Journal of Advanced Research Design

Journal homepage:
https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Adaptive Quantum Behaved Flower Pollination and Tabu Search Algorithm for Energy Efficient Task Management in Edge-Cloud Continuum

Nasiru Muhammad Dankolo[1,3], Nor Haizan Mohamed Radzi[1], Noorfa Haszlinna Mustaffa[1], Farkhana Muchtar[1*], Muhammad Zafran Muhammad Zaly Shah[1], Aryati Bakri[1], Mohd Kufaisal Mohd Sidik[1], Carolyn Salimun@Jackson[2]

1 Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
2 Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia
3 Kebbi State University of Science and Technology, Aliero Nigeria

ARTICLE INFO

Article history:
Received 10 February 2025
Received in revised form 24 March 2025
Accepted 16 June 2025
Available online 18 August 2026

Keywords:
Edge-cloud; energy consumption; flower pollination

ABSTRACT

Efficient task management in a dynamic and resource-constrained edge-cloud environments is essential for minimizing execution delays and reducing energy consumption. This paper presents a novel hybrid optimization algorithm, integrating an Adaptive Flower Pollination Algorithm (FPA) with Tabu Search, to address these challenges. The proposed approach introduces a diversity-based adaptive mechanism for global and local search and leverages Tabu Search to refine solutions and prevent convergence to suboptimal points. Extensive simulations demonstrate that the proposed algorithm outperforms state-of-the-art models in reducing delays and energy usage while balancing resource utilization in both edge and cloud environments. These results highlight the significant improvements achieved over baseline algorithms, providing an effective solution for task scheduling in edge-cloud systems.

1. Introduction

The advent of edge-cloud computing has transformed the distributed computing by bringing computation and data storage closer to the data source typically at the network edge [1]. This paradigm shift addresses the limitations of traditional cloud-centric models, such as high latency and network congestion, making it well-suited for applications that require real-time processing and low-latency communication, such as the Internet of Things (IoT) [2,3], autonomous vehicles and smart city infrastructures. However, as the volume and complexity of data generated by these edge devices continue to grow, efficiently managing task execution in the edge-cloud continuum has emerged as a significant challenge [4]. In this regards, two primary objectives are crucial: minimizing task execution delay and reducing energy consumption [5]. These objectives are inherently conflicting, as

[*] *Corresponding author.*
E-mail address: farkhana@utm.my

https://doi.org/10.37934/ard.135.1.218337

achieving lower delays often requires increased computational resources, which can elevate energy usage, particularly at the resource-constrained edge nodes [6]. Therefore, optimizing task management to balance these competing objectives is essential for the sustainable operation of edge-cloud systems.

Existing task scheduling and resource allocation strategies are mostly metaheuristic-based approaches [7-9]. Metaheuristic-based optimization methods, such as Particle Swarm Optimization (PSO) [10] and Genetic Algorithms (GA) [11], often underperform in dynamic environments due to their static parameter tuning and premature convergence to local optima. These methods are either too rigid, resulting in suboptimal performance, or they lack the ability to effectively balance exploration (searching for new, better solutions) and exploitation (refining existing solutions), leading to premature convergence or inefficient resource utilization. The Flower Pollination Algorithm (FPA), a nature-inspired optimization method, has shown promise in addressing complex optimization problems due to its simplicity and ability to balance exploration and exploitation [12]. However, the standard FPA can still suffer from drawbacks such as premature convergence to local optima and slow convergence rates in high-dimensional or multi-modal search spaces, limiting its effectiveness in dynamic environments like the edge-cloud continuum.

To address these challenges, this paper proposes an Adaptive Quantum Behaved Flower Pollination Algorithm (QFPA) hybridized with Tabu Search. The adaptive QFPA incorporates a Quantum Potential Field that dynamically adjusts its search strategy based on solution diversity, enhancing the algorithm's ability to explore and exploit the search space effectively. Additionally, the integration of Tabu Search provides a robust local search mechanism that helps refine promising solutions, avoid local optima, and maintain solution diversity. The main contributions of this paper are as follows:

i. Proposed an Adaptive QFPA that dynamically balances exploration and exploitation through a Quantum Potential Field.
ii. Integrating Tabu Search as a local search strategy to refine solutions further and prevent premature convergence.
iii. Evaluating the performance of the proposed hybrid algorithm in minimizing task execution delay and reducing energy consumption in the edge-cloud continuum.

2. Literature Review

Task management in edge-cloud computing has garnered significant attention due to the challenges of minimizing execution delays and energy consumption in dynamic, resource-constrained environments. Several metaheuristic algorithms have been proposed to address these challenges, offering varying degrees of success. This section critically examines the limitations of existing approaches and positions the proposed hybrid algorithm within this context.

Metaheuristic methods such as PSO, GA and Grey Wolf Optimizer (GWO) have demonstrated flexibility in solving optimization problems. However, these methods often suffer from premature convergence and suboptimal performance in highly dynamic environments. For instance, PSO's reliance on static parameter tuning limits its adaptability to fluctuating resource demands, leading to inefficient task scheduling in edge-cloud systems [10]. Similarly, GA often requires extensive computational time for convergence, making it less suitable for real-time applications [11]. Recent advancements have explored hybrid algorithms to address these limitations. For example, Hamed *et al.,* [13] integrated Cooperative Search with GA for dynamic scheduling in heterogeneous environments, demonstrating improved task completion times but struggling with energy efficiency in larger systems. Similarly, Najafizadeh *et al.,* [14] proposed a hybrid approach combining deadline-aware scheduling with resource optimization. While this method effectively reduces delays, its performance diminishes under high variability in task arrival rates [14].

Mohammadzadeh *et al.,* [15] introduced an Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. This algorithm targets minimizing both execution delay and energy consumption by optimizing task scheduling based on a dual focus: reducing total execution time and minimizing energy use while maintaining task delay constraints. The approach includes enhancements to genetic operations, such as selection, crossover and mutation, along with a catastrophic strategy to avoid premature convergence. This strategy allows the algorithm to escape local optima and better balance energy consumption and delay. However, the algorithm's current testing conditions, limited to static tasks with strict deadlines, may not fully represent the dynamic environments where new tasks appear spontaneously and demand real-time adjustments. Other algorithms have also been developed to address these objectives. For instance, Too *et al.,* [16] proposed the Multi-Objective Binary Salp Swarm Algorithm (BSSA), which uses binary coding to handle discrete joint request offloading and computational resource scheduling. The algorithm aims to enhance system efficiency by balancing execution delays and energy consumption, thus achieving fair resource allocation. Despite showing improvements in performance through simulations, scalability challenges remain when these algorithms are applied to larger systems or environments where task requests and network conditions fluctuate frequently.

Karaja *et al.,* [17] offered a bi-level multi-objective scheduling approach using an enhanced NSGA-II algorithm to manage dynamic task allocation in heterogeneous multi-cloud environments. Their approach focuses on minimizing execution delays while adhering to strict energy consumption budgets. However, while their method demonstrates efficiency in controlled simulations, the scalability of the algorithm in extremely large-scale or highly dynamic environments has not been comprehensively addressed, potentially limiting its applicability in more volatile operational contexts. In a similar vein, Gabi *et al.,* [18] proposed the Fruit Fly-based Simulated Annealing Optimization Scheme (FSAOS) for dynamic resource scheduling in mobile edge-cloud settings. This algorithm specifically targets the trade-offs between execution delay and execution cost by combining local and global search capabilities to prevent premature convergence. However, while the algorithm has shown promise in simulated environments, its effectiveness in real-world scenarios, which often involve complex and variable conditions, remains uncertain and prone to overfitting.

Abdullahi *et al.,* [8] enhanced the Symbiotic Organisms Search (SOS) algorithm, termed CMABFSOS, to improve task scheduling in cloud computing environments with a focus on minimizing both execution delay and execution cost. The algorithm incorporates adaptive mechanisms for managing benefit factors and constraints, but these enhancements could introduce additional computational complexity, potentially increasing the time and resources required to reach convergence, especially in large-scale systems. Efforts have also been made to integrate different metaheuristic strategies to optimize these objectives. Movahedi *et al.,* [19] proposed an efficient population-based multi-objective task scheduling approach in fog computing systems and Zhang *et al.,* [20] proposed Deadline-Aware Dynamic Task Scheduling in Edge-Cloud. This hybrid approach is designed to minimize both execution delay and energy consumption while avoiding local optima through a more diversified search process. However, balancing these objectives without one negatively impacting the other remains a challenge, and the trade-offs involved are not fully detailed in the study.

Li, *et al.,* [21] developed the Multi-Objective Optimization Artificial Fish Swarm Algorithm (MOOAFSA) [21] to optimize task scheduling in secure cloud environments by focusing on execution time and energy usage. This algorithm attempts to achieve effective and secure scheduling by balancing these objectives; however, its performance may be highly sensitive to initial parameter settings. The study does not extensively explore the robustness of the algorithm under varying configurations, which could impact its effectiveness in different contexts.

Despite these advancements, the limitations of current metaheuristic algorithms in handling the complexities of edge-cloud environments persist. Many of these algorithms struggle to adapt to dynamic conditions where task priorities and resource availability can change rapidly, leading to suboptimal performance in terms of task execution delay and energy consumption [22-24]. Moreover, the need for extensive parameter tuning and the tendency towards premature convergence to local optima remain significant challenges in applying these algorithms to real-world edge-cloud applications [25]. Given these challenges, the FPA emerges as a promising alternative due to its simplicity and easy to implement [26]. While FPA has shown potential in solving complex optimization problems, it still requires improvements to adapt effectively to the dynamic and multi-objective nature of edge-cloud computing. The integration of an Adaptive Quantum Potential Field, which dynamically adjusts search strategies based on solution diversity, combined with Tabu Search as a local search enhancement, represents a novel approach to overcome the limitations identified in the current literature and achieve more efficient task management in edge-cloud environments.

3. System Model and Problem Description

The edge-cloud continuum consists of a hierarchical computing architecture where computational resources are distributed across the cloud, edge servers, and user devices [23]. The system model is designed to leverage both centralized cloud resources and decentralized edge resources to optimize task execution based on latency, energy consumption, and resource availability. In the edge-cloud continuum model, there are three distinct entities which are the Users, Edge Servers and the Cloud Servers. Users generate tasks from their devices (e.g., smartphones, IoT devices) that require processing. These tasks could range from simple data analytics to complex machine learning inferences. Edge Servers are Situated closer to the users; edge servers offer low-latency computation and storage capabilities. They act as intermediaries, handling time-sensitive tasks that require quick processing and minimal delay. While the Cloud Servers provide extensive computational power and storage, suitable for tasks that are less sensitive to latency but require significant processing resources. In this model, tasks generated by user devices are dynamically offloaded to edge or cloud servers based on several factors, such as the nature of the task, the current workload of the edge and cloud resources, network conditions, and the optimization objectives (e.g., minimizing delay and energy consumption).

Task generation in the edge-cloud continuum begins at the user devices, such as smartphones, IoT sensors, or other computing devices, which continuously produce tasks requiring varying levels of computational processing. These tasks can range from simple data analytics to complex machine learning inferences, characterized by parameters like task size, computation intensity, and deadlines. Each task is generated based on user activity or automated triggers and needs to be processed promptly to meet the application requirements. Once a task is generated, it is packaged with its relevant parameters and sent to the task management system, which decides how and where to execute it based on current system conditions, resource availability, and the optimization objectives of minimizing delay and energy consumption.

The task management system evaluates the tasks by considering factors such as the urgency of the task, the required computational resources, the network conditions, and the current workload of both edge and cloud servers. Based on this evaluation, the system determines the most suitable resource for executing the task. For tasks that require low latency or have tight deadline constraints, the task may be processed at a nearby edge server to minimize transmission delay and provide quick responses. Conversely, for tasks that are computationally intensive but less sensitive to delay, the cloud servers, with their extensive computational capabilities, may be chosen for execution. If the task is assigned to an edge server, it is transmitted through a local network connection, minimizing transmission time. For cloud-bound tasks, data transmission occurs over a potentially longer network path, impacting both the transmission delay and energy consumption.

After allocation, the task waits in the queue of the assigned resource, where it experiences queuing delay depending on the current workload and service rate of the server. Once the task reaches the front of the queue, it undergoes execution, which involves processing based on its computation intensity and the resource capabilities. Execution delay is influenced by the speed and efficiency of the allocated resource, and once processing is complete, the results are sent back to the user device. This entire process, from task generation to result delivery, is dynamic and influenced by multiple factors that determine the overall system performance, particularly the balance between task execution delay and energy consumption. Therefore, the task execution Delay and energy consumption models can be derived from the system.

3.1 Task Execution Delay Model

The task execution delay $D(T_i)$ experienced by task T_i is composed of three components which are transmission delay $D_{transmit}(T_i)$, queuing delay $D_{queue}(T_i)$ and execution delay $D_{exec}(T_i)$ as in Eq. (1).

$$D(T_i) = D_{transmit}(T_i) + D_{queue}(T_i) + D_{exec}(T_i) \tag{1}$$

Where, $D_{transmit}(T_i)$ is the time taken to transmit the task from the user device to the edge server or cloud server. It depends on the network bandwidth B and the size of the task T_s as in Eq. (2).

$$D_{transmit}(T_i) = \frac{T_s}{B} \tag{2}$$

The queuing delay on the other hand is the time the task spends waiting in the queue before being processed. It is influenced by the current workload of the selected resource in Eq. (3).

$$D_{queue}(T_i) = \frac{n}{\mu - \lambda} \tag{3}$$

Where, n is the number of tasks in the queue, μ is the service rate and λ is the arrival rate of tasks. The execution delay $D_{exec}(T_i)$ is the time taken to execute the task once it reaches the front of the queue. It depends on the computational resources available (e.g., CPU speed) and the computation intensity C_i of the task in Eq. (4).

$$D_{exec}(T_i) = \frac{C_i}{R} \tag{4}$$

Where, R is the processing speed of the allocated resources.

3.2 Energy Consumption Model

The total energy consumption $E(T_i)$ for executing a task T_i includes the energy consumed during transmission $E_{transmit}(T_i)$ and the energy consumed during the execution $E_{exec}(T_i)$. The transmission energy $E_{transmit}(T_i)$ is energy required to transmit the task from the user device to the edge or cloud server, given by Eqs. (5) and (6).

$$E(T_i) = E_{transmit}(T_i) + E_{exec}(T_i) \tag{5}$$

$$E_{transmit}(T_i) = P_{trans} \times D_{trans}(T_i) \tag{6}$$

Where, P_{trans} is the power consumed for transmission. The execution energy $E_{exec}(T_i)$ is the energy consumed during the execution of the task on the edge or cloud server or both and is given in Eq. (7)

$$E_{exec}(T_i) = P_{exec} \times D_{exec}(T_i) \tag{7}$$

Where, P_{exec} is the power consumed for execution.

3.3 Multi-Objective Function

The goal is to formulate a multi-objective optimization problem that minimizes both the total task execution delay and the total energy consumption. Let F_1 represent the objective function for minimizing delay and F_2 for minimizing energy consumption as follows in Eqs. (8) and (9).

$$F_1 = \sum_{i=1}^{N} D(T_i) \tag{8}$$

$$F_2 = \sum_{i=1}^{N} E(T_i) \tag{9}$$

Where, N is the total number of tasks. The multi-objective optimization can be expressed as Eq. (10).

$$F(x) = \min(F_1, F_2) \tag{10}$$

and is subjected to deadline constraints $D(T_i) \leq D_l$. The proposed hybrid algorithm, combining the Adaptive Quantum Behaved Flower Pollination Algorithm (QFPA) with Tabu Search, aims to find an optimal or near-optimal solution for this multi-objective problem.

4. Proposed Algorithm

4.1 Overview of the Flower Pollination Algorithm (FPA)

The FPA is a nature-inspired optimization algorithm based on the pollination process of flowering plants [27]. Pollination is the process by which pollen is transferred from the male part of a flower to the female part, enabling fertilization and reproduction. FPA mimics this process to explore and exploit the search space effectively, balancing between global and local search strategies to find optimal or near-optimal solutions. The FPA consists of two main processes: global pollination and local pollination. The global pollination step is inspired by cross-pollination, where pollen is carried over long distances by biotic and abiotic agents like insects or wind, enabling exploration across the search space. The local pollination step, on the other hand, mimics self-pollination, where pollen transfer occurs within the same plant or nearby flowers, leading to a more localized search. The mathematical model of the FPA can be summarized with the following equations.

Global pollination is performed using Lévy flights, a type of random walk characterized by step lengths that follow a Lévy distribution. The global pollination step is expressed in Eq. (11).

$$x_i^{t+1} = x_i^t + \gamma L(x_i^t + g^*) \tag{11}$$

Where, x_i^t is the solution vector (flower) at iteration t, g^* is the current best solution, γ is the step size scaling factor, and L is the represent the Levy flight distribution defined by Eq. (12).

$$L \approx \frac{\lambda}{\Gamma(1+\lambda)\sin\left(\frac{\pi\lambda}{2}\right)} \cdot \frac{1}{s^{1+\lambda}} \tag{12}$$

Where, s is the step size, λ is constant parameter typically set to 1.5. The local pollination is modelled as in Eq. (13).

$$x_i^{t+1} = x_i^t + \in \left(x_j^t + x_k^t\right) \tag{13}$$

Where, x_j^t and x_k^t are two solutions chosen at random from the population, and ∈ is a random number drawn from a uniform distribution in the range [0,1].

A switch probability $p \in [0,1]$ determines whether global or local pollination is applied. Typically, a value like p = 0.8 is chosen to favor global search in the initial iterations and gradually focuses more on local search as the algorithm progresses. The FPA effectively balances exploration and exploitation; however, it can suffer from drawbacks such as premature convergence to local optima and slow convergence rates in high-dimensional and multi-modal search spaces and poor solution initialization.

4.2 Solution Initialization

To enhance the initialization of solutions in the FPA, the Chaotic Circle Map is used. The Chaotic Circle Map introduces chaos theory principles into the initialization phase, allowing the algorithm to generate a more diverse and well-distributed set of initial solutions [28]. This improves the algorithm's ability to explore the solution space effectively and avoid premature convergence to local optima. The Chaotic Circle Map is a simple yet effective mathematical model that exhibits chaotic behavior, generating sequences that can be used to initialize solutions in optimization algorithms. The Circle Map is defined as follows:

$$x_{n+1} = \left(x_n + \Omega - \frac{K}{2\pi}\sin(2\pi x_n)\right) \bmod 1 \tag{14}$$

where x_n is the current value in the sequence, Ω is a constant representing the angular frequency, K is a control parameter that determines the degree of chaos in the sequence and $\bmod 1$ ensures the value remains within the range [0,1]. The procedure for solution initiation is given in algorithm 1 (Table 1).

Table 1
Algorithm 1

```
1.  Algorithm 1: Solution initialization using Circle Map
2.      //Set parameters for Chaotic Circle Map:
1.  Omega (Ω) = 0.3, K = 0.5, Initial value $x_0$ in range [0,1], Population size N, Solution space
    boundaries [a, b]
3.      //Generate chaotic sequence:
2.  Initialize $x = x_0$
3.  For i = 1 to N:
4.  x = (x + Ω - (K / (2π)) * sin (2π * x)) mod 1
5.  chaotic_values[i] = x
4.      //Map chaotic values to the solution space:
6.  for each chaotic value $x_i$ in chaotic_values []
7.  $xsolution_i = a + (b - a) * x_i$
8.      // Initialize FPA population
9.  Return mapped solutions as the initial population for FPA
```

The use of the Chaotic Circle Map for solution initialization introduces a higher degree of diversity in the initial population, which enhances the FPA's ability to explore the solution space more comprehensively. The chaotic sequence generated by the Circle Map ensures that the initial solutions are not only well-distributed but also cover a wider range of potential solutions, increasing the likelihood of finding the global optimum. This initialization technique reduces the chances of premature convergence by promoting a more thorough exploration of the search space from the very beginning of the optimization process.

4.3 Quantum Behaved Flower Pollination Algorithm (QFPA)

The Quantum Behaved Flower Pollination Algorithm (QFPA) introduces quantum computing principles to enhance the standard FPA’s search capabilities. The need for QFPA arises from the limitations of the standard FPA in dynamic and complex environments, such as premature convergence and suboptimal exploration of the solution space. QFPA leverages the concepts of quantum mechanics, such as superposition and probability amplitudes, to allow more diverse and flexible exploration of the search space. This is particularly important for multi-objective optimization problems, where maintaining a diverse set of solutions is critical for balancing conflicting objectives. The key improvement of QFPA over the standard FPA is the integration of a Quantum Potential Field, which dynamically adjusts the probability distribution governing the search direction based on the diversity of the solutions. This allows the algorithm to adaptively balance exploration and exploitation, enhancing its ability to avoid local optima and accelerate convergence towards the global optimum.

In QFPA, each solution is represented in a quantum state, allowing it to exist in multiple states simultaneously, enhancing the diversity of the search. The position update of each solution is controlled by the Quantum Potential Field, which is modelled using a Gaussian probability distribution as presented in Eq. (15).

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{15}$$

Where, P(x) represents the probability of selecting a particular solution x, μ is the mean position of the current set of solutions, and σ is the standard deviation that dynamically adjusts based on the diversity of the solutions. The dynamic adjustment of σ enables the algorithm to expand or narrow the search space adaptively. Hence, the QFPA enhanced global and local search steps are given in Eq. (16) and Eq. (17) respectively.

$$x_i^{t+1} = x_i^t + \gamma L(x_i^t + g^*).P(x) \tag{16}$$

$$x_i^{t+1} = x_i^t + \in \left(x_j^t + x_k^t\right).P(x) \tag{17}$$

The pseudo code for QFPA is given in algorithm 2 (Table 2).

Table 2
Algorithm 2

```
5.   Algorithm 2: QFPA procedure
1.   Generate initial solution using Algorithm 1, set the initial values for parameters N, $\gamma$,
     $p$, $\mu$, and σ
2.   For t= to N
6.        //Quantum Potential Field Update
3.   calculate the diversity of the current solutions
4.   Adjust σ dynamically based on the diversity
5.   If diversity is high, increase σ to encourage exploration; if diversity is low, decrease σ
     to focus on exploitation.
7.        //Decide whether to perform a global search or a local search
6.   if rand $<$ p
7.   Global and Local Search using Eq. (16)
8.   Else
9.   Local Search using Eq. (17)
8.        //Evaluation
10.  Evaluate each solution using multi-objective function
11.  Update the current best solution $g^*$ if a better solution is found.
12.  Repeat
13.  Return best solution
14.  End
```

4.4 Adaptive QFPA with Tabu Search

To further enhance the search process and refine the solutions identified by QFPA, Tabu Search (TS) is incorporated into the algorithm. The rationale behind this integration is that while QFPA excels in maintaining diversity and exploring the global solution space, it may still struggle with fine-tuning solutions and avoiding premature convergence in local optima. TS complements QFPA by providing a structured local search mechanism that intensively explores the neighborhood of promising solutions, ensuring that high-quality solutions are not missed due to restrictive search conditions. To incorporate TS into the QFPA, we need to modify the solution update equations to account for the local search optimization provided by TS. The integration of TS involves refining the current best solution obtained from the QFPA's global and local search processes by exploring its neighborhood more thoroughly [29]. The updated equations reflect how TS is applied to the QFPA framework to achieve this.

After the initial population of solutions is generated and updated using the QFPA's global and local search mechanisms, the best solution g^* from the current population is selected for further refinement using TS. The position of the current best solution is perturbed to explore its neighborhood. The new candidate solutions, denoted as x_{new}, are generated using the Eq. (18).

$$x_{new} = x_{current} + \Delta x \tag{18}$$

Where, $x_{current}$ is the current best solution, and Δx represents a small perturbation vector applied to explore the local neighborhood. This perturbation is typically generated randomly within a small range to ensure that the local search is focused around the current solution. The fitness of each new candidate solution x_{new} is evaluated using the multi-objective function that aims to minimize both task execution delay and energy consumption. If x_{new} provides a better objective value than $x_{current}$, and it is not in the tabu list, it becomes the new current solution.

To incorporate the memory aspect of TS, a tabu list T is maintained to store the recently visited solutions, ensuring that the algorithm does not revisit these points. The updated candidate solution is checked against the tabu list, and if it is not present in T or satisfies the aspiration criterion (i.e., it offers a significant improvement in the objective function), it is accepted as the new best solution. The equation for the local search update within the TS phase thus becomes:

$$x_{best} = \begin{cases} x_{new}, & if\ F(x_{new}) < F(x_{current}) and\ x_{new} \notin T, \\ x_{new}, & if\ F(x_{new}) < F(x_{best})\ (aspiration\ condition) \\ x_{current}, & otherwise \end{cases} \tag{19}$$

Here, F(x) denotes the multi-objective fitness function, and x_{best} represents the current best solution after applying TS. This equation ensures that TS can dynamically refine the solution space while integrating smoothly with the adaptive global search behavior of QFPA, leading to a more robust and effective optimization process. The pseudo code for the Hybrid QFPA-TS is presented in algorithm 3 (Table 3).

Table 3
Algorithm 3

```
9.  Algorithm 3: Adaptive QFPA-TS Algorithm
10. Set population size N, initialize solution using Algorithm 1, Set QFPA parameters: Iter, γ,
    p, μ and σ. Set Tabu Search parameters: T, K,
1.  For each solution $x_i$ in X
2.  Calculate $F(x_i)$ : Eq. (10)
3.  Identify the current best solution g* in the population X
4.  Repeat until convergence or maximum iterations reached:
5.  Calculate the diversity of the population X
6.  Adjust Quantum Potential Field parameter σ based on diversity:
7.  For each solution $x_i$ in X:
8.  Generate a random number r in [0, 1]
9.  If r < p:
10. Perform Quantum-Enhanced Global Search: Eq. (16)
11. Else:
12. Perform Quantum-Enhanced Local Search: Eq. (17)
13. For each solution $x_i$ in X:
14. Calculate the fitness function $F(x_i)$: Eq. (10)
15. Update current best solution g* if a better solution is found
16. Select the current best solution g* for Tabu Search
17. Initialize tabu list and iteration counter for TS
18. While not reaching TS maximum iterations or convergence:
19. Generate a new candidate solution: using Eq. (18)
20. Evaluate the fitness F $(x_{new})$: Eq. (10)
21.     If $(x_{new})$is not in the tabu list and F $(x_{new})$ < F $(x_{current})$:
22.         Update $x_{current} = x_{new}$
```

```
23.             Add $\mathrm{x_{new}}$ to the tabu list
            Else if $\mathrm{x_{new}}$ is in the tabu list but satisfies aspiration criterion:
              Update $\mathrm{x_{current}}$= $\mathrm{x_{new}}$
26.           Add $\mathrm{x_{new}}$ to the tabu list
27.         Update tabu list by removing the oldest entry if size exceeds the limit
28.   Update population X with refined solution from Tabu Search
29.   End Repeat
30.   Return the best solution g* and corresponding fitness value
```

4.5 Execution of the Proposed Adaptive QFPA-TS Algorithm

The execution of the Adaptive QFPA-TS algorithm in the edge-cloud continuum begins by initializing the algorithm parameters, which include defining the population size, setting parameters for the Quantum Behaved Flower Pollination Algorithm (QFPA), such as step size, switch probability and quantum potential field parameters, as well as configuring the TS parameters like the size of the tabu list and the maximum number of iterations for the local search. The initial population of solutions is generated using the Chaotic Circle Map (algorithm 1), which introduces chaotic behavior to ensure a diverse and well-distributed starting point for the optimization process. Each solution in this initial population represents a potential strategy for allocating tasks across the edge and cloud resources in the continuum. The fitness of each solution is evaluated using a multi-objective function (Eq. (10)) that considers both task execution delay and energy consumption.

The solution with the best fitness value is identified as the current best solution, which will serve as the focal point for further refinement and optimization. Once the initial population is evaluated, the algorithm adjusts the Quantum Potential Field dynamically to balance exploration and exploitation (step 7). This adjustment is based on the diversity of the current set of solutions. If the diversity is high, indicating a broad range of potential task allocation strategies, the quantum potential field parameter is adjusted to promote more exploration, allowing the algorithm to search for novel and potentially more efficient resource allocation scenarios. Conversely, if the diversity is low, indicating that the solutions are converging towards a particular area in the search space, the quantum potential field is adjusted to focus on exploitation, refining the current solutions to find the best possible allocation strategy. This dynamic adjustment mechanism enables the QFPA to adapt to the changing search landscape and maintain an effective balance between exploring new solutions and refining existing ones.

In the next phase, the QFPA performs both global and local searches (step 8-13) to explore different task allocation strategies. For each solution in the population, a random decision is made to determine whether to perform a global search or a local search. If the global search is chosen, the algorithm uses quantum-enhanced Lévy flights to explore distant regions of the search space (Eq. (16), simulating a broader search for potential solutions by making larger, stochastic jumps. This approach helps identify promising areas that may contain optimal or near-optimal solutions. If the local search is chosen, the algorithm performs a quantum-enhanced local search by making smaller, more focused adjustments to the solution (Eq. (17)), allowing it to exploit known good configurations to refine the task allocation strategy. Both global and local searches are guided by the quantum potential field (Eq. (15)), ensuring that the search process remains adaptive and responsive to the diversity of the solutions.

After performing the global and local searches, the fitness of the updated population is evaluated again using the multi-objective function (step 15). If a better solution is found, the current best solution is updated to reflect this improvement. At this stage, the algorithm moves into the TS phase for local refinement (step 17-28). TS begins by selecting the current best solution obtained from the

QFPA phase and focuses on intensively exploring its neighborhood by generating small perturbations around it (step 20). The new candidate solutions generated by these perturbations are evaluated based on the multi-objective fitness function, and if a candidate solution offers a better objective value than the current best solution and is not in the tabu list, it is accepted as the new current best solution. To ensure the search does not revisit recently explored solutions, a tabu list is maintained to record these solutions.

However, if a solution in the tabu list provides a significant improvement (satisfies the aspiration criterion), it is accepted regardless of its tabu status. This local search process continues until either the maximum number of iterations for TS is reached, or no further improvement is found. The refined solution obtained from TS is then used to update the population of solutions for the next iteration of the QFPA global and local search. The algorithm iteratively alternates between the global search phase of QFPA, where broad exploration is performed, and the local refinement phase of TS, where the focus is on intensifying the search around the most promising solutions. This iterative framework allows the algorithm to continuously improve the quality of solutions by adapting to the dynamic nature of the edge-cloud continuum environment.

5. Experiment and Results Analysis

In this section, we present the experimental setup and result analysis of our proposed algorithm, comparing its performance against Quantum Particle Swarm Optimization (QPSO) [10], Grey Wolf Optimizer (GWO) [30], and the Standard Flower Pollination Algorithm (FPA). The experiment is conducted using the Edge-CloudSim simulation environment, configured to simulate a realistic edge-cloud computing scenario with varying numbers of offloading tasks. We evaluate the algorithms based on key performance metrics, including delay, energy consumption, makespan and resource utilization.

5.1 Experiment

To conduct the experiment, the Edge-CloudSim simulation environment is used to evaluate the performance of our proposed algorithm in comparison with QPSO, GWO and the FPA. The focus of the evaluation is on several key metrics: delay, energy consumption, makespan and resource utilization. Delay will measure the total time taken from task generation to completion, while energy consumption will quantify the amount of energy used by both edge and cloud servers during task execution. Makespan will represent the total time required to complete all offloaded tasks. Resource utilization will indicate the percentage of computational resources used by the servers. The Edge-CloudSim environment is configured with specific settings to simulate a realistic edge-cloud computing scenario, ensuring that all algorithms are evaluated under identical conditions. The settings include the number of servers, their capacities, network bandwidths, and task parameters. This is presented in detail in Table 4. The simulation involves varying the number of offloading tasks and observing the behavior of each algorithm in response to these changes.

Table 4
Edge-Cloudsim environment settings

Parameter	Value
Number of Cloud servers	2
Cloud server CPU capacity	60-80 (uniform distribution)
Edge link bandwidth	100 Mbps
Cloud link bandwidth	200 Mbps

Number of edge servers	30
Edge server CPU capacity	40-60 (uniform distribution)
Number of mobile devices	100
Task CPU request	Oct-20
Task Data Size	10-20 MB
Task tolerable delay	10-15 ms
Number of Offloading tasks	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000

The parameter settings for the benchmark algorithms are adopted from the original benchmark papers to ensure a fair comparison. The parameter settings of each algorithm are presented in Table 5.

Table 5
Algorithms parameter settings

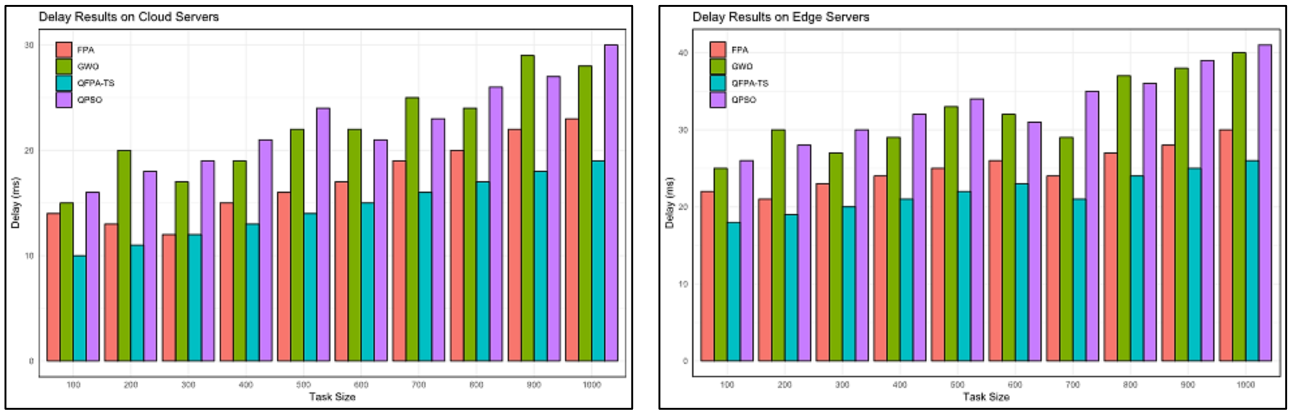
Algorithm	Parameter	Value
QPSO	Population size	30
	Maximum iterations	100
	Cognitive coefficient	1.5
	Social coefficient	1.5
	Inertia weight	0.9-0.4
GWO	Population size	30
	Maximum iterations	100
	Alpha (Î±)	0.5
	Beta (Î²)	0.3
FPA	Population size	30
	Maximum iterations	100
	Probability switch (p)	0.8
	Step size	0.1
QFPA-TS	Population size	30
	Maximum iterations	100
	Adaptive probability function	Based on solution diversity
	TS maximum iterations	50
	Tabu list size	10

5.2 Results Analysis

In this section, we present the results of our experiments, organized into subsections for each performance metric: delay, energy consumption, makespan and resource utilization. Each subsection provides a detailed analysis of the corresponding metric, comparing the performance of our proposed algorithm with the baseline algorithms QPSO, GWO and FPA. The results are discussed to highlight the strengths and weaknesses of each algorithm, emphasizing the effectiveness of our proposed approach in optimizing task management in the edge-cloud continuum under various conditions.

5.2.1 Delay analysis

The delay results of the Adaptive QFPA-TS algorithm, when compared to the baseline algorithms GWO, QPSO and the FPA demonstrate a significant improvement in task execution times in both edge and cloud environments, as shown in Figures 1(a) and (b). In the edge environment, the Adaptive QFPA-TS outperforms all baselines, primarily due to its dynamic balance between exploration and exploitation. The adaptive quantum potential field enables the algorithm to explore the solution space efficiently, avoiding premature convergence to suboptimal solutions. Additionally, the incorporation of TS intensively refines solutions, further reducing task delays, which gives the algorithm an edge over GWO and QPSO, which lack these adaptive and local search mechanisms.



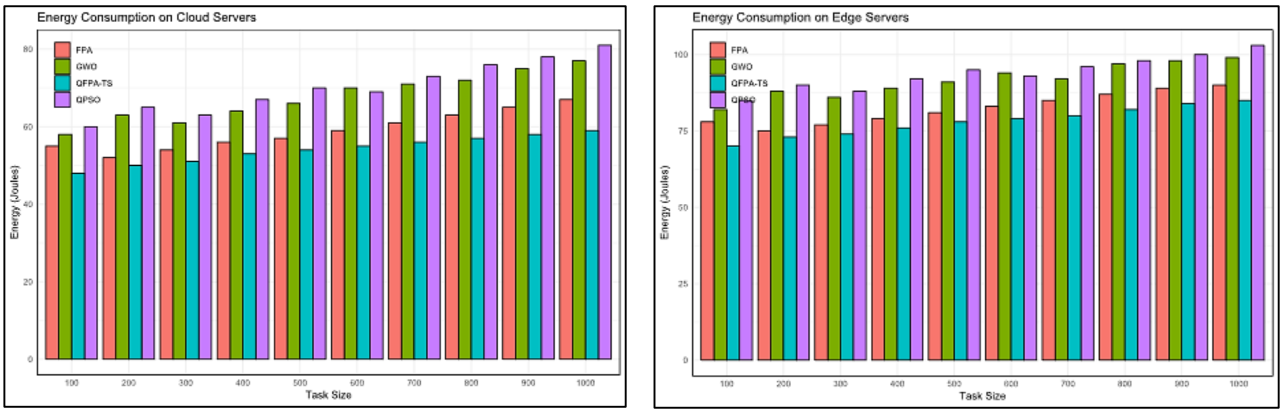
(a) (b)

Fig. 1. Task execution delay (a) Cloud servers (b) Edge servers

On cloud resources, the Adaptive QFPA-TS also shows better performance in minimizing task execution delays, although the margin of improvement over the baseline algorithms is less pronounced compared to the edge environment. This difference is due to the cloud's higher computational capacity, which naturally reduces delays. However, the Adaptive QFPA-TS maintains its advantage by effectively balancing task execution and data transmission times through its quantum-enhanced global search and TS-based local refinement. The GWO and QPSO perform well in the cloud but cannot match the Adaptive QFPA-TS's adaptability in dynamically allocating resources. The standard FPA, without enhancements, lags behind in both environments, especially on the edge, where it struggles to adapt to rapid changes in resource availability.

5.2.2 Energy consumption analysis

The energy consumption performance results of the Adaptive QFPA-TS algorithm also reveal notable differences in both edge and cloud environments compared to the base line algorithms. As illustrated in Figures 2. On the edge, the Adaptive QFPA-TS demonstrates a significant reduction in energy consumption compared to all baseline algorithms. This is primarily due to its ability to dynamically manage task allocation by considering both the execution cost and the energy overhead associated with each task. The quantum-inspired adaptive search of the QFPA allows it to explore various allocation scenarios efficiently, finding those that minimize the overall energy usage. Furthermore, the integration of Tabu Search helps refine these allocations by locally optimizing around the most promising solutions, which is particularly advantageous in the edge environment, where energy resources are limited.



(a) (b)

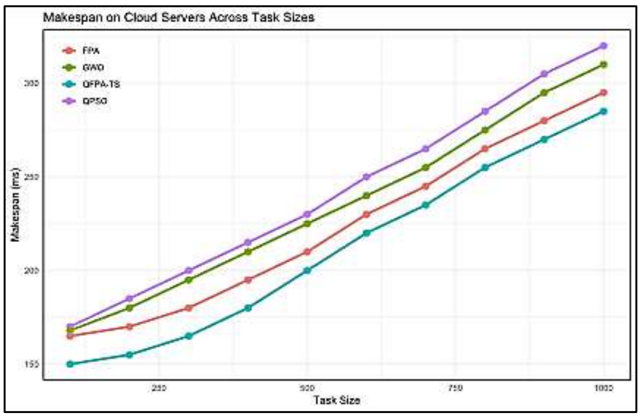
Fig. 2. Energy consumption (a) Cloud servers (b) Edge servers

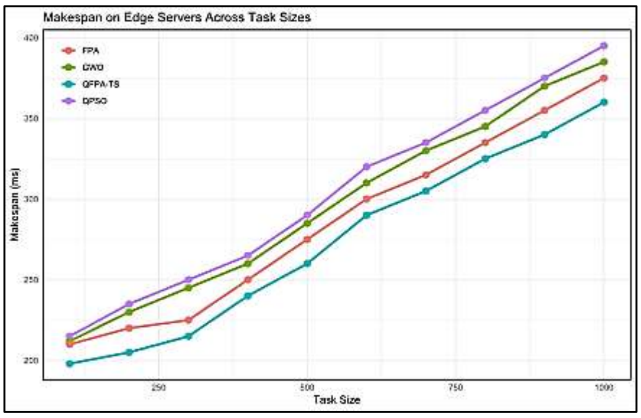
In the cloud environment, depicted in Figure 2(a), the Adaptive QFPA-TS also achieves lower energy consumption than GWO, QPSO, and FPA, but the margin of improvement is relatively smaller than in the edge environment. The cloud's greater computational capacity tends to offset energy savings, reducing the overall consumption benefits. However, the Adaptive QFPA-TS still maintains an edge by effectively balancing the computational load and minimizing the energy-intensive data transmission required for offloading tasks to cloud servers. GWO and QPSO, while demonstrating reasonable performance, do not achieve the same level of energy efficiency as the Adaptive QFPA-TS because they lack mechanisms to dynamically adjust allocation based on both execution cost and energy consumption. The standard FPA, without the benefits of quantum adaptation or local search refinement, also shows higher energy consumption in both environments compared to the Adaptive QFPA-TS, particularly on the edge.

5.2.3 Makespan analysis

Compared to the baseline algorithm, the makespan results for the Adaptive QFPA-TS algorithm demonstrate its effectiveness in optimizing task completion time across both edge and cloud environments, as shown in Figure 3. On the edge, the Adaptive QFPA-TS shows a substantial reduction in makespan compared to all baselines. This improvement stems from the algorithm's ability to dynamically manage task scheduling and resource allocation, balancing the load more efficiently across limited edge resources. The adaptive quantum potential field facilitates a diverse exploration of potential scheduling scenarios, while TS fine-tunes these strategies by intensively searching the local neighborhood of promising solutions, thereby achieving a more balanced distribution of tasks and minimizing the total time required to complete all tasks.

In the cloud environment, as illustrated in Figure 3(a), the Adaptive QFPA-TS also exhibits a notable decrease in makespan compared to GWO, QPSO and FPA, though the relative improvement is less dramatic than on the edge. The cloud's extensive computational resources naturally contribute to a lower makespan overall, but the Adaptive QFPA-TS maintains its advantage by optimizing the allocation of tasks in a way that reduces bottlenecks and improves parallel processing. GWO, while achieving reasonable makespan reductions, often converges prematurely due to its less dynamic search strategy, resulting in less optimal task allocations. QPSO performs better in some cases but lacks the refinement mechanisms needed to consistently minimize makespan across different load levels. The standard FPA, without the enhancements of quantum-inspired adaptation or TS refinement, shows the higher makespan values against QFPA-TS in both environments, particularly on the edge, where the lack of dynamic adjustment leads to inefficient task distribution.



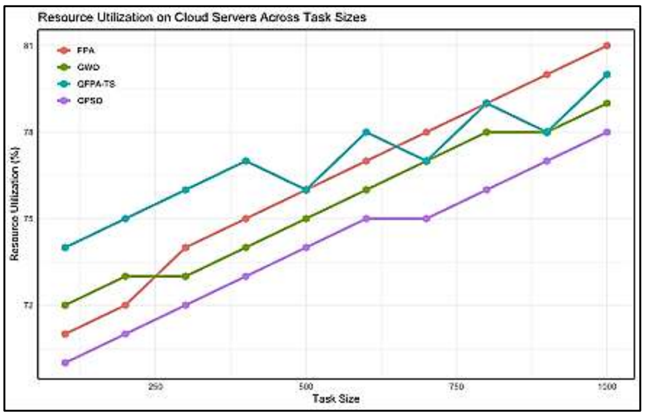


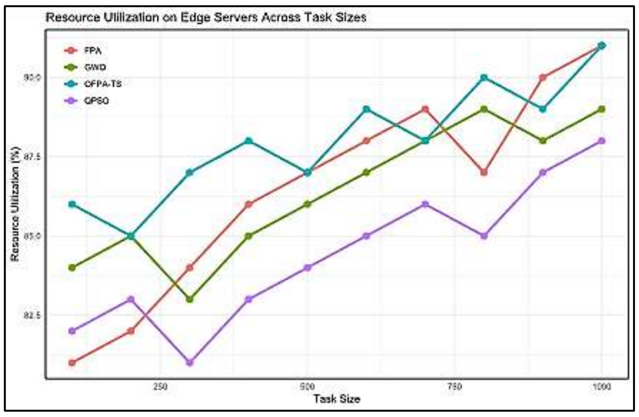
(a) (b)

Fig. 3. Makespan time (a) Cloud servers (b) Edge servers

5.2.4 Resource utilization

The resource utilization results also illustrate performance of the QFPA-TS in effectively managing computational resources in both edge and cloud environments, as presented in Figure 4. In the edge environment, the Adaptive QFPA-TS achieves higher and more balanced resource utilization compared to the baselines. This advantage stems from the algorithm's ability to dynamically allocate tasks across the available edge resources, leveraging both global exploration through the adaptive quantum potential field and local refinement *via* TS. The combination of these mechanisms allows for an optimized distribution of workload, reducing idle time and ensuring that all available resources are effectively used, which is crucial in environments with limited capacity like the edge.





(a) (b)

Fig. 4. Resource utilization (a) Cloud servers (b) Edge servers

In the cloud environment, as shown in Figure 4(a), the Adaptive QFPA-TS also demonstrates a higher resource utilization than GWO, QPSO, and FPA, although the relative improvement is somewhat less noticeable compared to the edge. The cloud's abundant computational resources generally lead to higher baseline utilization levels; however, the Adaptive QFPA-TS maintains an edge by preventing resource underutilization and ensuring that tasks are distributed in a way that maximizes the use of available processing power. GWO tends to achieve suboptimal utilization due to its static hierarchy-based approach, which can lead to uneven task distribution. QPSO, while better than GWO, lacks the adaptive refinement capabilities of TS, leading to occasional resource bottlenecks. The standard FPA also struggled in balancing the resources.

5.2.5 Percentage improvement

The percentage improvement of an algorithm refers to the relative increase in performance (in this case, delay, resource utilization and energy consumption) between the baseline algorithm (QPSO, GWO and FPA) and the enhanced version (QFPA-TS). It helps in quantifying how much better the improved algorithm is compared to the previous one. This percentage improvement is calculated using Eq. (20) and the percentage improvement is presented in Table 6.

$$\text{Improvement\%} = \frac{\text{Baseline} - \text{Proposed}}{\text{Baseline}} \times 100 \tag{20}$$

Table 6
Percentage improvement

Metric	Against QPSO (%)	Against GWO (%)	Against FPA (%)
Delay (ms)	33.75371	31.1569	12.44463
Energy consumption (J)	16.92538	14.74175	5.194264
Makespan (ms)	9.942674	8.060856	4.414982

The percentage improvement results demonstrate that QFPA-TS consistently outperforms QPSO, GWO, and FPA in reducing delay, energy consumption, and makespan across all task sizes. QFPA-TS achieves the highest improvements in delay, with an average reduction of 33.75% over QPSO, 31.16% over GWO, and 12.44% over FPA, showcasing its robust handling of latency-sensitive tasks. The energy consumption results further highlight the efficiency of QFPA-TS, achieving reductions of 16.93%, 14.74%, and 5.19% compared to QPSO, GWO, and FPA, respectively. These improvements underscore its suitability for energy-constrained environments. While the improvements in makespan are slightly lower, with reductions of 9.94% over QPSO, 8.06% over GWO, and 4.41% over FPA, they are still significant and demonstrate QFPA-TS's ability to maintain efficient task execution. Overall, the results emphasize the effectiveness of QFPA-TS in optimizing task scheduling for edge-cloud environments by reducing latency and energy usage while ensuring timely task completion, making it a reliable solution for such systems.

6. Conclusions

This research presents a novel hybrid optimization approach, the Adaptive Quantum Behaved Flower Pollination Algorithm with Tabu Search (QFPA-TS), designed to optimize task management in the edge-cloud continuum by minimizing task execution delay, energy consumption, makespan, and maximizing resource utilization. The integration of quantum-inspired global search mechanisms with adaptive adjustments, alongside a robust local search refinement using TS, allows the proposed algorithm to dynamically balance exploration and exploitation, thereby overcoming the limitations of traditional optimization methods like GWO, QPSO and standard FPA.

The results of this study demonstrate that the Adaptive QFPA-TS algorithm consistently outperforms these baseline algorithms across multiple performance metrics in both edge and cloud environments. In the edge environment, where resources are limited and the need for rapid adaptation is critical, the Adaptive QFPA-TS shows significant improvements in reducing task execution delays, lowering energy consumption, minimizing makespan and maximizing resource utilization. The algorithm's ability to adjust dynamically to varying conditions enables it to maintain optimal performance in scenarios characterized by high variability and constrained resources. In the cloud environment, while the relative performance gains are less noticeable due to the inherent capacity and resource availability, the Adaptive QFPA-TS still provides a distinct advantage by ensuring efficient task allocation and minimizing energy-intensive data transmissions. Hence, the Adaptive QFPA-TS proves to be a robust and versatile solution for managing tasks in the edge-cloud continuum, demonstrating substantial improvements over existing methods. Its unique combination of quantum-inspired global search, adaptive parameter adjustments, and local refinement capabilities offers a powerful framework for multi-objective optimization in dynamic, resource-constrained environments. This approach not only enhances the efficiency and effectiveness of resource management but also paves the way for future research into adaptive optimization techniques in emerging computing paradigms. The findings suggest that the proposed hybrid algorithm can be a valuable tool for optimizing complex distributed computing environments, supporting the development of more efficient and sustainable edge-cloud systems.

Acknowledgement
This research was not funded by any grant.

References

[1] Dankolo, Nasiru Muhammad Dankolo, Nor Haizan Mohamed Radzi, Noorfa Haszlinna Mustaffa, Mohd Shukor Talib, Zuriahati Mohd Yunos, and Danlami Gabi. "Efficient task scheduling approach in edge-cloud continuum based on flower pollination and improved shuffled frog leaping algorithm." *Baghdad Science Journal* 21, no. 2 (SI) (2024): 0740-0740. https://doi.org/10.21123/bsj.2024.10084

[2] Fuaad, Mohamad Adrian Mohd, Qairel Qayyum Muhamad Ridhuan, Wan Muhammad Alif Firdaus Wan Hanapi, and Shelena Soosay Nathan. "LifeGuardian; A smart bracelet for visually impaired elderly." *Journal of Advanced Research in Computing and Applications* 36, no. 1 (2024): 20-28. https://doi.org/10.37934/arca.36.1.2028

[3] Ali, Tasnuva, Azni Haslizan Ab Halim, and Nur Hafiza Zakaria. "3D lightweight cryptosystem design for IoT applications based on composite S-box." *International Journal of Computational Thinking and Data Science* 3, no. 1 (2024): 40-54. https://doi.org/10.37934/ctds.3.1.4054

[4] Shafiq, Dalia Abdulkareem, N. Z. Jhanjhi, and Azween Abdullah. "Load balancing techniques in cloud computing environment: A review." *Journal of King Saud University-Computer and Information Sciences* 34, no. 7 (2022): 3910-3933. https://doi.org/10.1016/j.jksuci.2021.02.007

[5] Al Shamaa, Saleh, Nabil Harrabida, Wei Shi, and Marc St-Hilaire. "Particle swarm optimization with enhanced neighborhood search for task scheduling in cloud computing." In *2022 IEEE Cloud Summit*, p. 31-37. IEEE, 2022. https://doi.org/10.1109/CloudSummit54781.2022.00011

[6] Gabi, Danlami, Abdul Samad Ismail, Anazida Zainal, Zalmiyah Zakaria, Ajith Abraham, and Nasiru Muhammed Dankolo. "Cloud customers service selection scheme based on improved conventional cat swarm optimization." *Neural Computing and Applications* 32 (2020): 14817-14838. https://doi.org/10.1007/s00521-020-04834-6

[7] Saif, Faten A., Rohaya Latip, Zurina Mohd Hanapi, and Kamarudin Shafinah. "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing." *IEEE Access* 11 (2023): 20635-20646. https://doi.org/10.1109/ACCESS.2023.3241240

[8] Abdullahi, Mohammed, Md Asri Ngadi, Salihu Idi Dishing, and Shafi'I. Muhammad Abdulhamid. "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing." *Journal of ambient intelligence and humanized computing* 14, no. 7 (2023): 8839-8850. https://doi.org/10.1007/s12652-021-03632-9

[9] Usman, Mohammed Joda, Lubna A. Gabralla, Ahmed Aliyu, Danlami Gabi, and Haruna Chiroma. "Multi-objective hybrid flower pollination resource consolidation scheme for large cloud data centres." *Applied Sciences* 12, no. 17 (2022): 8516. https://doi.org/10.3390/app12178516

[10] Dong, Shi, Yuanjun Xia, and Joarder Kamruzzaman. "Quantum particle swarm optimization for task offloading in mobile edge computing." *IEEE Transactions on Industrial Informatics* 19, no. 8 (2022): 9113-9122. https://doi.org/10.1109/TII.2022.3225313

[11] Wang, Shudong, Yanqing Li, Shanchen Pang, Qinghua Lu, Shuyu Wang, and Jianli Zhao. "A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline." *Scientific Programming* 2020, no. 1 (2020): 3967847. https://doi.org/10.1155/2020/3967847

[12] Guevara, Judy C., and Nelson LS Da Fonseca. "Task scheduling in cloud-fog computing systems." *Peer-to-Peer Networking and Applications* 14, no. 2 (2021): 962-977. https://doi.org/10.1007/s12083-020-01051-9

[13] Hamed, Ahmed Y., M. Kh Elnahary, Faisal S. Alsubaei, and Hamdy H. El-Sayed. "Optimization Task Scheduling Using Cooperation Search Algorithm for Heterogeneous Cloud Computing Systems." *Computers, Materials & Continua* 74, no. 1 (2023). http://dx.doi.org/10.32604/cmc.2022.032215

[14] Najafizadeh, Abbas, Afshin Salajegheh, Amir Masoud Rahmani, and Amir Sahafi. "Multi-objective task scheduling in cloud-fog computing using goal programming approach." *Cluster Computing* 25, no. 1 (2022): 141-165. https://doi.org/10.1007/s10586-021-03371-8

[15] Mohammadzadeh, Ali, Mohammad Masdari, and Farhad Soleimanian Gharehchopogh. "Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm." *Journal of Network and Systems Management* 29, no. 3 (2021): 31. https://doi.org/10.1007/s10922-021-09599-4

[16] Too, Jingwei, Abdul Rahim Abdullah, and Norhashimah Mohd Saad. "A new quadratic binary harris hawk optimization for feature selection." *Electronics* 8, no. 10 (2019): 1130. https://doi.org/10.3390/electronics8101130

[17] Karaja, Mouna, Abir Chaabani, Ameni Azzouz, and Lamjed Ben Said. "Efficient bi-level multi objective approach for budget-constrained dynamic bag-of-tasks scheduling problem in heterogeneous multi-cloud environment." *Applied Intelligence* 53, no. 8 (2023): 9009-9037. https://doi.org/10.1007/s10489-022-03942-1

[18] Gabi, Danlami, Nasiru Muhammad Dankolo, Abubakar Atiku Muslim, Ajith Abraham, Muhammad Usman Joda, Anazida Zainal, and Zalmiyah Zakaria. "Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme." *Neural Computing and Applications* 34, no. 16 (2022): 14085-14105. https://doi.org/10.1007/s00521-022-07260-y

[19] Movahedi, Zahra, Bruno Defude, and Amir Mohammad Hosseininia. "An efficient population-based multi-objective task scheduling approach in fog computing systems." *Journal of Cloud Computing* 10, no. 1 (2021): 53. https://doi.org/10.1186/s13677-021-00264-4

[20] Zhang, Yu, Bing Tang, Jincheng Luo, and Jiaming Zhang. "Deadline-aware dynamic task scheduling in edge–cloud collaborative computing." *Electronics* 11, no. 15 (2022): 2464. https://doi.org/10.3390/electronics11152464

[21] Li, Chunlin, Chengyi Wang, and Youlong Luo. "An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment." *The Journal of Supercomputing* 76 (2020): 6941-6968. https://doi.org/10.1007/s11227-019-03133-9

[22] Wang, Ying, Zhile Yang, Yuanjun Guo, Bowen Zhou, and Xiaodong Zhu. "A novel binary competitive swarm optimizer for power system unit commitment." *Applied Sciences* 9, no. 9 (2019): 1776. https://doi.org/10.3390/app9091776

[23] Zhang, Miao, Peng Jiao, Yong Peng, and Quanjun Yin. "Efficient Dynamic Deployment of Simulation Tasks in Collaborative Cloud and Edge Environments." *Applied Sciences* 12, no. 3 (2022): 1646. https://doi.org/10.3390/app12031646

[24] Patra, Manoj Kumar, Sanjay Misra, Bibhudatta Sahoo, and Ashok Kumar Turuk. "GWO-based simulated annealing approach for load balancing in cloud for hosting container as a service." *Applied Sciences* 12, no. 21 (2022): 11115. https://doi.org/10.3390/app122111115

[25] Fu, Kaihua, Wei Zhang, Quan Chen, Deze Zeng, and Minyi Guo. "Adaptive resource efficient microservice deployment in cloud-edge continuum." *IEEE Transactions on Parallel and Distributed Systems* 33, no. 8 (2021): 1825-1840. https://doi.org/10.1109/TPDS.2021.3128037

[26] Kaur, Harpreet, and Munish Rattan. "Improved offline multi-objective routing and wavelength assignment in optical networks." *Frontiers of Optoelectronics* 12 (2019): 433-444. https://doi.org/10.1007/s12200-019-0850-4

[27] Yang, Xin-She. "Flower pollination algorithm for global optimization." In *International Conference on Unconventional Computing and Natural Computation*, p. 240-249. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-32894-7_27

[28] Gölcük, İlker, and Fehmi Burcin Ozsoydan. "Fuzzy flower pollination algorithm with chaos for global optimization." In *International Online Conference on Intelligent Decision Science*, p. 406-414. Cham: Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-66501-2_33

[29] Umam, Moch Saiful, Mustafid, and Suryono. "A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem." *Journal of King Saud University-Computer and Information Sciences* 34, no. 9 (2022): 7459-7467. https://doi.org/10.1016/j.jksuci.2021.08.025

[30] Li, Xiaobo, Qiyong Fu, Qi Li, Weiping Ding, Feilong Lin, and Zhonglong Zheng. "Multi-objective binary grey wolf optimization for feature selection based on guided mutation strategy." *Applied Soft Computing* 145 (2023): 110558. https://doi.org/10.1016/j.asoc.2023.110558