Cloud Task Scheduling using Particle Swarm Optimization and Capuchin Search Algorithms

Gang WANG[1*], Jiayin FENG[2], Dongyan JIA[3], Jinling SONG[4], Guolin LI[5]
Hebei Normal University of Science & Technology, Qinhuangdao 066004, China

***Abstract*—Cloud providers offer heterogeneous virtual machines for the execution of a variety of tasks requested by users. These virtual machines are managed by the cloud provider, eliminating the need for users to set up and maintain their hardware. This makes accessing the computing resources necessary to run applications and services more accessible and cost-effective. The task scheduling problem can be expressed as a discrete optimization issue known as NP-hard. To address this problem, we propose a hybrid meta-heuristic algorithm using the Capuchin Search Algorithm (CapSA) and the Particle Swarm Optimization (PSO) algorithm. PSO excels in global exploration, while CapSA is adept at fine-tuning solutions through local search. We aim to achieve better convergence and solution quality by integrating both algorithms. Our proposed method's performance is thoroughly evaluated through extensive experimentation, comparing it to standalone PSO and CapSA approaches. The findings reveal that our hybrid algorithm outperforms the individual techniques in terms of both total execution time and total execution cost metrics. The novelty of our work lies in the synergistic integration of PSO and CapSA, addressing the limitations of traditional optimization methods for cloud task scheduling. The proposed hybrid approach opens up intriguing directions for future research in dynamic task scheduling, multi-objective optimization, adaptive algorithms, integration with emerging technologies, and real-world deployment scenarios.**

Keywords*—*Cloud computing; virtualization; task scheduling; optimization; resource utilization; capuchin search algorithm; particle swarm optimization

I. INTRODUCTION

Cloud computing is a well-known computing model that hosts and delivers various services via the Internet [1]. It enables users to access computing resources on demand, thus reducing the cost of ownership and IT management. Cloud computing services are provided pay-as-you-go and typically include storage, software, analytics, and networking [2]. The cloud computing paradigm has allowed companies to move, process, and run some of the services and applications in cloud environments, providing convenient access to a wide range of resources easily and conveniently. In addition, these services are tailored to each customer's requirements [3]. Originally, due to the emergence of big data, conventional hardware could not handle heterogeneous workloads flowing onto the infrastructure. Consequently, many IT companies are migrating their infrastructure to the cloud to handle these diverse and heterogeneous tasks. Cloud computing models offer several potential benefits, including flexibility, highly resilient virtual architectures, on-demand services, elasticity, and scalability. Multi-tenant computing environments share resources among users. A scheduler module checks the resource status and allocates it under user requests [4]. Scheduling plays a crucial role in achieving optimal real-time performance in cloud computing. The scheduling algorithms map the tasks into the cloud environment and utilize the available resources, thereby reducing latency and response time for requests and increasing resource utilization and system throughput [5].

The fusion of IoT, big data, artificial intelligence (AI), machine learning (ML), deep learning, feature and channel selection, meta-heuristic algorithms, and association rule mining with cloud computing has ushered in a new era of technological possibilities. IoT connects a vast array of devices, generating immense volumes of data that can be harnessed in the cloud for analysis and processing [6]. Big data, with its inherent complexity and scale, finds a natural fit in cloud computing, which offers the necessary storage and computational capabilities to extract valuable insights [7, 8]. AI and ML form the backbone of intelligent cloud applications, enabling systems to learn from data patterns and make informed decisions [9-11]. Deep learning, a subset of ML, is especially powerful in cloud computing for tasks such as image recognition, natural language processing, and complex data analysis [12-14]. Feature and channel selection play a crucial role in optimizing cloud-based applications by identifying pertinent data attributes and sources, leading to improved efficiency and accuracy [15]. Meta-heuristic algorithms, with their ability to efficiently solve complex optimization problems, facilitate resource allocation, load balancing, and task scheduling in cloud environments [16-19]. Association rule mining is essential for discovering meaningful patterns and relationships within vast datasets, empowering businesses to make data-driven decisions and gain a competitive edge [20]. The integration of these concepts in cloud computing is transformative for businesses and industries alike. Cloud-based IoT solutions facilitate real-time data analysis and decision-making, enabling predictive maintenance, personalized services, and improved operational efficiency. The ability to process and store big data in the cloud ensures data accessibility, scalability, and cost-effectiveness for organizations. Furthermore, AI and ML capabilities in the cloud enable innovative applications like virtual assistants, recommendation systems, and fraud detection.

Many researchers have addressed the problem of scheduling tasks; however, it has remained an NP-hard problem. This means that scheduling tasks is a computationally difficult problem, and finding an optimal solution is not feasible in a reasonable time. As a result, numerous heuristics have been created to solve this issue effectively [21]. Cloud

environments consist of geographically distributed data centers. These data centers are connected by high-speed networks, making it possible to transfer data between them quickly. This enables cloud providers to use distributed scheduling algorithms that can use multiple data centers to solve the task scheduling problem more efficiently. There are thousands of servers in each data center. Each server has an array of virtual machines equipped with various resources, such as storage, CPU, and memory. To execute tasks, groups of virtual machines are allocated to cloud users. It is a complex task to schedule the appropriate resources for the task. In order to assign tasks to virtual resources, it is necessary to examine their characteristics with respect to dependency, length, and size. By factoring in the total execution time for all tasks, task scheduling algorithms can distribute the workload across virtual machines.

In this research paper, we introduce a novel hybrid meta-heuristic algorithm that combines the strengths of the Capuchin Search Algorithm (CapSA) and the Particle Swarm Optimization (PSO) algorithm. The decision to merge these two approaches was motivated by their complementary characteristics in tackling optimization problems. While PSO is renowned for its efficient global exploration capabilities, CapSA excels in local search strategies, making it adept at fine-tuning solutions. The integration of PSO and CapSA allows us to harness their unique strengths synergistically, aiming for improved convergence and solution quality. During the experimentation phase, our hybrid approach demonstrated superior performance to standalone PSO and CapSA, as evident in total execution time and total execution cost metrics. Furthermore, our evaluation of existing methods for cloud task scheduling revealed specific limitations hindering their effectiveness in this context. Traditional optimization algorithms often struggle to find solutions in large search spaces because they lack robust exploration capabilities. On the other hand, local search algorithms may get trapped in local optima, preventing them from achieving global optimality. Our proposed hybrid PSO-CapSA approach effectively addresses these limitations. By leveraging PSO's global exploration capabilities, the algorithm conducts a diverse search across the solution space, mitigating the risk of premature convergence to suboptimal solutions. Additionally, CapSA's local search enhances the precision of the algorithm by refining solutions and escaping local optima. This combination empowers our method to tackle cloud task scheduling challenges more effectively than existing approaches.

The paper is arranged in the following manner. Section II summarizes the literature. Section III discusses the problem statement and the proposed algorithm. Section IV illustrates the findings and analyses. Section V presents the conclusion.

II. Related Work

Dai, et al. [22] propose a novel task-scheduling algorithm that incorporates multiple quality of service criteria into the scheduling process, like reliability, security, expenditure, and time. It combines genetic as well as Ant Colony Optimization (ACO) algorithms. ACO employs the genetic algorithm to generate an initial pheromone efficiently. A four-dimensional quality of service objective is evaluated utilizing a designed fitness function. The optimum resource is then identified using the ACO algorithm. The proposed algorithm was implemented on several real-world tasks and demonstrated significant improvements in the quality of service. The consequences demonstrated that the suggested algorithm achieved better scheduling than existing algorithms.

Tang, et al. [23] proposed the DVFS-enabled Energy-efficient Workflow Task Scheduling (DEWTS) algorithm to reduce energy consumption and ensure the quality of service adhering to deadlines. DEWTS is built upon a dynamic voltage frequency scaling approach that assigns appropriate processing speeds to tasks based on their deadlines. By combining potentially inefficient processors, DEWTS can utilize the slack time repeatedly after servers have been merged by reclaiming the slack time. This ensures that peak performance is maintained and the power wasted is minimized. The DEWTS algorithm can effectively manage servers with different deadlines and dynamically adjust the processors' frequency and voltage. After calculating the starting scheduling sequence for all tasks, DEWTS determines the total makespan and deadline by applying the Heterogeneous-Earliest-Finish-Time (HEFT) algorithm. The underutilized processors are combined by terminating the last node and reallocating the assigned tasks to the processors according to the number of running tasks and the energy consumption of each processor. Results from the experiments demonstrate that DEWTS reduces entire power consumption by up to 46 percent for a variety of parallel applications while balancing scheduling performance based on randomly generated DAG workflows.

Keshanchi, et al. [24] introduced a powerful and enhanced genetic algorithm to optimize task-scheduling solutions. Based on model-checking techniques, they have developed a behavioral modeling approach to verify the algorithm's validity. Next, Linear Temporal Logic (LTL) functions are used to extract the expected specifications. A Labeled Transition System (LTS) is used to obtain optimal results in the validation process. The algorithm was tested on various tasks and provided better performance and scalability than existing methods. The results were validated using the LTS model, and the algorithm was effective in generating optimal task-scheduling solutions. According to the verification outcomes, the validity of the suggested algorithm is assessed with respect to some reachability, fairness, specifications, and deadlock-free performance. Statistical analysis, as well as simulation findings, indicate that the designed approach is superior to traditional heuristic algorithms. The proposed algorithm has been successfully implemented in a deployed system, demonstrating its effectiveness and scalability. The results of the verification process show that the algorithm achieves the desired results with minimal computational cost.

Lin, et al. [25] presented a power efficiency model for cloud servers. To optimize energy consumption in cloud environments, they propose a heuristic task scheduling algorithm (ECOTS) that utilizes the power efficiency of the server to guide task scheduling. Multiple vital factors are taken into account by ECOTS, including degradation of performance, power efficiency models for servers, and resource requirements for tasks, with the goal of reducing the energy consumption of the system without compromising performance. ECOTS is

characterized by its simplicity in terms of time and space and the ability to search globally to find an effective scheduling strategy. An evaluation of the effectiveness of ECOTS was conducted by simulating a heterogeneous cluster environment. It has been demonstrated that the ECOTS algorithm achieves the highest energy efficiency level.

The energy efficiency of task scheduling in cloud data center architectures was studied by Sharma and Garg [26], who also created a new hybrid meta-heuristic algorithm according to the harmony-inspired genetic algorithms. It combines the exploration capabilities of a genetic algorithm with the exploitation capabilities of harmony searches to provide rapid convergence while intelligently sensing both local and global optimal regions without wasting time in local or global optimal regions. Key goals include reducing the time required for computation and the energy it consumes. In contrast, secondary objectives include reducing energy consumption and scheduling execution overhead.

Alsaidy, et al. [27] propose a heuristic algorithm for developing the initialization of the Particle Swarm Optimization (PSO) algorithm. The PSO is initially configured using the minimum completion time (MCT) and longest job to fastest processor (LJFP) algorithms. Both algorithms are tested in terms of their efficiency to minimize total energy consumption, degree of imbalance, and total execution time. A comparison is also made between the proposed algorithms and recent methods. The simulation outcomes demonstrate that the suggested algorithms are more effective and superior to traditional PSO and comparative algorithms.

Saravanan, et al. [28] used the enhanced wild horse optimization algorithm and the levy flight algorithm for task scheduling. This method generates a multi-objective fitness function by optimizing resource utilization and minimizing the makespan. The simulation results indicated that the suggested method outperformed others in a variety of situations. The algorithm was successful in achieving better task scheduling compared to traditional algorithms. It achieved better outcomes regarding completion time, cost, as well as energy efficiency. The proposed method was also found to be scalable and robust, able to handle a large number of tasks and resources.

III. Proposed Method

This section begins with a description of the problem statement. Then, the standard PSO and CapSA algorithms are discussed to determine the basis for the suggested algorithm. This section also examines the fitness function incorporated into the proposed algorithm.

A. Problem Statement

The process of scheduling cloud-based tasks is illustrated in Fig. 1. Initially, users' tasks are placed in a queue. The tasks are then retrieved from the queue and allocated to the available cloud resources. Once the tasks are finished, the results are sent back to the users. This procedure continues until each task is completed. Generally, static and dynamic are the two kinds of task scheduling algorithms. Static scheduling algorithms require detailed information regarding the environment and the tasks. Dynamic scheduling algorithms monitor the system continuously and are capable of balancing workloads. In the following way, the task scheduling problem is described. The task scheduling problem consists of assigning n tasks of users to m heterogeneous virtual machines based on some constraints in order to optimize some objective functions. We have observed that task scheduling is an objective-driven strategy that involves allocating computing resources to specific tasks periodically to maximize one or more objectives. This process requires analysis of data related to the task, such as the resource requirements, the time needed to complete the task, and the task's priority. Optimizing task scheduling can result in significant cost savings by allocating resources efficiently. Furthermore, effective task scheduling can help improve the system's overall performance. In the cloud computing environment, the scheduling policy should address this problem from two different perspectives, customer and cloud provider perspectives. The customer's objective is to lower the cost of executing the tasks, and the cloud provider's objective is to maximize the utilization of the infrastructure. Thus, cost and performance should be considered when designing the scheduling policy. In this regard, scheduling is a major concern in the cloud environment, as it directly impacts the performance of the cloud system as well as the cloud service consumer.

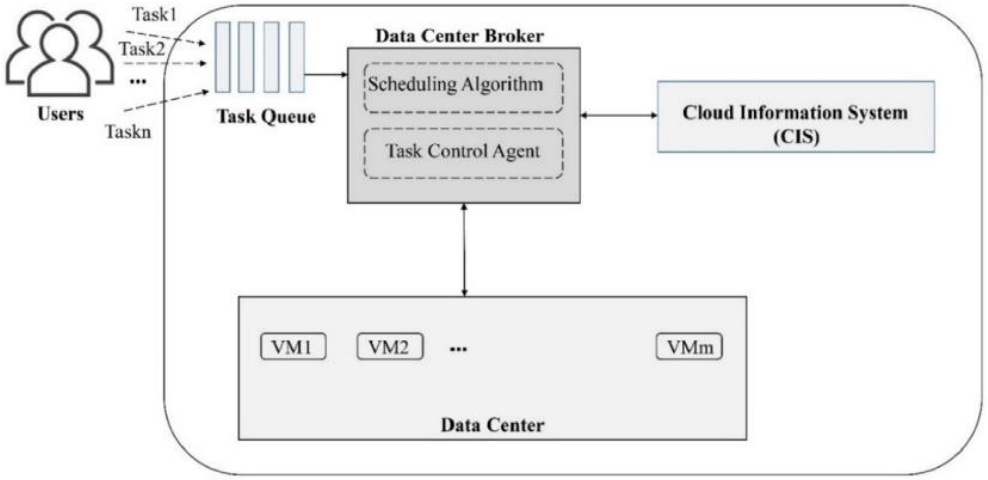

Fig. 1. Task scheduling in cloud computing.

Assume that $VM = [VM_1, VM_2, \dots, VM_m]$ represents the set of virtual machines available within a data center. These virtual machines are provided to cloud users in order to fulfill their task requests. Each virtual machine is equipped with processing power expressed in Millions of Instructions Per Second (MIPS). A cloud broker allocates tasks to appropriate virtual machines according to details of existing resources and requirements of tasks. The collection of tasks to execute on the virtual machines at the data center is expressed as $T = [T_1, T_2, \dots, T_n]$. Tasks are defined by their length and processing requirements, which are specified by Millions of Instructions (MIs).

B. Fitness Function

Fitness functions represent the desired objectives to be optimized. They measure the performance of the problem being solved and direct the search process toward finding the best solution. Fitness functions are generally represented as mathematical functions, which can be used to evaluate the candidate solutions. Fitness functions may be multi-objective from two perspectives: prior and posterior. In an a priori approach, the fitness function is designed to optimize multiple objectives from the beginning. In a posteriori approach, a single-objective fitness function is used, and the multiple objectives are satisfied through post-processing. According to the prior strategy, objectives are given a weight that reflects their significance in producing a single-value function, referred to as a fitness value. The posterior approach produces non-dominant solutions. This paper uses priori principles to develop the fitness function. Total execution time and total execution cost are included in the fitness function. In mathematical terms, the fitness function considered is expressed by Eq. 1.

$$F = a_1 \times TEC + a_2 \times TET \quad (1)$$

Cloud computing is designed to meet users' functional needs while reducing costs. Consequently, a scheduling algorithm should provide users access to their necessary applications at a minimal cost. Costs associated with storage, communication, and execution are all contained in the cost of cloud computing. Execution cost consists of the price per unit interval applied by the virtual machine and the execution time of all tasks executed by that virtual machine. Eq. 2 can be used to calculate the total execution cost of a workflow.

$$TEC_W = \sum_{i \in W, i=1}^{k} \frac{ET_{i,j}}{\tau} \times CO_j : j \in VM_j \quad (2)$$

In Eq. 2, $ET_{i,j}$ measures the time taken to execute task T_i by jth VM, τ is the period during which the user utilizes the resources, and CO_j represents the cost of a type-i VM instance for a unit of time in the cloud data center. The total execution time or makespan is a key metric used to measure the performance of a task scheduling approach. It is the sum of the longest completion times of tasks within a workflow. Optimizing the makespan is an important part of workflow scheduling. Eq. 3 can be used to calculate the makespan of a workflow.

$$TET_W = \max\{CT_i | i = 1,2, \dots, m\} \quad (3)$$

In Eq. 3, CT_i is represented the completion time of task T_i in the workflow. In other words, it is characterized as the variation between task T_i's start and end times. Eq. 4 is used to calculate the completion time. According to Eq. 5, the waiting time of task T_i equals the total completion time of its predecessors. Eq. 6 calculates the execution time of task T_i on VM_j. PE_{unit} represents the size of each core in MIPS, $Num(PE_j)$ indicates how many cores are allocated to the virtual machine VM_j, as well as SZ_{Task} indicates the size of task T_i in MI.

$$CT_i = \begin{cases} ET_i & \text{if pred } (T_i) = 0 \\ WK_i + ET_i & \text{if pred } (T_i) \neq 0 \end{cases} \quad (4)$$

$$WK_i = \begin{cases} 0 & \text{if pred } (T_i) = 0 \\ \max(CT_i) & \text{if pred } (T_i) \neq 0 \end{cases} \quad (5)$$

$$ET_{i,j} = \frac{SZ_{Task}}{Num(PE_j) \times PE_{unit}} \quad (6)$$

C. Proposed Algorithm

The proposed algorithm combines the PSO and Capuchin Search Algorithm (CapSA). The algorithm consists of running the PSO algorithm during the first half of the total iterations, initializing the most optimal solution produced by the PSO algorithm (gbest) to CapSA, and running CapSA for the second half of the total iterations. The best CapSA solution is the most efficient assignment of tasks to virtual machines. After the total iterations are finished, the algorithm takes the best solution from PSO and CapSA and finds the best fitness value. This solution is returned as the final output of the proposed algorithm.

In order to apply any algorithm to a workflow scheduling problem, the problem must be modeled in terms of the tasks that need to be completed, their precedence relationships, the resources and time needed to complete each task, and other factors. Once the problem is accurately modeled, the appropriate algorithm can be applied to find an optimal solution. This problem can be viewed as a mapping between user tasks and virtual machines. As shown in Fig. 2, an array can represent the proposed algorithm's solution. Tasks and assigned VMs are represented by an array's index and array's values, respectively. Population refers to the set of solutions. In the first iteration, the population is first initialized using a random solution. As the algorithm is iterated, the solution is improved. Fig. 3 illustrates a random population initialization. Each population is evaluated to determine its fitness. The population with the highest fitness is considered the most optimal solution. Selection processes are then used to identify solutions for reproduction.

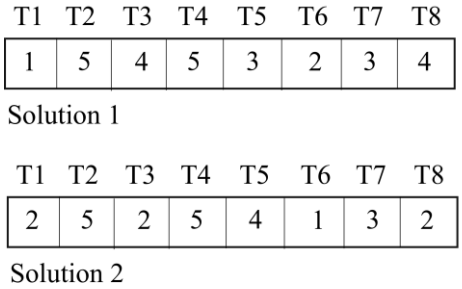

Fig. 2. A solution array.

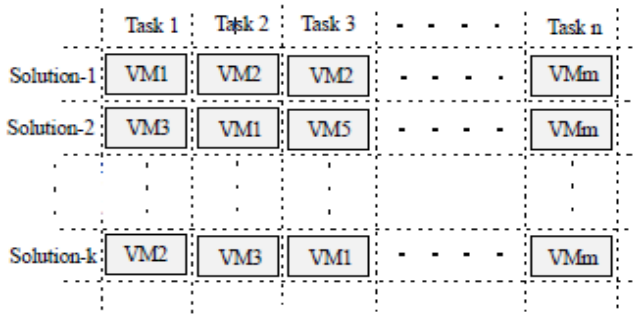

Fig. 3. Initializing the population.

Using Eq. 7, the algorithm determines the execution time and assigns them to the execution time matrix. The element value demonstrates the execution time; for example, $ET_{1,1}$ represents the execution time of task T_1 on VM_1. According to Eq. 8, the cost matrix contains the execution cost of each virtual machine. $C_1, C_2, \dots, C_m$ correspond to the unit execution costs of the virtual machines $VM_1, VM_2, \dots, VM_m$.

$$ET - Mtx = \begin{matrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{matrix} \begin{pmatrix} ET_{1,1} & ET_{1,2} & \dots & ET_{1,m} \\ ET_{2,1} & ET_{2,2} & \dots & ET_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ ET_{n,1} & ET_{n,2} & \dots & ET_{n,m} \end{pmatrix} \quad (7)$$

$$Cost - Mtx = (C_1, C_2, \dots, C_m) \quad (8)$$

The PSO algorithm is an evolutionary algorithm derived from bird swarms or fish schools. It optimizes an issue by iteratively seeking to enhance a candidate solution concerning a given quality measure. It works by moving a population of candidate solutions, known as particles, throughout the search space in accordance with straightforward mathematical formulas. The movement of particles is guided by their best-known position and the best-known positions in the search space, which are updated as other particles find better positions. Position pi and velocity vi are the two parameters that describe a particle. Pbest and gbest are two parameters that affect the particle's position. Gbest represents the best position of neighboring particles, while Pbest represents the best position visited by the particle. Position and velocity are updated after each iteration of the algorithm. The velocity is updated using Eq. 9.

$$v_{id}^{t} = wv_{id}^{t-1} + c1r1(pbest_{id}^{t} - x_{id}^{t}) + c2r2(gbest_{id}^{t} - x_{id}^{t}) \quad (9)$$

In Eq. 9, v_{id}^{t} corresponds to the velocity of the ith particle in the dth dimension on iteration t. Eq. 10 is used to update the particle's position. In Eq. 10, p_{id}^{t} displays the position of particle i in the dth dimension at time t.

$$p_{id}^{t+1} = p_{id}^{t} + v_{id}^{t} \quad (10)$$

The CapSA is inspired by capuchin monkeys' dynamic behavior when navigating between branches and riverbanks for food [29, 30]. It uses three great navigation methods: jumping, swinging, and climbing. According to CSA, capuchin populations can be categorized into two important groups: the leaders and the followers. The leaders guide their followers and keep track of each other. Three principal concepts are used by capuchin leaders and swarm members when looking for food sources: explore independently, cooperate in locating better food sources, and lead by example. In the search for food sources, the Alpha males, as well as Alpha females, lead the other group members. By assisting the other group members in discovering food sources, the Alpha male serves as a leader. An iterative solution is determined using the above strategies. In order to select the best features, CapSA follows the following steps:

- CapSA initialization: Like other meta-heuristic algorithms, CapSA generates a predetermined number of individuals (i.e., capuchins) to serve as its population. Each individual represents a potential answer to the task scheduling problem in this paper. A capuchin array can be viewed as a d-dimensional matrix. The matrix for the initial population is shown in Eq. 11.

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & \cdots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^n & x_2^n & \cdots & \cdots & x_d^n \end{bmatrix} \quad (11)$$

- In Eq. 11, n stands for the number of capuchins, d demonstrates the number of variables, xid indicates the dimension of the ith capuchin, and x signifies the position of the capuchins. Eq. 12 is used to compute the first situation of each capuchin.

$$X^i = ub_j + t \times (ub_j - lb_j) \quad (12)$$

- In Eq. 12, ub_j and lb_j define upper and lower bounds for the jth capuchin, respectively, as well as t varies uniformly from 0 to 1.

- Generation of solutions by capuchins: In CapSA, a new population originates from the position of the capuchin, the best capuchin, and F, which represents the food source. In a d-dimensional search domain, capuchins should iteratively update this food source. The following equation is used in our proposed algorithm to generate new solutions. In Eq. 13, P_{bf} is the probability of the balance generated by the tail of the capuchin during leaping movements, θ is the angle at which capuchins jump, v_j^i is the velocity of the ith capuchin in the dimension j, g is gravity equal to 9.81, F_j refers to the food's position in the dimension, and x_j^i refers to the position of the alpha capuchins and their following capuchins within the dimension j. A capuchin's jump angle can be determined using Eq. 14, where r is a uniformly generated number from 0 to 1.

$$x_j^i = F_j + \frac{P_{bf}(v_j^i)^2 \sin(2\theta)}{g} \quad (13)$$

$$\theta = \frac{3}{2} r \quad (14)$$

IV. Experimental Results

In this section, the proposed algorithm's performance is compared with previous algorithms under Montage workflows

with 50 and 100 tasks. The proposed algorithm was simulated and evaluated using the WorkflowSim -1.1 toolkit, an extension of Cloudsim. Comparisons are made among the proposed algorithm as well as standalone PSO and CapSA. Table I provides a summary of the simulation parameters that were utilized for evaluating the algorithm.

1) Total execution time comparison (50 tasks): As depicted in Fig. 4, the algorithms were compared regarding the total execution time as iterations increased from 50 to 500. The proposed algorithm consistently outperformed CapSA in all iterations. While slight degradations were observed in comparison with PSO at several iterations, the proposed algorithm exhibited an average improvement in performance. These findings indicate that the proposed algorithm is more effective than both CapSA and PSO in completing tasks in a shorter time. Moreover, the results demonstrate that the proposed algorithm maintains consistent performance as the number of iterations increases.

2) Execution costs comparison (50 tasks): Fig. 5 illustrates the comparison of execution costs among the algorithms. The proposed algorithm exhibited improvements of 9.7%, 1.4%, 12.6%, 10.6%, 3.8%, 15.2%, 11.1%, 14.4%, 19.9%, and 13.3% compared to CapSA for all considered iterations. In terms of PSO, the proposed algorithm demonstrated a significant drop in total execution cost of up to 10%. Overall, the experimental results indicate that the proposed algorithm is more efficient than both CapSA and PSO. Additionally, the proposed algorithm exhibited faster convergence compared to these algorithms.

3) Total execution time comparison (100 tasks): Fig. 6 demonstrates that for 100 tasks, the proposed algorithm consistently reduced the total execution time compared to CapSA and PSO in all iterations.

4) Execution costs comparison (100 tasks): In Fig. 7, the algorithms were compared based on their total execution costs for 100 tasks. The proposed algorithm outperformed CapSA in all iterations, with a decrease of 0.6%, 0.3%, 0.2%, 1%, 0.9%, 1.2%, 1.4%, 0.4%, 1.1%, and 1.3% for all considered iterations. Compared to PSO, the proposed algorithm showed improvements of 0.11%, 0.04%, 0.02%, and 0.15% for iterations 200, 250, 300, and 500, respectively. These results indicate that the proposed algorithm is more effective and efficient in finding better solutions for the given problem compared to CapSA and PSO. Furthermore, it consistently produces better results as the number of iterations increases. Overall, the presented results demonstrate the superior performance of the proposed hybrid algorithm in terms of both total execution time and execution costs when compared to the standalone PSO and CapSA algorithms..

TABLE I. SIMULATION PARAMETERS

Parameter	Value
VM policy	Time shared
Number of processors	1
Bandwidth	1000
Ram	512 MB
MIPS	1000
Number of virtual machines	5
Number of tasks	50-100

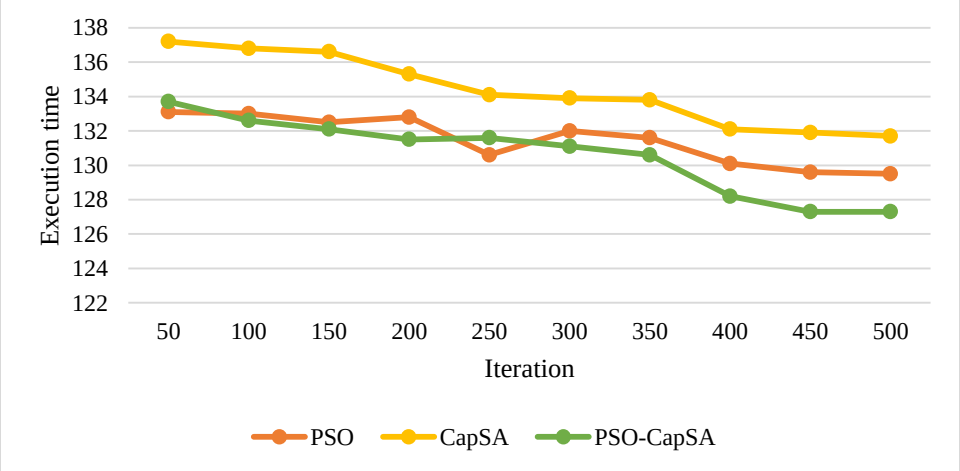

Fig. 4. Execution time for 50 tasks.

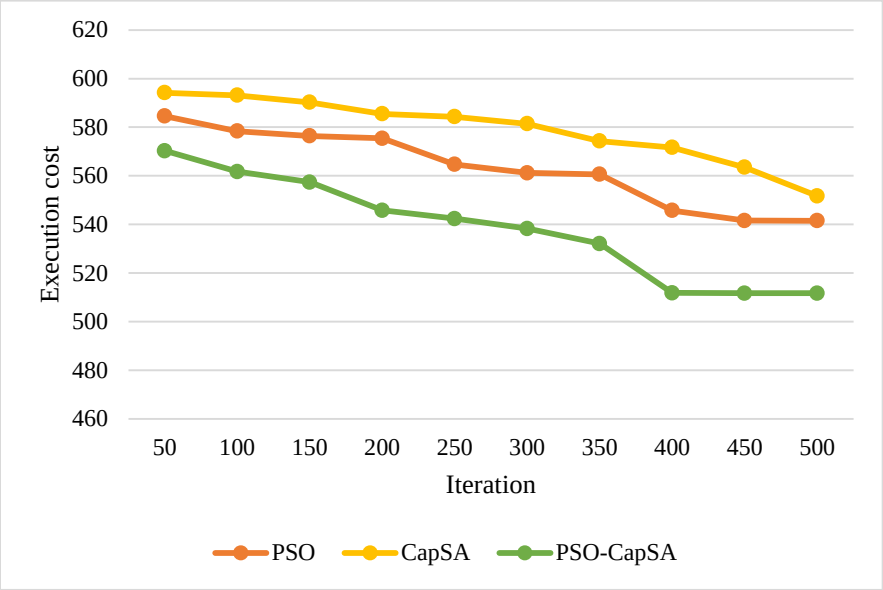

Fig. 5. Execution cost for 50 tasks.

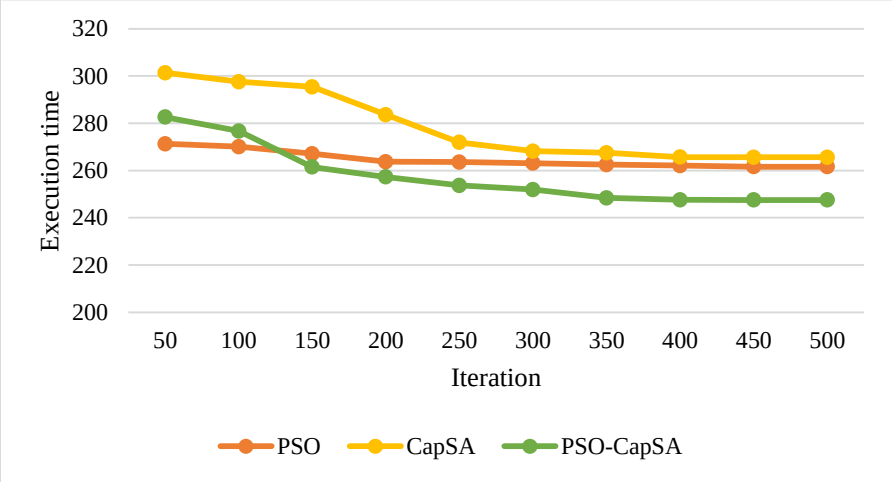

Fig. 6. Execution time for 100 tasks.

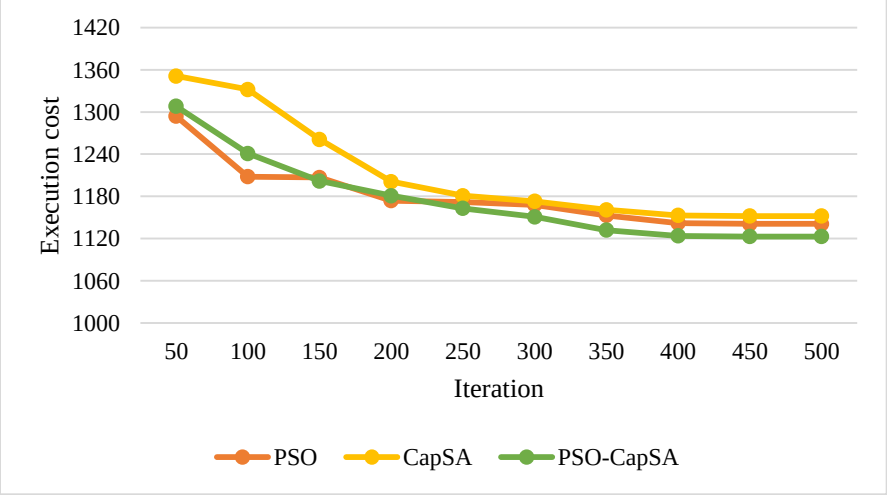

Fig. 7. Execution cost for 100 tasks.

V. CONCLUSION

Cloud services offer a wide range of solutions to the varying computing challenges in the real world. In order to accomplish a variety of tasks, users access various cloud services. Services provided by different service providers are used to handle such tasks. In this paper, a novel hybrid meta-heuristic algorithm according to the PSO algorithm as well as CapSA is proposed. The proposed algorithm uses PSO and CapSA to employ complementary global and local search strategies. Based on simulation results from the Cloudsim simulator, our algorithm outperformed standalone PSO and CapSA on both total execution cost and total execution time measures. There are several potential open problems in this domain for further research. Extending the proposed algorithm to handle dynamic task scheduling scenarios, where tasks and resource availability fluctuate in real-time, will be essential for addressing real-world dynamic workload challenges. This adaptability will ensure the algorithm remains efficient and effective in dynamic cloud environments. Exploring multi-objective optimization techniques to optimize cloud task scheduling for various performance metrics, such as energy consumption, resource utilization, and quality of service (QoS), can lead to more comprehensive and versatile solutions. Exploring the integration of the proposed algorithm with emerging technologies, such as edge computing and AI-driven resource management, can open up new possibilities for efficient and intelligent task scheduling. This integration will leverage the advancements in these fields to enhance the scheduling process and overall cloud system performance. Conducting real-world deployments and large-scale experiments to assess the scalability and applicability of the proposed algorithm in practical cloud computing environments will be valuable for validating its effectiveness.

REFERENCES

[1] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[2] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[3] O. Ali, A. Shrestha, J. Soar, and S. F. Wamba, "Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review," International Journal of Information Management, vol. 43, pp. 146-158, 2018.

[4] V. Hayyolalam, B. Pourghebleh, and A. A. Pourhaji Kazem, "Trust management of services (TMoS): Investigating the current mechanisms," Transactions on Emerging Telecommunications Technologies, vol. 31, no. 10, p. e4063, 2020.

[5] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective Task Scheduling in cloud-fog computing using goal programming approach," Cluster Computing, vol. 25, no. 1, pp. 141-165, 2022.

[6] A. Peivandizadeh and B. Molavi, "Compatible authentication and key agreement protocol for low power and lossy network in IoT environment," Available at SSRN 4194715, 2022.

[7] M. Ilbeigi, A. Morteza, and R. Ehsani, "Emergency Management in Smart Cities: Infrastructure-Less Communication Systems," in Construction Research Congress 2022, pp. 263-271.

[8] M. Javidan, H. Esfandi, and R. Pashaie, "Optimization of data acquisition operation in optical tomography based on estimation theory," Biomedical optics express, vol. 12, no. 9, pp. 5670-5690, 2021.

[9] M. Bagheri et al., "Data conditioning and forecasting methodology using machine learning on production data for a well pad," in Offshore Technology Conference, 2020: OTC, p. D031S037R002.

[10] B. M. Jafari, X. Luo, and A. Jafari, "Unsupervised Keyword Extraction for Hashtag Recommendation in Social Media," in The International FLAIRS Conference Proceedings, 2023, vol. 36.

[11] C. Han and X. Fu, "Challenge and Opportunity: Deep Learning-Based Stock Price Prediction by Using Bi-Directional LSTM Model," Frontiers in Business, Economics and Management, vol. 8, no. 2, pp. 51-54, 2023.

[12] R. Soleimani and E. Lobaton, "Enhancing Inference on Physiological and Kinematic Periodic Signals via Phase-Based Interpretability and Multi-Task Learning," Information, vol. 13, no. 7, p. 326, 2022.

[13] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Journal of Water Reuse and Desalination, 2022.

[14] S. Vairachilai, A. Bostani, A. Mehbodniya, J. L. Webber, O. Hemakesavulu, and P. Vijayakumar, "Body Sensor 5 G Networks Utilising Deep Learning Architectures for Emotion Detection Based On EEG Signal Processing," Optik, p. 170469, 2022.

[15] M. Javidan, M. Yazdchi, Z. Baharlouei, and A. Mahnam, "Feature and channel selection for designing a regression-based continuous-variable emotion recognition system with two EEG channels," Biomedical Signal Processing and Control, vol. 70, p. 102979, 2021.

[16] S. Aghakhani and M. S. Rajabi, "A new hybrid multi-objective scheduling model for hierarchical hub and flexible flow shop problems," AppliedMath, vol. 2, no. 4, pp. 721-737, 2022.

[17] S. Aghakhani, A. Larijani, F. Sadeghi, D. Martín, and A. A. Shahrakht, "A Novel Hybrid Artificial Bee Colony-Based Deep Convolutional Neural Network to Improve the Detection Performance of Backscatter Communication Systems," Electronics, vol. 12, no. 10, p. 2263, 2023.

[18] S. Mahmoudinazlou and C. Kwon, "A Hybrid Genetic Algorithm for the min-max Multiple Traveling Salesman Problem," arXiv preprint arXiv:2307.07120, 2023.

[19] S. Mahmoudinazlou and C. Kwon, "A Hybrid Genetic Algorithm with Type-Aware Chromosomes for Traveling Salesman Problems with Drone," arXiv preprint arXiv:2303.00614, 2023.

[20] M. Shahin et al., "Cluster-based association rule mining for an intersection accident dataset," in 2021 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube), 2021: IEEE, pp. 1-6.

[21] Y. Kumar, S. Kaul, and Y.-C. Hu, "Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey," Sustainable Computing: Informatics and Systems, vol. 36, p. 100780, 2022.

[22] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing," in 2015 7th international conference on intelligent human-machine systems and cybernetics, 2015, vol. 2: IEEE, pp. 428-431.

[23] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," Journal of Grid Computing, vol. 14, pp. 55-74, 2016.

[24] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing," Journal of Systems and Software, vol. 124, pp. 1-21, 2017.

[25] W. Lin, W. Wang, W. Wu, X. Pang, B. Liu, and Y. Zhang, "A heuristic task scheduling algorithm based on server power efficiency model in cloud environments," Sustainable computing: informatics and systems, vol. 20, pp. 56-65, 2018.

[26] M. Sharma and R. Garg, "HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers," Engineering Science and Technology, an International Journal, vol. 23, no. 1, pp. 211-224, 2020.

[27] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 6, pp. 2370-2382, 2022.

[28] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, p. 24, 2023.

[29] M. Mohseni, F. Amirghafouri, and B. Pourghebleh, "CEDAR: A cluster-based energy-aware data aggregation routing protocol in the internet of things using capuchin search algorithm and fuzzy logic," Peer-to-Peer Networking and Applications, pp. 1-21, 2022.

[30] M. Braik, A. Sheta, and H. Al-Hiary, "A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm," Neural computing and applications, vol. 33, pp. 2515-2547, 2021.