Hybrid Artificial Bee Colony and Bat Algorithm for Efficient Resource Allocation in Edge-Cloud Systems

Jiao GE, Bolin ZHOU, Na LIU*
Cangzhou Normal University, Hebei Cangzhou 061001, China

***Abstract*—Integrating edge and cloud computing systems builds up a powerhouse, a framework for realizing real-time data processing and conducting large-scale computation tasks. However, efficient resource allocation and task scheduling are outstanding challenges in these dynamic, heterogeneous environments. This paper proposes an innovative hybrid algorithm that amalgamates the features of the Bat Algorithm (BA) and Artificial Bee Colony (ABC) to meet such challenges. The ABC algorithm's solid global search capabilities and the BA's efficient local exploitation are merged for efficient task scheduling and resource allocation. Dynamic adaptation of the proposed hybrid algorithm accommodates such conditions by balancing exploration and exploitation through periodic solution exchanges. Experimental evaluations highlight that the proposed algorithm can minimize execution time and costs involving resource utilization by guaranteeing proper management of task dependencies using a Directed Acyclic Graph (DAG) model. Compared to the available methods, the proposed hybrid technique generates better performance metrics concerning reduced makespan, improved resource utilization, and lower computational delays concerning resource optimization in an edge-cloud context.**

Keywords*—Cloud computing; edge computing; resource allocation; optimization; task scheduling

I. Introduction

Edge and cloud computing have changed how computation is performed. Real-time processing and elasticity of resources can now be achieved. Edge computing minimizes latency by executing data near the source and is well-suited to the Internet of Things (IoT), intelligent cities, and autonomous vehicle deployments [1]. In contrast, cloud computing can offer substantial resources for computation and storage but entails more significant latency for processing data remotely [2]. The integration of edge and cloud systems into a unified continuum-field edge benefits the two paradigms in an optimum manner: task execution, resource allocation, and network performance [3]. This is an increasingly crucial hybrid approach for meeting growing demands in today's computational ecosystems [4].

Despite the potential, resource allocation and task scheduling in the edge-cloud ecosystem are still challenging. Most existing algorithms suffer from exploration or exploitation, resulting in suboptimal task allocation, increased delays, and inefficient resource use [5]. Besides, most methodologies cannot function effectively in dynamic and heterogeneous environments where tasks may have dependencies, network conditions may change, and diverse resource constraints create a highly complex optimization problem [6, 7]. These limitations are a challenge to innovate in finding ways to best perform in edge-cloud systems.

To counter these challenges, the present study develops a novel hybrid algorithm combining the Artificial Bee Colony (ABC) and Bat Algorithm (BA). ABC efficiently explores the diverse solution space, while the BA exploits the local search area [8, 9]. The proposed hybrid algorithm dynamically balances the two aspects: exploration and exploitation. It can explore more candidate solutions of better quality and achieve superior performance in resource allocation and task scheduling. A Directed Acyclic Graph (DAG) model manages task dependencies, ensuring efficient execution while avoiding deadlocks. The hybrid algorithm also incorporates periodic information exchange between ABC and BA populations to enhance adaptability and convergence. This research has made the following primary contributions:

- Proposing a hybrid ABC-BA algorithm suitable for dynamic edge-cloud environments;
- Developing with DAG-based task dependency management to ensure efficient task scheduling;
- Extensive performance evaluations suggest minimum execution time, cost, and makespan over the existing methods;

The rest of the study proceeds by reviewing the literature relevant to the subject matter and identifying the limitations in Section II. Section III discusses the system model and architecture and addresses design principles. Section IV outlines basic concepts and background that constitute the background necessary for a study. Section V explains the proposed ABC-BA hybrid algorithm's design principle, working mechanism, and merits. A general simulation and comparative analysis can be presented in Section VI. Finally, the paper summarizes the results found in Section VII, along with implications and further development directions of the research.

II. Literature Review

Xia and Shen [10] have proposed a hybrid approach for allocating resources in mobile edge cloud systems, merging Ant Colony Optimization (ACO) and Genetic Algorithm (GA). This approach maximizes system utility while balancing economic costs, energy conservation, and task latency. ACO can generate initial populations, while GA improves solutions using crossover and mutation.

Chafi, et al. [11] proposed a comparative study between GA and Particle Swarm Optimization (PSO) in edge-fog cloud architectures for resource allocation. Using the FogWorkflowSim simulator, this work underlines the logical structure of GA's fruition and the heuristic behaviors of PSO while showing efficiency for optimization in resource allocation under specific constraints.

Haghighat Afshar, et al. [12] developed a new metaheuristic ERSGWO in mobile edge computing resource allocation, introducing a novel integration between the Grey Wolf Optimizer (GWO) and Reptile Search Algorithm (RSA). The algorithm is enhanced by introducing an intermediate neighborhood exploration phase that boosts agent performance against getting stuck in local optima. Test outcomes show that the performance improvement attained is as high as 97.7% across 12 scenarios.

Yu, et al. [13] present a decomposition-driven multi-objective optimization algorithm (MOEA/D-EoD) for mobile edge computing systems. Using estimation-of-distribution models, the proposed algorithm can optimize continuous and discrete decision variables associated with resource allocation and task offloading. It has been shown to perform very well in multi-user, multi-server collaborative edge systems.

Yin, et al. [14] presented an effective convergent firefly algorithm, called the ECFA, for coordinating sensitive tasks in a cloud-edge environment. This approach introduces novel concepts of boundary traps to enhance exploration and improve convergence. Testing on several cloud-edge scheduling problems demonstrates its better performance than the standard Firefly algorithm. Wang, et al. [15] introduced Quantum Particle Swarm Optimization (QPSO) for device-edge-cloud cooperative computing (DE3C). QPSO outweighs the other metaheuristics on user experience or resource efficiency but has shortcomings in large-scale problems.

Salehnia, et al. [16] suggested a Multiple-Objective Moth-Flame Optimization (MOMFO) algorithm for task scheduling in IoT-based fog-cloud systems. This approach reduced energy consumption and CO2 emissions, along with task delays, while improving IoT system performance. Khiat, et al. [17] presented a genetic-based scheduling algorithm (GAMMR) to lower latency and energy consumption. Several datasets simulated with GAMMR performed better than the standard genetic algorithms by 3.4%.

Although some of the methods summarized in Table I provide valuable insights into resource allocation and task scheduling, significant gaps still need to be addressed. Various methods, such as ERSGWO and MOEA/D-EoD, are algorithm-specific, limiting their adaptability to diverse and dynamic edge-cloud environments. Besides, hybrid methods in ACO-GA and ECFA predominantly suffer from an inability to balance exploration and exploitation, returning suboptimal solutions in multi-objective contexts.

TABLE I. AN OVERVIEW OF PREVIOUS RESOURCE ALLOCATION AND TASK SCHEDULING METHODS

Reference	Contribution	Limitation
[10]	Proposed a hybrid ACO-GA algorithm for resource allocation, maximizing utility while reducing cost and latency.	Computational overhead due to the complexity of combining ACO and GA makes it less practical for large-scale and highly dynamic edge-cloud scenarios.
[11]	Compared GA and PSO for resource allocation in edge-fog cloud environments, highlighting strengths and weaknesses.	GA requires significant computational resources and a large number of iterations to converge, while PSO's performance heavily depends on parameter tuning and may lead to premature convergence.
[12]	Developed ERSGWO, a hybrid RSA-GWO algorithm, enhancing exploration and avoiding local optima for MEC environments.	The algorithm's design focuses exclusively on MEC-specific scenarios, making it less generalizable to broader edge-cloud or fog-cloud architectures.
[13]	Designed MOEA/D-EoD, a multi-target algorithm for task offloading and resource allocation in collaborative MEC systems.	The approach is limited by its scalability issues, particularly when handling a large number of mobile users and servers in complex, real-world edge-cloud environments.
[14]	Introduced ECFA for task scheduling in cloud-edge systems, enhancing convergence and robustness.	Sensitive to parameter settings, requiring extensive tuning for different applications, and prone to falling into local optima in highly dynamic conditions.
[15]	Applying QPSO for task scheduling in DE3C systems improves resource efficiency and customer satisfaction.	While effective for small to medium-sized problems, QPSO struggles with scalability and fails to deliver consistent performance for large-scale task scheduling.
[16]	Proposed MOMFO for IoT task scheduling, reducing energy consumption, CO2 emissions, and task delays.	The algorithm is tailored for fog-cloud systems, limiting its application to more generalized edge-cloud environments or scenarios with highly dynamic task demands.
[17]	Designed GAMMR to optimize latency and energy in fog-cloud systems, achieving improved scheduling efficiency.	Achieves only marginal improvements (3.4%) over the standard GA, making it less competitive in scenarios demanding significant enhancements in performance.

Scalability remains an issue, as most approaches like GAMMR and QPSO represent low efficiency when dealing with large-scale problems. Besides, approaches like MOMFO and MOEA/D-EoD need to be better generalized for different architectures. Along this line of thought, this paper presents a hybrid ABC-BA algorithm that incorporates ABC's global exploration capability and the efficient exploitation of BA. The proposed design targets the shortcomings identified and aims to elevate adaptability, scalability, and resource optimization in an edge-cloud system.

III. SYSTEM MODEL

As depicted in Fig. 1, the resource allocation system model integrates edge resources, cloud resources, and mobile users in an edge-cloud environment. Mobile users initiate requests for task executions, which are forwarded by a local dispatch system to the edge servers managed by the orchestrator of each edge. An edge orchestrator comprises three modules: a task analyzer and policy enhancement module, a task scheduler, and a global resource manager. Table II summarizes the mathematical symbols and notations used throughout this study.

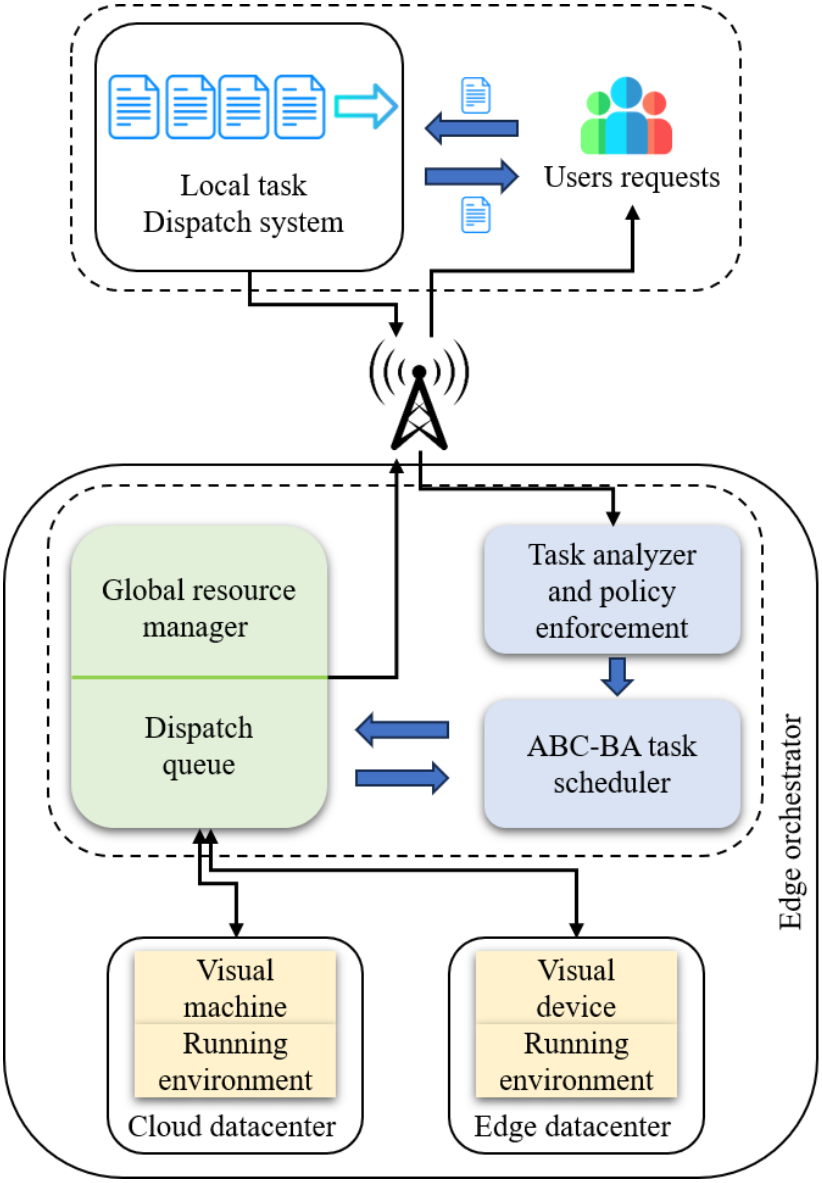

Fig. 1. System model.

TABLE II. SYMBOLS AND DESCRIPTIONS

Symbol	Description	Symbol	Description
p_l	Probability of selecting individual ℓ in ABC phase	t	Number of iterations
gc	General number of cycles	t_{BA}	Remaining iterations for bat algorithm
N	Population size	sc	Success control parameter
z_j^l	Component of individual ℓ for ABC algorithm	mnc	Maximum number of changes between BA and ABC
$f(x)$	Target function	v_i^t	Velocity of the i^{th} bat at time t
x^*	Best solution in the population	x_i^t	Position of the i^{th} bat at time t
max_i	Maximum iteration	D	Dimension of the problem space
A	Loudness in the bat Algorithm	h_j	Hourly resource leasing price
r	Pulse emission rate	t_{ij}	Task execution time
t_{ABC}	Remaining iterations for ABC algorithm	c_{ij}	Communication time
C	Total cost	M	Makespan

The task analyzer subjects the tasks to a task dependency model that enforces execution to avoid delays and deadlocks. Another essential component is the global resource manager, responsible for monitoring real-time network metrics to make resource allocation decisions. The scheduler utilizes the hybrid ABC-BA algorithm to assign tasks to edge or cloud resources based on execution order, task requirements, and current network performance. ABC-BA couples the global searching capability of the ABC algorithm with the fast local searches of the BA to provide optimal task allocation and adaptation to dynamic conditions.

The task scheduling algorithm takes the request from the task requests by going through QoS requirements and constraints. It decides whether tasks should be executed on cloud resources, edge resources, or even both to minimize execution time and cost. Resource utilization rates, such as the usage hour rate for a cloud virtual machine or the per-minute rate for an edge server, form the basis for calculating the execution cost of a task. This model can upload dependent and independent tasks on which different cost structures, like Amazon EC2 and Microsoft Azure, differ for various providers.

Makespan measures the total time needed to accomplish all tasks, influenced by task execution time on edge or cloud devices and communication delays, calculated as follows:

$$M = max\left(\underset{t_i \in C_i}{max}(t_{ic} + c_{ic}) + \underset{t_i \in E_i}{max}(t_{ie})\right) \quad (1)$$

Where t_{ie} is the execution time on edge VMs, t_{ic} is the execution time on cloud VMs, and c_{ic} is the communication delay between edge and cloud. Cost is mathematically modeled as follows:

$$C = \sum_{i=1}^{k}\sum_{j=1}^{n} Cost_{ij}, \quad where \quad Cost_{ij} = h_j \times t_{ij} \quad (2)$$

Where h_j denotes the resource price per hour, and t_{ij} represents the task execution duration.

A Directed Acyclic Graph (DAG) represents the dependencies between tasks to ensure the order is respected and prevent deadlocks. Therefore, nodes represent tasks, and an edge implies dependency. The finish time of each task and the dependency constraint are computed as follows.

$$F_i = S_i + D_i, \quad \forall t_i \in T \quad (3)$$

$$S_j \geq F_i, \quad \forall (t_i, t_j) \in E \quad (4)$$

Where S_i is the start time, D_i is the duration, and F_i is the finish time of task t_i.

The network model optimizes interactions between edge devices, servers, and cloud resources. Each edge device connects to one edge server, computed as:

$$\sum_{j \in E} z_{ij} = 1, \quad \forall ED_i \in D \quad (5)$$

Where z_{ij} indicates a connection between edge device ED_i and edge server ES_j. Edge servers can connect to multiple cloud servers:

$$\sum_{j \in C} w_{ij} \geq 0, \quad \forall ES_i \in E \quad (6)$$

The flow conservation constraint ensures data stability across nodes:

$$\sum_{k:(k,i)\in L} f_{ki} - \sum_{k:(i,k)\in L} f_{ik} = b_i, \quad \forall i \in D \cup E \cup C \quad (7)$$

Where b_i represents data flow demand, and f_{ik} denotes data flow between nodes. This model ensures efficient task execution by integrating the hybrid ABC-BA algorithm into resource and network management in the edge-cloud continuum.

IV. BACKGROUNDS

A. Artificial Bee Colony Algorithm

The ABC algorithm takes inspiration from the foraging behavior of honeybee colonies. Presented by Karaboga and Akay [18], it was used to solve continuous optimization problems. This algorithm works in iterative phases, comprising several steps to effectively achieve optimum solutions for optimization problems. In the initial phase, a population of N_p individuals (candidate solutions) is generated using Eq. 8, randomly within a specified range $[-100, 100]^m$, where m represents the problem's dimensionality.

$$z_j^l = z_{min} + (z_{max} - z_{min}) \cdot rand(), \quad l \in \{1,2,\dots,N_p\}, j \in \{1,2,\dots,m\} \quad (8)$$

z_{min} and z_{max} denote the bounds, and rand() is a random scalar in the range [0 , 1).

Employed bees search for improved solutions near their current positions. A new solution v_j is generated as follows:

$$v_j = z_j^l + (2 \cdot rand() - 1) \cdot \left(z_j^l - z_j^k\right) \quad (9)$$

Where k is a random individual index ($k \neq l$). The fitness of each candidate solution is evaluated using the objective function, defined as:

$$fit_l = \begin{cases} \dfrac{1}{1+f_l}, & if\ f_l \geq 0 \\ 1 + |f_l|, & otherwise \end{cases} \quad (10)$$

Each solution's probability of being selected for further exploration is determined as:

$$p_l = \frac{fit_l}{\sum_{t=1}^{N_p} fit_t}, \quad l \in \{1,2,\dots,N_p\} \quad (11)$$

Onlooker bees exploit food sources based on probabilities calculated in the previous step. Similar to the employed bees, a new solution is generated using Eq. 12.

$$v_j = z_j^l + (2 \cdot rand() - 1) \cdot \left(z_j^l - z_j^k\right) \quad (11)$$

If a food source cannot be improved after a defined limit of attempts, the corresponding bee becomes a scout and generates a new random solution as follows:

$$[v_j = z_{\min} + (z_{\max} - z_{\min}) \cdot \text{rand}() \quad (12)$$

B. Bat Algorithm

The BA is a heuristic algorithm inspired by echolocation patterns, where bats employ sound waves to locate their prey and avoid collision with objects. The delay between high-frequency sound pulse transmission and reception determines bats' distance from their prey. This echolocation capability provides the basis for BA's mechanism of exploration and exploitation in search spaces. This algorithm follows the following underlying principles:

- Echolocation is used by bats to determine their distance from prey.
- Bats move at a velocity v_i toward a position x_i spanning a frequency spectrum $[f_{min}, f_{max}]$, making sounds at various frequencies λ and loudness A to discover their prey.
- Bats adjust their signal's wavelength and pulse rate dynamically while calculating distances.
- The loudness drops from a maximum (A_0) to a minimum value (A_{min}) as a bat approaches its prey, while the pulse emission rate r rises.

In a D-dimensional sphere, the frequency, velocity, and position of the i^{th} bat are updated as follows:

Frequency calculation:

$$f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta \quad (13)$$

Where β is a random number in [0, 1].

Velocity update:

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) \cdot f_i \quad (14)$$

Where x^* is the current global best position.

Position update:

$$x_i^{t+1} = x_i^t + v_i^t \quad (14)$$

For local exploitation, a new solution is generated around the best current solution using a random step as follows:

$$x_i^{t+1} = x_i^t + \epsilon \cdot \vec{A}^t \quad (15)$$

Where ϵ is a random value in $[-1, 1]$, and $\vec{A}^t$ represents the average loudness of all bats at time t.

As bats converge toward the prey, the loudness decreases exponentially while the pulse emission rate increases as follows:

$$A_i^{t+1} = \alpha \cdot A_i^t \quad (16)$$

$$r_i^{t+1} = r_0 \cdot [1 - e^{-\gamma t}] \quad (17)$$

Where α and γ are constants that control the rate of adaptation.

BA dynamically alters the relationship between exploration on the global scale and exploitation on the local scale by dynamically adjusting its parameters. Frequency guides the range of movements, while loudness and pulse rate control the tradeoff between exploring new areas and refining existing solutions. These allow the algorithm to maintain a balance between exploration and exploitation.

V. Proposed Hybrid Algorithm

The proposed hybrid algorithm, ABC-BA, is an amalgamation of the strengths of BA and ABC in an attempt to balance the tradeoffs between exploitation and exploration. Integrating the two algorithms, ABC-BA enhances global search capabilities, population diversity, and solution quality. ABC-BA rectifies BA structural defects by dynamically updating the pulse emission rate (r), along with loudness (A). As iterations progress, r increases, allowing the algorithm to focus on local search, while A decreases to refine the solutions. To enhance the search capability, an inertia weight coefficient (w) is introduced into the velocity formula of BA.

$$v_i^t = \omega . v_i^{t-1} + (x_i^t - x^*) \cdot f_i \quad (18)$$

This coefficient improves global search during initial iterations and gradually emphasizes local search as iterations progress.

The hybrid algorithm divides the population into two equal parts. The BA processes one part, while the other part is treated by the ABC. Both algorithms work independently; however, at certain periods, they share their information with the other to enhance the performance of the whole. After a predetermined iteration cycle (sc), the performance of ABC and BA is assessed based on the number of new solutions generated. If BA performs better (based on ba_sn), its best solutions replace the worst solutions in the ABC group and vice versa for ABC:

- ba_ni: Number of new solutions generated by BA.
- bee_ni: Number of new solutions generated by ABC.
- ba_sn and bee_sn: Counters tracking successful exchanges.

Solutions are exchanged between BA and ABC based on success rates. The parameter ac determines the number of individuals to replace, calculated as follows:

$$mnc = \left(\frac{\max^{(f)} - \text{iteration}}{sc} \right) \cdot 0.6 \quad (19)$$

The hybrid algorithm's complexity depends on the separate complexities of BA and ABC. For a problem size D, the worst-case computational complexity for fitness evaluation is (D), while the complexity of the algorithms is (D·N), where N is the population size. During parallel execution, the complexity is:

$$t \cdot \left(O\left(P \cdot \frac{N}{2} \right) + O\left(2 \cdot P \cdot \frac{N}{2} \right) \right) \quad (20)$$

If one algorithm dominates, the complexity becomes:

$$t_1 \cdot \left(O\left(P \cdot \frac{N}{2} \right) + t_2 . O\left(2 \cdot P \cdot \frac{N}{2} \right) \right) \quad (21)$$

Where t_1 and t_2 are the respective iteration counts for BA and ABC.

By incorporating BA and ABC, the ABC-BA algorithm combines ABC's global search capability with BA's detailed local exploitation advantageously. Such synergy avoids early

convergence of the algorithm to poor suboptimal solutions, improves solution quality, and effectively adapts the algorithm to dynamic optimization problems. Significant exploration and exploitation improvements are shown in the proposed algorithm, especially for large-scale, complex search spaces. Fig. 2 shows the pseudo-code of the proposed hybrid algorithm.

Initialization:
Set population size (N), number of cycles (gc), maximum iterations (max_i), and problem dimension (D).
Define the objective function $f(x)$.
Initialize the population ($\{x_1, x_2, \ldots, x_N\}$) in D-dimensional space.
Define algorithm parameters:
Set BA and ABC parameters.
Determine the initial best solution (x^*) in the population.
Determine exchange parameters:
Set the success control parameter (sc).
Compute the maximum number of changes (mnc) based on Eq. 19.
Start the iterative process:
Begin with iteration $G = 1$.
While $G \leq gc$.
Reset counters:
Initialize counters for new solutions generated (ba_ni= 0, bee_ni=0) and successful exchanges (ba_sn=0, bee_sn=0).
Apply BA to first half of population:
For $t = 1$ **to** max_i:
For $i = 1$ **to** $N/2$:
Generate a new solution (x_{new}) using BA.
If $f(x_{new}) < f(x^*)$, update $x^*$and increment ba_ni.
Apply ABC to second half of population:
For $i = 1$ **to** $N/2$:
Generate a new solution (x_{new}) using ABC.
If $f(x_{new}) < f(x^*)$, update $x^*$and increment bee_ni.
Information exchange between algorithms:
If $mod(t, sc) == 0$:
Compute ba_ni and bee_ni.
If $ba\_ni \geq bee\_ni$:
Replace worst solutions in ABC population with best solutions from BA.
Increment ba_sn.
Else
Replace worst solutions in BA population with best solutions from ABC.
Increment bee_sn.
Check algorithm dominance:
If $ba\_sn \geq mnc$:
Execute remaining iterations using BA.
Else if $bee\_sn \geq mnc$:
Execute remaining iterations using ABC.
Increment iteration:
Increment G and repeat the process.
Output results:
Display the best solution (x^*) after completing all iterations.

Fig. 2. Pseudo-code of hybrid algorithm.

VI. Performance Evaluation

This section compares the performances of ABC-BA against two benchmark algorithms, Fruit fly Simulated Annealing Optimization Scheme (FSAOS) [19] and Quantum-behaved Particle Swarm Optimization (QPSO) [20], implemented using an Edge-CloudSim simulator. In this regard, two edge data centers and one cloud data center were configured by setting different resources to make the scenarios realistic. In ABC-BA, maximum iterations were 100, the population size was 50, and a dynamic switch probability was adopted to balance exploration with exploitation. The experiments were conducted based on task execution metrics: makespan, cost, resource utilization, and task count ranging from 100 to 1000. Makespan measures the total task completion time, while cost evaluates resource utilization expenses. Simulation findings highlight ABC-BA's capability to optimize resource allocation and task scheduling across edge and cloud environments.

As shown in Fig. 3 and 4, ABC-BA consistently outperformed FSAOS and QPSO in terms of makespan. On the cloud, ABC-BA had an increased trend in makespan from 74.69 for 100 tasks to 503.85 for 1000 tasks, while QPSO and FSAOS presented a much larger increase. Similarly, in the case of the edge environment, ABC-BA had lower makespan values, ranging from 31.66 to 339.93, compared to QPSO (41.16 to 441.91) and FSAOS (47.49 to 490.56). These results demonstrate ABC-BA's efficiency in minimizing task execution time, critical for latency-sensitive applications like

IoT and real-time analytics. The findings emphasize its scalability and robustness under increasing workloads.

Cost analysis further demonstrated ABC-BA's superiority in resource efficiency. In the cloud environment, as shown in Fig. 5, ABC-BA achieved the lowest costs, starting at 25.21 and increasing to 333.43, outperforming QPSO and FSAOS, which exhibited costs ranging from 30.25 to 400.12 and 55.86 to 485.06, respectively. On the edge, as shown in Fig. 6, ABC-BA's costs increased from 12.51 to 126.11, while QPSO.

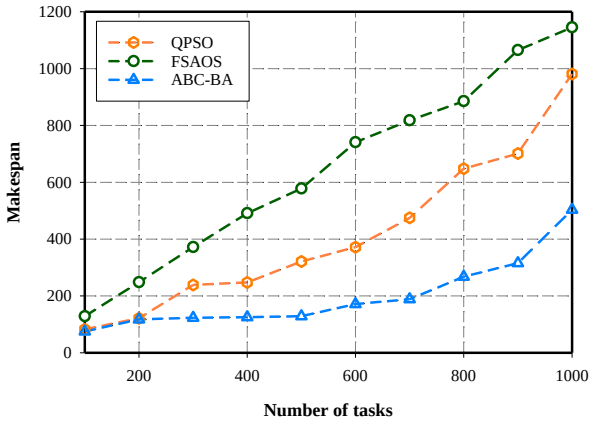

Fig. 3. Makespan results on cloud servers.

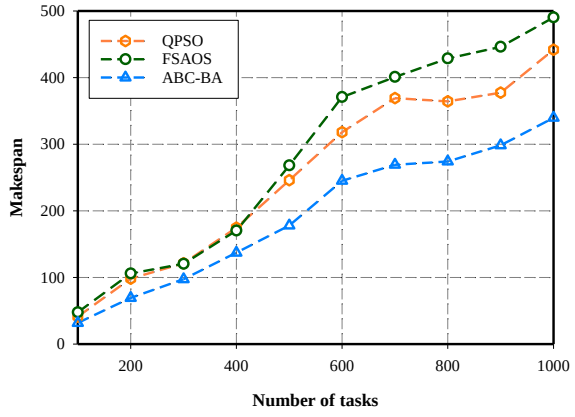

Fig. 4. Makespan results on edge servers.

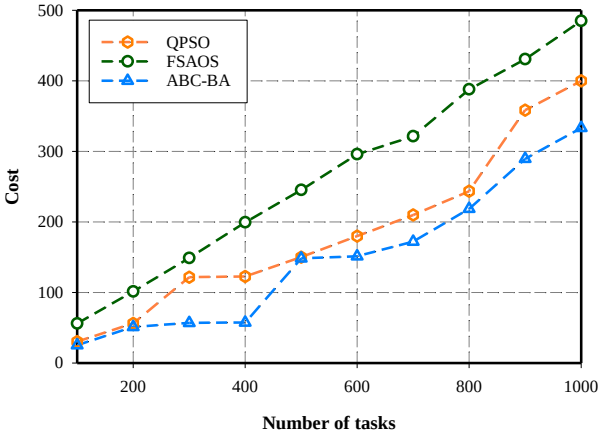

Fig. 5. Task execution on cloud servers.

Ranged from 15.46 to 151.34, and FSAOS ranged from 17.54 to 162.30. The results highlight ABC-BA's ability to minimize operational expenses by optimizing resource allocation. This cost efficiency makes ABC-BA an ideal choice for resource-constrained edge environments and large-scale cloud systems, ensuring reduced execution time and financial overheads while maintaining high performance.

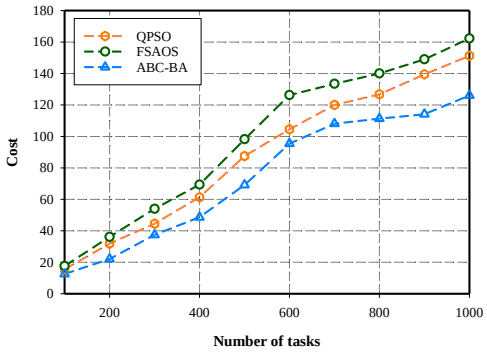

Fig. 6. Task execution on edge servers.

The resource utilization analysis highlights the efficiency of the ABC-BA algorithm in both cloud and edge environments. As shown in Fig. 8, on the edge, the resource utilization values for ABC-BA steadily increase from 1.90 at 100 tasks to 19.05 at 1000 tasks. As shown in Fig. 7, utilization starts at 0.17 in the cloud and rises to 1.68 for the same task sizes. Comparatively, the benchmark algorithms, QPSO and FSAOS, demonstrate higher resource usage in both environments, reflecting less efficient optimization.

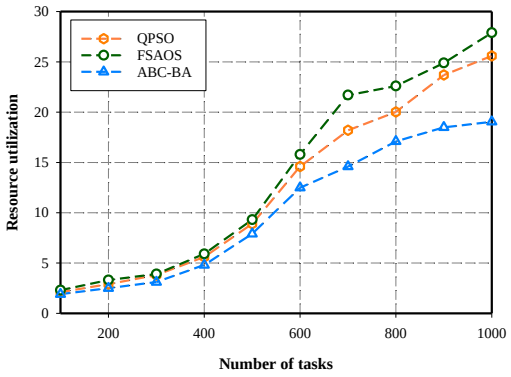

Fig. 7. Resource utilization on edge servers.

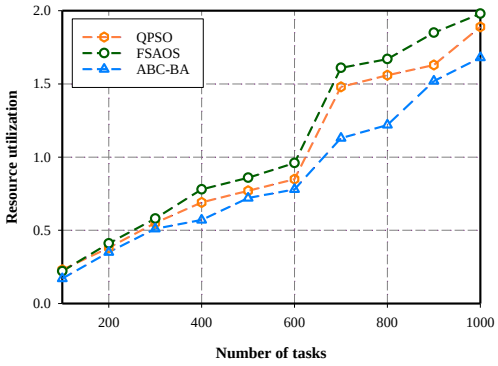

Fig. 8. Resource utilization on cloud servers.

VII. CONCLUSION

This paper proposed a hybrid ABC-BA algorithm for resource allocation and task scheduling in edge-cloud environments. By leveraging the strengths of ABC and BA, ABC-BA addressed the vital challenges of exploration-exploitation balance, reduction of time taken to execute tasks, and optimization of resource utilization. The testing evidence proved that the proposed algorithm performed better than the benchmark methods regarding makespan, cost, and resource utilization. ABC-BA has constantly outperformed the comparatives for a wide range of simulations in both cloud and edge computing under various task size and resource constraints, proving its robustness and scalability.

Future works may implement an improved ABC-BA, considering other optimization algorithms like multi-objective algorithms or deep reinforcement learning methods that might improve the performance even further. The implementation of ABC-BA using real-world edge-cloud environments on different workloads and their integration with sophisticated network models will be of greater value in eliciting insight into the practical workability proposed in this approach. More generally, the proposed ABC-BA algorithm may be able to point out a new frontier in resource optimization and task scheduling, fostering further research in the direction of more efficient and scalable edge-cloud solutions. Additionally, future research will incorporate robust fault tolerance mechanisms to address node failures, including dynamic task reallocation, node health monitoring, and redundancy protocols. By integrating these strategies, we aim to enhance the resilience and practical applicability of the proposed algorithm in unpredictable real-world environments.

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] 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, doi: https://doi.org/10.1002/cpe.6698.

[3] T. Wang, Y. Liang, X. Shen, X. Zheng, A. Mahmood, and Q. Z. Sheng, "Edge computing and sensor-cloud: Overview, solutions, and directions," ACM Computing Surveys, vol. 55, no. 13s, pp. 1-37, 2023, doi: https://doi.org/10.1145/3582270.

[4] G. Baranwal, D. Kumar, and D. P. Vidyarthi, "Blockchain based resource allocation in cloud and distributed edge computing: A survey," Computer Communications, 2023, doi: https://doi.org/10.1016/j.comcom.2023.07.023.

[5] V. Hayyolalam, B. Pourghebleh, A. A. Pourhaji 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, pp. 471-498, 2019, doi: https://doi.org/10.1007/s00170-019-04213-z.

[6] S. Zhang, J. He, W. Liang, and K. Li, "MMDS: A secure and verifiable multimedia data search scheme for cloud-assisted edge computing," Future Generation Computer Systems, vol. 151, pp. 32-44, 2024, doi: https://doi.org/10.1016/j.future.2023.09.023.

[7] W. Liang et al., "TMHD: Twin-Bridge Scheduling of Multi-Heterogeneous Dependent Tasks for Edge Computing," Future Generation Computer Systems, vol. 158, pp. 60-72, 2024, doi: https://doi.org/10.1016/j.future.2024.04.028.

[8] Ş. Öztürk, R. Ahmad, and N. Akhtar, "Variants of Artificial Bee Colony algorithm and its applications in medical image processing," Applied soft computing, vol. 97, p. 106799, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106799.

[9] T. Agarwal and V. Kumar, "A systematic review on bat algorithm: Theoretical foundation, variants, and applications," Archives of Computational Methods in Engineering, pp. 1-30, 2022, doi: https://doi.org/10.1007/s11831-021-09673-9.

[10] W. Xia and L. Shen, "Joint resource allocation at edge cloud based on ant colony optimization and genetic algorithm," Wireless Personal Communications, vol. 117, no. 2, pp. 355-386, 2021, doi: https://doi.org/10.1007/s11277-020-07873-3.

[11] S.-E. Chafi, Y. Balboul, M. Fattah, S. Mazer, and M. El Bekkali, "Enhancing resource allocation in edge and fog-cloud computing with genetic algorithm and particle swarm optimization," Intelligent and Converged Networks, vol. 4, no. 4, pp. 273-279, 2023, doi: https://doi.org/10.23919/ICN.2023.0022.

[12] M. Haghighat Afshar, K. Majidzadeh, M. Masdari, and F. Fathnezhad, "An Energy-Aware Resource Allocation Framework based on Reptile Search Algorithm and Gray Wolf Optimizer for Mobile Edge Computing," Arabian Journal for Science and Engineering, pp. 1-32, 2024, doi: https://doi.org/10.1007/s13369-024-09718-8.

[13] C. Yu, Y. Yong, Y. Liu, J. Cheng, and Q. Tong, "A Multi-Objective Evolutionary Approach: Task Offloading and Resource Allocation Using Enhanced Decomposition-Based Algorithm in Mobile Edge Computing," IEEE Access, 2024, doi: https://doi.org/10.1109/ACCESS.2024.3444607.

[14] L. Yin, J. Sun, J. Zhou, Z. Gu, and K. Li, "ECFA: an efficient convergent firefly algorithm for solving task scheduling problems in cloud-edge computing," IEEE Transactions on Services Computing, 2023, doi: https://doi.org/10.1109/TSC.2023.3293048.

[15] B. Wang, Z. Zhang, Y. Song, M. Chen, and Y. Chu, "Application of Quantum Particle Swarm Optimization for task scheduling in Device-Edge-Cloud Cooperative Computing," Engineering Applications of Artificial Intelligence, vol. 126, p. 107020, 2023, doi: https://doi.org/10.1016/j.engappai.2023.107020.

[16] T. Salehnia et al., "An optimal task scheduling method in IoT-Fog-Cloud network using multi-objective moth-flame algorithm," Multimedia Tools and Applications, vol. 83, no. 12, pp. 34351-34372, 2024, doi: https://doi.org/10.1007/s11042-023-16971-w.

[17] A. Khiat, M. Haddadi, and N. Bahnes, "Genetic-based algorithm for task scheduling in fog–cloud environment," Journal of Network and Systems Management, vol. 32, no. 1, p. 3, 2024, doi: https://doi.org/10.1007/s10922-023-09774-9.

[18] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," Applied mathematics and computation, vol. 214, no. 1, pp. 108-132, 2009, doi: https://doi.org/10.1016/j.amc.2009.03.090.

[19] D. Gabi et al., "Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme," Neural Computing and Applications, vol. 34, no. 16, pp. 14085-14105, 2022, doi: https://doi.org/10.1007/s00521-022-07260-y.

[20] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: adaptive PSO-based task scheduling approach for cloud computing," Sensors, vol. 22, no. 3, p. 920, 2022, doi: https://doi.org/10.3390/s22030920.