Enhanced Task Scheduling Algorithm Using Harris Hawks Optimization Algorithm for Cloud Computing

Fang WANG
Computer School, Hubei University of Education, Wuhan 430205, China

***Abstract*—Amongst the most transformational technologies nowadays, cloud computing can provide resources such as CPU, memory, and storage over secure internet connections. Due to its flexibility and resource availability with guaranteed QoS, cloud computing allows comprehensive business and research adoptions. Despite the rapid development, resource management remains one of the significant challenges, especially handling task scheduling efficiently in this environment. Task scheduling strategically assigns tasks to available resources so that Quality of Service (QoS) metrics are effectively related to response time and throughput. This paper proposes an Enhanced Harris Hawks Optimization (EHHO) algorithm for scheduling cloud tasks to mitigate the common limitations found in existing algorithms. EHHO integrates a dynamic random walk strategy, enhancing exploration capabilities to avoid premature convergence and significantly improving scalability and resource allocation efficiency. Simulation outcomes reveal that EHHO minimizes makespan by up to 75%, memory usage by up to 60%, execution time by up to 39%, and cost by up to 66% compared to state-of-the-art algorithms. These benefits demonstrate that EHHO can optimize resource allocation while being highly scalable and reliable. Consistent performance over various stacks such as Kafka, Spark, Flink, and Storm further evidences the superiority of EHHO in handling complex scheduling challenges in dynamic cloud computing environments.**

Keywords*—*Cloud computing; optimization; task scheduling; Harris Hawks Optimization; resource allocation; quality of service

I. INTRODUCTION

The Internet of Things (IoT) symbolizes change whereby many devices, from simple sensors and actuators to various everyday objects, are connected via the Internet to communicate and share information [1]. Objects like sensors and actuators form communication grids in healthcare, manufacturing, and smart cities. However, with the growth of IoT applications, volumes of generated data are huge and require immense processing and colossal storage. More importantly, real-time analytics implies vast demand [2]. Thus, cloud computing has become the backbone of IoT systems for extendable resources and rugged data management capabilities beyond IoT devices [3, 4].

Cloud computing enables customers to use the services with the help of the Internet on a pay-per-usage basis [5]. Cloud services include within their ambit a broad range of services, namely Software as a Service (SaaS), Platform as a Service (Paas), Communication as a Service (CaaS), Data storage as a Service (DaaS), and Infrastructure as a Service (IaaS) [6]. These services allow cloud providers to provide utility-based resources whose usage supports diversified needs for IoT.

The physical servers and switches in the backbone layer of cloud computing are operated and scaled by the cloud service provider effectively as per user requirements [7]. It efficiently allocates hardware resources at a blistering pace. In terms of software, the supervisor runs these hardware resources, a hypervisor, middleware, etc. [8]. The operating system implements hardware functionalities and develops user and application communication [9]. It allows the hypervisor to create Virtual Machines (VMs) on cloud servers with specified hardware configurations and software stacks [10]. This, in turn, enables a further increase in service availability because virtualization facilitates easy service migration even during hardware failures. It is also accompanied by a tremendous rise in hardware utilization compared to the non-virtualized environment [11]. Recent advancements in reinforcement learning applications, particularly in mobile robotics, have showcased its ability to enhance decision-making and resource management in dynamic environments. Similarly, reinforcement learning's adaptive capabilities, as demonstrated in SLAM tasks, highlight its potential for optimizing virtualization and scalability in cloud computing infrastructures [12].

The middleware arranges the running and interaction of tasks on cloud servers transparently. The three fundamental types of software infrastructure are PaaS, SaaS, and IaaS [13]. IaaS allows users to create multiple VMs on servers as needed, enhancing computational resource utilization. SaaS permits users to store and access unlimited amounts of data in a minute on remotely located servers. PaaS provides secure, reliable communication services and an application development platform accessible via APIs. Lastly, the application tier enables the user to use applications stored in the cloud through the Internet, allowing quick and easy access without installation or updates locally.

Effective task scheduling is vital for managing resource allocation, execution time, and QoS in cloud-supported IoT environments. Scheduling can be classified into two types: static and dynamic approaches. In static scheduling, tasks are assigned to available machines based on a predefined strategy, whereas in dynamic scheduling, instantaneous conditions are considered to adjust resource allocations. Real-time scheduling techniques ensure priority tasks with tight timing constraints.

Task scheduling in cloud computing is complex and an NP-hard problem, directly influencing the system performance regarding resource utilization, response time, and energy consumption. In this regard, we propose an Enhanced Harris Hawks Optimization (EHHO) algorithm to optimize task scheduling problems in cloud environments. EHHO features a

novel dynamic random walk to reinforce exploration and avoid premature convergence issues, enhancing scalability, resource allocation, and energy management.

By integrating these enhancements, EHHO provides a high-performance solution for complex scheduling requirements in cloud-supported IoT systems, contributing to more efficient, reliable, and cost-effective service delivery. The following sections detail EHHO's methodology, implementation, and performance advantages over existing algorithms, underscoring its potential as a leading approach to resource management in cloud computing.

The remainder of this paper is arranged as follows: Section 2 discusses related research and highlights gaps in existing scheduling approaches. Section 3 presents the problem statement and explains the challenges of cloud task scheduling. Section 4 presents the proposed algorithm. Section 5 summarizes the experimental results and performance analysis. Section 6 discusses the practical implications and challenges. Finally, Section 7 concludes the paper and suggests future directions.

II. Related Work

Shukri, et al. [14] formulated an Enhanced Multi-Verse Optimizer (EMVO) to optimize task scheduling in cloud computing contexts. The developed algorithm incorporates a new mechanism to reserve the most optimal solution from each iteration and inject it back into the population after a predefined interval to leverage better exploration and exploitation capabilities. The proposed approach minimizes task execution time and considers factors like task length, cost, and power consumption. The combination of local and global search and the core components of MVO has caused EMVO to overcome the weaknesses inherent in traditional task scheduling algorithms. Comparisons with the original Particle Swarm Optimization (PSO) and MVO have revealed the efficiency of the proposed EMVO in decreasing the makespan while improving resource utilization.

Natesan and Chokkalingam [15] developed a new Mean Grey Wolf Optimization (MGWO) algorithm to solve cloud computing scheduling issues. The study aims to optimize energy consumption. MGWO performance was evaluated using the Cloudsim toolkit under baseline workload conditions. From the simulation results, it could be revealed that MGWO substantially outperforms competing algorithms in optimizing these crucial performance metrics.

Mapetu, et al. [16] proposed a new binary PSO algorithm to cope with cloud computing load and task scheduling issues. The suggested technique embraces a formula that minimizes the overall difference in execution time between different VMs while keeping some optimization criteria. A dedicated particle position updating strategy was adopted for enhanced load balancing. The numerical evidence verifies that the algorithm performs better than the previous meta-heuristic and heuristic approaches for optimizing load balancing and task scheduling.

Liu [17] developed an effective task scheduling approach using an adaptive Ant Colony Optimization (ACO) algorithm in cloud computing contexts. Pheromone adaptation is introduced into the procedure to accelerate convergence; thus, prematurity can be reduced. In the cloud environment, a multi-objective optimization function, which minimizes cost and time for task execution, reduces load imbalance and maximizes resource utilization, is implemented by optimized ACO. It has been proved by comparison analysis that, compared with traditional ACO, the proposed approach can always guarantee better performance in solution quality, convergence speed, and overall system efficiency, especially in handling large-scale task scheduling challenges.

Zhou, et al. [18] presented a hybrid task scheduling method based on an improved Genetic Algorithm (GA) combined with a greedy algorithm. This algorithm was designed to converge on optimal solutions in lower iteration numbers of the search process than compared approaches. It aimed at response time, completion time, and QoS performance metrics. Experimental results demonstrate that hybrid GA performs much better than existing algorithms in task-scheduling optimization than existing algorithms.

Abualigah and Diabat [19] proposed, incorporating the combination of Ant Lion Optimization (ALO) adapted to the concept of Differential Evolution (DE) to address many-objective task-scheduling issues in cloud computing settings. Elite-based DE enhanced ALO's exploitation and exploration capability, saving it from premature convergence. The effectiveness of the suggested algorithm has been tested on modeled and real-world datasets using the Cloudsim simulation environment. Additionally, experiments proved that the hybrid ALO method outperformed the other optimization algorithms regarding continuous convergence rate, especially for large-scale scheduling problems.

Panda, et al. [20] introduced a new multi-paired task scheduling algorithm for cloud computing by utilizing the Hungarian algorithm. With this approach, the logic efficiently resolves unbalanced workloads based on the pairing strategy for task scheduling. The algorithm outperformed the Hungarian Algorithm with Converse Lease Time, the Hungarian Algorithm with Lease Time, and First-Come-First-Served baselines in large-scale simulations.

Tamilarasu and Singaravel [21] proposed an Improved Coati Optimization Algorithm (ICOA) for critical challenges in cloud computing, namely lengthy scheduling times, excessive costs, and unbalanced VM loads. A task distribution and scheduling scheme involving VMs, time, and cost, was developed. A dual-objective fitness function is employed to optimize resource utilization and makespan. In the ICOA, an exploitation strategy has been incorporated to prevent the solution from converging prematurely and, hence, to enhance local search capabilities. Simulation results demonstrated the superiority of the ICOA over conventional metaheuristic task scheduling algorithms at improving makespan, success rate, turnaround efficiency, and overall system availability.

Abualigah, et al. [22] offered an enhanced hybrid optimization algorithm for cloud task scheduling, which combines Jaya algorithm strengths with Synergical Swarm Optimization (SSO) and a Levy flight. This new approach efficiently balances exploration and exploitation to accelerate and prevent premature convergence. In integrating the best of Jaya and SSO, this algorithm uses their complementary

analytics capabilities to drive an optimal assignment of tasks and allocate resources. The experimental investigation against existing methodologies confirmed the algorithm's superior scalability, convergence speed, solution quality, and performance.

Behera and Sobhanayak [23] proposed a hybrid meta-heuristic approach using GA and Gravitational Search Algorithm (GSA) for multi-target optimization of task scheduling in cloud computing. The authors addressed the NP-hard challenge of efficiently managing an exponentially growing search space while enhancing system performance. The proposed approach leveraged strengths from GA and GSA in improving the Quality of Service (QoS) measures: energy consumption, resource utilization, and makespan. As tested under CloudSim with standard, real-time, and artificial workloads, it improved degree of imbalance by about 12%, resource utilization by 9%, response time by 7%, and energy consumption by 6%.

Khademi Dehnavi, et al. [24] proposed a hybrid GA for efficient and dependable task scheduling across heterogeneous cloud computing environments. The method models an NP-hard optimization problem in scheduling to minimize costs, time, and failures. HGA introduces two novel mutation and crossover operators in global search. It also implements a localized "Walking around" step to improve solutions. Simulation runs on twelve scenarios revealed significant cost reductions compared to state-of-the-art techniques: a 14.1% reduction in makespan, an 18.7% monetary cost reduction, and a 42.3% decrease in failure cost.

Gong, et al. [25] introduced the Enhanced Marine Predator Algorithm (EMPA) for the task scheduling challenges in the cloud computing environment. This approach incorporates the operators of the Whale Optimization Algorithm (WOA) operators, nonlinear inertia weight coefficients, and Golden Sine strategies to minimize makespan while optimizing resource utilization. Simulation runs using synthetic and GoCJ datasets showed that EMPA outperformed GWO, SCA, PSO, and WOA in makespan, resource utilization, and degree of imbalance, positioning EMPA as a very effective scheduling solution in cloud environments.

Pabitha, et al. [26] suggested a new scheduling algorithm, the Chameleon and Remora Search Optimization Algorithm (CRSOA), to tackle the task-scheduling issues arising in cloud environments due to uncertain user demands. In this proposed technique, the Chameleon Search Algorithm (CSA) is combined with the Remora Search Optimization Algorithm (RSOA) to deliver an efficient resource utilization approach that takes into consideration parameters like MIPS and network bandwidth to ensure load balancing while imposing minimal scheduling cost and, at the same time, reduced makespan. The experimental results show that the makespan reduction achieved by CRSOA is 18.9%, cost reduction is 22.1%, and the improvement in load balancing is 20.5% against baseline metaheuristic algorithms.

TABLE I. AN OVERVIEW OF RECENT TASK SCHEDULING ALGORITHMS FOR CLOUD COMPUTING

References	Algorithm	Pros	Cons
[14]	Enhanced multi-verse optimizer	Efficiently reduces makespan and improves resource utilization through enhanced exploration and exploitation mechanisms.	Limited focus on real-time dynamic scheduling challenges.
[15]	Mean grey wolf optimization	Optimizes energy consumption and makespan effectively under baseline workloads.	Does not account for task heterogeneity or scalability in large datasets.
[16]	Binary particle swarm optimization	Superior load balancing with tailored particle updating strategies reduces execution time variance.	Lacks emphasis on cost-efficiency and energy consumption.
[17]	Adaptive ant colony optimization	Enhanced convergence rate and solution quality; minimized cost, execution time, and load imbalance.	Limited applicability for large-scale dynamic task scheduling.
[18]	Hybrid genetic algorithm with greedy	Faster convergence with improved QoS metrics such as response time and completion time.	Focuses primarily on search process efficiency, with limited exploration capabilities.
[19]	Ant lion optimization with differential evolution	Enhanced convergence rates for many-objective problems; robust against premature convergence.	Complexity increases the computational costs for large-scale scheduling tasks.
[20]	Multi-paired task scheduling	Effectively handles unbalanced workloads; superior in minimizing layover times.	Limited applicability to multi-objective or heterogeneous scheduling scenarios.
[21]	Improved coati optimization algorithm	Dual-objective optimization improves makespan and system availability and prevents premature convergence.	No explicit consideration of energy efficiency metrics.
[22]	Jaya with synergistic swarm optimization	Balances exploration and exploitation; achieves high solution quality and convergence speed.	Performance under real-time or uncertain environments is not evaluated.
[23]	Genetic algorithm and gravitational search algorithm	Improves energy consumption, makespan, and resource utilization; suitable for QoS optimization.	Focus on standard workloads with limited scalability for heterogeneous tasks.
[24]	Hybrid genetic algorithm	Cost-efficient with significant reductions in makespan, monetary cost, and failure cost.	Limited application to real-time dynamic and heterogeneous scheduling problems.
[25]	Enhanced marine predator algorithm	Superior makespan reduction, resource utilization, and imbalance handling.	Lack of scalability for highly complex or real-time cloud scheduling tasks.
[26]	Chameleon and remora search optimization algorithm	Effectively minimizes scheduling costs and makespan under uncertain user demands.	High computational complexity for large-scale environments.
[27]	Horse herd–squirrel search algorithm	Demonstrates significant advantages in cost, energy, and makespan reduction.	Limited evaluation under multi-cloud or distributed cloud environments.

Parthasaradi, et al. [27] proposed a hybrid meta-heuristic for cloud computing task scheduling: the Horse Herd–Squirrel Search Algorithm (HO–SSA). This protocol combined SSA and the Horse Herd Optimization Algorithm (HOA) to increase cost efficiency, energy utilization, and scheduling performance. Furthermore, the proposed HO–SSA showed significant superiority and reduced up to 22.2% regarding tasks' cost scheduling costs, 9.68% regarding energy consumption, and makespan compared with SSA, HOA, and TSA.

As summarized in Table I, recent task-scheduling algorithms excel at optimizing specific metrics such as makespan, resource utilization, or cost. However, deficiencies remain in addressing scalability, real-time scheduling, and energy efficiency in heterogeneous and dynamic cloud environments. Although algorithms like EMPA have proven efficient in resource utilization, and huge cost reductions have been achieved in HO-SSA, few have provided a balanced approach to large-scale and multi-cloud comprehensive optimization problems for energy efficiency, load balancing, and QoS. Furthermore, most of those methods lack real-time adaptability to uncertain user demands. Under such gaps, the proposed algorithm operates under an integrated dynamic exploration and exploitation strategy, aiming for optimal resource allocation scalability while enhancing performance in various cloud environments.

III. Problem Statement

Scheduling tasks in cloud computing environments is critical for effective and efficient execution, whereby resources are assigned according to user requests. Multiple layered architectures have been developed in cloud computing to offer these utility-based services. Fig. 1 depicts this kind of layered architecture. Each layer addresses specific functionalities, from data storage and processing to application development and communication support, and enables IoT applications to operate efficiently without major investments in local infrastructure. Table II provides a list of abbreviations and symbols used throughout the paper.

TABLE II. Symbols and Definitions

Symbol	Description	Symbol	Description
T	Cloud tasks	c_{ij}	Binary variable indicating task i assigned to virtual machine j
V	Cloud virtual machines	x_{ij}	Association between a virtual machine and a task
n	Total number of tasks	LB	Lower bound of the solution space
m	Total number of virtual machines	CT	Convergence time
VR	Collection of virtual machine resources	J	Randomization factor
$MIPS$	Millions of instructions per second capability of a CPU	$X_m(i)$	Updated position after applying random walk strategy
CU_j	Compute units capacity of the j^{th} virtual machine	$O(x)$	Objective function for optimization x
L_i	Task duration for the i^{th} task	$x(t+1)$	Position of a hawk in the next iteration
ET_{ij}	Execution time for the i^{th} task on the j^{th} virtual machine	$x_{random}(t)$	Random position of a hawk
BT_j	Busy period of the j^{th} virtual machine	$x_{rabbit}(t)$	Position of the prey
E	Rabbit's escaping energy	$x_{mean}(t)$	Average position of the hawk population
t	Current iteration number	UB	Upper bound of the solution space
Max_iter	Maximum number of iterations	c	Random walk deviation control constant
$\Delta x(t)$	Difference between prey and hawk positions	$rand$	Random number between 0 and 1

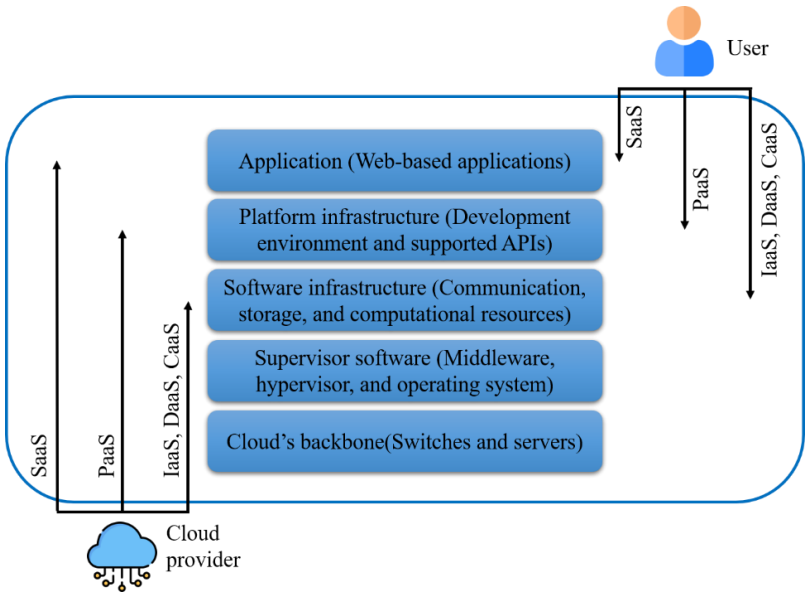

Fig. 1. Multi-layer design of cloud computing.

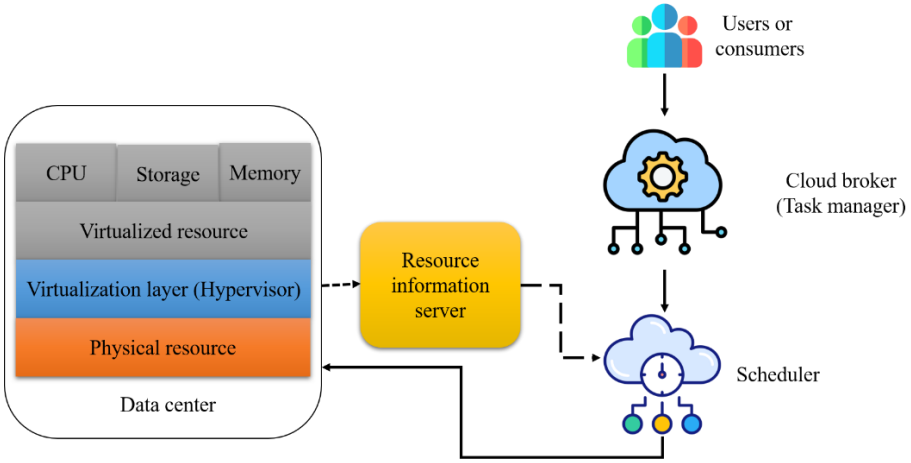

Fig. 2. Proposed framework for task scheduling.

The quality of the service will be directly affected by the following scheduling algorithm, affecting parameters such as execution time and operational costs. The existing frameworks comprise a cloud broker and Resource Information Servers (RIS) to ensure optimum scheduling and provide runtime information about resource availability and VM capabilities. While the above systems consolidate data from physical and virtualized infrastructures, significant research gaps exist in scheduling. A schematic of this framework is provided in Fig. 2.

Most traditional approaches fail to balance exploration and exploitation and thus lead to premature convergence or suboptimal resource allocation in a dynamic, real-time environment. In addition, most methodologies cannot adapt to heterogeneous workloads or consider uncertain factors such as fluctuating resource demand and VM performance. Other multi-objective optimizations, like minimal makespan, energy consumption, cost, and maximal resource utilization, have also been inadequately performed by most current strategies.

These gaps highlight the need for advanced algorithms capable of dynamic decision-making, enhanced exploration of solution spaces, and robust handling of diverse workloads. The EHHO algorithm addresses these challenges by integrating dynamic random walk strategies and stochastic adjustments to produce superior task scheduling performance, ensuring scalability and efficiency in complex cloud environments.

Cloud data centers feature an extensive range of actual machines containing functional VMs. The VMs function as the underlying infrastructure for the execution of user tasks. Tasks assigned to a particular VM are based on the task's requirements. Two alternative concepts of scheduling are common inside the cloud environment. The initial step involves identifying and allocating servers specifically intended for supporting VMs. The specific scheduling variant significantly enhances data center productivity, reduces power usage, and optimizes resource utilization. The impact of such a phenomenon is notably significant on cloud service vendors' operational activities.

On the other hand, the second classification of scheduling is concerned with assigning VMs for task execution. It is common practice to divide large tasks into separate components and assign each one to a separate VM for execution under the virtualization setup. In the present scenario, the choice of VMs is contingent upon users' particular service requirements and the current condition of VMs. The implications of this specific scheduling method substantially affect the duration of job completion and the financial expenditure related to task execution. The scheduling paradigm exhibits a notable resonance among users, particularly concerning service quality and budgetary factors.

The responsibility for coordinating user tasks onto VMs based on user requirements and QoS factors usually falls on the data center broker and the cloud information service in a cloud computing environment. They are crucial in ensuring that user tasks are allocated to suitable VMs that meet the desired performance, resource availability, and other criteria. This coordination helps optimize resource utilization and deliver efficient cloud services. Users prefer minimizing costs associated with service expenses, whereas cloud providers strive to reduce energy consumption while maintaining optimal server performance and capacity utilization. These issues arise from the direct impact of these elements on the time of task performance.

As the duration of a task lengthens, there is a corresponding rise in cost expenditures and energy use. Therefore, the primary focus of this scholarly inquiry is the reduction of makespan. As has been previously analyzed, the complex issue of task scheduling is classified as one of the NP-hard issues. Despite the notable effectiveness of evolutionary algorithms in addressing NP-hard problems, their convergence rate tends to be prolonged due to the exhaustive examination of all probable, plausible solutions. As a result, the prompt achievement of convergence is regarded as a subordinate goal in this research. Given the presence of m VMs and n tasks within the environment, a set of tasks (T) and VMs (V) can be expressed by Eq. (1) and (2).

$$T = \{t_1, t_2, t_3, \dots, t_n\} \quad (1)$$

$$V = \{v_1, v_2, v, \dots, v_m\} \quad (2)$$

The assignment of these tasks to VMs produces a significant number of potential patterns, which can be expressed as *nm* possible scenarios. Suppose *VR* represents a collection of VM resources, indicated as $VR = (vr_1, vr_2, vr_3, ..., vr_k)$. These features include the central processing unit's (CPU) capability to execute Millions of Instructions Per Second (MIPS), the availability of bandwidth, the capacity of Random Access Memory (RAM), and the capacity of storage. The task completion duration depends on the specific allocation of resources to the selected VMs. Cloud service providers utilize specific measurements known as Compute Units (CUs) to measure the capacity of VMs. For example, a solitary Amazon CU possesses processing capabilities that align with a frequency range of up to 1.2 GHz, similar to an Xeon and Opteron processor. The calculation of the duration of task execution and the projected operational expenditure for the workflow depends on CUs. Therefore, the execution time for the t^{th} task is given in the form of Eq. (3).

$$ET_{ij} = \frac{L_i}{CU_j} \tag{3}$$

Where i and j represent indices within integer numbers sets ranging from 1 to n and 1 to m, respectively. Here, CU_j corresponds to the computational unit associated with the jth VM, whereas Li signifies the execution time of the i^{th} task. The active time of a VM is defined as the interval during which tasks are being processed on the VM. Specifically, the phase of intense utilization throughout the active time of the j^{th} VM is represented by Eq. (4).

$$BT_j = \sum_{i=1}^{n} ET_{ij} \times c_{ij} \tag{4}$$

c_{ij} is limited to binary values, 0 or 1, x_{ij} represents the relationship between tasks and VMs, where 1 implies that task t_i is allocated to the j^{th} VM. Since VMs operate concurrently, the workflow's overall duration, or makespan, is calculated by the most prolonged time any single VM remains occupied. The makespan can be expressed using Eq. (5).

$$M = \max(BT_j) \tag{5}$$

Evolutionary algorithms aim to find the optimal solution by systematically navigating the problem domain. The time needed for the algorithm to converge depends on the solution space properties and the number of iterations executed. As the solution space expands, the convergence time increases. This relationship between convergence time and the solution space size can be expressed mathematically by Eq. (6).

$$CT \propto (l_x, k) \tag{6}$$

CT refers to the convergence time, k denotes the number of iterations necessary to identify an optimal solution, and l_x represents the length of the optimal solution x,. The objective function O, used to determine the solution x, can be formulated based on Eq. (3) and (4) in form of Eq. (7).

$$O(x) = \min(M), \min(CT) \tag{7}$$

IV. Proposed Method

Harris Hawk cooperative hunting and tracking procedures inspire the HHO algorithm. These birds employ strategic tactics of surprise jumps and seven killings to capture their prey. In cooperative attacks, some hawks coordinate in pursuit of a rabbit that has exposed itself after revealing its whereabouts for pursuit and quick capture. With hunting, however, there would be successive quick dives next to the prey, based on how it would react and the chance of its fleeing. Harris's hawks have various hunting techniques under their wings, each for different circumstances and different maneuvers of prey to evade them. If the top hawk in a hunting activity fails to track the rabbits, then another member of the team should replace that hawk and foil possible escape. It is here that the rabbit, once the hunt starts, cannot regain its defense mechanism, and the team's combined effort prevents it from escaping. The most experienced hawk makes the final catch of exhausted prey to share among the team members.

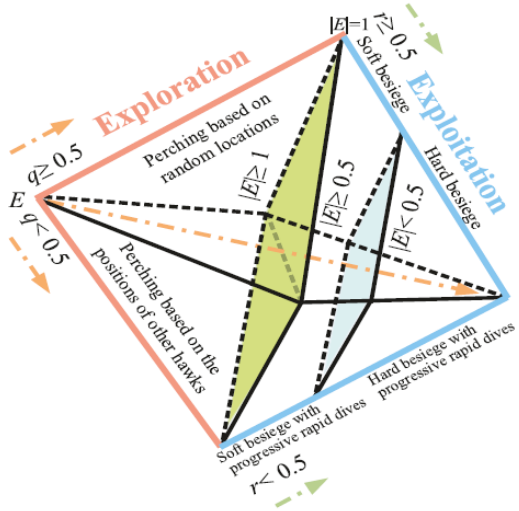

Fig. 3. HHO steps.

Fig. 3 visually represents the different phases of the HHO algorithm and reflects hawk predatory behavior: locating, circling, and ultimately capturing prey. HHO's mathematical formulation is structured accordingly, including the exploration, transition, and exploitation phases. Within this conceptual framework, each Harris's hawk symbolizes a possible solution to a particular problem, while the target prey symbolizes the ideal solution to be identified. Falcons use two exploration strategies to find prey. In the first strategy, hawks choose locations according to other hawks' positions and prey locations. In the second tactic, hawks sit randomly on tall trees. Eq. (8) mathematically models these two exploration methods with equal probability and uses random numbers to simulate their occurrence.

$$x(t+1) = \begin{cases} x_{random}(t) - r_1 |x_{random}(t) - 2r_2 x| & q \geq 0.5 \\ x_{rabbit}(t) - x_{mean}(t) - & \\ r_3\big(LB + r_4(UB - LB)\big) & q < 0.5 \end{cases} \tag{8}$$

Eq. (9) calculates the average position of the hawk population. The algorithm dynamically transitions between exploration and exploitation phases based on a metric termed 'rabbit energy,' defined by Eq. (10). When the rabbit's escaping energy $|E|$ exceeds 1, the hawks engage in a more extensive exploration of the search space; otherwise, the algorithm

transitions to the exploitation phase. Eq. (11) to (14) establish whether the hawks execute a soft siege or a hard siege, depending on the rabbit's energy level and its likelihood of escape. In a soft siege, the hawks simulate the rabbit's successful escape by performing repetitive diving maneuvers. Conversely, a hard siege employs a distinct computational strategy to model the scenario.

$$x_{mean}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t) \quad (9)$$

$$E = 2E_0\left(1 - \frac{t}{Max\_iter}\right) \quad (10)$$

$$x(t+1) = \Delta x(t) - E|J.x_{rabbit}(t) - x(t) \quad (11)$$

$$\Delta x(t) = x_{rabbit}(t) - x(t) \quad (12)$$

$$J = 2(1 - random) \quad (13)$$

$$x(t+1) = x(t) - E|\Delta x(t)| \quad (14)$$

Eq. (15) to (18) regulate the rapid dives employed during the soft siege, employing Lévy movements to simulate the prey's evasive behavior. Eq. (15) and (16) mathematically model the hawks' actions during the diving phase. Subsequently, Eq. (17) and (18) define the characteristics of the final rapid dives performed during the soft siege and the associated factors, *k* and *z*, utilized throughout the hard siege phase.

$$k = x_{rabbit}(t) - E|J.x_{rabbit}(t) - x(t)| \quad (15)$$

$$z = k + RandomVector.L(dim) \quad (16)$$

$$x(t+1) = \begin{cases} k & if \quad f(k) < f(x(t)) \\ z & if \quad f(z) < f(x(t)) \end{cases} \quad (17)$$

$$k = x_{rabbit}(t) - E|J.x_{rabbit}(t) - x_{mean}(t)| \quad (18)$$

The HHO algorithm incorporates four pursuit strategies during the exploitation phase to enhance exploration capabilities. While heightened exploration is beneficial in identifying diverse solution spaces, it can inadvertently precipitate premature convergence and local optima. To counteract this, standard stochastic strategies such as Gaussian random walk, Brownian motion, and Levy flight are often integrated into optimization algorithms. These strategies introduce controlled stochasticity, allowing the algorithm to balance exploitation with exploration. By generating random deviations, these methods keep the algorithm from becoming stuck in suboptimal solutions and boost its overall performance.

Initialize parameters:
- Generate an initial population of hawks with random positions.
- Define maximum iterations (Max_iter), iteration counter (t = 1), and necessary constants.
- Define the fitness function for the optimization problem.

Evaluate initial fitness:
- Calculate the fitness of each hawk in the initial population.
- Identify the prey position (the current best solution based on fitness).

While t ≤ Max_iter, perform the following steps:

a. Calculate Rabbit Energy (E):
- Compute $E = 2E_0(1 - t/Max\_iter)$, where E_0 is a random value in $[-1,1]$.

b. Determine phase:
- If $|E| > 1$, proceed with the exploration phase.
- Otherwise, proceed with the exploitation phase.

c. Exploration phase:
- Update the hawks' positions using:
  - Strategy 1: Update positions based on other hawks and the prey location.
  - Strategy 2: Allow hawks to perch randomly on tall trees.
- Use probabilistic rules (Equation 8) to determine the update strategy.

d. Exploitation phase:
- If prey escapes (soft siege):
  - Simulate evasive behavior using rapid dives with Lévy flights (Equations 15-18).
- Else (hard siege):
  - Aggressively update positions based on prey location (Equations 11-14).

e. Dynamic random walk (if fitness stagnates):
- Compute the deviation using Equation 19.
- Update the hawks' positions using Equation 20.

f. Update fitness and prey position:
- Recalculate the fitness for the updated hawk positions.
- Update the prey position if a better solution is found.

g. Increment iteration:
- Increase t by 1.

Output the results:
- Return the best position (prey) and its corresponding fitness value.

Fig. 4. Pseudocode of the proposed algorithm.

The paper proposes a dynamic random walk strategy to enhance the HHO algorithm. The pseudocode of the proposed algorithm is depicted in Fig. 4. The magnitude of the random walk deviation decreases over time. This ensures a balance between exploration (larger deviations in early iterations) and exploitation (smaller deviations in later iterations). The random walk is activated only when the fitness value of a hawk remains unchanged compared to the previous iteration. This indicates potential stagnation in the search process. The deviation is calculated using a time-dependent formula involving a random number and the present iteration relative to the maximum number of iterations. The proposed random walk strategy can be mathematically expressed using Eq. (19).

$$Deviation = (c \times rand - {}^{c}/_{2}) \times \cos({}^{\pi}/_{2} \times ({}^{t}/_{T}) \quad (19)$$

Where *Deviation* is the value added to the hawk's position, *c* is a constant controlling the maximum deviation, *rand* is a random number between 0 and 1, *t* is the ongoing iteration, and *T* is the total number of iterations. Eq. (20) is used to model the process.

$$x_m(i) = X(i) + \left(c \times rand - \frac{c}{2}\right) \times cos\left(\frac{\pi}{2} \times \left(\frac{t}{T}\right)^2\right) \times (X(i) - X_{rabbit}) \quad (20)$$

Experimental results indicate that a value of *c* equal to six yielded optimal performance. Applying the random walk strategy produces a novel position, denoted as $X_m(i)$. A subsequent greedy selection process, as formalized in Eq. (21), determines the most suitable position for the ensuing iteration.

$$X(t+1) = \begin{cases} X_m(t+1), & f(X_m(t+1)) < f(X(t+1)) \\ X(t+1), & f(X_m(t+1)) \geq f(X(t+1)) \end{cases} \quad (21)$$

V. RESULTS

The proposed algorithm (EHHO) algorithm was simulated using the CloudSim toolkit, which offers robust support for on-demand resource provisioning and versatile features, including multi-objective optimization, dynamic resource scaling, application modeling, and cloud deployment simulation. Kafka's built-in load-balancing mechanism was employed. To evaluate EHHO's performance, it was compared against ALO, GA, ACO, PSO, MGWO, and EMVO algorithms using metrics such as execution time, cost, memory storage, and makespan. Experimental parameters are detailed in Table III.

The platform selection for evaluating the EHHO algorithm, including Kafka, Spark, Flink, and Storm, was driven by their unique characteristics that align with the requirements of task scheduling in cloud environments. Kafka was chosen for its real-time reporting capabilities, enterprise-level security, and efficient cloud monitoring, making it ideal for scenarios requiring immediate feedback and load balancing. Spark's in-memory computation and scalability enable high-speed processing for large datasets, while Flink's event-driven architecture supports dynamic and continuous task scheduling. Storm, known for its low-latency processing, is particularly suitable for time-critical scheduling tasks. These platforms were selected to demonstrate EHHO's adaptability and performance across workloads, real-time requirements, and resource management conditions, ensuring comprehensive evaluation in diverse cloud scenarios.

Tables IV and V show the execution time and cost results for different algorithms and platforms. EHHO consistently demonstrated superior performance, achieving the lowest execution time (610 ms) and cost (60) on the Kafka platform. Tables VI and VII summarize memory storage and makespan results, with EHHO again exhibiting optimal performance, recording minimum makespan values and memory consumption across all platforms. To ensure a fair comparison, all algorithms employed a maximum iteration of 100 and a population size of 100. Specific parameter settings for each algorithm are detailed below:

- EHO: alpha = 0.5, beta = 1, upper bound = 0.9, number of clans = 10, set elitism = 2, lower bound = 0.3.
- MGWO and EMVO: number of appliances = 12, coefficient vector = 1, TDR = 1, WEP = 0.2.
- PSO: maximum initial velocity = 15, minimum initial velocity = 5, alpha = 0.8, beta = 0.8.
- ACO: time factor = 2, saving matrix factor = 2, visibility coefficient = 3, pheromone concentration coefficient = 1.
- GA: mutation probability = 0.02, crossover probability = 0.60, number of demes = 6.
- ALO: number of dimensions = 5, lower bound = 0.1, upper bound = 0.8.

Kafka consistently outperforms other platforms regarding cost, execution time, makespan, and memory storage for all algorithms. Its real-time reporting capabilities, enterprise-level security, efficient cloud monitoring, and superior processing speed contributed to these results. Fig. 5 to 8 provide visual representations of the comparative performance of the algorithms as measured by cost, execution time, memory storage, and makespan, respectively. The simulations validate the superiority of the EHHO algorithm in optimizing resource allocation and performance across various metrics and platforms. Its ability to effectively balance workload and esource utilization resulted in significant improvements compared to traditional optimization algorithms.

TABLE III. SIMULATION PARAMETERS

Element	Parameter	Value
Task	Task length	1000
	Task count	1000
VM	Service provider count	5
	VM count	1000
	MIPS	500
	Bandwidth	500
	Processing element count	2
Datacenter	Datacenter count	10
	Host count	2

TABLE IV. SIMULATION RESULTS FOR COST

Platform	EMVO	MGWO	PSO	ACO	GA	ALO	EHHO
Kafka	151	150	123	178	174	155	60
Spark	175	165	139	189	178	170	79
Flink	186	174	145	202	184	192	85
Storm	191	187	153	190	191	204	101

TABLE V. SIMULATION RESULTS FOR EXECUTION TIME

Platform	EMVO	MGWO	PSO	ACO	GA	ALO	EHHO
Kafka	835	893	792	785	911	774	610
Spark	897	946	862	888	1080	803	649
Flink	906	956	874	901	1123	875	716
Storm	964	979	889	909	1201	895	727

TABLE VI. SIMULATION RESULTS FOR MEMORY USAGE

Platform	EMVO	MGWO	PSO	ACO	GA	ALO	EHHO
Kafka	502	421	530	443	398	382	305
Spark	531	488	631	555	479	457	317
Flink	690	548	659	630	525	536	332
Storm	696	571	722	730	840	514	336

TABLE VII. SIMULATION RESULTS FOR MAKESPAN

Platform	EMVO	MGWO	PSO	ACO	GA	ALO	EHHO
Kafka	109	120	140	124	210	240	52
Spark	121	146	164	142	231	243	55
Flink	140	156	175	185	275	275	82
Storm	143	187	184	191	281	286	94

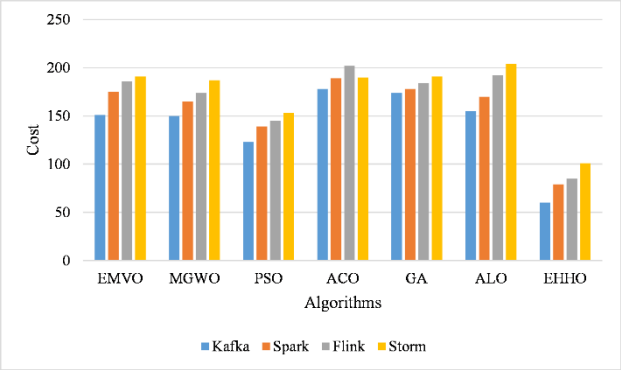

Fig. 5. Cost comparison.

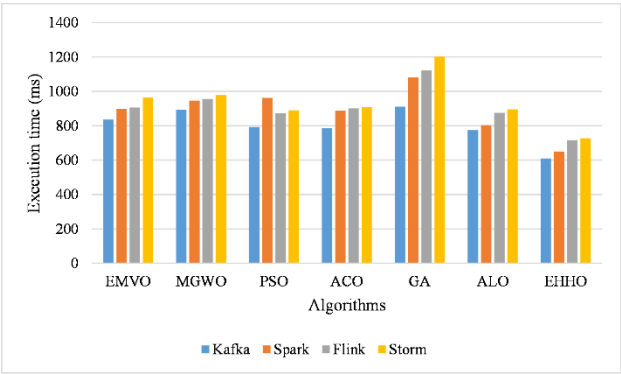

Fig. 6. Execution time comparison.

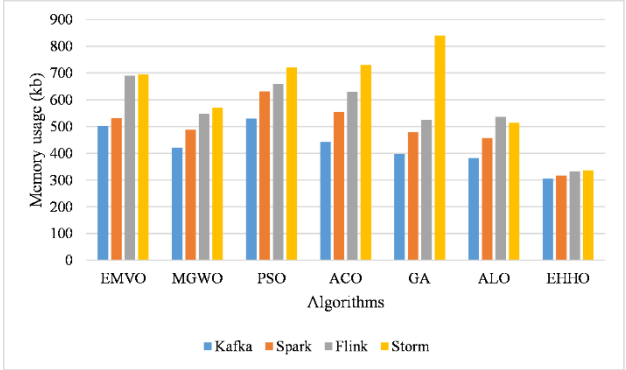

Fig. 7. Memory usage comparison.

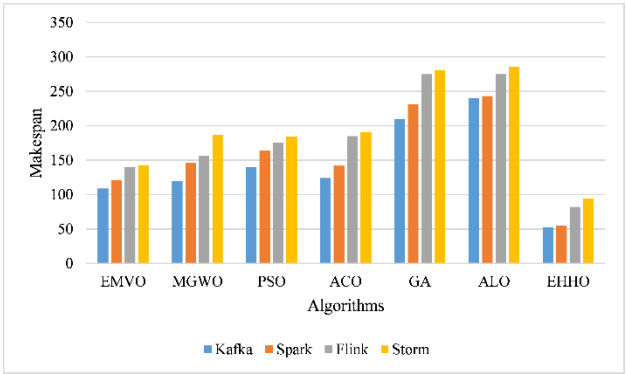

Fig. 8. Makespan comparison.

VI. DISCUSSION

The EHHO algorithm demonstrates significant advancements in task scheduling within cloud computing environments. Integrating a dynamic random walk strategy has notably improved the algorithm's exploration and exploitation power, leading to superior performance in various metrics compared to other optimization algorithms. The experimental findings reveal that EHHO consistently achieves lower execution times, costs, memory usage, and makespan across multiple platforms, including Kafka, Spark, Flink, and Storm. These findings underscore the robustness and efficiency of EHHO in optimizing resource allocation and handling complex scheduling problems in cloud computing.

A key factor contributing to EHHO's success is its ability to avoid premature convergence, a common issue in traditional meta-heuristic algorithms. By incorporating stochastic strategies such as Gaussian random walk, Brownian motion, and Levy flight, EHHO maintains equilibrium between global exploration and local exploitation. This balance ensures that the algorithm can explore diverse solution spaces without falling into a local optimum, thereby enhancing solution quality. The dynamic adjustment of the random walk deviation over time further refines this balance, enabling EHHO to effectively adapt to different stages of the optimization process.

Moreover, the simulation results highlight the exceptional functionality of the Kafka platform concerning makespan, execution time, cost, and memory usage. Kafka's real-time reporting capabilities, enterprise-level security, efficient cloud monitoring, and superior processing speed contribute to these outcomes. These characteristics make Kafka a suitable environment for deploying EHHO, allowing it to fully leverage its optimization potential. The comparative analysis with other platforms reinforces the importance of selecting an appropriate infrastructure to maximize the benefits of advanced optimization algorithms like EHHO in cloud computing.

In summary, the EHHO algorithm effectively responds to the complex task scheduling challenges in cloud computing. Its enhanced exploration and exploitation mechanisms, coupled with the optimal performance on platforms like Kafka, position EHHO as a leading approach for efficient resource management. Researchers could explore ways to improve the EHHO algorithm, such as integrating additional stochastic strategies or refining the random walk parameters, to achieve even greater performance improvements. Additionally, investigating the algorithm's scalability and applicability to other optimization problems could expand its utility in broader contexts.

The EHHO algorithm can seamlessly integrate with popular cloud services such as AWS, Azure, and Google Cloud to optimize task scheduling and resource management. By leveraging these platforms' capabilities, EHHO can enhance the efficiency of IaaS by dynamically allocating VMs and managing compute resources. In PaaS, EHHO can streamline application deployments by optimizing workload distribution across scalable infrastructure. For SaaS, the algorithm ensures reduced latency and cost-effective resource utilization, improving overall service delivery. The ability of EHHO to adapt to real-time cloud environments and balance workloads makes it a crucial component for maximizing the performance and scalability of cloud-based services, further solidifying its relevance in modern cloud computing ecosystems.

Despite its promising performance in task scheduling, the EHHO algorithm has certain constraints. Its reliance on predefined parameters, such as random walk deviation and iteration limits, may limit adaptability across varying real-time scenarios and dynamic workloads. Additionally, while EHHO demonstrates superior results on metrics like makespan, cost, and memory usage, its scalability to handle significantly larger task datasets or highly heterogeneous environments remains untested. The simulations, primarily conducted using the Kafka platform, suggest a dependency on specific infrastructure capabilities such as real-time reporting and efficient monitoring, raising concerns about performance consistency on less advanced platforms. Furthermore, while the dynamic random walk strategy improves exploration and exploitation, fine-tuning these adjustments for broader applications remains challenging. Addressing these constraints, particularly scalability and infrastructure independence, will be critical for maximizing EHHO's potential in diverse cloud environments.

VII. CONCLUSION

Effective task scheduling is paramount to the optimal performance of cloud computing systems. Unlike traditional computing environments, cloud-based task scheduling necessitates considering diverse parameters, including computational costs, processing capabilities, and task duration. In this research, we introduced the EHHO algorithm to tackle the complex challenge of task scheduling in cloud computing environments. Leveraging the CloudSim toolkit for simulations, EHHO demonstrated superior performance over traditional algorithms like PSO, ACO, GA, ALO, MGWO, and EMVO across critical metrics, including cost, execution time, makespan, and memory storage. Integrating a random walk approach significantly improved the algorithm's exploration capabilities, effectively preventing premature convergence to local optima and ensuring more efficient resource allocation. With its robust load balancing, high security, real-time analysis, and scalability, Kafka's platform further highlighted the algorithm's efficiency. Our findings underscore EHHO's potential for optimizing the operational efficiency of cloud computing systems, making it a viable solution for better task scheduling and resource management in diverse and dynamic cloud environments.

Future research on the EHHO could include focusing on its scalability and adaptability given real-time scheduling scenarios in highly dynamic cloud environments. By integrating adaptive random walk strategies, deviation parameters can dynamically change depending on task complexity and resource availability in real-time. The following extension of EHHO for multi-cloud or hybrid cloud infrastructures with cross-platform scheduling and resource allocation would increase its applicability. Testing its performance with more diverse and larger datasets and optimization of computational efficiency for real-world runtime applications may position EHHO as a more robust and versatile solution to complex challenges in cloud computing.

REFERENCES

[1] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," Journal of Network and Computer Applications, vol. 97, pp. 23-34, 2017, doi: https://doi.org/10.1016/j.jnca.2017.08.006.

[2] M. Elrifaee, T. Zayed, E. Ali, and A. H. Ali, "IoT contributions to the safety of construction sites: a comprehensive review of recent advances, limitations, and suggestions for future directions," Internet of Things, p. 101387, 2024.

[3] V. Hayyolalam, B. Pourghebleh, A. A. P. Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[4] Q. Li, J. Huang, S. Li, and C. Huang, "A Sustainable Data Encryption Storage and Processing Framework via Edge Computing-Driven IoT," Engineering Letters, vol. 32, no. 7, 2024.

[5] 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.

[6] 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.

[7] A. Mohamed et al., "Software-defined networks for resource allocation in cloud computing: A survey," Computer Networks, vol. 195, p. 108151, 2021.

[8] L. Rosa, L. Foschini, and A. Corradi, "Empowering Cloud Computing With Network Acceleration: A Survey," IEEE Communications Surveys & Tutorials, 2024.

[9] C. Wang and D. Wang, "Managing the integration of teaching resources for college physical education using intelligent edge-cloud computing," Journal of Cloud Computing, vol. 12, no. 1, p. 82, 2023.

[10] R. Zolfaghari, A. Sahafi, A. M. Rahmani, and R. Rezaei, "Application of virtual machine consolidation in cloud computing systems," Sustainable Computing: Informatics and Systems, vol. 30, p. 100524, 2021.

[11] T. Sun, C. Ma, Z. Li, and K. Yang, "Cloud Computing-based Parallel Deep Reinforcement Learning Energy Management Strategy for Connected PHEVs," Engineering Letters, vol. 32, no. 6, 2024.

[12] M. D. Tezerjani, M. Khoshnazar, M. Tangestanizadeh, and Q. Yang, "A Survey on Reinforcement Learning Applications in SLAM," arXiv preprint arXiv:2408.14518, 2024, doi: https://doi.org/10.48550/arXiv.2408.14518.

[13] 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.

[14] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," Expert Systems with Applications, vol. 168, p. 114230, 2021.

[15] G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," ICT Express, vol. 5, no. 2, pp. 110-114, 2019.

[16] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," Applied Intelligence, vol. 49, pp. 3308-3330, 2019.

[17] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," Optik, vol. 258, p. 168677, 2022.

[18] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," Neural Computing and Applications, vol. 32, pp. 1531-1541, 2020.

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

[20] S. K. Panda, S. S. Nanda, and S. K. Bhoi, "A pair-based task scheduling algorithm for cloud computing environment," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 1, pp. 1434-1445, 2022.

[21] P. Tamilarasu and G. Singaravel, "Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment," Journal of Engineering Research, 2023.

[22] L. Abualigah et al., "Improved Jaya Synergistic Swarm Optimization Algorithm to Optimize Task Scheduling Problems in Cloud Computing," Sustainable Computing: Informatics and Systems, p. 101012, 2024.

[23] I. Behera and S. Sobhanayak, "HTSA: A novel hybrid task scheduling algorithm for heterogeneous cloud computing environment," Simulation Modelling Practice and Theory, vol. 137, p. 103014, 2024.

[24] M. Khademi Dehnavi, A. Broumandnia, M. Hosseini Shirvani, and I. Ahanian, "A hybrid genetic-based task scheduling algorithm for cost-efficient workflow execution in heterogeneous cloud computing environment," Cluster Computing, pp. 1-26, 2024.

[25] R. Gong, D. Li, L. Hong, and N. Xie, "Task scheduling in cloud computing environment based on enhanced marine predator algorithm," Cluster Computing, vol. 27, no. 1, pp. 1109-1123, 2024.

[26] P. Pabitha, K. Nivitha, C. Gunavathi, and B. Panjavarnam, "A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing," Sustainable Computing: Informatics and Systems, vol. 41, p. 100944, 2024.

[27] V. Parthasaradi, A. Karunamurthy, C. Hussaian Basha, and S. Senthilkumar, "Efficient Task Scheduling in Cloud Computing: A Multiobjective Strategy Using Horse Herd–Squirrel Search Algorithm," International Transactions on Electrical Energy Systems, vol. 2024, no. 1, p. 1444493, 2024.