ECOA: An Enhanced Chimp Optimization Algorithm for Cloud Task Scheduling

Yue WANG
Hebei Chemical & Pharmaceutical College, Hebei, 050026, China

***Abstract*—Effective scheduling of tasks is a key concern in cloud computing because it considerably affects system functionality, resource usage, and execution efficiency. The present study proposes an Enhanced Chimp Optimization Algorithm (ECOA) to address such problems by overcoming the disadvantages of traditional scheduling methods. The proposed ECOA combines three innovative components: 1) the highly disruptive polynomial mutation enhances population diversity, 2) the Spearman rank correlation coefficient promotes the refinement of inferior solutions, and 3) the beetle antennae operator facilitates more efficient local exploitation. These changes significantly enhance the equilibrium between exploration and exploitation, decrease the chance of premature convergence, and are a better solution. Extensive experiments on benchmark datasets prove that ECOA outperforms traditional algorithms concerning makespan, imbalance degree, and resource utilization. The obtained results confirm that the proposed ECOA has excellent potential for better performance in task scheduling in dynamic and large-scale cloud environments, as it represents a promising optimization solution for complex problems in cloud computing.**

Keywords*—Cloud computing; task scheduling; resource utilization; chimp optimization

I. INTRODUCTION

Cloud computing has reshaped how computing capabilities are accessed and used, bringing revolutionary expansion, adaptability, and affordability [1]. It enables companies and individuals to dynamically allocate resources to optimize their processes for optimal performance without requiring investment in physical infrastructure [2]. From big data analytics to intricate Internet of Things (IoT) and Artificial Intelligence (AI) systems, cloud computing forms the pillar of contemporary technology platforms. Effective management of these resources is the key for ensuring Quality Of Service (QoS) and optimal system performance during operations [3]. The recent development of AI-powered quality control, including employing a hybrid vision transformer and Convolutional Neural Networks (CNNs) ensembles for industrial defect inspection, exemplifies the revolutionary potential of intelligent computing infrastructures in real-time, high-precision applications [4].

Scheduling tasks in the cloud is one of the biggest challenges since this environment is dynamic, with heterogeneous resources [5]. Scheduling involves mapping incoming tasks to the available resources and must consider many constraints, such as dependencies between tasks, resource capacities, or QoS requirements [6]. Poor resource scheduling leads to inefficient exploitation of resources, prolonged execution of tasks, and increased business expenses [7]. Further complications in large-scale and real-time applications aggravate these challenges, which require innovative ways of handling resource allocation duties [8]. Similar scheduling concerns are also evident in other networked environments, such as LTE-Advanced systems, where dynamic and QoS-aware approaches are essential for optimizing multi-carrier resource allocation and improving user experience [9].

While traditional scheduling methods and heuristic algorithms have widely been put into practice, they still often suffer from such flaws as premature convergence, inefficiency in handling complex solution spaces, and poor adaptability against dynamic cloud environments [10]. Several metaheuristic algorithms proposed in the literature present promising results, but they usually show poor performance regarding the harmony between exploitation and exploration.

Recent studies have highlighted the growing role of machine learning in analyzing complex economic systems and decision-making under uncertainty, further underscoring the need for adaptive and intelligent optimization approaches in dynamic scenarios [11]. Similarly, in smart grid and energy-aware systems, integrating energy storage and photovoltaic solutions has demonstrated the importance of efficient resource management and real-time optimization, reinforcing the relevance of such capabilities in cloud-based scheduling contexts [12]. Despite being efficient, the original version of the Chimp Optimization Algorithm (COA) has deficiencies in population diversity and needs to improve in local optimum; therefore, it cannot guarantee global optimization when tackling a large-scale task-scheduling problem.

To remedy these defects, this study suggests an Enhanced Chimp Optimization Algorithm (ECOA) for cloud computing task scheduling. The enhanced algorithm applies three main strategies: the highly disruptive polynomial mutation strategy to keep the variety in populations, Spearman's rank correlation coefficient for refining less-fit solutions, and the beetle antennae operator for improving local exploitation.

With the integration of enhancements, ECOA gives a better balance between exploitation and exploration, resulting in improved scheduling efficiency and scalability. ECOA obtains promising results in general because many benchmark datasets are used under considerable experiments to establish its great strength in the techniques adopted in task scheduling during this dynamic cloud environment.

The study is structured into four main sections. Section II offers an overview of state-of-the-art task scheduling methods in cloud computing. Section III introduces the proposed ECOA. Section IV includes extensive experimental analysis of standard

benchmarks with a comparison of the performance of the developed algorithm with previous algorithms. Section V critically discusses the practical relevance of the algorithm, behavior over different datasets, and limitations. Section VI concludes the primary results of the study, highlights the scope of the proposed algorithm's potential in large-scale and time-evolving cloud systems, and provide directions for further research.

II. Related Works

Kashikolaei, et al. [13] proposed a hybrid load-balancing algorithm combining Firefly Algorithm (FA) and Imperialist Competitive Algorithm (ICA) to address NP-hard challenges in cloud computing. This enhanced the scheduling speed, load balancing, CPU time, and makespan. The local search capability of FA strengthened the global search capability of ICA and achieved significant improvements in performance metrics.

Velliangiri, et al. [14] suggested a Hybrid Electro Search with Genetic Algorithm (HESGA) for optimizing cloud task scheduling. In local optimizations, the hybrid approach uses the genetic algorithm, and the authors utilize electro-search to calculate optimal global solutions that would improve load balancing, makespan, resource utilization, and cost.

Mangalampalli, et al. [15] developed a Cat Swarm Optimization (CSO) approach for task scheduling to minimize total power cost, energy usage, migration time, and makespan. This approach prioritizes tasks and virtual machines, improving energy efficiency and execution time by considering realistic workload datasets.

Malathi and Priyadarsini [16] proposed a hybrid Lion Optimizer and Genetic Algorithm (LO-GA) for load balancing in cloud environments. The proposed two-stage approach utilized the lion optimizer for the task and virtual machine selection probabilities, whereas the modified genetic algorithm performs the global search. As a result, this hybrid approach significantly improved resource utilization and turnaround time.

Ghafari and Mansouri [17] introduced an Enhanced African Vulture Optimization Algorithm (E-AVOA-TS) for task scheduling in fog-cloud computing. This method guarantees minimum energy usage, makespan, and cost by sending tasks sensitive to latency to fog environments. Simulation tests have been performed on benchmark datasets and have shown the exceptional performance of E-AVOA-TS over existing algorithms.

Abualigah, et al. [18] presented an Improved Jaya Synergistic Swarm Optimization Algorithm (IJSSOA) integrating Levy flight mechanisms to optimize task scheduling. By combining Jaya's exploitation capabilities with SSO's collaborative strategy, the algorithm improved scalability, convergence rate, and outcome, achieving 88% accuracy and a 10% enhancement over the original method.

Boroumand, et al. [19] presented a new methodology to coordinate tasks in cloud computing, leveraging the features of the Synergistic Swarm Optimization (SSO) and Jaya algorithms. In the suggested algorithm, Jaya's exploitation power has been combined with SSO's collaborative approach to optimize the quality of the solution, convergence rate, and scalability.

Despite some interesting advances in the state-of-the-art, as shown in Table I, existing methods have notable scalability, efficiency, and adaptability limitations across different cloud environments. ICA+FA and HESGA, for example, consider only a single objective, such as makespan or load balancing, while they do not consider multi-objective optimizations. Some proposals, however, such as CSO and LO-GA, consider multiple objectives in their optimization but suffer from high computational complexity and limited scalability.

TABLE I. An Overview of Cloud Task Scheduling Methods

Algorithm	Metrics improved	Strength	Weakness
ICA + FA [13]	Makespan, CPU time, load balancing	Combines the global exploration ability of ICA with the local search efficiency of FA, achieving improved scheduling speed and resource utilization.	It requires fine-tuning of parameters for scalability and limited exploration in high-dimensional spaces.
HESGA [14]	Makespan, load balancing, resource utilization	Combines GA's ability to find local optima with Electro Search's global optimization, achieving superior task scheduling performance in multi-cloud environments.	Limited analysis of energy consumption and migration time; performance is dataset-dependent.
CSO [15]	Makespan, migration time, energy consumption	Addresses multiple objectives, including energy efficiency and power cost, using realistic workloads, providing a holistic improvement in task scheduling metrics.	Computational complexity increases with large-scale tasks that require careful parameter adjustment.
LO-GA [16]	Turnaround time, resource utilization	Utilizes lion optimizer for task and VM selection probabilities and a modified GA for global optimization, improving resource allocation and reducing bottlenecks.	High computational resources are needed for hybrid optimization, but there is limited scalability for diverse tasks.
E-AVOA-TS [17]	Makespan, cost, energy consumption	Prioritizes latency-sensitive and latency-tolerant tasks effectively using a fog-cloud hierarchy, achieving superior task scheduling efficiency and energy savings.	Limited exploration of scalability for extremely large-scale systems; complexity in implementation.
IJSSOA [18]	Makespan, convergence speed, scalability	Integrates Jaya and SSO algorithms with Levy flight for robust exploration-exploitation trade-offs, achieving high-quality solutions and fast convergence.	Focuses primarily on benchmark datasets; limited exploration of dynamic real-world task variability.
IJSSOA (Levy Flights Integration) [19]	Makespan, resource utilization, scalability	Balances exploration and exploitation effectively with Levy flights, enabling escape from local optima and providing scalable performance improvements.	Limited emphasis on energy consumption and multi-objective optimization across diverse environments.

Furthermore, algorithms like IJSSOA show exploration-exploitation trade-offs confined to benchmark datasets only and lack robustness for dynamic real-world complexities, including variability in resources and heterogeneity of tasks. To fill these gaps, we propose ECOA, which integrates superior mutation strategies and optimization mechanisms. In this way, the proposed approach will ensure improved scalability, resource utilization, and scheduling efficiency while remaining adaptable to dynamically large-scale cloud environments.

III. PROPOSED METHODOLOGY

A. Problem Formulation

Cloud task scheduling stands for the activity of performing a scheduler for user-defined computational tasks on available physical servers or virtual machines for cloud computing. Scheduling optimizes critical performance indicators, including execution time, resource utilization, and energy efficiency. Due to its complexity and constraints, task scheduling is an NP-hard optimization problem, requiring heuristic and metaheuristic algorithms for effective solutions.

Tasks (T) are defined as a set $T = \{t_1, t_2, \ldots, t_n\}$, where each task t_i is characterized by its computational requirements, execution time, and dependencies. Resources (R) are represented as $R = \{r_1, r_2, \ldots, r_m\}$, each having availability, memory, and processing capacity characteristics. The binary variable x_{ij} indicates whether task t_i is allocated to resource r_j as in Eq. (1):

$$x_{ij} = \begin{cases} 1, & if\ task\ t_i\ is\ assigned\ to\ resource\ r_j \\ 0, & otherwise \end{cases} \quad (1)$$

The goal is to allocate tasks effectively, maintaining balanced resource utilization while meeting performance targets. Task scheduling is constrained by two key factors: dependency and resource. Some tasks depend on others, represented as D_i, a sequence of tasks necessary to be accomplished in advance of t_i starts execution. Resources have finite capacity, denoted as U_j, limiting the simultaneous execution of tasks on a given resource.

The task scheduling problem minimizes several performance metrics, including resource utilization, makespan, and total execution time. Resource utilization ensures the best use of resources by reducing the time spent idling and thus balancing the workloads. The makespan concerns the time from the beginning of the first task to the end of the last task. Total execution time refers to the overall time all tasks take to complete. The entire VM count in the cloud setup is calculated using Eq. (2):

$$h = \sum_{i=1}^{Nph} N_{vmi} \quad (2)$$

where, Nph represents the total number of physical hosts, and N_{vmi} stands for the number of VMs on the i^{th} physical host. The average number of VMs on each physical host is calculated as in Eq. (3):

$$v_{ij} = \frac{1}{N_{vmi}} \sum_{j=1}^{N_{vmi}} 1 \quad (3)$$

The expected completion time for a task k on a VM j is given by Eq. (4):

$$ETC(j_k) = \frac{L_k}{P_j} \quad (4)$$

where, L_k is the length of the task (in millions of instructions), and P_j is the processing performance of VM j. The execution time for all tasks assigned to a VM j is computed using Eq. (5):

$$ET_j = \sum_{k=1}^{N} x(j_k).ETC(j_k) \quad (5)$$

where, $x(j_k)$ is the decision variable indicating task assignment, finally, the makespan is determined as in Eq. (6):

$$Makespan = max(ET_j) \quad (6)$$

Heuristic and metaheuristic algorithms, formulated in an optimization setting, efficiently explore the vast solution space. These algorithms repeatedly evaluate various mappings of tasks to resources, considering constraints to optimize objective functions. The challenge is to find the trade-off between the two objective functions: minimizing makespan and maximizing resource utilization. As the problem scales, dynamic resource availability, and task variability necessitate robust, adaptive approaches to achieve near-optimal solutions within reasonable time frames.

B. Enhanced Chimp Optimization Algorithm

COA is a swarm intelligence-based metaheuristic inspired by the cooperative hunting strategies of chimpanzees. This algorithm divides chimpanzees into four hierarchical roles: attacker, barrier, chaser, and driver, based on their importance in the optimization process. While the original COA has shown promise in solving various optimization problems, it suffers from limited population diversity during initialization and often gets trapped in local optima during the exploitation phase. COA is a metaheuristic inspired by swarm intelligence which is inspired by the cooperative hunting strategies of chimpanzees. In this algorithm, chimpanzees fall into four hierarchical roles based on their importance in the optimization process: attacker, barrier, chaser, and driver.

The original COA has produced promising results for solving different optimization problems. However, it suffers from limited population diversity during initialization and often gets trapped in local optima during exploitation. To overcome these limitations, the Enhanced COA (ECOA) integrates three advanced mechanisms: Highly Disruptive Polynomial Mutation (HDPM), Spearman's rank correlation coefficient, and the Beetle Antennae Operator (BAO). These enhancements improve the balance between exploration and exploitation, ensuring superior optimization performance.

As shown in Fig. 1, in the original COA, the position of the chimp is updated by calculating the weighted average of the contributions from the four roles. The mathematical model for

updating the position of the chimp is given as Eq. (7) and Eq. (8):

$$X_{chimp}(t+1) = \frac{X_1 + X_2 + X_3 + X_4}{4} \tag{7}$$

where,

$$\begin{aligned} X_1 &= X_{Attacker}(t) - \alpha_1 . d_{Attacker} \\ X_2 &= X_{Barrier}(t) - \alpha_2 . d_{Barrier} \\ X_3 &= X_{Chaser}(t) - \alpha_3 . d_{Chaser} \\ X_4 &= X_{Driver}(t) - \alpha_4 . d_{Driver} \end{aligned} \tag{8}$$

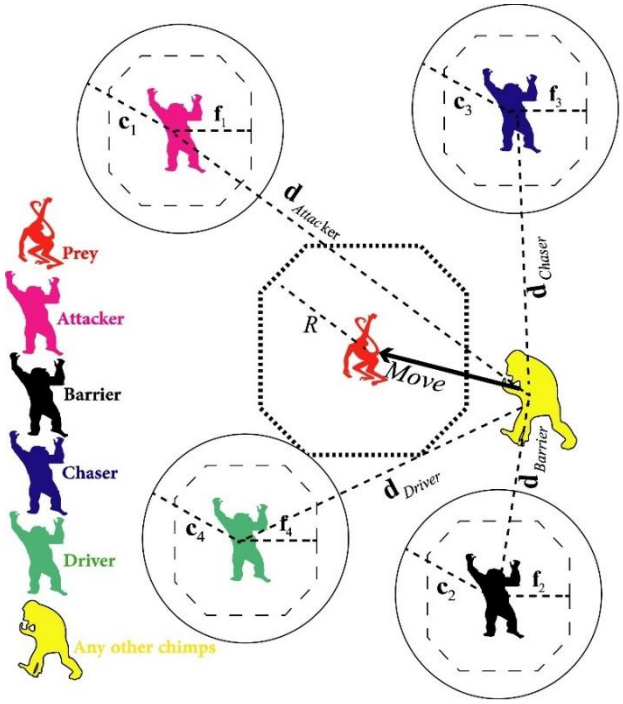

Fig. 1. Position updating in COA

The coefficients $a1$, $a2$, $a3$, and $a4$ represent dynamic adjustment factors that decrease nonlinearly over iterations, while d_{Attacker}, d_{Barrier}, d_{Chaser}, and d_{Driver} are the distances between the chimp and the prey, modeled as in Eq. (9):

$$d_{Attacker} = |c . X_{Attacker}(t) - m . X(t)| \tag{9}$$

where, c is a scaling factor (c=2r_2, with $r_2 \in [0,1]$) and m is a chaotic map vector that introduces randomness to the search process.

The position update ensures that the chimps move collectively toward the global optimum while maintaining role-specific contributions.

HDPM addresses the limitation of population diversity in the initialization phase by enhancing the global exploration capability of the algorithm. Traditional polynomial mutation fails to utilize boundary variables effectively, which HDPM resolves by introducing a mutation operator that can handle boundaries efficiently. The updated position of a chimp X_{new} is calculated as in Eq. (10):

$$X_{new} = X + \delta_k . (ub - lb) \tag{10}$$

where, ub and lb are the upper and lower bounds of the search space, respectively. The mutation factor δ_k is computed using Eq. (11):

$$\delta_k = \begin{cases} [2r + (1-2r).(1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1, & if \\ 1 - [2(1-r) + 2(r-0.5).(1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}, \end{cases} \tag{11}$$

This mutation mechanism ensures a diverse initial population, enabling the algorithm to explore the search space more comprehensively in the early stages.

Spearman's rank correlation coefficient (ρ) measures the relationship between the fitness of the attacker chimp (leader) and the driver chimps (followers). The value of ρ determines whether the driver chimps require position updates to enhance their contribution. The coefficient is calculated using Eq. (12):

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)} \tag{12}$$

where, d_i is the difference between the ranks of two variables (e.g., attacker and driver fitness), and n is the dimension of the problem. If $\rho \leq 0$, the driver's position is refined using the BAO to prevent stagnation.

BAO enhances local exploitation by simulating the sensory behavior of beetles. The beetle uses its antennae to search left and right areas for better solutions. The search direction is determined using a normalized random vector as in Eq. (13):

$$\vec{b} = \frac{rnd(n,1)}{\|rnd(n,1)\|} \tag{13}$$

The beetle explores the search space using Eq. (14):

$$\begin{aligned} X_r(t) &= X(t) + d(t).\vec{b} \\ X_l(t) &= X(t) - d(t).\vec{b} \end{aligned} \tag{14}$$

where, $d(t)$ represents the step size calculated as in Eq. (15):

$$\begin{aligned} d(t) &= \frac{\delta(t)}{C} \\ \delta(t) &= K.\delta(t-1) \end{aligned} \tag{15}$$

where, K = 0.95 and C = 2. The beetle's new position is updated as in Eq. (16):

$$X(t+1) = X(t) + \delta(t).\vec{b}.sign\left(f(X_r(t)) - f(X_l(t))\right) \tag{16}$$

This mechanism allows the driver chimps to exploit fine-grained while avoiding local optima. ECOA operates through a systematic workflow designed to achieve optimal solutions effectively. The population is initialized using HDPM to ensure diversity within the search space. Chimps are assigned specific roles based on their fitness evaluations, such as attacker, barrier, chaser, or driver. The position updates occur in subsequent steps, starting with calculating Spearman's rank correlation coefficient

(ρ) for driver chimps. If $\rho \leq 0$, the BAO is applied to refine the driver's position, enhancing local exploitation.

The positions of all chimps are updated using role-specific equations to facilitate global optimization. Afterwards, the fitness of all updated positions is reevaluated, and roles are reassigned accordingly. This iterative approach continues until the algorithm converges on an optimal solution or the maximum number of iterations, ensuring a robust balance between exploitation and exploration. Fig. 2 illustrates the proposed algorithm in the form of a flowchart.

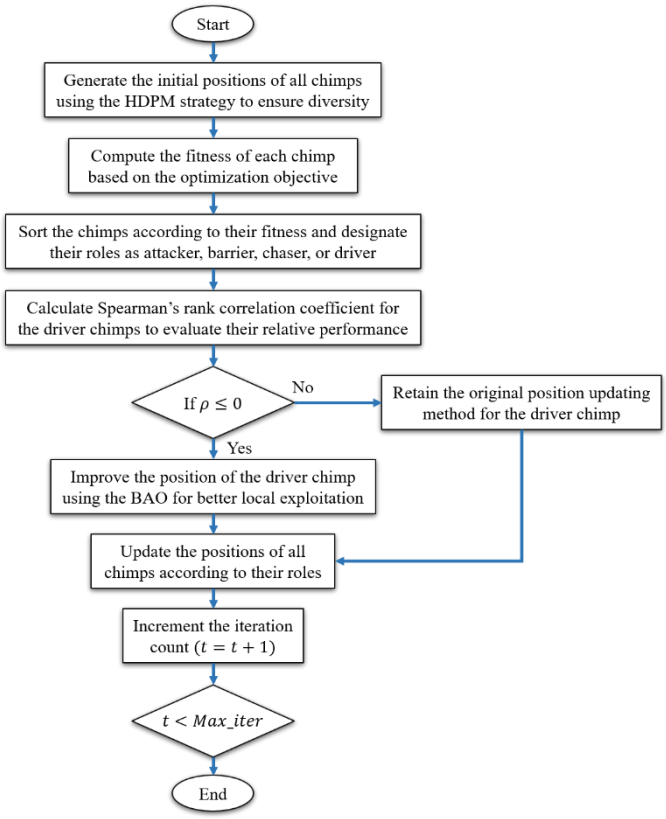

Fig. 2. Flowchart of ECOA.

The integration of HDPM, Spearman's correlation, and the BAO enables ECOA to overcome the limitations of the original COA. The algorithm maintains a better equilibrium between exploration and exploitation, making it more robust for solving complex optimization problems in cloud computing, engineering, and other domains. This comprehensive framework ensures faster convergence, higher solution quality, and adaptability to dynamic environments.

IV. RESULTS

The proposed ECOA was implemented and tested on a system whose specifications are given in Table II. The setup chosen was sufficient to computationally test the performance of ECOA over a wide variety of synthetic datasets. Synthetic datasets were used to evaluate task scheduling performance. Each dataset consisted of tasks from 100 to 500, and task lengths were randomly chosen between 1000 and 2000 MI. Virtual machines had processing capacities from 100 to 1000 MIPS. These datasets provide controlled experimentation under various workloads and resource configurations.

TABLE II. SYSTEM SETUP FOR TESTING THE PROPOSED ECOA

Parameter	Specification
Operating system	Windows 11 64-bit
CPU	Intel(R) Core(TM) i7-3770 @ 3.90 GHz
SDD	240 GB
Memory	32 GB DDR4

ECOA was tested against some state-of-the-art optimization algorithms, namely Geyser-Inspired Algorithm (GIA) [20], Prairie Dog Optimization Algorithm (PDOA) [21], Dwarf Mongoose Optimization Algorithm (DMOA) [22], Reptile Search Algorithm (RSA) [23], and Arithmetic Optimization Algorithm (AOA) [24]. These algorithms' parameter settings are set to the values indicated in their respective source studies.

The most important performance metric in task scheduling is called makespan, referring to the total time taken to execute all the tasks. Fig. 3 depicts the values of makespan for each of the tasks. It can be noticed that the value of makespan rises with the increase in the number of tasks in all algorithms, which reflects an increase in computational complexity and resource demand.

However, ECOA reliably reached lower makespan values than other algorithms and thus showed better efficiency in scheduling. For example, the higher makespan value by algorithms like PDOA and GIA indicated that the scheduling solution was not optimal. AOA-LPO has increased stability, considering a variation of loads.

Fig. 4 depicts the Average Resource Utilization (ARU) over a range of tasks. ECOA maintained high values of ARU, reflecting efficient utilization of computational resources. This remained consistent even with increasing task counts, thus further proving the adaptability of the algorithm to larger workloads. Other algorithms, such as PDOA and DMOA, also maintained relatively stable ARU values, while RSA and GIA exhibited large variances, indicating sensitivity to workload size.

Fig. 5 depicts the Diversity Index (DI), which represents the distribution variety of the tasks. For most practical cases, the value of DI is preferred to be lower when the distribution among resources is even. ECOA had a steady DI for different workloads, indicating the ability to maintain load balance effectively. Other algorithms had varying DI values with increased workload; hence, there was a drop in efficiency while managing task diversity.

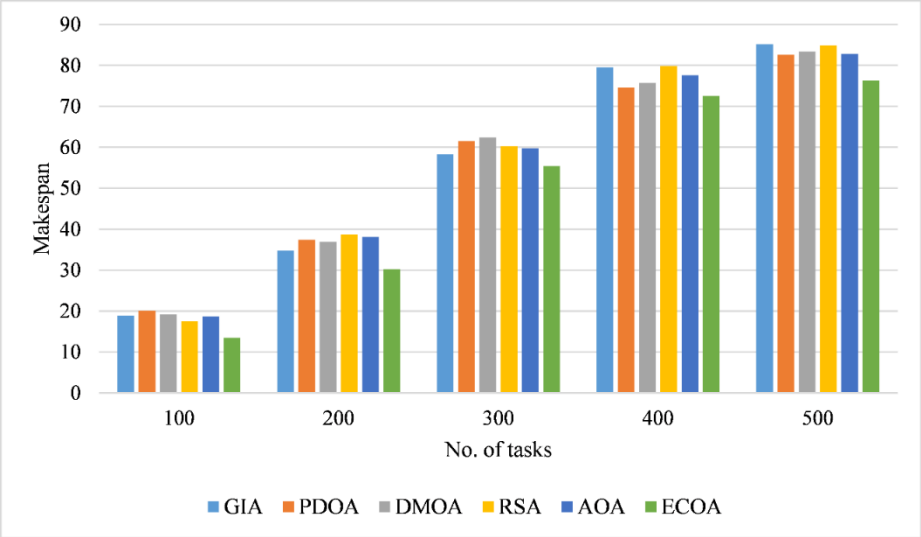

Fig. 3. Makespan comparison

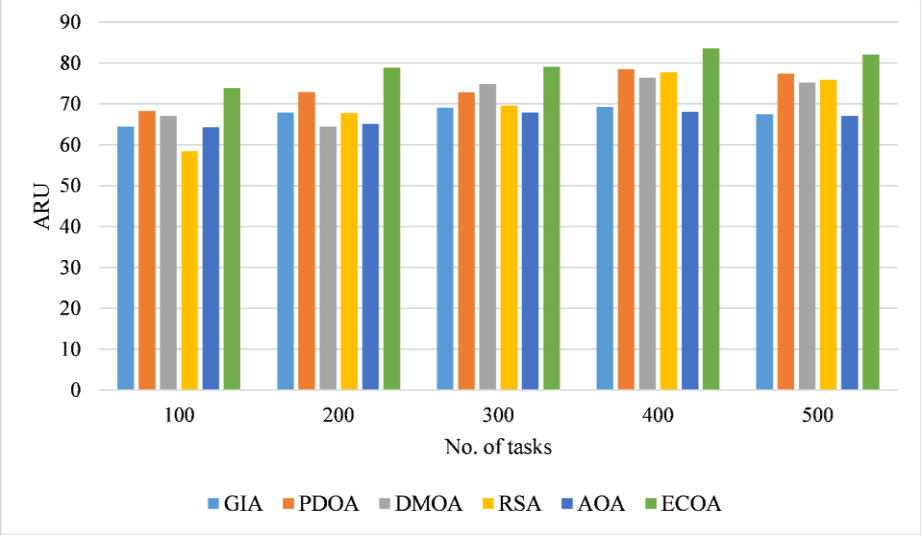

Fig. 4. ARU Comparison

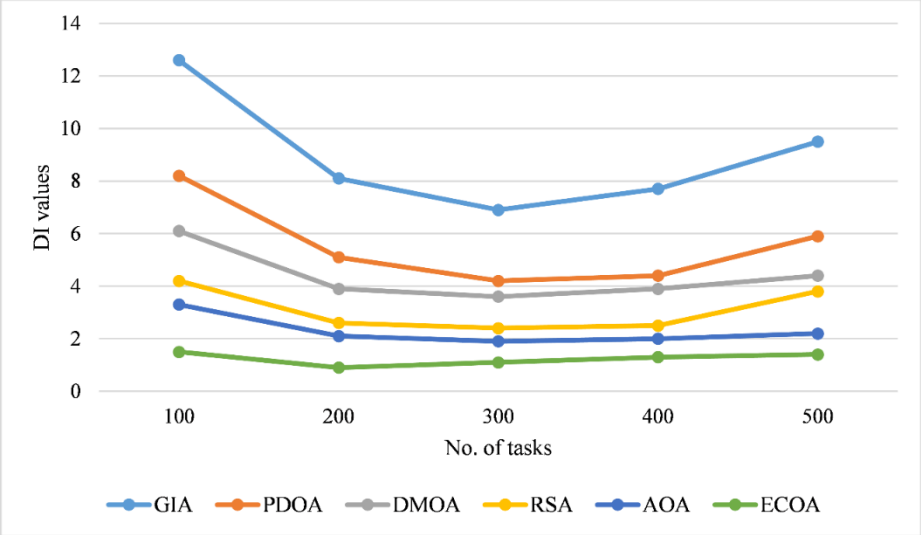

Fig. 5. DI Comparison

V. DISCUSSION

The outcomes demonstrate the effectiveness of ECOA in task scheduling optimization for cloud computing. Through improved makespan, increased resource utilization, and balanced task diversity, ECOA performed better than other algorithms for various workload conditions. Incorporating HDPM, Spearman's rank correlation, and the BAO enabled ECOA to adjust to multiple workload situations while delivering stable performance dynamically.

In addition, the results emphasize the significance of using task scheduling algorithms optimized for a particular workload and resource setup. Although AOA and LPO were effective across workloads, some algorithms, including SSOA and GIA, were workload-sensitive. These observations can inform the design and choice of algorithms for effective cloud resource management.

It is noteworthy that ECOA consistently reduces makespan and optimizes average resource utilization in light and heavy task loads, indicating that it is not tuned to the specific conditions of the dataset but is capable of generalized optimization. On the contrary, enhanced performance becomes even stronger in loads with large task diversity and resource constraints, where the balance between exploration and exploitation becomes paramount. Thus, the proposed algorithm best suits complex, large-scale, and heterogeneous scenarios yet still performs competitively under simpler conditions.

The theoretical contributions made by ECOA directly resulted in practical improvements to cloud computing setups. Cloud computing providers are tasked with efficiently assigning computational resources while satisfying dynamic user demands. If the scheduling of tasks is inefficient, it can result in idle servers, increased power consumption, and failure to fulfill service-level agreements. The proposed method of ECOA overcomes these issues by providing a scalable, adaptive, and low-overhead method that enhances the utilization of resources, minimizes task execution time, and preserves load balance in different intensities of workload. These attributes are highly desired in real-world applications, including multi-tenant data centers, IoT-hardened infrastructures, and edge-cloud hybrid setups since performance degradation in these setups would incur considerable costs in finance and operations. Hence, the theoretical design of ECOA proves to be academically sound and meets practical demands in contemporary cloud computing systems.

Notwithstanding the encouraging performance realized by ECOA, a few limitations should be noted. First, the experimental validations were performed on simulated datasets in a controlled setup, which might not reflect the real-world complexities and variations in cloud environments. Consequently, the algorithm's performance in heterogeneous, time-unpredictable, and large-scale production environments still awaits further confirmation. Second, the implementation focuses mainly on single-objective optimization, aiming for makespan and resource consumption; multi-objective trade-offs, e.g., energy efficiency, service level agreements breaching, or economic cost, were not extensively analyzed.

In addition, while incorporating operators such as polynomial mutation and BAO enhances convergence behavior, the algorithm can introduce some computational overhead in time-sensitive or limited-resource deployments. Future directions must compare ECOA in production of cloud platforms, extend it to multi-objective cases, and analyze its scalability over dynamic loads and different infrastructure setups.

VI. CONCLUSION

This study proposed the ECOA, a new metaheuristic algorithm, to handle cloud computing task scheduling problems. Accordingly, Spearman's rank correlation coefficient, HDPM, and the BAO mechanism were adopted in the ECOA, effectively improving the harmony between exploration and exploitation. The suggested algorithm has been extensively evaluated against some of the latest optimization methods using synthetic data sets and yielded superior results on all key metrics, such as makespan, resource utilization, and task distribution diversity. The experimental results proved that ECOA showed consistently lower makespan values, higher average resource utilization, and stable diversity indices for the increasing workload size.

The dynamic adaptability of ECOA under changing cloud environments brings out its robustness and scalability for effective cloud resource management. ECOA delivers an effective and scalable framework for optimizing task scheduling in cloud computing. Further extension can be done by its application on real-world datasets and finding its performance in multi-objective optimization problems like minimizing energy consumption along with the time of execution of tasks. Exploring various directions for integrating ECOA within a distributed and edge computing environment would also be exciting.

REFERENCES

[1] M. Shariq et al., "Anonymous and reliable ultralightweight RFID-enabled authentication scheme for IoT systems in cloud computing," Computer Networks, vol. 252, p. 110678, 2024.

[2] A. Zhu, H. Lu, S. Guo, Z. Zeng, M. Ma, and Z. Zhou, "SyRoC: Symbiotic robotics for QoS-aware heterogeneous applications in IoT-edge-cloud computing paradigm," Future Generation Computer Systems, vol. 150, pp. 202-219, 2024.

[3] D. Wang, "Improved Cat Swarm Optimization Algorithm for Load Balancing in the Cloud Computing Environment," International Journal of Advanced Computer Science and Applications, vol. 14, no. 7, 2023.

[4] A. Hosseinzadeh, M. Shahin, M. Maghanaki, H. Mehrzadi, and F. F. Chen, "Minimizing wastevia novel fuzzy hybrid stacked ensembleof vision transformers and CNNs to detect defects in metal surfaces," The International Journal of Advanced Manufacturing Technology, pp. 1-26, 2024, doi: 10.1007/s00170-024-14741-y.

[5] B. Ya-meng, W. Yang, and W. Shen-shen, "Deadline-aware Task Scheduling for Cloud Computing using Firefly Optimization Algorithm," International Journal of Advanced Computer Science and Applications, vol. 14, no. 5, 2023.

[6] F. S. Prity, M. H. Gazi, and K. A. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," Cluster computing, vol. 26, no. 5, pp. 3037-3067, 2023.

[7] S. Gurusamy and R. Selvaraj, "Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network," Expert Systems with Applications, vol. 249, p. 123554, 2024.

[8] A. Ahmed, M. Adnan, S. Abdullah, I. Ahmad, N. Alturki, and L. J. Menzli, "An Efficient Task Scheduling for Cloud Computing Platforms Using Energy Management Algorithm: A Comparative Analysis of Workflow Execution Time," IEEE Access, 2024.

[9] S. E. Mahdimahalleh and V. T. Vakili, "Optimizing Scheduling Techniques for Enhanced Carrier Aggregation in LTE-Advanced Networks," European Journal of Electrical Engineering and Computer Science, vol. 8, no. 6, pp. 26-32, 2024, doi: https://doi.org/10.24018/ejece.2024.8.6.675.

[10] R. Nithiavathy, S. Janakiraman, and M. Deva Priya, "Adaptive Guided Differential Evolution - based Slime Mould Algorithm - based efficient Multi - objective Task Scheduling for Cloud Computing Environments," Transactions on Emerging Telecommunications Technologies, vol. 35, no. 1, p. e4902, 2024.

[11] M. B. Bagherabad, E. Rivandi, and M. J. Mehr, "Machine Learning for Analyzing Effects of Various Factors on Business Economic," Authorea Preprints, 2025, doi: https://doi.org/10.36227/techrxiv.174429010.09842200/v1.

[12] A. Kermani et al., "Energy management system for smart grid in the presence of energy storage and photovoltaic systems," International Journal of Photoenergy, vol. 2023, no. 1, p. 5749756, 2023, doi: https://doi.org/10.1155/2023/5749756.

[13] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G.-B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," The Journal of Supercomputing, vol. 76, no. 8, pp. 6302-6329, 2020.

[14] S. Velliangiri, P. Karthikeyan, V. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," Ain Shams Engineering Journal, vol. 12, no. 1, pp. 631-639, 2021.

[15] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Multi objective task scheduling in cloud computing using cat swarm optimization algorithm," Arabian journal for science and engineering, vol. 47, no. 2, pp. 1821-1830, 2022.

[16] K. Malathi and K. Priyadarsini, "Hybrid lion–GA optimization algorithm-based task scheduling approach in cloud computing," Applied Nanoscience, vol. 13, no. 3, pp. 2601-2610, 2023.

[17] R. Ghafari and N. Mansouri, "E-AVOA-TS: Enhanced African vultures optimization algorithm-based task scheduling strategy for fog–cloud computing," Sustainable Computing: Informatics and Systems, vol. 40, p. 100918, 2023.

[18] 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.

[19] A. Boroumand, M. Hosseini Shirvani, and H. Motameni, "A heuristic task scheduling algorithm in cloud computing environment: an overall cost minimization approach," Cluster Computing, vol. 28, no. 2, p. 137, 2025.

[20] M. Ghasemi, M. Zare, A. Zahedi, M.-A. Akbari, S. Mirjalili, and L. Abualigah, "Geyser inspired algorithm: a new geological-inspired meta-heuristic for real-parameter and constrained engineering optimization," Journal of Bionic Engineering, vol. 21, no. 1, pp. 374-408, 2024.

[21] A. E. Ezugwu, J. O. Agushaka, L. Abualigah, S. Mirjalili, and A. H. Gandomi, "Prairie dog optimization algorithm," Neural Computing and Applications, vol. 34, no. 22, pp. 20017-20065, 2022.

[22] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Dwarf mongoose optimization algorithm," Computer methods in applied mechanics and engineering, vol. 391, p. 114570, 2022.

[23] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer," Expert Systems with Applications, vol. 191, p. 116158, 2022.

[24] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," Computer methods in applied mechanics and engineering, vol. 376, p. 113609, 2021.