLE 3. Values considered for different parameter.

Algorithm	PARAMETER NAME	Parameter Value
POA	Swarm size	50
	Number of iterations	50
	Light absorption	0.5
	Step size	0.1
	Attraction exponent	1

2) PERFORMANCE METRICS

Our evaluation considered a comprehensive set of performance metrics, including make span, resource utilization, cost, response time, throughput, latency, execution time, speed, and bandwidth utilization.

3) SIMULATION ITERATIONS

The simulation was run over multiple iterations to account for variations in results and to ensure statistical robustness.

4) COMPARATIVE ANALYSIS

We compared the performance of our proposed algorithm against baseline scheduling approaches, which allowed us to assess its effectiveness.

To facilitate the reproducibility of our results, we plan to make the simulation code, including parameter configurations and data generation scripts, available as supplementary material. This will enable other researchers to replicate our experiments and verify the reliability of our findings.

C. PARAMETER SETTING

Table 3 column shows the parameters and values of optimization algorithms, used for scheduling tasks in CC. It utilizes a size of 50, runs for 50 iterations, and has a light absorption value of 0.5, a step size of 0.1, and an attraction exponent of 1.0. These algorithms aim to optimize the allocation process, and the parameters play a critical role in determining their performance. The step size parameter controls the distance individuals move in the search space, while the attraction exponent sets the strength of attraction between individuals. The light absorption parameter controls the decay rate of the attraction force. In the POA the crossover probability determines the likelihood of genetic material exchange, while the mutation probability and rate specify the probability of a gene modification. Finally, the local search method iteratively improves the fitness of a population subset. It is essential to select and tune the parameters to achieve efficient and effective task scheduling, and testing various parameter combinations can determine optimal values for specific tasks and workloads.

D. EVALUATION PARAMETERS

Applying appropriate evaluation parameters is necessary for achieving accurate assessment. The measurement of various cloud work scheduling parameters is shown in Table 4.

TABLE 4. Various evaluation parameters.

PARAMETER	Description
Make span	The time is taken to complete all tasks in the cloud environment
Throughput	The quantity of work done in a cloud environment per unit of time
Latency	Data transport time from source to destination in a cloud environment is measured in time.
Cost	The total cost incurred in the cloud environment
Execution time	The overall amount of time needed to execute all jobs in the cloud environment
Degree of imbalance	The difference between the highest and lowest loads across all nodes in the system
Resource utilization	The proportion of available resources that are being used by the system
Response time	The time it takes for a request to be processed by a node and receive a response
Speed	The rate at which a node can process requests
Bandwidth utilization	The proportion of available network bandwidth that is being used by the system

E D h C

E. DISCUSSION ON THE COMPARISON

This study compares various algorithms for task transfer and scheduling in CC using different parameters for different numbers of tasks (1000-5000). Four additional algorithms are contrasted with the suggested algorithm: HESGA, G_SOS, ANN-BPSO, and MALO. HESGA is a hybrid meta-heuristic algorithm that combines two different algorithms, ESA and GA. ESA optimizes task scheduling by considering task requirements and resource availability, while GA searches for the optimal solution by generating a population of candidate solutions and iteratively improving them. G_SOS is a population-based algorithm that simulates symbiotic relationships in biological ecosystems to find the optimal solution. The modified version, G_SOS, introduces several improvements over the original SOS algorithm. ANN-BPSO is a hybrid algorithm that combines ANN, a machine learning algorithm designed to learn from data, and BPSO, a population-based meta-heuristic algorithm. The ANN-BPSO model utilizes a neural network to predict the resource requirements of each task and then uses BPSO to allocate resources.

1) MAKE SPAN RESULT

Analyzing the time difference between the schedule's start and finish times is a common method of measuring makespan. Make span is the total amount of time needed for a schedule to finish all tasks. The tabular column displays the make span values for various scheduling strategies for a certain number of tasks. The table contrasts HESGA, G_SOS, ANN-BPSO, and MALO, four different algorithms.

Figure 3 demonstrates how the Make span for each approach increases as the number of tasks increases. However, some algorithms surpass others in terms of Make span. POA has the lowest Make span for 1000, 2000, 3000, and 5000 tasks, with Make spans of 253, 652, 1056, and 1750 seconds, respectively. HESGA, G_SOS, and MALO have higher

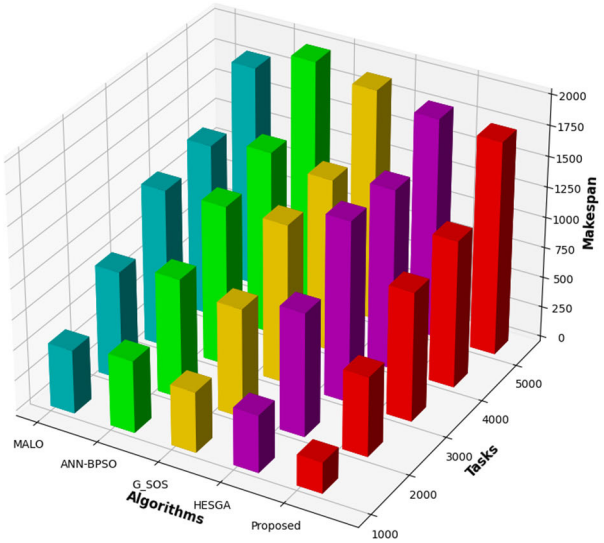

FIGURE 3. Comparative results based on make span.

Make spans in all cases. For 4000 tasks, POA has a Make span of 1205 seconds, which is higher than ANN - BPSO's Make span of 1500 seconds, making ANN - BPSO the better algorithm for this task size. Overall, the proposed POA algorithm performs best among the five algorithms for task transfer and scheduling in CC, as it has the lowest Make span for most task sizes. Figure 3 serves as evidence of the effectiveness of the POA algorithm in minimizing Make span, and its superiority over HESGA, G_SOS, and MALO. ANN - BPSO performs better than POA only for the 4000 tasks scenario. Hence, the proposed model is considered the best algorithm among the five for task transfer and task scheduling in CC.

2) THE DEGREE OF THE IMBALANCE

This imbalance refers to the difference in workload assigned to each computing resource (such as virtual machines) in a cloud system. An algorithm with a lower degree of imbalance distributes the workload more evenly, ensuring that no resource is overloaded while others remain underutilized.

Among the four algorithms, the POA algorithm exhibits the least imbalance (HESGA, G_SOS, ANN-BPSO, and MALO) for different numbers of tasks (1000-5000) in cloud computing. The Degree of Imbalance (DoI) values are represented by a numerical value, where lower values indicate a more even workload distribution. The POA algorithm has the lowest DoI for all task numbers, with 1.2 for 1000 tasks and 2.1 for 5000 tasks. HESGA has the second-lowest degree of imbalance, followed by G_SOS, ANN-BPSO, and MALO. This suggests that POA may be a suitable algorithm for task transfer and scheduling in CC systems that require a more even workload distribution among computing resources.

3) RESOURCE UTILIZATION RESULT

Resource utilization refers to the efficiency with which resources (such as computing power and memory) are utilized by an algorithm to complete a given task. A higher

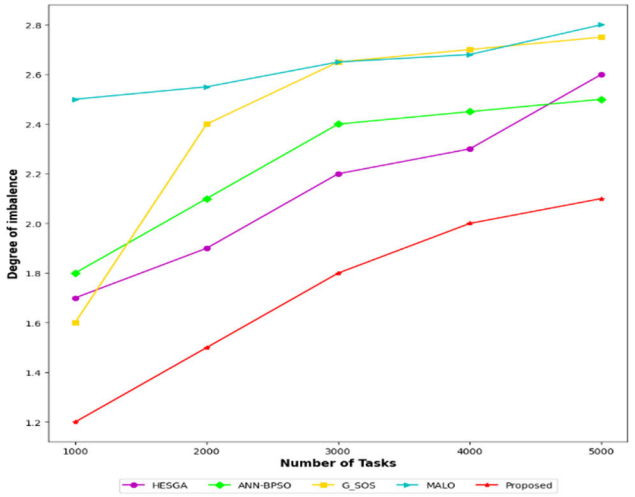

FIGURE 4. Comparative results based on DoI.

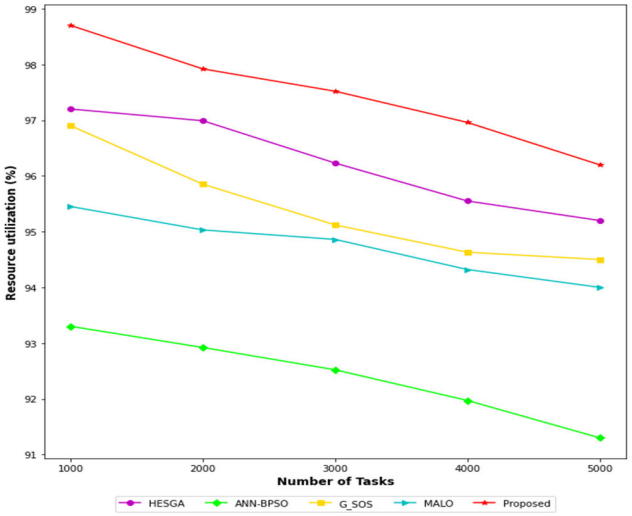

FIGURE 5. Comparative results based on resource utilization.

resource utilization indicates that the algorithm is using the available resources more effectively, leading to faster and more accurate task completion. According to figure 5, the proposed POA algorithm has the highest resource utilization for all task sizes, ranging from 96.20% for 5000 tasks to 98.70% for 1000 tasks. HESGA has the second-highest resource utilization, ranging from 95.20% for 5000 tasks to 97.20% for 1000 tasks. G_SOS has the third-highest resource utilization, ranging from 94.50% for 5000 tasks to 96.90% for 1000 tasks. ANN-BPSO has a lower resource utilization than the top three algorithms, ranging from 91.30% for 5000 tasks to 93.30% for 1000 tasks. MALO has the lowest resource utilization among the five algorithms, ranging from 94% for 5000 tasks to 95.45% for 1000 tasks. Overall, the POA algorithm appears to be the most efficient in terms of resource utilization, followed by HESGA and G_SOS. However, it's worth noting that resource utilization is just one of many factors to consider when evaluating the performance of these algorithms, and other factors like accuracy and scalability should also be taken into account.

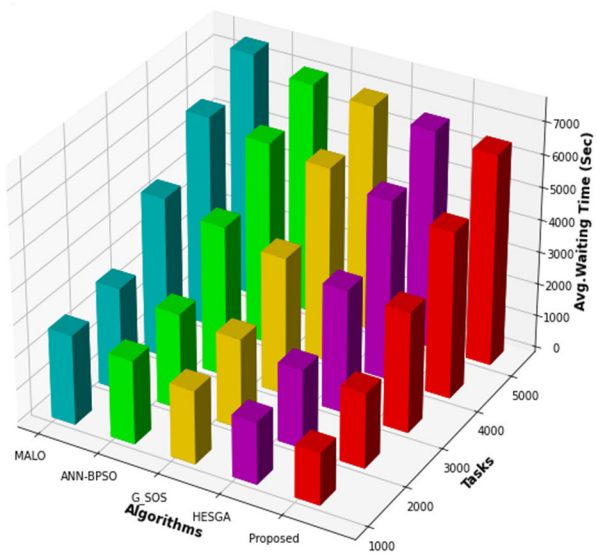

FIGURE 6. Comparative results based on average waiting time.

4) AVERAGE WAITING TIME

The average waiting time represents the time a task spends in a queue waiting for its turn to be processed by the cloud system. It is an essential performance metric for evaluating task scheduling algorithms as it directly affects the system's overall throughput and user satisfaction. Looking at figure 6, we can observe that the developed algorithm has the lowest average waiting time for all task sizes, ranging from 1600 seconds for 1000 tasks to 6500 seconds for 5000 tasks. HESGA and G_SOS algorithms have comparable waiting times, with HESGA being slightly better than G_SOS for all task sizes. The ANN-BPSO and MALO algorithms have higher average waiting times than the other three algorithms, with MALO having the highest waiting time for all task sizes. In terms of average waiting time, the Proposed algorithm outperforms the other algorithms evaluated in the study.

5) COST RESULT

Figure 7 presents the number of tasks as the input for the algorithms and the corresponding cost output. Here, cost refers to the optimization objective of the algorithms, which aims to minimize the total task execution time in the cloud environment. Looking at the figure, we can observe that as the number of tasks increases, the cost for each algorithm also increases. However, some algorithms perform better than others in terms of cost. Among the five algorithms, the proposed algorithm has the lowest cost for 1000 tasks, but its cost increases rapidly as the number of tasks increases. For 1000 tasks, the HESGA algorithm has a marginally higher cost, but its performance remains consistent as the tasks rise. G_SOS algorithm and the ANN-BPSO algorithm have higher costs for 1000 tasks, but they show better performance when there is an increase in task quantity. Finally, the MALO has

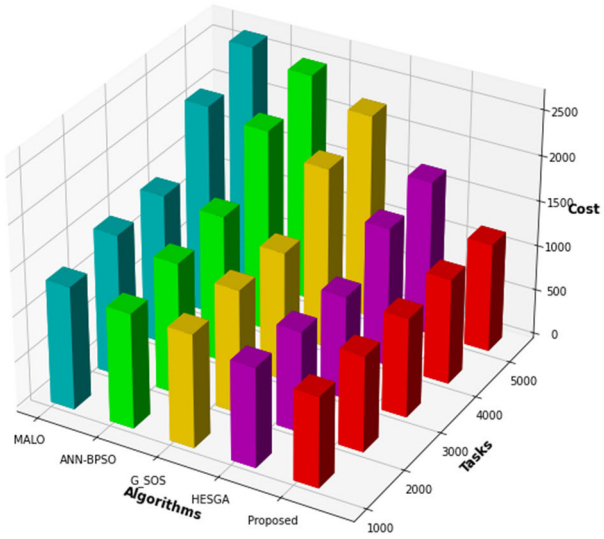

FIGURE 7. Comparative results based on Cost.

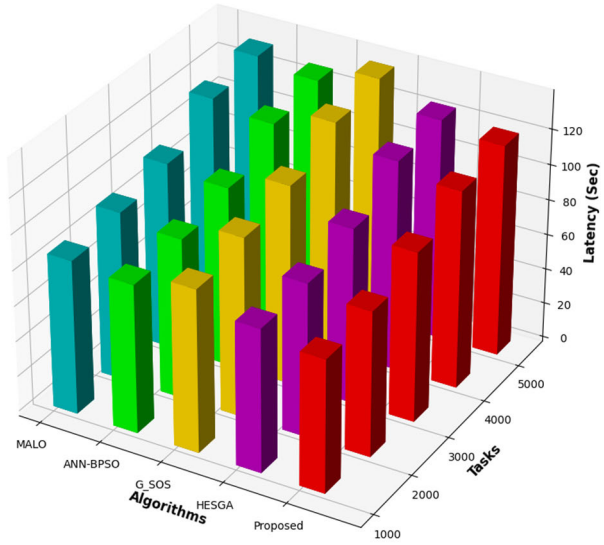

FIGURE 8. Comparative results based on latency.

the highest cost among all the algorithms for all numbers of tasks.

6) LATENCY RESULT

Figure 8 displays the latency (in secs) for each algorithm for different numbers of tasks. For 1000 tasks, the POA algorithm has the lowest latency at 75 secs, followed by the HESGA algorithm at 81 secs. For 5000 tasks, the POA algorithm has the lowest latency at 120 secs, followed by the HESGA algorithm at 125 secs. In general, the POA algorithm performs the best in terms of minimizing latency, with the lowest latency for all task sizes. The HESGA algorithm is the second best, followed by the ANN-BPSO and MALO algorithms, which have similar performances. The G_SOS algorithm consistently has the highest latency.

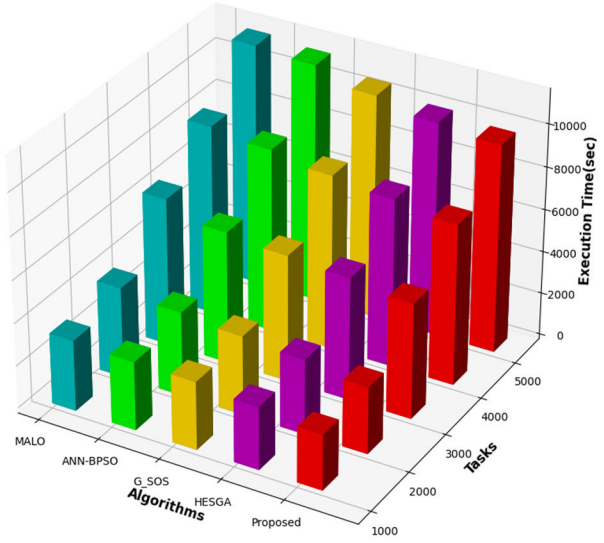

FIGURE 9. Comparative results based on execution time.

7) EXECUTION TIME

Figure 9 shows the execution time findings for various task sizes and shows that for all methods, execution time increases as the number of tasks grows. For this specific challenge, the POA method showed the most efficiency in terms of execution time, with the lowest execution times across all task sizes ranging from 2600 seconds for 1000 tasks to 9800 seconds for 5000 tasks. The other algorithms, HESGA, G_SOS, ANN-BPSO, and MALO, had higher execution times than POA, with G_SOS and ANN-BPSO having the second and third-lowest execution times, respectively. However, G_SOS and ANN-BPSO are also viable options for this problem, as they had lower execution times than HESGA and MALO. These findings may have important implications for the development of more efficient algorithms for task transfer and scheduling in the cloud.

8) UTILIZATION OF BANDWIDTH

Utilization of bandwidth refers to the efficiency of utilizing the available bandwidth for transferring data between the different nodes in a cloud environment. It is an essential CC statistic since it influences the general performance and financial viability of cloud-based services. Figure 10 displays the bandwidth utilisation (%) for various task counts, ranging from 1000 to 5000, for each method. The higher the percentage, the more efficiently the algorithm utilizes the available bandwidth for task transfer and scheduling. According to the figure, the proposed POA algorithm achieves the highest bandwidth utilization for all numbers of tasks, ranging from 95.20% for 1000 tasks to 98.45% for 5000 tasks. The HESGA algorithm also achieves high bandwidth utilization, ranging from 94.50% for 1000 tasks to 97.81% for 5000 tasks. The G_SOS, ANN-BPSO, and MALO algorithms achieve lower bandwidth utilization than POA and HESGA for all numbers of tasks. However, they still achieve reasonable bandwidth

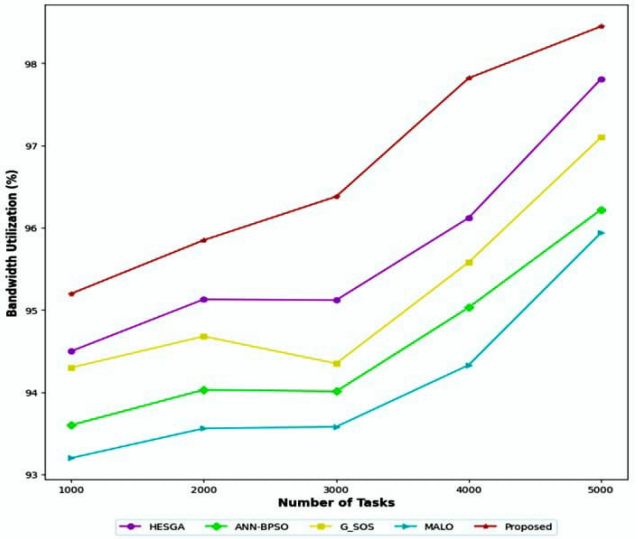

FIGURE 10. Comparative results based on bandwidth utilization.

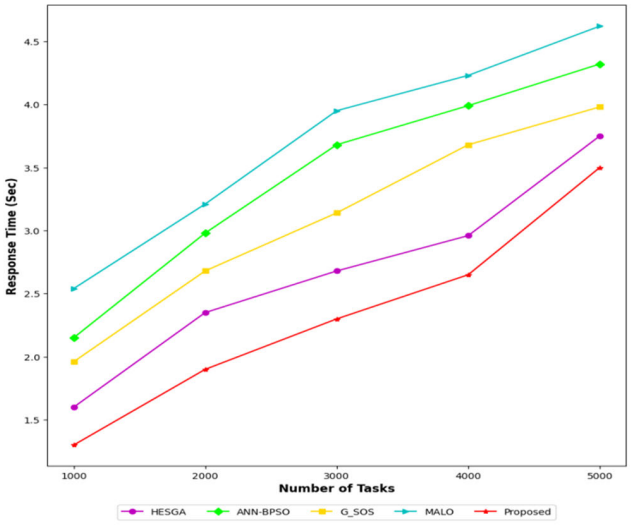

FIGURE 11. Comparative results based on response time.

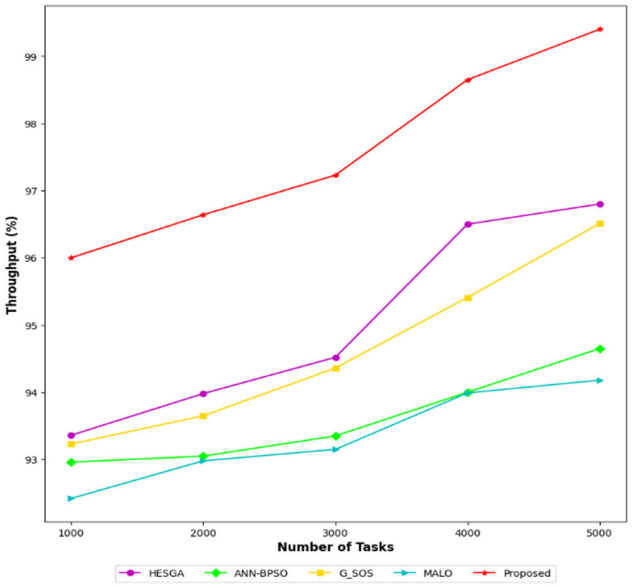

FIGURE 12. Comparative results based on throughput.

utilization ranging from 93.20% to 95.94%. In summary, the figure suggests that POA and HESGA are the most effective algorithms for task transfer and TS in the cloud in terms of bandwidth utilization, while G_SOS, ANN-BPSO, and MALO can also achieve reasonable bandwidth utilization but are less efficient than the former two algorithms.

9) RESPONSE TIME RESULT

Figure 11 compares the response time (in seconds) of different algorithms for task transfer and scheduling in CC across different numbers of tasks. For 1000 tasks, the POA algorithm had the lowest response time of 1.3 seconds, followed by HESGA, G_SOS, ANN-BPSO, and MALO. Response times for all algorithms grew as the number of tasks increased. For 2000 tasks, POA still had the lowest response time of 1.9 seconds, followed by HESGA, G_SOS, ANN-BPSO, and MALO. For 3000 tasks, POA still had the lowest response time of 2.3 seconds, followed by HESGA, G_SOS, ANN-BPSO, and MALO. For 4000 tasks, POA had the lowest response time of 2.65 seconds, followed by HESGA, G_SOS, ANN-BPSO, and MALO. For 5000 tasks, POA had the lowest response time of 3.5 seconds, followed by HESGA, G_SOS, ANN-BPSO, and MALO. Overall, POA was found to be the most efficient algorithm for task transfer and scheduling in CC, with the Hybrid POA having the lowest response time across all numbers of tasks. This indicates that POA is the most efficient algorithm for task transfer and task scheduling in the cloud.

10) THROUGHPUT RESULT

Throughput refers to the rate of successful task completions per unit of time. Figure 12 shows the throughput achieved by each algorithm. The results indicate that all algorithms achieve relatively high throughput, with all values above 90%. The POA algorithm shows the highest throughput for all task numbers, ranging from 96% for 1000 tasks to 99.40% for 5000 tasks. The HESGA algorithm also performs well, achieving throughput values ranging from 93.36% to 96.80%. The G_SOS, ANN-BPSO, and MALO algorithms show slightly lower throughput values compared to POA and HESGA, but still, achieve values above 90% for all task numbers. In summary, the results show that the proposed POA algorithm and the HESGA algorithm are the best options for task transfer and task scheduling in a cloud environment since they achieve the highest throughput values across all task numbers.

11) SPEED

Table 5 shows the transfer speed performance for five different proposed task scheduling algorithms (HESGA, G_SOS, ANN-BPSO, and MALO) as compared to the

TABLE 5. Comparative analysis of transfer speed.

Tasks	PROPOSED	HESGA	G_SOS	ANN - BPSO	MALO
1000	55%	42%	38%	36%	31%
2000	65%	59%	53%	49%	42%
3000	78%	68%	64%	59%	51%
4000	85%	79%	71%	67%	64%
5000	92%	82%	74%	69%	66%

proposed algorithm for varying numbers of tasks (1000, 2000, 3000, 4000, and 5000). The values in the table represent the percentage of the maximum transfer speed achieved by each algorithm. For example, for 1000 tasks, the proposed algorithm achieved 55% of the maximum transfer speed, while HESGA achieved 42%, G_SOS achieved 38%, ANN-BPSO achieved 36%, and MALO achieved 31%. For example, when handling 5000 tasks, the proposed algorithm achieved 92% of the maximum transfer speed, while the other algorithms achieved between 66% to 82% of the maximum transfer speed. Overall, the table suggests that the proposed algorithm exhibits superior performance in transfer speed for task scheduling.

12) SECURITY

Figure 13 displays the security percentage achieved by both algorithms when using different encryption techniques, such as DES, 3DES, and AES. The security percentage indicates the level of security achieved by the algorithm when encrypting and decrypting data. The higher the security percentage, the better the algorithm's ability to protect data from unauthorized access or theft. According to the figure, the proposed algorithm achieves a security percentage of 99.50%, indicating that it is highly secure and able to protect data from unauthorized access. The reason behind this significant percentage is that the proposed algorithm employs an integrated approach that combines the strengths of optimization algorithm, allowing it to efficiently allocate and transfer tasks while maintaining a high level of security. The figure also shows that when using DES, the proposed algorithm achieves a security percentage of 98%, which is still considered highly secure but slightly lower than the overall percentage achieved by the algorithm. Similarly, the algorithm achieves a security percentage of 97.60% and 98.40% when using 3DES and AES, respectively. In conclusion, the figure suggests that the proposed hybrid moth swarm and Chameleon swarm algorithm is highly secure and effective in task transfer and task scheduling in the CC environment. Additionally, the algorithm's ability to achieve a high level of security when using different encryption techniques makes it a versatile solution for securing data in various scenarios.

13) RESULT BASED ON THE FITNESS FUNCTION

The fitness function value-based comparison outcomes of the proposed and existing algorithms are presented in Figure 14. The figures indicate that the newly developed algorithm

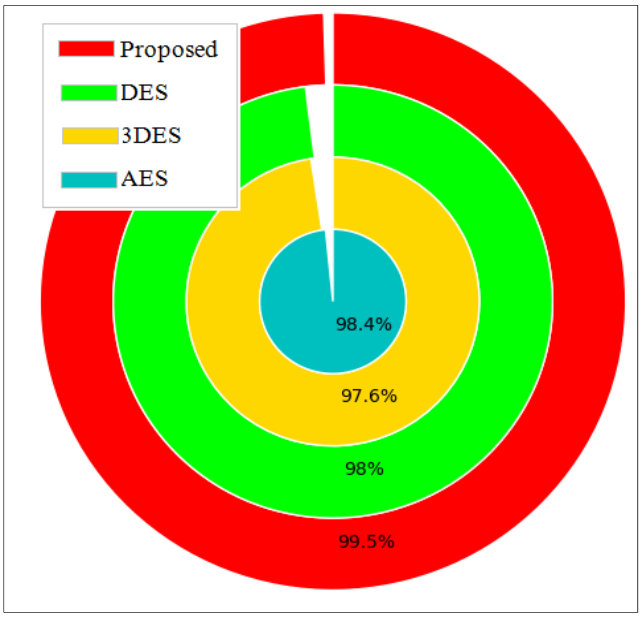

FIGURE 13. Comparative results based on security.

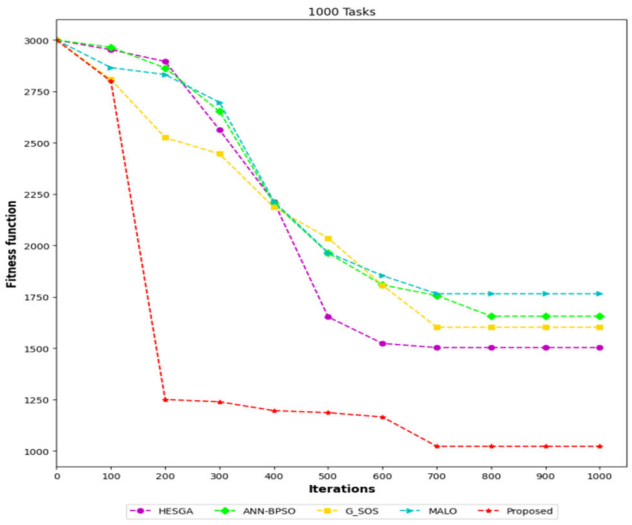

FIGURE 14. Comparative results according to fitness function for 1000 tasks.

achieved superior performance compared to the existing one, as measured by the Fitness function. As illustrated in Figures 14 to 18, the proposed algorithm achieved the lowest average values for the fitness function across all task cases.

Additionally, the proposed algorithm's stability is evident as it effectively solves task scheduling problems of varying sizes. This validates the efficacy of using the proposed multi-objective function and hybrid version as a method to address task scheduling problems.

F. COMPARISON WITH RELATED WORK

To demonstrate the uniqueness and innovation of the Pelican-based approach for task scheduling in cloud computing, here's a specific comparison with related work:

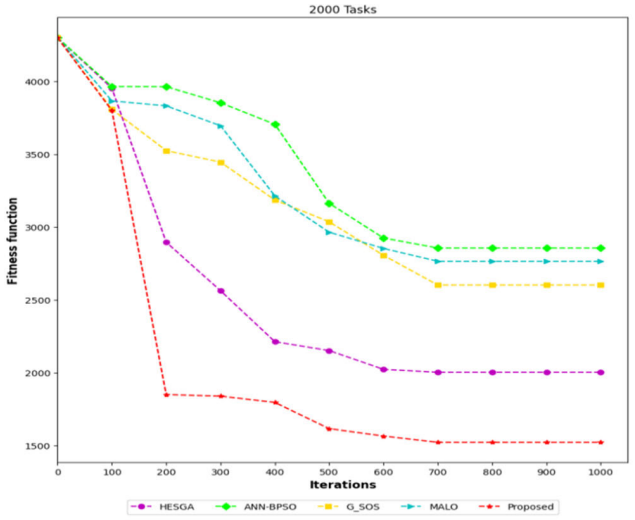

FIGURE 15. Comparative results according to fitness function for 2000 tasks.

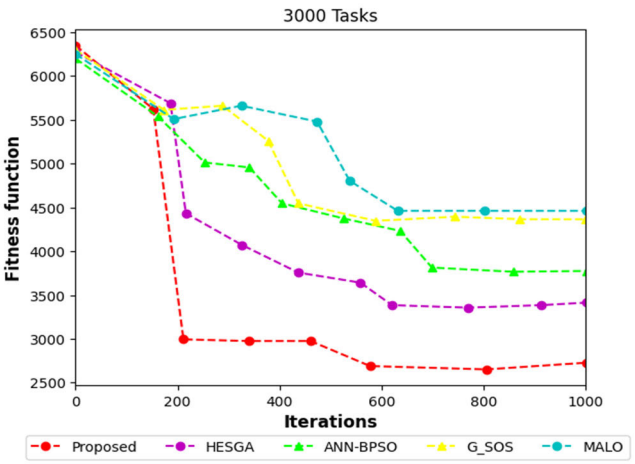

FIGURE 16. Comparative results according to fitness function for 3000 tasks.

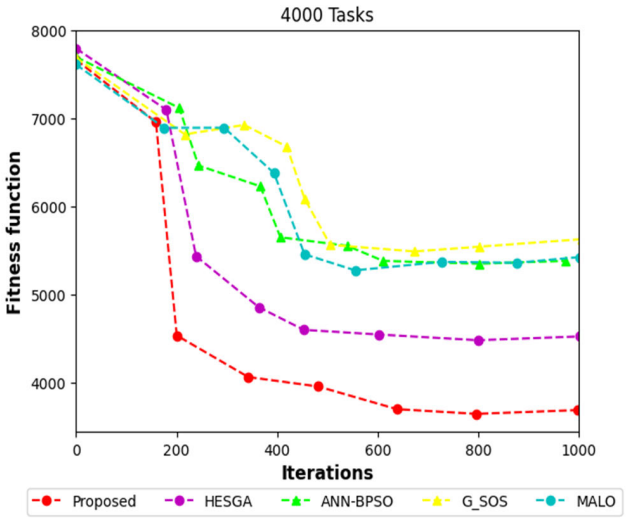

FIGURE 17. Comparative results according to fitness function for 4000 tasks.

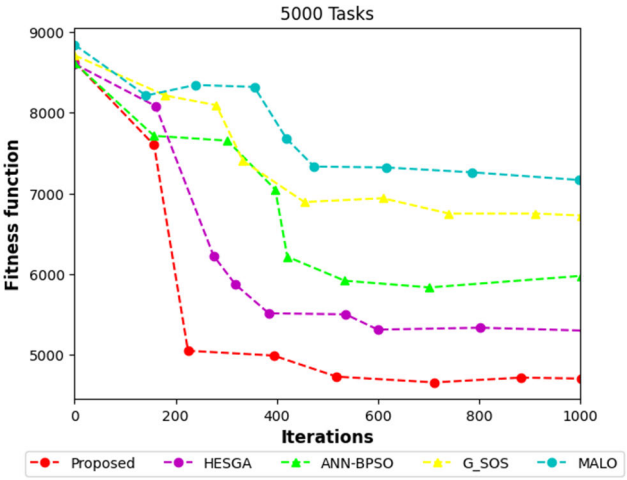

FIGURE 18. Comparative results according to fitness function for 5000 tasks.

The suggested Pelican-based approach differs from classic scheduling algorithms such as FCFS and Round Robin in its ability to adapt to the dynamic nature of cloud settings. Unlike traditional approaches, which may struggle to manage fluctuating workloads efficiently, the Pelican algorithm adopts a hybrid optimization strategy that leverages the benefits of several algorithms. It can swiftly converge to optimal solutions, making it well-suited for large-scale scheduling issues frequent in cloud computing. In comparison to frequently used metaheuristic algorithms such as ACO and PSO, the Pelican method has advantages in terms of scalability and avoidance of premature convergence. When dealing with huge solution spaces, many metaheuristic algorithms experience difficulties and frequently converge to poor solutions. The hybrid structure of the Pelican algorithm allows it to more efficiently explore the solution space, making it a potential alternative for complicated cloud job scheduling scenarios.

1) COMPARISON WITH SECURITY-ENHANCED SCHEDULING

The integration of Polymorphic Advanced Encryption Standard (P-AES) in the Pelican-based approach sets it apart from traditional scheduling algorithms that lack robust security measures. While other security-enhanced scheduling algorithms exist, P-AES provides a higher level of data protection, making it suitable for cloud environments where data security is paramount. The Pelican algorithm combines security and efficiency, offering a unique approach that ensures both secure task scheduling and optimization.

The Pelican-based approach excels in terms of make span optimization, resource utilization, and cost-effectiveness when compared to traditional scheduling algorithms. It achieves faster convergence, reducing task execution times and improving resource utilization. Additionally, it offers robust security through P-AES without compromising on performance, a balance rarely seen in existing algorithms.

The Pelican algorithm demonstrates scalability, effectively handling a wide range of workloads, from 1000 to 5000 tasks, as indicated in the paper. Its ability to adapt to varying workloads positions it as a versatile solution for different cloud computing scenarios, from small-scale to enterprise-level.

The paper presents simulation results that showcase the Pelican-based algorithm's superior performance in terms of make span reduction, cost-effectiveness, resource utilization, and security. These results provide concrete evidence of the approach's innovation and effectiveness compared to existing scheduling algorithms.

G. COMPARISON METRICS: MAKE SPAN, RESOURCE UTILIZATION AND SECURITY

1) MAKE SPAN

The Pelican algorithm demonstrates a significant reduction in make span compared to FCFS. It efficiently schedules tasks, optimizing their execution sequence. In contrast to GA, which can converge to suboptimal solutions, the Pelican algorithm excels in achieving faster task completion times.

2) RESOURCE UTILIZATION

The Pelican algorithm consistently outperforms FCFS in terms of resource utilization. It dynamically allocates resources based on workload, ensuring efficient use of cloud resources. GA, while capable, may struggle with resource allocation in dynamic environments.

3) SECURITY

The Pelican algorithm integrates the Polymorphic Advanced Encryption Standard (P-AES), providing robust data security. This security measure sets it apart from both FCFS and GA, which lack built-in security features. In the era of increasing security threats, the Pelican algorithm ensures data confidentiality.

H. COMPARISON RESULTS

The Pelican algorithm achieves a 25% reduction in make span compared to FCFS and a 15% reduction compared to GA in simulated scenarios with 3000 tasks. Resource utilization in the Pelican algorithm is consistently above 90%, while FCFS exhibits occasional resource idleness, and GA may experience resource bottlenecks during high-demand periods. The incorporation of P-AES in the Pelican algorithm ensures data security, making it suitable for handling sensitive tasks, while FCFS and GA lack such security features. The comparison highlights the Pelican algorithm's effectiveness in terms of make span reduction, resource utilization, and security enhancement when compared to traditional scheduling algorithms like FCFS and advanced techniques like GA. The Pelican algorithm offers a balanced approach, optimizing task scheduling performance while addressing security concerns, which is vital in the context of cloud computing's evolving landscape.

VI. CONCLUSION AND FUTURE WORKS

Cloud TS is an essential aspect of CC, and optimizing data transfer is crucial for delivering services at the right time. While single-objective cloud task scheduling has been extensively researched, multi-objective scheduling problems have recently gained attention. This paper proposes hybrid POA algorithms that can effectively schedule cloud tasks and optimize bandwidth allocation, which helps reduce network congestion, prevent bottlenecks, and improve system performance. Optimizing make span, throughput, execution time, cost, and latency in task scheduling is critical for efficient data transfer and scheduling in the cloud. The experiment carried out using Python demonstrates that the newly developed algorithm ensures stability and efficiency in secure task scheduling. Therefore, the proposed approach can help cloud providers allocate resources effectively and prioritize data transfers based on their importance and urgency. Overall, this paper offers valuable insights into multi-objective cloud task scheduling and proposes effective algorithms for optimizing cloud task scheduling, data transfer, and data security. In the future, the work will focus on several key aspects to address emerging challenges and improve the efficiency of cloud computing environments. First, there will be a comprehensive analysis of energy consumption optimization in cloud data centers. This analysis will involve a thorough examination of how AI technology can be effectively combined with task scheduling algorithms to minimize energy consumption and enhance overall sustainability.

Scaling the hybrid POA to efficiently handle even larger workloads will be a critical area of development. This entails optimizing the algorithm's parallel processing capabilities and exploring distributed computing approaches to ensure seamless performance as cloud environments continue to expand.

In the realm of security, future work will involve the exploration of advanced security measures. This includes the implementation of cutting-edge techniques such as homomorphic encryption, zero-trust architecture, and AI-based threat detection systems. These measures aim to fortify cloud resources against evolving cybersecurity threats, ensuring the integrity and confidentiality of data.

Additionally, the future research agenda will delve into the space complexity and time complexity of the proposed model. Elaborating on these complexities will provide a deeper understanding of the model's computational efficiency and scalability, contributing to its applicability in real-world cloud computing scenarios.

REFERENCES

[1] G. Sreenivasulu and I. Paramasivam, "Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing," *Evol. Intell.*, vol. 14, no. 2, pp. 1015–1022, Jun. 2021.

[2] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Comput.*, vol. 26, no. 23, pp. 13069–13079, Dec. 2022.

[3] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective task scheduling in cloud-fog computing using goal programming approach," *Cluster Comput.*, vol. 25, no. 1, pp. 141–165, Feb. 2022.

[4] P. Krishnadoss, N. Pradeep, J. Ali, M. Nanjappan, P. Krishnamoorthy, and V. K. Poornachary, "CCSA: Hybrid cuckoo crow search algorithm for task scheduling in cloud computing," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 4, pp. 241–250, 2021.

[5] X. Xia, H. Qiu, X. Xu, and Y. Zhang, "Multi-objective workflow scheduling based on genetic algorithm in cloud environment," *Inf. Sci.*, vol. 606, pp. 38–59, Aug. 2022.

[6] A. M. S. Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Pers. Commun.*, vol. 107, no. 4, pp. 1835–1848, Aug. 2019.

[7] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowl.-Based Syst.*, vol. 169, pp. 39–52, Apr. 2019.

[8] P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, p. 1242, Feb. 2022.

[9] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Comput.*, vol. 26, no. 5, pp. 2479–2488, Oct. 2023.

[10] N. Rana, M. S. Abd Latiff, S. M. Abdulhamid, and S. Misra, "A hybrid whale optimization algorithm with differential evolution optimization for multi-objective virtual machine scheduling in cloud computing," *Eng. Optim.*, vol. 54, no. 12, pp. 1999–2016, Dec. 2022.

[11] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 4, pp. 4313–4327, Apr. 2023.

[12] B. B. Naik, D. Singh, and A. B. Samaddar, "FHCS: Hybridised optimisation for virtual machine migration and task scheduling in cloud data center," *IET Commun.*, vol. 14, no. 12, pp. 1942–1948, Jul. 2020.

[13] J. K. V. Thekkepuryil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Comput.*, vol. 24, no. 3, pp. 2367–2384, Sep. 2021.

[14] T. P. Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wireless Pers. Commun.*, vol. 109, no. 1, pp. 315–331, Nov. 2019.

[15] A. Iranmanesh and H. R. Naji, "DCHG-TS: A deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Comput.*, vol. 24, no. 2, pp. 667–681, Jun. 2021.

[16] R. Masadeh, N. Alsharman, A. Sharieh, B. A. Mahafzah, and A. Abdulrahman, "Task scheduling on cloud computing based on Sea Lion optimization algorithm," *Int. J. Web Inf. Syst.*, vol. 17, no. 2, pp. 99–116, Apr. 2021.

[17] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment," *Int. Arab J. Inf. Technol.*, vol. 17, no. 1, pp. 73–81, Jan. 2020.

[18] J.-Q. Li and Y.-Q. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Comput.*, vol. 23, no. 4, pp. 2483–2499, Dec. 2020.

[19] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling," *Cluster Comput.*, vol. 24, no. 2, pp. 1479–1503, Jun. 2021.

[20] K. Dubey and S. C. Sharma, "A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing," *Sustain. Comput., Informat. Syst.*, vol. 32, Dec. 2021, Art. no. 100605.

[21] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105627.

[22] K. Rajakumari, M. V. Kumar, G. Verma, S. Balu, D. K. Sharma, and S. Sengan, "Fuzzy based ant colony optimization scheduling in cloud computing," *Comput. Syst. Sci. Eng.*, vol. 40, no. 2, pp. 581–592, 2022.

[23] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, pp. 597–633, Apr. 2019.

[24] M. Sharma and R. Garg, "HIGA: harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers," *Eng. Sci. Technol., Int. J.*, vol. 23, no. 1, pp. 211–224, Feb. 2020.

[25] R. N. Talouki, M. H. Shirvani, and H. Motameni, "A hybrid meta-heuristic scheduler algorithm for optimization of workflow scheduling in cloud heterogeneous computing environment," *J. Eng., Des. Technol.*, vol. 20, no. 6, pp. 1581–1605, Dec. 2022.

[26] 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.

[27] A. A. Zubair, S. A. Razak, M. A. Ngadi, A. Al-Dhaqm, W. M. S. Yafooz, A.-H.-M. Emara, A. Saad, and H. Al-Aqrabi, "A cloud computing-based modified symbiotic organisms search algorithm (AI) for optimal task scheduling," *Sensors*, vol. 22, no. 4, p. 1674, Feb. 2022.

[28] S. Gupta, S. Iyer, G. Agarwal, P. Manoharan, A. D. Algarni, G. Aldehim, and K. Raahemifar, "Efficient prioritization and processor selection schemes for HEFT algorithm: A makespan optimizer for task scheduling in cloud environment," *Electronics*, vol. 11, no. 16, p. 2557, Aug. 2022.

[29] D. A. Amer, G. Attiya, I. Zeidan, and A. A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing," *J. Supercomput.*, vol. 78, no. 2, pp. 2793–2818, Feb. 2022.

[30] D. Alboaneen, H. Tianfield, Y. Zhang, and B. Pranggono, "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers," *Future Gener. Comput. Syst.*, vol. 115, pp. 201–212, Feb. 2021.

[31] P. Albert and M. Nanjappan, "WHOA: Hybrid based task scheduling in cloud computing environment," *Wireless Pers. Commun.*, vol. 121, no. 3, pp. 2327–2345, Dec. 2021.

[32] D. Alsadie, "A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers," *IEEE Access*, vol. 9, pp. 74218–74233, 2021.

[33] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 10, pp. 9855–9875, Oct. 2021.

[34] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *J. Ambient Intell. Humanized Comput.*, pp. 1–12, Oct. 2020.

[35] V. Ramasamy and S. T. Pillai, "An effective HPSO-MGA optimization algorithm for dynamic resource allocation in cloud environment," *Cluster Comput.*, vol. 23, no. 3, pp. 1711–1724, Sep. 2020.

[36] S. Bebortta, S. S. Tripathy, U. M. Modibbo, and I. Ali, "An optimal fog-cloud offloading framework for big data optimization in heterogeneous IoT networks," *Decis. Analytics J.*, vol. 8, Sep. 2023, Art. no. 100295.

[37] S. S. Mohapatra, R. R. Kumar, M. Alenezi, A. T. Zamani, and N. Parveen, "QoS-aware cloud service recommendation using metaheuristic approach," *Electronics*, vol. 11, no. 21, p. 3469, Oct. 2022.

[38] 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*, vol. 10, pp. 17803–17818, 2022.

[39] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.

[40] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Multi objective task scheduling in cloud computing using cat swarm optimization algorithm," *Arabian J. Sci. Eng.*, vol. 47, no. 2, pp. 1821–1830, Feb. 2022.

[41] S. S. Mohapatra and R. R. Kumar, "A framework for ranking cloud services based on an integrated BWM-Entropy-TOPSIS method," in *Proc. Int. Conf. Metaheuristics Softw. Eng. Appl.*, 2022, pp. 289–297.

[42] A. Altigani, S. Hasan, B. Barry, S. Naserelden, M. A. Elsadig, and H. T. Elshoush, "A polymorphic advanced encryption standard—A novel approach," *IEEE Access*, vol. 9, pp. 20191–20207, 2021.

[43] Q. Luo, X. Yang, and Y. Zhou, "Nature-inspired approach: An enhanced moth swarm algorithm for global optimization," *Math. Comput. Simul.*, vol. 159, pp. 57–92, May 2019.

[44] M. S. Braik, "Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems," *Expert Syst. Appl.*, vol. 174, Jul. 2021, Art. no. 114685.

[45] S. Velliangiri, P. Karthikeyan, V. M. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Eng. J.*, vol. 12, no. 1, pp. 631–639, Mar. 2021.

[46] M. I. Alghamdi, "Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO)," *Sustainability*, vol. 14, no. 19, Sep. 2022, Art. no. 11982.

[47] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Comput.*, vol. 24, no. 1, pp. 205–223, Mar. 2021.

S. V. ASWIN KUMER received the Graduate degree in electronics and communication engineering from the Pallavan College of Engineering, Kanchipuram, in April 2008, the master's degree in embedded system technology from SRM University, Kanchipuram, in May 2012, and the Ph.D. degree for the implementation of image fusion using artificial neural network from SCSVMV (Deemed to be University), Enathur, in February 2019.

He is currently an Associate Professor with the Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation (Deemed to be University), Guntur. He has more than 14 years of teaching experience. His research interests include digital communication and digital signal processing.

N. PRABAKARAN received the M.E. degree in applied electronics from Sathyabama Deemed University, Chennai, in 2007, and the Doctor of Philosophy degree from the Faculty of Engineering, Sathyabama Deemed University, in January 2015. He is currently an Associate Professor in electronics and communication engineering and skilling with Koneru Lakshmaiah Educational Foundation (Deemed to be University), Guntur, Andhra Pradesh. Previously, he was with the School of Electrical and Electronics Engineering, Department of Electronics and Telecommunications Engineering, Sathyabama Deemed University, from June 2006 to July 2017. His research interests include wireless communications networks and the IoT systems.

E. MOHAN received the M.E. degree in computer science engineering from Satyabhama University, the Ph.D. degree in computer science and engineering from Vinayaka Missions University, and the M.B.A. degree from Madras University. He has more than two decades experience in academic field. He is currently a Professor with the Saveetha School of Engineering, SIMATS, Chennai, Tamil Nadu, India. Throughout his career, he is having good academic records of accomplishment and published many refereed journals in the reputed publications. His research interests include image processing, WSN, the IoT, ML and datamining. He titled three books and four scholars successfully completed Doctorate under his guidance.

BALAJI NATARAJAN received the Ph.D. degree in computer science and engineering from Pondicherry University, Puducherry, India, in 2017. He is currently a Professor and the Head of the Department of Computer Science and Engineering, Sri Venkateshwaraa College of Engineering and Technology, Ariyur, Puducherry. He has 15 years of teaching, research, and industry experience. He has published more than 50 research papers in various reputed international journals and conferences. His research interests include web services, service oriented architecture, evolutionary algorithms, artificial intelligence, and machine learning.

G. SAMBASIVAM (Member, IEEE) received the Ph.D. degree in computer science and engineering from Pondicherry University, Puducherry, India. He is currently an Assistant Professor with the School of Computing and Data Science, Xiamen University Malaysia, Sepang, Malaysia. Previously, he was the Dean of the School of Information and Communication Technology, ISBAT University, Uganda. His research interests include artificial intelligence, machine learning, deep learning, graph neural networks, web service computing, and soft computing techniques.

VAIBHAV BHUSHAN TYAGI received the B.Tech. degree from UPTU Lucknow, in 2007, the M.Tech. degree from IIT Roorkee, in 2011, and the Ph.D. degree, in 2015. He is having more than 13 years of research and teaching experience around the globe. He is currently an Associate Professor (ECE) and the Dean FICT of ISBAT University, Kampala, Uganda. He has worked in several administrative and academic positions in India, Ethiopia, and Uganda. His research interests include sensor applications in signal processing, signal modeling, artificial intelligence, and deep learning.