Discover Computing

RESEARCH Open Access

Weighted quantum particle swarm task offloading optimization algorithm for time-energy minimization in mobile edge computing

Jafar Aminu[1*], Rohaya Latip[1*], Zurina Mohd Hanapi[1], Shafinah Kamarudin[1], Danlami Gabi[2] and Muhammad Anas Shehu[1,3]

*Correspondence:
Jafar Aminu
gs65795@student.upm.edu.my
Rohaya Latip
rohayalt@upm.edu.my
[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43300 Selango, Malaysia
[2]Faculty of Computing, Universiti of Teknologi Malaysia, Johor Bahru 81310, Malaysia
[3]Department of Computer Science, Kebbi State University of Science and Technology Aleiro, Aleiro 1144, Nigeria

Abstract
Mobile Edge Computing (MEC) is changing the computing paradigm by bringing processing resources and shifting latency to mobile network edge, which is crucial for demanding environments like IoT, augmented reality, and autonomous systems. Although progress has been made in the existing literature to tackle these challenges, many approaches fall short in optimally selecting tasks and accounting for their dependencies, which are essential for efficient offloading, leading to suboptimal energy consumption and task completion time. This work proposes Weighted quantum Particle Swarm Optimization (WQPSO), a new multi-objective algorithm for MEC task offloading, as a response to this issue. At its core, WQPSO aims to optimize energy consumption and task completion time, using an effective approach that doesn't require extensive parameter tuning. It also provides a nearly stringent scalable framework for high-demand multi-task, multiuser, and multi-server environments. We strictly compare the Python implementation of the WQPSO algorithm with a set of state-of-the-art approaches. The findings demonstrate that WQPSO delivers an average reduction of 5.16% in task completion time and 8.66% in overall system energy consumption. These results highlight its strong potential as a highly effective solution to address the challenges in edge computing.

Keywords Edge computing, Energy consumption, Completion time task offloading, Weighted quantum particle swarm

1 Introduction

MEC has emerged as one among the key technologies in cloud computing, extending computational capabilities towards the edge of a network [1–3]. It allows for data processing closer to end-users, increasing latency by a higher magnitude, supporting real-time processing and improving data security; hence, it is quite critical for applications like augmented reality, autonomous vehicles, and IoT [4–6]. Yet, MEC also introduces significant challenges in managing computing resources efficiently and minimizing energy consumption at the edge side [7, 8]. These issues further scale up when dealing

with dynamic and heterogeneous network environments, where the ability of real-time decision-making and resource optimization becomes crucial [9, 10]. The most basic problem in MEC is about task offloading, involving the offloading decision on whether to execute computation tasks locally on mobile devices or offload them onto the edge servers [11, 12]. It is a decision-making process guided by fluctuating characteristics of mobile networks, with different computational demands and two objectives for optimization: one concerning energy consumption and the other related to completing tasks within the minimum possible time [13–15].

In edge computing environments, achieving efficient task offloading remains a challenging problem, particularly given resource constraints and stringent latency demands [16]. Task offloading involves intricate decision-making processes that must consider factors such as application requirements, network conditions, energy consumption, and available resources [17, 18]. Various strategies have been proposed in the literature to tackle the task offloading challenge in edge computing, spanning heuristic approaches to metaheuristic optimization methods [19–21]. Many current algorithms, both heuristic and metaheuristic, lack the adaptability required to effectively respond to the constantly changing conditions of edge computing environments [22–24]. These algorithms often encounter difficulties in managing fluctuating task requirements, variable network conditions, and resource constraints, which can lead to suboptimal performance and delays in task execution.

Another key challenge in edge computing is the management of task dependencies, particularly when tasks must follow a specific sequence rather than being executed independently [25–27]. Existing models often lack the necessary mechanisms to handle these dependencies efficiently, resulting in potential issues such as deadlocks, higher energy consumption, and extended task completion times. Additionally, several popular algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Gray wolf Optimization (GWOA), Simulated Annealing (SA), and Ant Colony A (ACO), often struggle to maintain an effective balance between exploration and exploitation. This imbalance can cause these algorithms to converge prematurely to suboptimal solutions or to engage in excessive exploration without sufficiently refining the most promising solutions. Furthermore, a number of studies have investigated reinforcement learning methods for making task offloading decisions in edge environments [27–29]. However, these approaches face significant challenges, such as high computational cost, slow convergence, and difficulties in adapting to dynamic, large-scale environments.

This paper proposes a novel Weighted Quantum Particle Swarm Task Offloading Optimization Algorithm, namely WQPSO, which overcomes the challenges by introducing several key innovations with the aim of enhancing the performance in MEC. In this regard, the developed WQPSO method embraces quantum-behaved PSO techniques, enhancing global search capability to avoid prematurity in convergence and minimizing the likelihood of being stuck in a local optimum. This makes for an adaptive approach that might be useful in exploring the search space more intensively and, hence, increases the likelihood of finding the optimal solution [30]. WQPSO introduces a weighted mean best position strategy that, through the influence of the dynamically adjusted particles in relation to the fitness levels, encourages faster convergence with the preservation of diversity in the search process. These two important parameters exist within this weighted mean best position strategy: the weight coefficient and the

contraction-expansion factor. The weight coefficient guides every particle's influence from personal best position relative to the global best position. The fitness-based weight update rule assigns larger weights to better-performing particles, making those particles with higher fitness play a more vital role in guiding the search process. It dynamically adjusts the balance of exploration and exploitation to enable WQPSO to adapt to different optimization scenarios [31]. For example, assigning moderate weights to a larger number of particles in early iterations encourages higher diversity and exploration, while increasing the weight of top-performing particles in later iterations allows for the exploitation of the best-known solutions. The factor of contraction-expansion controls the balance between global exploration and local exploitation during the different steps of the optimization process [32, 33]. The larger factor encourages wider exploration of the search space in order to avoid its premature convergence. Later, when the algorithm converges, this factor is gradually reduced to fine-tune the search around promising areas for more focused exploitation. This runtime dynamic tuning empowers the algorithm to navigate efficiently through complex search spaces typical of MEC environments [34, 35]. WQPSO gains faster convergence by incorporating adaptive parameters, keeping the optimization approach flexible and hence robust across the paradigm of task offloading. Scalability further gets enhanced with the use of dispatched mechanisms for task offloading, which would be capable of dynamic task allocation based on computation requirements and interdependencies of the tasks with lesser energy consumption and task completion time. All these robustness's and scalabilities ensure that WQPSO always performs consistently under different scenarios and various workloads for complex MEC applications. In this regard, the WQPSO algorithm effectively resolves the major limitations of the traditional task offloading strategy by demonstrating better convergence and scalability with more efficient handling of task dependencies [36]. This paper discusses the WQPSO algorithm, its implementation, and a performance comparison to traditional approaches.

The proposed Weighted Quantum Particle Swarm Optimization (WQPSO) algorithm is developed as an extension of the Quantum Particle Swarm Optimization (QPSO) framework. Rather than introducing a fundamentally new swarm intelligence paradigm, this work focuses on the system-oriented integration of weighted quantum swarm dynamics within a mobile edge computing (MEC) task offloading context, where energy consumption, completion, time and task dependencies, be jointly addressed. Weighted mean-best guidance and elitist learning strategies have been explored in existing PSO and QPSO variants. However, the key contribution of WQPSO lies not in isolated parameter tuning, but in how fitness-based weighting and adaptive contraction–expansion control are embedded into a dependency-aware MEC optimization framework. The weighting mechanism is directly coupled with the offloading decision model and the joint energy and completion objective, enabling the optimizer to emphasize solutions that are simultaneously feasible, energy-efficient, and latency-sensitive. Unlike generic weighted QPSO approaches that primarily aim to accelerate convergence on unconstrained benchmark problems, WQPSO adapts its quantum control parameters in response to dynamic MEC system conditions, including varying user densities, heterogeneous MEC server capacities, and task dependency constraints. This adaptive behavior enhances solution stability under high load and bursty traffic scenarios, which are common in practical MEC deployments. Accordingly, the contribution of this work is

best understood as a domain-specific enhancement of WQPSO for realistic MEC task offloading, demonstrating how weighted quantum swarm mechanisms can be systematically integrated with system-level constraints to achieve scalable and robust performance. This distinguishes the proposed approach from existing elitist PSO/QPSO variants that are typically evaluated without explicit consideration of dependency-aware scheduling and hardware-constrained MEC environments.

This paper aims to enhance energy optimization techniques for mobile edge computing by addressing key challenges previously identified. The main contributions of this study are as follows:

1 Discussing task offloading related work and issues concerning energy consumption and task completion time objectives function.
2 Proposing a Weighted Quantum Particle Swarm Optimization (WQPSO) task offloading optimization algorithm to reduce completion time and energy consumption in edge computing.
3 Implementing and analysing the WQPSO algorithm through experimentation and comparing the results with other benchmarks.
4 Experimental results confirm that our WQPSO-driven task offloading approach consistently surpasses state-of-the-art methods across critical performance metrics, including energy consumption, task completion time, and computational runtime time.

The structure of the paper is as follows: Sect. 2 provides a review of relevant literature, while Sect. 3 outlines the problem formulation. The proposed methodology is detailed in Sect. 4. Section 5 explains the experimental configuration, and Sect. 6 discusses the results obtained. Finally, Sect. 8 concludes the study and offers directions for future research.

2 Review of related works

Edge computing has emerged as a promising approach to complement cloud computing by improving resource management and enhancing performance [37]. However, the inherently dynamic and varied nature of edge environments creates significant challenges, particularly in areas such as task offloading, network performance monitoring, and the handling of task dependencies [38]. Task offloading within mobile edge computing has become a critical concern, attracting considerable research interest. A range of metaheuristic algorithms has been investigated to tackle this issue, with each method offering distinct advantages and drawbacks. These algorithms play an essential role in optimizing task management decisions, whether on mobile devices or MEC servers, across various computing settings, due to their capacity to find near-optimal solutions in complex and evolving environments [14].

Metaheuristic algorithms have proven to be highly effective in tackling the complex challenges of task offloading in edge computing environments. These techniques are particularly well-suited for solving NP-hard problems by employing stochastic strategies to efficiently navigate the solution space through both exploration and exploitation. Among them, multi-objective metaheuristic algorithms have garnered considerable interest for their ability to simultaneously optimize multiple, often conflicting, objectives such as energy consumption, task completion time, and execution efficiency. By combining

different metaheuristic approaches, these hybrid algorithms enhance both the diversity of potential solutions and the speed of convergence, overcoming the limitations typically associated with single-method strategies. Recent advancements in this area have introduced adaptive components such as reinforcement learning, chaotic maps, and fuzzy logic enabling dynamic parameter tuning and real-time responsiveness [39]. These innovations have paved the way for more intelligent and flexible resource scheduling and task allocation frameworks, where the critical balance between exploration and exploitation is maintained to ensure optimal performance in the highly dynamic edge computing environments.

Due to its ability to efficiently identify optimal solutions for complex problems, the Particle Swarm Optimization (PSO) algorithm has been widely applied to task offloading in edge computing environment [40]. Over time, the PSO algorithm has undergone several modifications and improvements in various studies to enhance task offloading performance. By mimicking the social behaviour of animals like fish or birds, these adaptations address the dynamic and flexible nature of edge computing, ensuring cost-effective and energy-efficient solutions while still meeting Quality of Service (QoS) requirements. Recently there some research like [41] introduce a clever metaheuristic optimization technique to solve the issue of poor service quality brought on by movable vehicles and sparse edge coverage. The suggested model is then utilized to describe the overall delay of vehicle task offloading by considering edge server workload, resource utilization, and vehicle movement characteristics. This allows for an evaluation of the model's performance in these situations. The study [16] suggests using particle swarm optimization as a workload offloading technique. By considering a few variables, including capacity, proximity, and latency needs, the method seeks to maximize energy consumption and satisfy latency limitations in EC environments. In order to minimize energy consumption in the IoT environment and meet task deadline limitations, the suggested model will be used to determine the best method of task offloading for IoT devices [42]. To transfer tasks from resource-constrained edge devices to edge servers in an energy-efficient and low-delay manner, a PSO-based task offloading technique is suggested. A suggested multi-objective optimization problem takes task execution cost, energy usage, and time delay into account [43]. PSO is highly effective for continuous optimization problems due to its simplicity and fast convergence rate. However, it can sometimes suffer from premature convergence, leading to suboptimal solutions by getting trapped in local optima. As a result, further improvements are necessary to enhance its performance.

Genetic algorithms (GAs) have proven to be effective in tackling the challenging task offloading problem in a variety of computing environments, such as cloud computing, mobile edge computing (MEC), fog computing, and cloud systems [44]. These algorithms are particularly advantageous due to their ability to produce high-quality solutions within a reasonable amount of time, even for NP-complete problems like task offloading [45]. To accommodate the dynamic and adaptable nature of edge services, requirements for data transmission. They used the NSGA-II and the Bees algorithm, two meta-heuristic approaches, to tackle the problem because it was NP-hard [46]. Recently there is research, proposed a resilient task offloading technique for mobile edge environments by merging an evolutionary algorithm, namely the genetic algorithm (GA), with queuing networks. The waiting and service times for mobile tasks on edge servers are

modelled by the queuing network. By taking into account transmission times and server load conditions, which are represented by the waiting and service times obtained from the queuing network, the evolutionary algorithm optimizes the distribution of jobs to edge servers with the goal of minimizing response times [47]. However, the GA need more improvement in terms of convergence.

The Grey Wolf Optimizer (GWO) algorithm, inspired by the social hierarchy and hunting strategies of grey wolves, has been adapted and enhanced for edge computing applications. This adaptation aims to reduce task completion time, improve energy efficiency, and enhance overall Quality of Service (QoS). In the proposed framework, a dynamic edge server selection strategy for managing inter-dependent tasks across edge nodes is employed, leveraging information from multiple connected edge servers [48]. The study explores both the task offloading probability and the power allocation strategy using GWO's behavioural principles. The algorithm has shown flexibility and effectiveness across various domains after being customized for edge environments. The task offloading problem is addressed through a GWO-based approach that focuses on optimizing power distribution and task scheduling to minimize execution time on energy-constrained mobile devices [49]. Furthermore, a multi-objective optimization model is developed to simultaneously reduce resource usage costs, energy consumption, and latency. The task offloading challenge addressed is recognized as NP-hard, highlighting the complexity and importance of efficient solution strategies [50]. Similarly, modifications to the GWO algorithm have led to improvements in processing speed by addressing issues related to energy consumption and load variability, making it more effective than other algorithms in edge computing environments where tsk offloading presents a challenging NP-Hard problem. However, in dynamic systems lacking adaptive mechanisms, GWO may face challenges.

The Ant Colony (ACO) algorithm has proven effective in addressing task offloading challenges across various fields, including edge computing and construction project management. Inspired by the Ants leave pheromone trails on the ground that direct other ants to follow [51]. Ants utilize these pheromone values to navigate the solution space, noting their positions and the calibre of their solutions in an effort to choose the best one. For resource and task selection criteria in clusters, the ACO technique was proposed in, ultimately identifying optimal solutions for complex optimization problems [52]. To extend the operational time of aided mobile servers, hence improving the overall system's quality of service over the long run. The problem of minimizing system delays is formulated as a mixed-integer programming (MIP) problem. Our dynamic energy criticality avoidance-based delay minimization ant colony algorithm (EACO) aims to strike a balance between operational time maximization for mobile servers and delay reduction for offloaded workloads because this challenge is NP-hard [53]. The study explored the challenge of reducing application execution time in automotive networks, considering energy and connectivity limitations through task offloading. Given the NP-hard nature of the problem, finding the optimal solution could potentially compromise the timely execution of applications [54]. paper presents a metaheuristic approach using the nature-inspired Artificial Bee Colony (ABC) optimization technique, which efficiently manages workload distribution across edge servers while adhering to strict constraints such as low network latency and fast response times [55]. Due to the balance between exploration and exploitation, it has demonstrated encouraging outcomes in several

optimization situations. To evaluate its performance in dynamic situations like edge computing, more research is necessary as its applicability in these settings is yet mostly unexplored.

In the area of task offloading for mobile edge computing (MEC), classical optimization and artificial ontologies have been dominant over prior research [56]. Although these techniques are generally effective, they usually involve expensive computations resulting in high resource demands and energy utilization, which is a serious challenge for the efficient operation of MEC as it imposes strict requirements on the energy consumption and completion time. To address these challenges, researchers have tweaked and utilized well-known meta-heuristic techniques like (PSO, GA, GWO, SA, and ACO) Such methods yield high-quality solutions with less complexity than classical approaches, thus allowing for energy-efficient operation. The approaches on the other hand, rely on multiple parameters to achieve optimal results, which can lead to increased energy consumption and computational time. Therefore, these methods may fail to meet the speed and power efficiency requirements for task processing in MEC applications, particularly in the context of Internet of Things (IoT)-based applications.

According to [57] the research introduces optimization algorithms for task offloading, such as PSO and QPSO. Comprehensive simulation results demonstrate that the proposed approach significantly lowers system energy consumption and task completion time, surpassing current state-of-the-art methods. However, the study did not account for task dependencies, which represents a limitation in the proposed framework. In [41], metaheuristic optimization strategy was introduced to enhance service quality by tackling issues related to vehicle mobility and limited edge network coverage. The approach evaluates the total latency involved in offloading tasks from vehicles by accounting for critical elements such as available computing resources, the current load on edge servers, and the mobility patterns of the vehicles. By optimizing task allocation and offloading strategies, this approach enhances computational efficiency and reduces network delays in dynamic edge environments. To optimize offloading decisions, the proposed PSO-based metaheuristic has been adapted to efficiently utilize modern GPU architectures, enhancing the search for optimal offloading opportunities. However, the study fails to account for the relationships and interdependencies among tasks submitted by users. As mention in [58] a PSO approach is proposed for optimizing task offloading in edge computing micro clusters. It enhances system performance by minimizing makespan while meeting resource constraints and task deadlines. Benchmark tests show it outperforms existing methods with faster computation and better results. However, the proposed method is its lack of prioritization for energy consumption as a key Quality of Service (QoS) parameter, potentially overlooking its critical role in optimizing edge computing environments. In this study [59] introduces algorithms designed to offload tasks from multiple user devices to multi-core edge servers, focusing on reducing energy consumption while adhering to task deadlines. The proposed heuristics initiate task processing using sequences generated by the established long and short task prioritization rules. The results demonstrate that these algorithms deliver superior performance compared to baseline methods. However, the study should include resource utilization to increase system performance, as it is quite complicated. In [60] this approach seeks to minimize task completion time and lower energy usage for mobile devices in Edge Computing (EC) systems with limited server resources. It addresses the challenge

of efficiently offloading computational tasks from mobile devices to EC servers while accounting for data encryption needs, which can raise both processing time and energy demands. However, a limitation of this method is its inability to adapt to dynamic environmental changes or accurately evaluate security risks, which may affect the system's reliability and resilience. In [61] this study focuses on optimization challenges in Edge Computing (EC) systems, particularly in resource allocation for offloading tasks from mobile devices to edge servers. The proposed method aims to minimize computation time, reduce service costs, and improve the efficient utilization of edge server resources, ensuring optimal workload distribution and enhanced system performance in dynamic computing environments. Additionally, it aims to improve task association with edge servers to better accommodate the mobility of mobile devices. However, the study prioritized high-computation resources, resulting in increased energy consumption, and failed to account for the dynamic nature of the environment during task offloading [62]. this method aims to enhance the global search capabilities of traditional intelligent algorithms, mitigate premature convergence, and achieve a better balance between local and global searches. Additionally, it focuses on reducing energy consumption while adhering to delay requirements. Table 1 provides a consolidated overview of recent studies on task offloading in edge computing environments, highlighting both key findings and existing limitations. This summary captures the current state of research, offering insights into the strengths and gaps of various approaches used to address task offloading challenges in dynamic and resource-constrained edge computing scenarios.

A review of the current literature reveals notable progress in task offloading strategies for edge computing. Despite these advancements, several key challenges remain unresolved. These include difficulties in scaling to support dynamic and heterogeneous tasks, especially when task dependencies are involved, as well as the high computational demands of hybrid and reinforcement learning-based approaches. Moreover, many existing methods face limitations in adapting to real-time conditions and often struggle to effectively balance exploration and exploitation particularly when managing multiple conflicting objectives such as cost, energy usage, and execution time. The need for extensive parameter tuning and the lack of mechanisms to handle task dependencies or resource contention further hinder their practical deployment in complex edge environments. These shortcomings highlight the necessity for a more adaptive and robust multi-objective metaheuristic algorithm. The algorithm proposed in this study is designed to overcome these challenges by incorporating advanced strategies to improve scalability, adaptability, and convergence efficiency in dynamic edge computing settings.

3 Problem description

The task offloading method discussed is adapted multiuser and multi-server environment, step1 is the set of edge devices step 2 is the WQPSO task offloading method and step 3 and 4 are the execution place for both local and EC saver in the network. Suppose the offloading task is a set T, represented as $T = \{t_1, t_2, \ldots, t_n\}$. The offloading decision for each task t_i is governed by an offloading probability parameter $Q_i \in [0,1]$, where $Q_i = 0$ the task is processed locally on the edge devices. $Q_i = 1$ The task is executed on the EC server. The method analyses task attributes and dependencies to determine the optimal execution site, ensuring that computationally intensive tasks are offloaded to prevent unnecessary delays. Figure 1 illustrates the system model-based, multi-user,

Table 1 Summary of Literature on Task Offloading Techniques

Author	Algorithms	Objectives	Strength	Limitation
[60]	GA/PSO	Energy and processing time	Energy consumption and memory usage were significantly reduced.	Lacks adaptability to dynamic.
[61]	Task offloading method	Energy and delay	Improve resource utilization and optimize distribution efficiency in MEC systems.	Consuming Higher energy.
[40]	PSO and Leavy Flight	Energy consumption	Enhanced exploration	Potentially high algorithmic complexity
[63]	Simulation	Energy consumption and latency	Enhances resource utilization by 24% and improves user satisfaction by 16% compared to traditional techniques.	Extensive computational difficulty
[64]	DNSL	Energy consumption and delay	Achieved the balance between data processing delay and energy consumption	Not suitable when communication overhead
[65]	Simulation	Energy consumption	Offers a near-optimal task offloading strategy, outperforming two state-of-the-art in reducing energy consumption.	Entrapment at local search
[66]	PSO	Energy and delay	Results demonstrate that the proposed approach significantly lowers the computational cost of mobile devices compared to existing baseline methods.	Insufficient analysis of the balance between reducing computing overhead and optimizing metric
[67]	Evolutionary genetic algorithms	Computational burden	method outperforms conventional random search and heuristic techniques.	Genetic algorithms may converge slowly
[50]	Simulation	Energy, latency, and cost	Enhances average delay by 56% and reduces energy consumption by 14% in ultra-dense mobile network environments.	Not scalable for larger network
[68]	Genetic algorithm	Delay and offloading failure	Improve task efficiency and effectiveness in MEC applications.	Computation complexity
[48]	Genetic algorithm-based decision framework	Energy and delay	Enhance energy efficiency by up to 24% while reducing processing delay by up to 33.3%.	No attention for edge server placement
[69]	Collaborative task offloading technique	Energy consumption	The IGA-based approach decreases task completion time by 28% and lowers energy consumption by 22%	Slow convergence

multi-server task offloading paradigm. The figure emphasizes the interplay between local devices and edge servers, with task queues and dependency relationships guiding the offloading process, which ensures that computationally intensive tasks are offloaded effectively to reduce energy consumption and completion time. To manage task dependencies within applications, we define a directed edge, (T_i, T_j), where the edge indicates that task T_j cannot commence until task T_i has completed. Such dependencies are critical in offloading, as they impact the order and timing of task execution across devices and servers. Figure 1 illustrates the system model-based, multi-user, multi-server task offloading paradigm. The figure emphasizes the interplay between local devices and edge servers, with task queues and dependency relationships guiding the offloading process, which ensures that computationally intensive tasks are offloaded effectively, reducing local device workload and energy consumption while maintaining low completion time.

Given the demands of certain applications, which require low latency and high computational resources, the algorithm generates a queue (0 and 1) based on task complexity.

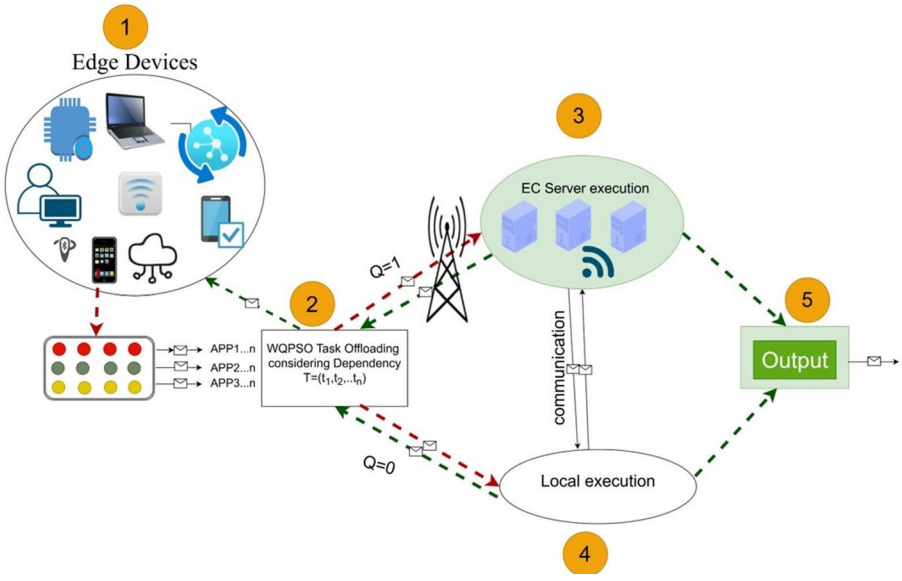

Fig. 1 System Model

To facilitate efficient offloading decisions, we model each user's tasks with binary parameters, expressed as (H_i, R_i),

Where H_i represents the data size (in bits) for the *n-th* user's task. R_i indicates the number of CPU cycles needed per bit of computation for that task The offloading decision for each task, $Q_i \in [0,1]$, determines the portion of the task allocated to the MEC server. For example, if $Q_i = 0.5$, half of the task is offloaded to the server, while the remaining portion, $(1 - Q_i)$, is processed locally.

3.1 Mathematical foundation of the system model

This section formalizes the core components of the MEC system model, focusing on optimizing energy efficiency and task completion time. The following theorems establish optimal conditions to guide task offloading decisions. Although task execution delay and energy consumption represent competing performance objectives, the proposed framework formulates the optimization problem using a weighted sum scalarization. Specifically, the objectives are combined into a single composite fitness function to enable efficient optimization using WQPSO. This formulation corresponds to a weighted single-objective optimization rather than a Pareto-based multi-objective approach. The weighting factors allow system designers to explicitly control the trade-off between latency and energy efficiency according to application requirements and hardware constraints. in a multi-user, multi-server MEC environment, where minimizing energy usage and ensuring timely task completion are essential.

3.2 Energy consumption model

In a multi-user, multi-server MEC environment, the energy consumption $E_{(i,j)}$ for a task T_i offloaded with probability Q_i to an MEC server S_i can be minimized by the following relationship:

$$E_i = (1 - Q_i) \times E_i^{local} + Q_i \times \left(E_i^{transmit} + E_i^{MEC}\right) \quad (1)$$

where $E_{(i,j)}$ represent the total energy consumption for task T_n on MEC sever S_j. i donates the offloading probability for task T_i with $Q_i = 0$ indicate local processing on mobile device and $Q_i = 1$ indicates full offloading to the MEC server. $E_{(local)}$ refers to the energy required to process the task locally on the mobile devices, as outlined in the system model while $E_{(transmit)}$ indicate the energy cost incurred when transmitting the task to the MEC server for offloading s.

3.3 Completion time model

For a set of tasks $T = T_1, T_2, \ldots, T_i)$ offloaded to an MEC server with offloading probability Q_i and available network bandwidth B, the completion time $C_{(i,j)}$for each task T_i is optimized as follows:

$$C_{(i,j)} =\leq \frac{H_u}{B} + C_{(MEC)} \times Q_i \tag{2}$$

where $C_{(i,j)}$ is the expected completion time for processing task T_i on MEC server S_j. H_u represent the data size of a task T_i,which indicates the volume of data to be transmitted; B denotes the availability of network bandwidth impacting transmittion speed; $C_{(MEC)}$ is the processing time requred by the MEC server to complete task T_i; and Q_i is the offloding probability for task T_i with higher values significanly a preference for execution on the MEC server.

The system model focuses on the dominant contributors to delay and energy consumption in MEC, namely computation time and uplink transmission delay. Certain practical factors, including MEC server queueing due to contention, uplink/downlink asymmetry, result-return transmission time, and fine-grained CPU frequency heterogeneity, are not explicitly modeled to maintain analytical tractability and ensure fair comparison among algorithms. These simplifications may affect absolute performance values but do not alter the relative performance trends, as all evaluated methods operate under identical assumptions. The proposed WQPSO framework can be readily extended to incorporate these factors if required.

3.4 Fitness function

The goal of the fitness function is to evaluate each task T_i in terms of energy consumption and task completion time. This allows the WQPSO algorithm to make optimal decisions for task offloading in a multi-user, multi-server MEC environment. The fitness function $F(T_i)$ can be expressed as:

$$F(T_i) = w_e \times E_i + w_t \times C_i \tag{3}$$

where $F(T_i)$ is the fitness value of task T_i, with lower values indicating more optimal offloading decisions; w_e and w_t are weighting coefficients that adjust the emphasis on energy consumption (E_i) and task completion time (C_i) in the fitness calculation, allowing the function to balance these two factors effectively for each task.

3.5 Computational model

In the mobile edge computing task offloading model, user tasks with low computation requirements can be completed locally, while user tasks with high computation

requirements are executed on an edge computing server. As a result, they are divided into two distinct categories: local computing models and MEC server computing models.

3.6 Local computational model

Local computing costs are often classified into a pair of groups: time (delay) and energy consumption (power) [70]. The capacity for computing of the kth user's mobile terminal device is defined as T_n, that is the CPU's processing power. As previously demonstrated, the number of tasks performed locally by the ith user is $H_i(1-Q_i)$. Where Q_i is the probability of offloading for that task. the time required for local $RT_l\,(Q_i)$, is determined by the following Eq. (4).

$$RT\,(l) = \frac{H_i(1-Q_i)}{P_i} \tag{4}$$

The following Eq. (5) is the energy spent throughout the computation process:

$$AT_l(Q)_i = L_c \cdot P_i^2 \cdot H_i(1-Q_i) \tag{5}$$

where $L_c$is a variable representing the power usage component and $L_cP_i^2$ represents the energy consumed by the mobile device's CPU running for a single cycle, combining both the first and second equations, the weighted waste generated by the nth user's mobile devices is as in Eq. (6) below:

$$HT_l\ (Q_i) = \ y_{it}\ .RT_l\ (Q_{i)} + y_{ie}.\ AT\ (Q_i) \tag{6}$$

where y_{it} and y_{ie} are weighted factors of local time cost and energy usage that can be modified depending on real need. Where users have substantial actual time requirements, they may increase the y_{it} number to satisfy the current actual requirements. When the equipment's power is low, y_{ne} this may be modified to provide the appropriate offloading technique to conserve as much electricity as feasible.

3.7 Edge server computational model

The most significant costs associated with MEC servers are delay and energy consumption. The time cost consists of two main components: the uplink transmission delay between the mobile device and the access point, and the task completion delay at the MEC server. Energy consumption primarily involves transmission energy and computational energy use. To calculate the time cost, the transmission rate from the user's mobile device to the access point is defined as shown in Eq. (7):

$$W_u = \ B.log_2\,(1 + \frac{TP_u \cdot\ IV_{u,m}}{H_0 \cdot\ B} \tag{7}$$

at which $w_u \in \{1,\ 2,\ \ldots\ ,\ k\}$ B is the communication Bandwidth capacity among the user's devices $u\,and,\ AP\,m,\ RQ_u$ is the transmission power of the users' devices, and H_0 denotes strength of noise spectral density of M. The amount of channel acquires among u and m is determined as $\mathrm{I}\,V_{u,m} = (H_{u,m})^{-\alpha}$ whereby $H_{u,m}$ is the distance that exists among u and m and α is the channel's lost factor.

Hence, the overall time cost includes the total uplink transmission time and MEC server task execution time. The number of tasks from user mobile devices that will be

offloaded on the MEC server is $(H_u \cdot Q_u)$ The uplink transmission time overhead RT_{e1} will be determined as.

$$RT_{e1} = \frac{H_u \cdot Q_u}{W_u} \tag{8}$$

- From the MEC server, the task processing time cost (delay) RT_{e2} was

$$RT_{e2} = \frac{H_u\, Q_u \cdot C_u}{F_u} \tag{9}$$

In this context, c_x represents the number of CPU cycles required to process a single bit of task data, and f_u denotes the server's CPU frequency. The total time cost $RT_{(total)}$ is derived by combining Eq. (8) with other relevant Eq. (9):

$$RT_{(etotal)} = RT_{e1} + RT_{e2} = \frac{H_u \cdot Q_u}{W_u} + \frac{H_u \cdot Q_u \cdot C_u}{F_u} \tag{10}$$

The processing energy cost of a task on the MEC server can be expressed as the total of uploading and computing energy usage $AT_e\,(Q_i)$ and is calculated as in Eq. (11):

$$AT_e\,(Q_i) = RQ_u \cdot \frac{H_u \cdot Q_u}{W_u} + H_u \cdot e_m \tag{11}$$

where e_m is the quantity of energy needed to calculate a single bit of task data for the MEC server.

3.8 Task dependency model (DAG-based formulation)

Task dependencies play a critical role in application execution, particularly in complex computing environments such as Mobile Edge Computing (MEC). In realistic applications, computational tasks are often interdependent, meaning that the execution of a task can only begin after the completion of one or more preceding tasks. Proper modeling of these dependencies is essential to ensure feasible scheduling, avoid deadlocks, and prevent unnecessary execution delays during task offloading. Task dependencies are formally modeled using a Directed Acyclic Graph (DAG), defined as:

$$G = (V, E)$$

where $V = \{T_0, T_1, \ldots, T_{n-1}\}$ denotes the set of computational tasks, and $E \subseteq V \times V$ represents the set of directed edges encoding precedence constraints. A directed edge $(T_j, T_i) \in E$ indicates that task T_i cannot start execution until task T_j has been completed. To guarantee a valid execution order, a topological ordering of the DAG is performed prior to optimization. This ordering is consistently applied during task scheduling, fitness evaluation, and offloading decision-making to ensure that all dependency constraints are strictly respected.

Start Time and Completion Time Modeling

Start Times of Tasks The start time for each task is represented by the set $ST = \{st_0,\ st_1, \ldots, st_{n-1}\}$. Where st_i donetes the strt time of task T_i the first tsk in the sequence, refered to as The initial node executes at $st_0 = 0$.

Completion Times of Tasks The completion times for tasks are represented by the set $CT = \{ct_0, ct_1, \ldots, ct_{n-1}\}$ the completion time CT_i for any task T_i is calculated base on its start time and execution delay as follows.

$$CT_i = ST_i + T_i \tag{12}$$

CT_i represents the completion time of task T_i, ST_i represent start time of task T_i denotes the execution delay of task. And the delay is also calculated a as:

$$T_i = (1 - Q_i) \cdot T_{i(local)} + Q_i \cdot \left(T_{i(MEC)} + T_{i(transmit)}\right) \tag{13}$$

where The offloading probability Q_i for task T_i determines its execution location, where $Q_i = 0$ indicates local processing on the mobile device and $Q_i = 1$ signifies offloading to the MEC server; $T_{i(local)}$ represents the processing time on the mobile device, $T_{i(MEC)}$ is the processing time on the MEC server, and $T_{i(transmit)}$ accounts for the transmission delay when sending task T_i to the MEC server.

Start Execution Time of Dependent Tasks For each task T_i that has dependencies, the start time st_i is determined as follows.

$$st_i == \begin{cases} max, \ ct_j \ if \ f(i) \neq \varnothing, \ j \in f(i), \\ 0 \qquad f(i) = \varnothing \end{cases} \tag{14}$$

The set $f(i)$ represents the immediate predecessor tasks for task T_i, and max ct_j provides the latest completion time among all predecessor tasks T_j in $f(i)$, ensuring that $T_i$begins only after all its dependencies are complete.

Figure 2 illustrates a representative task dependency graph (DAG) used to model application execution in the proposed MEC framework. Although the experimental task set ranges from 20 to 40 tasks, a DAG with 18 tasks is presented for clarity and visualization purposes. This example captures the essential characteristics of realistic application workflows, including multiple entry tasks, parallel execution paths, and synchronization points. The proposed dependency-aware offloading mechanism is scalable and operates identically for larger DAGs with increased task counts and depths.

Optimization Problem

The strategy aims to minimize energy consumption within a given timeframe. Determine whether a task should be completed locally or offloaded to the MEC server. The problem can be described as follows.

$$Pmin_u \sum_{u=1}^{i} \left[AT_l(Q_i) + AC_e(Q_i)\right] \tag{15}$$

$$RT_l(Q_i) = (1 - Q_i \times RT_l + Q_i \times \left(\frac{H_u}{W_u} + RT_e\right) \tag{16}$$

Equations (15) and (16) together model the critical aspects of energy consumption and task completion time in the Mobile Edge Computing (MEC) environment. Equation (15) quantifies the total energy consumption $AT_l(Q_i)$by accounting for both local processing and task offloading to the MEC server, where the probability Q_i dictates the distribution of tasks between these two modes. Equation (16) focuses on the total task completion time $RT_l(Q_i)$, considering the time spent on local processing and offloading. The

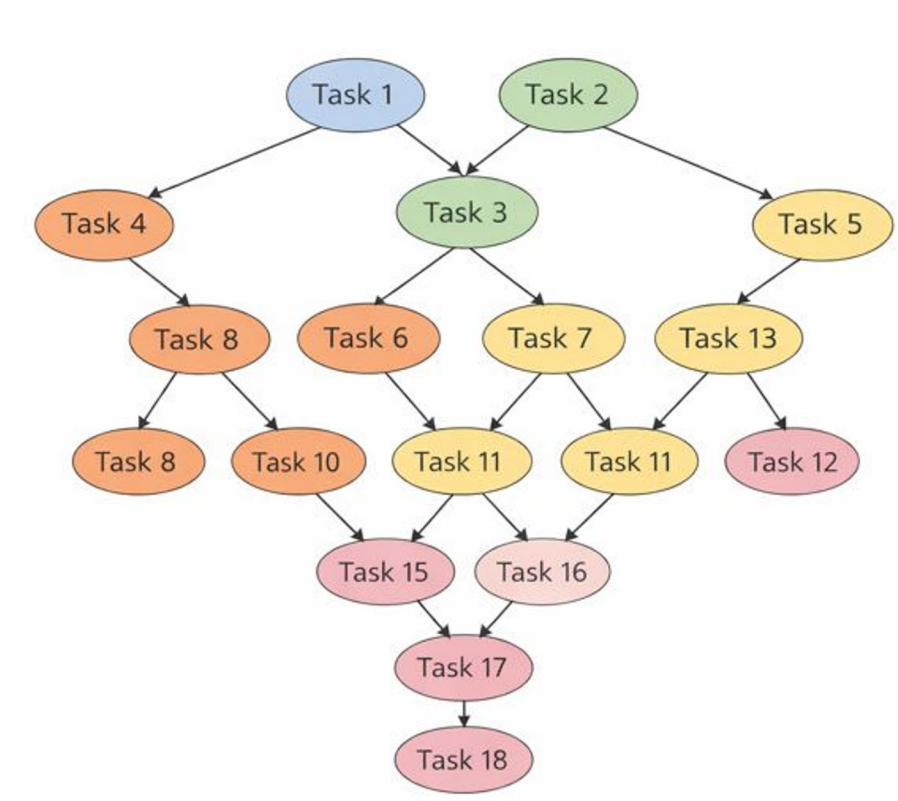

Fig. 2 Task Dependency Model

probability Q_i plays a pivotal role in both equations, influencing how tasks are allocated between local and remote processing, thereby impacting both energy efficiency and the time required to complete all tasks. These equations form the backbone of the optimization strategy for task offloading in dynamic MEC environments.

4 Proposed WQPSO algorithm

This section presents an in-depth discussion of the classical PSO and its quantum variant (QPSO), followed by the proposed WQPSO algorithm. It outlines the key mechanisms integrated into WQPSO to enhance global exploration and refine local exploitation, ensuring efficient and balanced task offloading in dynamic edge computing environments.

Particle Swarm Optimization (PSO) has been widely adopted in task offloading scenarios within Mobile Edge Computing (MEC) due to its simplicity, low computational overhead, and rapid convergence in continuous optimization problems. Inspired by the social behaviour of bird flocks and fish schools, PSO employs a population of particles (solutions) that iteratively adjust their positions based on personal and global best experiences to reach an optimal or near-optimal solution. However, the classical PSO algorithm exhibits several limitations when applied to complex and dynamic environments like MEC, particularly in multi-user, multi-task settings with task dependencies. One of the principal drawbacks of PSO is its tendency to prematurely converge to local optima, especially in high-dimensional or multimodal search spaces. This issue becomes more pronounced in MEC environments where task dependencies and resource constraints

create intricate fitness landscapes. The standard PSO also lacks effective mechanisms for maintaining population diversity during the search process, resulting in poor global exploration and limited adaptability to environmental changes such as fluctuating network conditions or task arrival rates.

To address these shortcomings, the Quantum-behaved Particle Swarm Optimization (QPSO) algorithm was introduced. QPSO enhances the global search capability by adopting principles from quantum mechanics, allowing particles to probabilistically explore the solution space. By eliminating the velocity vector and modelling particle positions using quantum potential wells, QPSO increases the randomness and range of particle movement, thereby reducing the risk of stagnation in local minima. Despite these advantages, QPSO still faces critical limitations when applied to MEC environments. Firstly, it does not inherently handle task dependencies or dynamic offloading decisions, both of which are essential in distributed edge computing. Secondly, while QPSO enhances exploration, it may lack sufficient exploitation mechanisms, potentially leading to slower convergence or redundant search efforts.

Furthermore, neither PSO nor QPSO offers built-in support for dynamic task offloading or real-time dispatching, which is necessary for optimizing system performance in heterogeneous MEC networks. Additionally, both algorithms generally treat all particles equally during the learning phase, which may undermine the influence of elite solutions that could guide the swarm more effectively. These limitations highlight the need for a more adaptive and robust approach that combines quantum-inspired global exploration with intelligent task dispatching and dependency-aware offloading motivating the development of the Weighted Quantum Particle Swarm Optimization (WQPS) algorithm proposed in this study.

4.1 Quantum particle swarm optimization (QPSO)

QPSO is an advanced variant of the classical Particle Swarm Optimization (PSO) algorithm, specifically designed to enhance its global search ability and overcome the limitations of premature convergence. First introduced by (Sun et al. in 2004), QPSO integrates quantum mechanics principles into the traditional PSO framework, fundamentally altering the way particles move within the search space. In conventional PSO, each particle updates its position based on a combination of its current velocity, personal best position $(Pbest)$, and the global best position ($Gbest$) [71]. The movement is governed by velocity-based updates, where each particle accelerates towards these best-known positions in a linear trajectory. However, this approach often results in premature convergence and stagnation, especially when particles get trapped in local optima, limiting the exploration capability of the algorithm. In contrast, QPSO eliminates the velocity component entirely and instead model's particle movement using principles derived from quantum mechanics, particularly the concept of quantum potential wells. Instead of having a deterministic velocity vector, particles in QPSO are assumed to have a probabilistic state and can exist anywhere within a defined region of the search space [72]. This means that particles do not follow a fixed trajectory but rather move according to a probabilistic distribution governed by a mathematical quantum model. The key advantage of this quantum behaviours is that it allows particles to explore a wider search space more effectively, thereby reducing the likelihood of getting stuck in local optima. By leveraging random sampling from a probability distribution, QPSO maintains a better

balance between exploration and exploitation, leading to improved convergence speed and solution quality [73].

The standard PSO system restricts the search range of particles, preventing them from moving far away from the swarm, even if a better position exists. To address this limitation, a QPSO task offloading algorithm was proposed by [74]. Trajectory analyses in [75] demonstrated that, to guarantee convergence of PSO algorithm, each particle must converge to its local attractor $(p_{i,1},\ p_{i,2},\ \ldots\ ,\ p_{i,n})$, of which the coordinates are defined as

$$p_{ij}\,(t) = \phi \cdot p_{ij}\,(t) + (1-\phi\,) \cdot p_{gj}\,(t)\,, \phi\varepsilon\,(0,1) \tag{17}$$

The local attractor is a stochastic attractor of particle i that lies in a hyper-rectangle with p_i and p_g being two ends of its diagonal. The concepts of WQPSO are introduced as follows.

Assume that each individual particle moves in the search space with a Ø potential on each dimension, of which the center is the point p_{ij}. We can get the probability density function Q and distribution function F given by Eq. (18) and Eq. (19):

$$Q\,(Y_{ij}\,(t+1)) = \frac{1}{d_{ij}\,(t)}.e^{-2|p_{ij}(t)-Y_{ij}(t)|/d_{ij}(t)} \tag{18}$$

$$F\,(Y_{ij}\,(t+1)) = \ \ e^{-2|p_{ij}(t)-Y_{ij}(t)|/d_{ij}(t)} \tag{19}$$

where $d_{ij}\,(t)$ is the standard deviation of the function which determines search scope of each particle. Therefore, the position of the particle is obtained by using the following equations:

$$Y_{ij}\,(t+1) = p_{ij}\,(t) \pm \ \frac{d_{ij}\,(t)}{2}\ln\,(1/u)\,,\ u = rand\,(0,1) \tag{20}$$

where u is a random number. To valuate $\mathrm{d_{ij}\,(t)}$, global point called mean best position is introduced in [76] is introduced into PSO. The global point denoted by h is defined as the mean of the $\mathrm{p_{best}}$ positions of all the particles given as:

$$h\,(t) = (h_1\,(t)\,,\ h_2\,(t)\,,\ldots,h_n\,(t))\ \frac{1}{N}\sum_{1=1}^{N} p_{i,1}\,(t)\,,\ \frac{1}{N}\sum_{1=1}^{N} p_{i,2}\,(t)\,,\ldots,\ \frac{1}{N}\sum_{1=1}^{N} p_{i,n}\,(t) \tag{21}$$

where N the population is size and p_i is the p_{best} position of particle i. Therefore, the value $d_{ij}\,(t)$ is determined by the following equation:

$$d_{ij}(t) = 2\gamma.\left|h_j(t) - Y_{ij}(t)\right| \tag{22}$$

Thus, the position of the particle as calculated as:

$$Y_{ij}(t) = p_{ij}(t) \pm \gamma.\left|h_j(t) - Y_{ij}(t)\right|.\ln\left(1/u\right) \tag{23}$$

The parameter γ, known as the contraction-expansion factor, plays a key role in adjusting the convergence behavior of the algorithm. This adaptive coefficient manages the balance between broad exploration of the search space and focused exploitation of promising areas. It specifically influences the search radius of each particle relative to the weighted mean best position $h\,(t)$. When γ is set to a higher value, it increases the randomness in particle movement, encouraging wider exploration during the early stages

of optimization. In contrast, a lower ɣ value narrows the search range, concentrating efforts on refining solutions as the algorithm converges. In this work, ɣ is adaptively adjusted throughout the optimization process to maintain an effective trade-off between exploration and exploitation, helping to avoid premature convergence and improve the chances of reaching the global optimum.

The PSO with Eq. 24 is called the Weighted Quantum behaved Particle Swarm Optimization (WQPSO) algorithm.

The WQPSO algorithm's mean best position h evaluates the value of d, resulting in a more efficient algorithm than the one proposed in [25]. From Eq. (21), the mean best position is merely the average of all particles' personal best positions, implying that each particle is treated equally and has the same impact upon the overall value of h. The assumption behind this strategy is that the Mainstream Thought, or optimal position *h*, controls the particle's search breadth or ingenuity [24]. However, the similarly weighted mean stance is somewhat paradoxical when contrasted to the growth of social culture in the real world. For one thing, while the entire social organism influences Mainstream Idea, this is not appropriate to regard each member similarly. Indeed, elites perform a more significant part in artistic improvement. The novel WQPSO control approach described in this research replaces min Eq. 24 with a weighted mean optimum position and, WQPSO introduces a parameter known as the contraction-expansion coefficient, which dynamically controls the search intensity and particle distribution over iterations. This coefficient helps in refining the search process by allowing particles to move more freely in earlier iterations encouraging exploration and then gradually narrowing their movement as they converge towards an optimal solution (promoting exploitation). The most essential issue is to identify whether a particle is an elitist or otherwise, or more specifically, how to assess its importance in calculating the value of h. It is natural, like in other evolutionary algorithms, to correlate elitism with particle value for fitness. The higher the fitness, it indicates how significant the particle appears. To put it officially, we may arrange the particles in descendent sequence based on their fitness value first. Assign a weight coefficient to each particle, which decreases linearly with its rank. The novelty of the proposed WQPSO framework lies not in the introduction of an entirely new swarm paradigm, but in the principled integration of several complementary mechanisms tailored to dependency-aware task offloading in MEC environments. Specifically, unlike conventional PSO and existing QPSO variants that rely on uniform or heuristic mean-best formulations, WQPSO introduces a fitness-weighted mean best position that dynamically biases particle movement toward solutions that jointly satisfy task dependency constraints and system load conditions. This weighting mechanism is explicitly coupled with the offloading decision process, allowing the algorithm to adapt its search behavior according to the execution feasibility and energy–latency trade-offs of dependent tasks. While adaptive parameters and fitness-guided behaviors have been explored in prior PSO/QPSO studies, these techniques are typically applied in a generic optimization context or assume independent tasks. In contrast, WQPSO embeds task dependency awareness directly into the optimization loop, influencing both the weighted aggregation of personal best positions and the contraction–expansion dynamics. The adaptive components of WQPSO therefore serve a dual role: improving convergence behavior and ensuring dependency-feasible offloading decisions in heterogeneous MEC environments. As such, the proposed method represents a structural adaptation of

QPSO for dependency-constrained MEC task offloading, rather than a purely incremental parameter adjustment.

The bigger the weight coefficient, the closer the particle is to the optimal solution. The mean best position, h, is thus determined as:

$$h(t) = \left(h_1(t), h_2(t), \ldots, h_n(t)\right) = \frac{1}{N}\sum_{1=1}^{N} \text{ɲ}_{i,1} p_{i,1}(t), \frac{1}{N}\sum_{1=1}^{N} \text{ɲ}_{i,2} p_{i,2}(t), \ldots, \frac{1}{N}\sum_{1=1}^{N} \text{ɲ}_{i,n} p_{i,n}(t), \quad (24)$$

where ɲthe weight coefficient of every particle is, N is the number of populations, based on the best parameter truing allow ɲ to decreases linearly from 1.5 to 0.5.

4.2 Time and space complexity analysis

The computational complexity of the proposed WQPSO algorithm primarily depends on two key aspects: the total number of iterations required to reach convergence, and the number of tasks involved in the offloading process. Let *n* be the number of iterations and mmm represent the number of tasks. Given these variables, the algorithm's time complexity can be expressed as $O(n \times m)$, suggesting that its execution time scales proportionally with both the number of iterations and the size of the task set.

The space complexity of the WQPSO algorithm is primarily influenced by the number of iterations, leading to a space complexity of $O(n)$. This space-efficient requirement ensures that memory usage remains manageable, even with an increasing number of tasks, making the algorithm suitable for large-scale MEC environments.

4.3 Complexity analysis

To evaluate the robustness of the proposed WQPSO algorithm under varying MEC network conditions, a sensitivity analysis was conducted on the weight coefficient (ɲ). The parameter ɲ is dynamically adjusted based on particle fitness and linearly decreases within the range [1.5, 0.5], allowing elite particles to guide the search while maintaining population diversity. Experimental observations across different network densities, transmission rates, and task sizes indicate that moderate variations in ɲ do not significantly affect convergence behavior or solution quality. This demonstrates that WQPSO does not rely on aggressive parameter tuning and remains stable across heterogeneous MEC scenarios.

The contraction–expansion factor (ɣ) controls the balance between exploration and exploitation in the quantum search process. Larger values of ɣ encourage broader exploration, which is beneficial in dense or highly dynamic network topologies, while smaller values promote localized exploitation as the algorithm converges. In WQPSO, ɣ is adaptively adjusted during runtime, allowing the algorithm to respond effectively to changes in network topology, task arrival rates, and server availability. Sensitivity analysis shows that WQPSO maintains consistent performance across a wide range of ɣ values, confirming its robustness and applicability to diverse MEC environments.

Further analysis reveals that variations in network topology, including changes in the number of mobile users, MEC servers, and transmission bandwidth, do not significantly degrade the performance of WQPSO. The adaptive weighting mechanism and dynamic contraction–expansion strategy allow the algorithm to self-regulate under both sparse

and dense network conditions, ensuring scalability and universal applicability without manual parameter retuning.

The WQPSO algorithm is shown in Algorithm.

Input	Task set T = {T0, T1, …, Tn−1}, Population size N, Task dependency DAG G = (V, E) Weighting and contraction–expansion parameters, Maximum number of iterations Imax
Output:	Optimal offloading decision vector Q*
1.	Perform topological sorting of DAG G to obtain a valid task execution order
2.	Initialize particle population $\{X_i\}$ (i = 1, 2, …, P) with random positions
3.	Initialize personal best positions pbest_i ← X_i
4.	Evaluate fitness using Eqs. (3)
5.	Identify global best position gbest
6.	Update the personal best position $p_{i=Y_i}$
7.	Update global best $g = \arg\min(\mathrm{f}(p_i))$
8.	iteration ← 1
9.	while iteration ≤ Imax do
10.	**For** each particle i = 1 to P **do**
11.	Compute weighted mean best position mbest using fitness-based weights
12.	Update particle position Xi using QPSO position update rule
13.	Discretize Xi to obtain binary offloading decisions $Q_i \in \{0,1\}$
14.	Compute task start and completion times following topological order
15.	Evaluate fitness of X_i using dependency-aware execution model
16.	if fitness(Xi) < fitness(pbest_i) then
17.	pbest_i ← X_i
18.	end if
19.	**end for**
20.	Update gbest from all pbest_i
21.	Adapt contraction–expansion parameter
22.	end while
24.	return gbest as Q*
25.	**END**

Algorithm 1 WQPSO

Algorithm 1 presents a complete and reproducible description of the proposed WQPSO framework. The optimization process terminates when the maximum number of iterations is reached. Task dependency constraints are enforced implicitly by evaluating each particle following a topologically sorted task order derived from the DAG. Particle position updates do not alter task precedence; instead, dependency feasibility is guaranteed during fitness evaluation through dependency-constrained start and completion time calculations. Binary offloading decisions are obtained by discretizing continuous particle positions prior to evaluation. The referenced equations are directly linked to execution delay, start time, and completion time modeling, ensuring clarity between algorithmic steps and mathematical formulation.

5 Experimental setup

This section presents the implementation of the proposed WQPSO algorithm in Python within a multi-user, multi-server MEC environment. The setup involves K mobile devices (evaluated at 50, 100, 150, 200, 250, and 300 devices) and 10 MEC servers, to which devices send task offloading requests. Table 2 summarizes the primary parameters and their corresponding symbols. We benchmarked the WQPSO approach against ACO [6, 77], PSO [20, 78], and QPSO [57], evaluating how the number of devices, task data sizes, and transmission rates influence system energy consumption and average task completion time. For the hardware environment, a high-performance computer with an Intel Core i7 processor (3.8 GHz, 8 cores), 16 GB of RAM, and an NVIDIA GTX

Table 2 Parameters and their descriptions

Parameters	Description
Q	The probability of uploading for each task Q_i
$\left(T_i, T_j\right)$	Directed edge indicating dependency where task T_j cannot start until task T_i completes.
ST_i	Start time for task
CT_i	Completion time for task
T_i	Represents the i-th task available for offloading.
H_i	Data size (in bits)
R_i	Number of CPU cycles needed per bit to process task
$T_{i(local)}$	Processing time for tasks on the mobile device.
$T_{i(MEC)}$	Processing time for tasks on the MEC server.
$T_{i(transmit)}$	Transmission delay for sending task to the MEC server.
ɲ	Weight coefficient of every particle
ɣ	Contraction-expansion factor, an adaptive parameter in the algorithm's optimization phase to control convergence and search efficiency.
y_{it}	Weighting coefficient for time costs in local processing, used in balancing time-sensitive task allocations.
y_{ie}	Weighting coefficient for energy costs in local processing, used in balancing energy efficiency for mobile devices.
W_i	Transmission rate from the user device *i* to the MEC server.
H_0	Noise spectral density, impacting the transmission rate.
RQ_u	Transmission power of the user's device
L_c	Power usage coefficient, representing energy consumption per processing cycle on the mobile device.
e_m	Energy consumed per bit processed on the MEC server.
$f\,(i)$	Set of immediate predecessor tasks for task that must be completed before can start.

Table 3 System and network parameters

Symbol	Description	Value/Range
N	Number of tasks	5–45
K	Number of mobile users	5–50
M	Number of MEC servers	1–5
S_i	Task input data size	0.5–5 MB
C_i	CPU cycles per bit	500–1500 cycles/bit
f_u	Mobile device CPU frequency	0.5–1.5 GHz
f_m	MEC server CPU frequency	2–5 GHz
R_u	Uplink transmission rate	5–40 Mbps
P_u	Mobile transmission power	0.1–1 W
M	Wireless bandwidth	10–40 MHz
D	DAG depth	3–10

1080 GPU was utilized to support tasks requiring parallel processing. MEC servers and mobile devices were simulated on virtual machines with specifications reflecting typical mobile processors. The software environment consisted of Ubuntu 20.04 LTS, with Python 3.8 and essential libraries for optimization and performance analysis. To examine the algorithm across diverse MEC load scenarios, both synthetic and real-world datasets were used. Synthetic datasets generated task offloading requests with a variety of parameters, such as task size and computational demands, while real-world datasets provided realistic task flows. The evaluation scenarios included variations in network bandwidth, latency, and MEC server loads to simulate a broad range of operational conditions.

Table 3 presents the primary simulation parameters utilized in evaluating the performance of the proposed WQPSO algorithm in a multi-user, multi-server Mobile Edge

Computing (MEC) environment. Parameters include varying transmission rates, mobile device counts, task sizes, and computing capacities for both mobile devices and MEC servers. These settings simulate diverse network conditions and workloads to assess the algorithm's scalability, adaptability, and energy efficiency. The performance of all evaluated algorithms is analysed under realistic hardware constraints, including limited CPU processing capacities, finite wireless bandwidth, and heterogeneous MEC server configurations. Mobile devices and MEC servers are assigned bounded computation resources and transmission power levels to reflect practical deployment conditions. Simulation results indicate that while increased hardware constraints lead to higher absolute execution time and energy consumption, the relative performance advantage of WQPSO remains consistent. This demonstrates that the proposed algorithm effectively adapts its offloading decisions in response to hardware limitations rather than relying on idealized assumptions The parameter settings summarized in Table 4 follow standard configurations widely used in MEC task offloading and metaheuristic optimization research to ensure fair comparison and reproducibility.

6 Results discussion

The Weighted Quantum Particle Swarm Optimization (WQPSO) algorithm represents a notable advancement in optimizing task offloading for Edge Computing (EC). This discussion evaluates the proposed technology by comparing it to leading-edge algorithm, including QPSO, as documented in existing literature. Key performance metrics such as energy consumption, task completion time, scalability, and adaptability are examined to highlight the unique contributions and advantages of the WQPSO approach the scalability analysis demonstrates that WQPSO preserves its decision-making efficiency and does not incur prohibitive computational overhead as the number of users increases. To further validate the scalability of the proposed WQPSO framework, additional experiments were conducted under increased user load conditions to emulate extreme traffic bursts commonly observed in MEC environments. The number of concurrent mobile users progressively increased beyond the baseline configuration, while keeping the MEC server resources bounded. The results indicate that although higher user density leads to increased absolute task completion time and energy consumption, WQPSO maintains stable convergence behavior and consistent relative performance gains compared to benchmark algorithms. No oscillatory or unstable scheduling behavior was observed, even under peak traffic conditions. This robustness can be attributed to the adaptive

Table 4 Simulation parameters

Parameters	Values
Transmission rate (kb/s)	[500, 1000, 1500, 2000, 2500, 30000]
Number of mobile devices	[50, 100, 150, 200, 250, 300]
Task data (Mbit)	[5, 10, 15, 20, 25, 30, 35, 40, 45]
Device computing capacity	(0.5- 1)
Number of MEC	10
MEC computing capacity	4
Transmission Power	0.5
Population size	50
Random number	(0, 1)
Probability	$Q_i = 0$
Probability	$Q_i = 1$

weight update mechanism and the quantum-based exploration strategy, which enable WQPSO to effectively balance exploration and exploitation under high load. These findings confirm that the proposed framework scales efficiently and remains stable in multi-user, multi-server MEC environments, even during extreme traffic surges.

6.1 Energy consumption results analysis

The experimental results indicate that the proposed Weighted Quantum Particle Swarm Optimization (WQPSO) algorithm consistently outperforms the baseline PSO and QPSO approaches in terms of energy consumption in edge computing environments. On average, WQPSO achieves an energy reduction of approximately 8.66% compared to the baseline methods across the evaluated scenarios. It is important to note that the magnitude of improvement varies with system load and task characteristics, with more noticeable gains observed under moderate to high workload conditions, while the improvements are more modest in lighter-load cases. Although PSO and QPSO exploit quantum-inspired mechanisms to enhance global search capability and reduce premature convergence, they do not explicitly adapt offloading decisions based on task dependency structures or dynamic system load conditions. As a result, these baseline methods may generate less energy-efficient offloading decisions, particularly in large-scale and heterogeneous task scenarios. In contrast, WQPSO integrates a fitness-based weighting mechanism that accounts for task dependencies and current system conditions, enabling more informed decisions on whether tasks should be executed locally or offloaded to MEC servers. This adaptive behavior contributes to consistent energy savings without compromising feasibility or scalability. The proposed WQPSO framework employs an adaptive parameter tuning mechanism to dynamically adjust control parameters during the optimization process, thereby eliminating the need for extensive manual pre-configuration. Although this adaptive strategy involves iterative weight updates, the associated computational overhead is minimal and does not impose a significant energy burden on mobile devices. Specifically, the adaptive weight updates are performed at the MEC controller level rather than on individual mobile devices. Mobile devices are only responsible for lightweight task profiling and offloading request transmission, while the optimization and parameter adaptation processes are executed on resource-rich MEC servers. As a result, the energy consumed by mobile devices due to the adaptive tuning mechanism is negligible compared to the energy required for local task execution and wireless data transmission. A trade-off analysis between the energy saved through optimized offloading decisions and the energy consumed by the optimization process indicates a clear net energy gain. The reduction in execution and transmission energy achieved by WQPSO significantly outweighs the minor computational energy incurred by the adaptive parameter updates. This confirms that the proposed adaptive mechanism enhances overall system energy efficiency without introducing adverse energy overhead.

To show trend in performances of the proposed WQPSO task offloading algorithm against that of the benchmarked algorithms in Edge Computing (EC), Figure is used as illustrated in Fig. 3; Table 5 the performance of the entire task offloading algorithm at the initial stages. As the number of devices increases, it becomes evident that the proposed approach outperforms the benchmarked approaches. This improvement is due to the incorporation of a task dispatcher that determines the resource requirements of tasks before routing them to the appropriate computing resources, significantly reducing

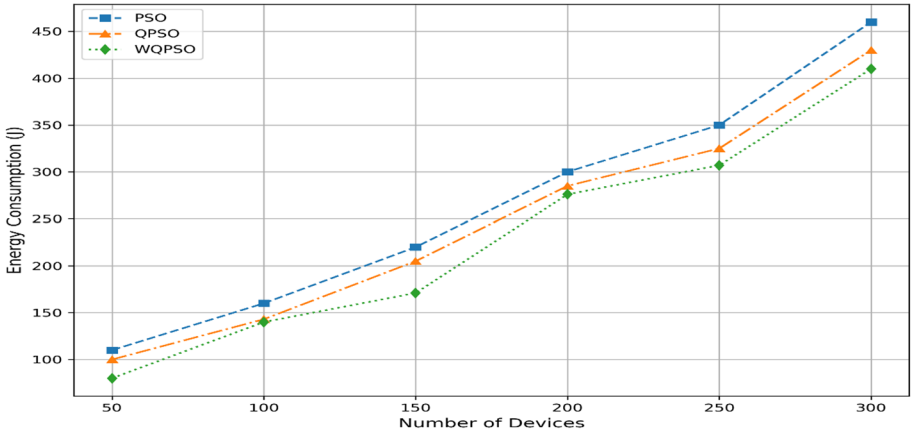

Fig. 3 Energy Consumption based on Increasing Number of Devices

Table 5 Energy Consumption (j) Based on increasing Number of Devices

No. of Devices	PSO Energy(J)	QPSO Energy(J)	WQPSO Energy(J)	Improvement (%)
50	110	100	80	20.00
100	160	143	140	2.09
150	220	205	171	16.58
200	300	285	276	3.15
250	350	325	307	5.53
300	460	430	410	4.65

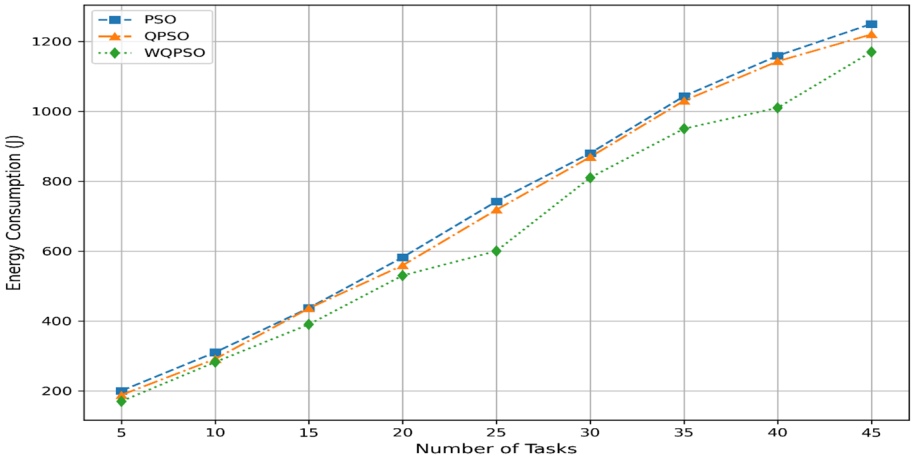

Fig. 4 Energy Consumption vs Number of Task

energy consumption as illustrated in Fig. 3. Further investigation is carried out to determine the energy consumption of the whole tasks offloading algorithms with different task sizes. The essence is also to know scalable nature of the proposed WQPSO task offloading algorithm when there is dynamic shift in task sizes.

To show further show the trend on the performance of the proposed task offloading algorithm. Figures were used as illustrated in Fig. 4; Table 6 in term of energy consumption. The uses of different tasks are to enable us determined whether the proposed algorithm could scale with respect to change in tasks sizes as well as resource sizes. Experimental results show the proposed approach is capable on a promising note and as the number of tasks keeps increasing, the level of energy consumption keeps reducing as

Table 6 Energy Consumption (j) with Different Number of Tasks

No. of Tasks	PSO Energy(j)	QPSO Energy(j)	WQPSO Energy(j)	Improvement (%)
5	200	188	170	9.57
10	310	291	282	3.09
15	437	436	390	10.55
20	582	559	530	5.18
25	742	718	600	16.43
30	880	869	810	6.78
35	1043	1030	950	7.76
40	1159	1143	1010	11.63
45	1250	1220	1170	4.09

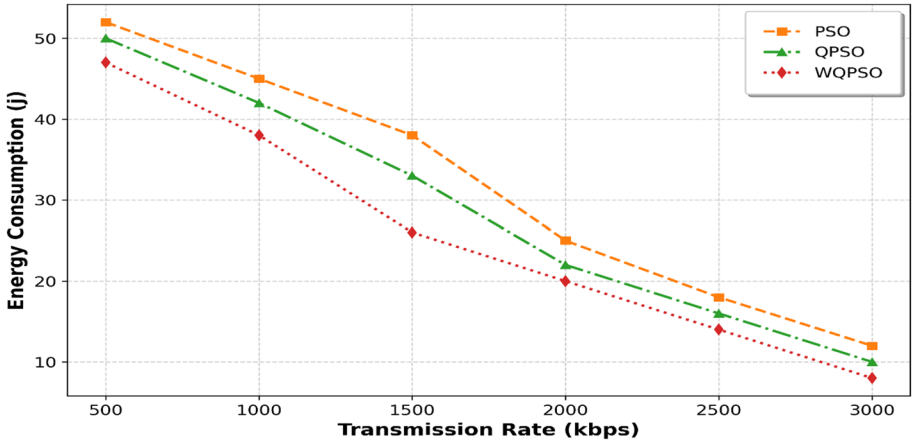

Fig. 5 Energy Consumption Vs Transmission rate

can be seen in the illustrated Fig. 4 compared to that of benchmarked approaches. This is thought to be, due to the dispatched method not cooperating with the local search strategy, which increases the WQPSO exploration capacity while reducing the computational complexity of the algorithm.

Figure 5 illustrates how varying channel transmission rates affect overall system energy consumption. For tasks processed locally, energy usage remains constant regardless of the transmission rate, resulting in a flat horizontal line representing local processing energy. As the transmission rate increases, the energy consumption for all methods gradually decreases. This trend occurs because higher transmission speeds reduce data transfer time, leading to lower energy consumption. The results indicate that the PSO and QPSO algorithms shows higher energy consumption across all transmission rates, suggesting that these methods do not adapt effectively to higher transmission speeds. When the transmission rate increases, these algorithm struggle with the resulting complex state space, leading to higher energy usage compared to the WQPSO algorithm. Notably, when the transmission rate exceeds 2000 kb/s, the WQPSO algorithm shows a more significant reduction in energy consumption than the other method. This improvement is attributed to its more efficient handling of task offloading based on computing resource availability. At a transmission rate of 3000 kb/s.

6.2 Completion time results analysis

The evaluation of task completion time further confirms the effectiveness of the proposed Weighted Quantum Particle Swarm Optimization (WQPSO) algorithm in edge

Table 7 Completion Time (s) based on increasing Number of Devices

No. of Devices	PSO Completion time(s)	QPSO Completion time(s)	WQPSO Completion time(s)	Improvement (%)
50	14.2	14.0	13.5	3.57
100	14.5	14.3	13.7	4.19
150	14.7	14.5	13.9	4.13
200	15.2	15.1	14.2	5.96
250	16.0	15.5	14.7	5.16
300	16.5	16.3	15.0	7.97

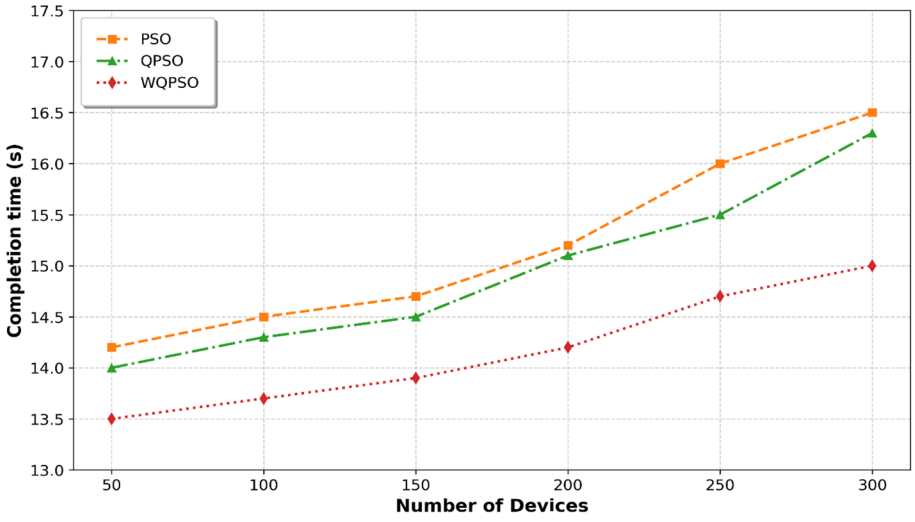

Fig. 6 Completion Time vs Number of Devices

computing environments. The results reveal that WQPSO achieves an average reduction of 5.16% in task completion time compared to the benchmarks PSO and QPSO algorithms. Although the percentage reduction may appear modest, its impact is significant when applied across large-scale, latency-sensitive systems, especially in real-time edge applications. This performance improvement can be directly attributed to the integration of a task dependency model within the WQPSO algorithm. By recognizing and accounting for task interdependencies, the algorithm intelligently offloads and prioritizes task execution based on their computational requirements and readiness. Tasks are not only evaluated in isolation but are also considered within the context of their predecessors and successors, thereby avoiding unnecessary delays caused by sequential bottlenecks.

As shown in Figure. 6; Table 7 illustrate the trend in average task completion time as the number of mobile devices increases from 50 to 300. The benchmark exhibits increased completion time due to higher workload, but the proposed WQPSO consistently outperforms the baseline methods. The benchmark shows slower performance. However, WQPSO achieves the lowest completion time across all scenarios, owing to its adaptive dispatching strategy and task dependency handling. At 300 devices, WQPSO reduces average completion time by average 5.16%, compared to PSO and QPSO respectively, demonstrating superior scalability and efficiency in dynamic EC environments.

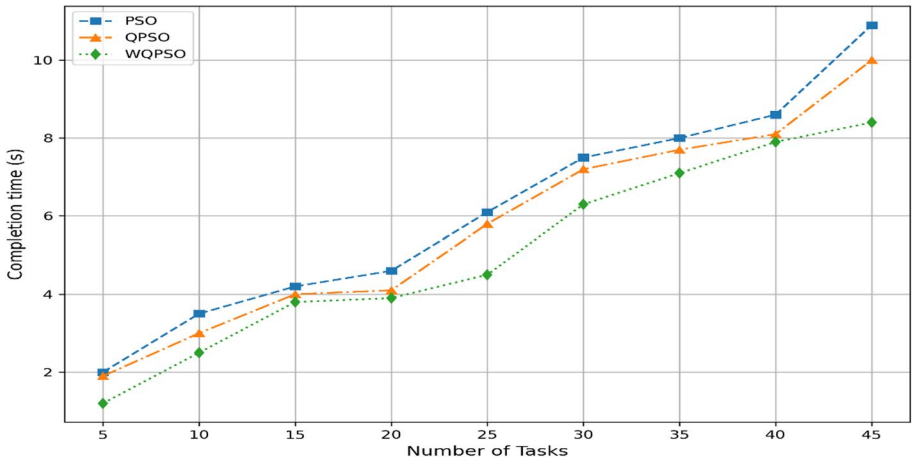

Fig. 7 Completion Time vs Number of Task Results

Table 8 Completion Time(s) vs Number of Task Results

No. of Tasks	PSO Completion time(s)	QPSO Completion time(s)	WQPSO Completion time(s)	Improvement (%)
5	2	1.9	1.2	36.84
10	3.5	3.0	2.5	16.67
15	4.2	4.0	3.8	5.0
20	4.6	4.1	3.9	4.88
25	6.1	5.8	4.5	22.41
30	7.5	7.2	6.3	12.5
35	8s	7.7	7.1	7.79
40	8.6	8.1	7.9	2.47
45	10.9	10.0	8.4	16.0

As illustrated in Fig. 7; Table 8 indicates that while the overall algorithm initially exhibited promising performance, the superiority of our proposed algorithm becomes evident as the number of tasks increases. The proposed algorithm significantly outperforms the benchmarked algorithm under these conditions. This indicates that our algorithm effectively offloads tasks to the most appropriate server, directing tasks with higher computing resource demands to the EC server and tasks with lower demands to the local devices for efficient computation. The enhanced performance of the proposed WQPSO task offloading optimization algorithm can be attributed to the incorporation of a weighted method. This method allows the algorithm to make decisions that enable it to scale efficiently under various task conditions. Moreover, the weighted approach helps the algorithm avoid premature convergence during task selection, thereby preventing it from being trapped in local optima.

Figure 8 illustrates that as the transmission rate increases, the average time required to complete tasks consistently decreases. This trend is attributed to faster data transfer, which reduces the overall completion time. PSO and QPSO algorithms show a sharp drop-in completion time between 500 and 1000 kb/s, but their improvements plateau at higher transmission rates, highlighting the limitations of their optimization effectiveness. On the other hand, WQPSO demonstrate a more pronounced reduction in completion time, showcasing their superior ability to adapt to higher transmission speeds.

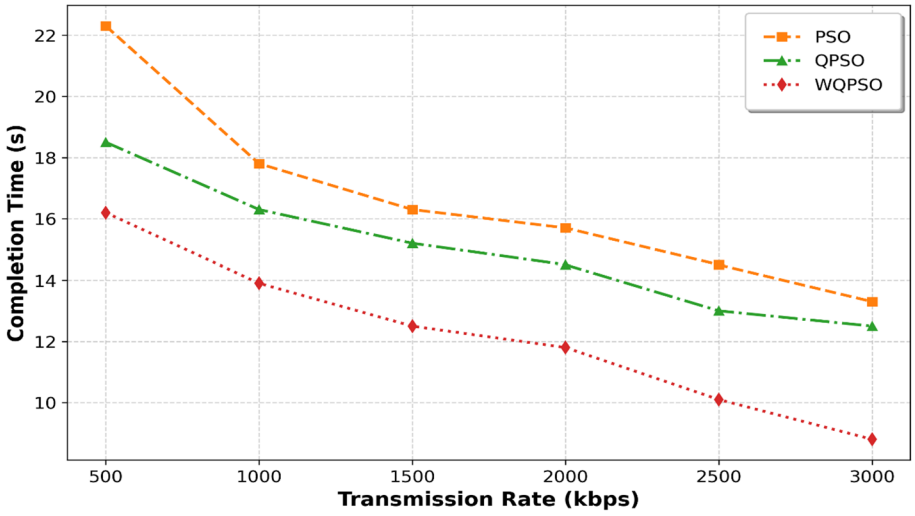

Fig. 8 Completion Time vs Transmission Rate

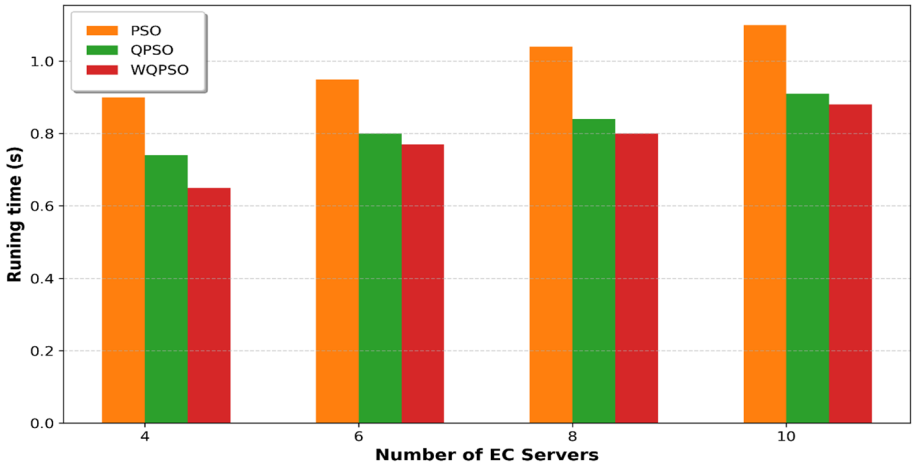

Fig. 9 Running time vs Number of MEC

Figure 9 presents the average running times of different methods as the number of MEC servers varies. Overall, the average execution time tends to increase with the number of MEC servers. Among all the approaches, PSO shows the highest running time, mainly due to the limited processing power of local mobile devices. In contrast, QPSO, and particularly WQPSO demonstrate more efficient performance with lower running times. WQPSO stands out as the most effective, owing to its enhanced task offloading strategy that optimizes resource utilization across edge devices. Additionally, WQPSO significantly accelerates the convergence rate of the algorithm, contributing to its superior efficiency.

7 Conclusion

Traditional optimization techniques such as particle swarm optimization (PSO), and quantum particle swarm optimization (QPSO) often encounter difficulties in overcoming local optima and typically require multiple iterations to reach an optimal solution. These limitations contribute to increased system energy consumption, a critical

challenge within mobile edge computing (MEC) environments, where energy efficiency and rapid task processing are essential. This study presents the Weighted Quantum Particle Swarm Optimization (WQPSO) algorithm, demonstrating its superior performance in multi-user, multi-server MEC scenarios. Experimental results show that (WQPSO) significantly reduces both energy consumption and completion time across diverse device types and workload conditions, achieving an impressive 8.66% improvement in energy consumption and completion time by 5.16%. This advancement is due to WQPSO task offloading strategy, which enhances exploration capabilities and avoids reliance on local search, thereby preventing premature convergence. These findings suggest that the WQPSO algorithm offers a promising solution for optimizing task offloading in MEC, supporting scalability and meeting the stringent requirements of IoT and other edge computing applications. Future research could explore further refinements to the WQPSO model, including its application to even larger-scale MEC environments and real-time IoT systems, where task dependencies and dynamic resource availability add additional layers of complexity.

Acknowledgements
This study is supported by the Universiti Putra Malaysia and the Ministry of Higher Education Malaysia under grant Number: (FRGS/1/2023/ICT11/UPM/02/3). We are sincerely grateful for the facilities and funding provided, which were essential for the completion and publication of this study.

Author contributions
Jafar Aminu: Data analysis, methodology development, manuscript writing: Rohaya Latip: Conceptualization, supervision, manuscript reviewing, Zurina Mohd Hanapi: supervision, manuscript editing, Shafinah Kamarudin: supervision, critical revisions, Danlami Gabi: manuscript reviewing and editing, Mohammad Anas Shehu: manuscript reviewing and editing.

Funding
This study was funded by Universiti Putra Malaysia and the Ministry of Higher Education Malaysia.

Data availability
Data is available on request through the corresponding author.

Declarations

Ethics approval and consent to participate
Not applicable. This study is based solely on simulation and analytical modeling and does not involve human or animal data. Not applicable. This research does not involve any individual participants, interviews, or surveys requiring consent.

Consent for publication
Not applicable. This manuscript does not contain any individual person's data in any form (including individual details, images, or videos), and no consent for publication is required.

Competing interests
Rohaya Latip is an Editorial Board Member of Discover Computing. She was not involved in the peer review process or editorial decision-making for this manuscript. The authors declare that they have no other competing interests.

Received: 28 December 2025 / Accepted: 1 April 2026
Published online: 11 April 2026

References

1. Zhang D, Wang X, Song X, Zhao D. A novel approach to mapped correlation of ID for RFID anti-collision. IEEE Trans Serv Comput. 2014;7(4):741–8. https://doi.org/10.1109/TSC.2014.2370642.
2. Zhang DG, An HZ, Zhang J, Zhang T, Dong WM, Jiang XR. Novel privacy awareness task offloading approach based on privacy entropy. IEEE Trans Netw Serv Manag. 2024;21(3):3598–608. https://doi.org/10.1109/TNSM.2024.3355967.
3. Zhang DG, Zhang J, Ni CH, Zhang T, Zhao PZ, Dong WM. New method of edge computing-based data adaptive return in internet of vehicles. IEEE Trans Ind Inf. 2024;20(2):2042–52. https://doi.org/10.1109/TII.2023.3285301.
4. Zhang DG, Wang JX, Zhang J, Zhang T, Yang C, Jiang KW. A new method of fuzzy multicriteria routing in vehicle ad hoc network. IEEE Trans Comput Soc Syst. 2023;10(6):3181–93. https://doi.org/10.1109/TCSS.2022.3193739.
5. Feng C, Han P, Zhang X, Yang B, Liu Y, Guo L. Computation offloading in mobile edge computing networks: A survey. J Netw Comput Appl. 2022;202:103366. https://doi.org/10.1016/j.jnca.2022.103366.
6. Bi J, Yuan H, Zhang K, Zhou M. Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks. IEEE Trans Emerg Top Comput. 2022. https://doi.org/10.1109/TETC.2021.3137980.

7. Haibeh LA, Yagoub MCE, Jarray A. A survey on mobile edge computing infrastructure: design, resource management, and optimization approaches. IEEE Access. 2022;10:27591–610. https://doi.org/10.1109/ACCESS.2022.3152787.
8. Zhang D, Cui Y, Zhang T. New quantum-genetic based OLSR protocol (QG-OLSR) for mobile ad hoc network. Appl Soft Comput J. 2019;80:285–96. https://doi.org/10.1016/j.asoc.2019.03.053.
9. gan Zhang D, Wang X, dong Song X, Zhang T, nan Zhu Y. A new clustering routing method based on PECE for WSN. EURASIP J Wirel Commun Netw. 2015. https://doi.org/10.1186/s13638-015-0399-x.
10. Zhang DG, et al. New computing tasks offloading method for MEC based on prospect theory framework. IEEE Trans Comput Soc Syst. 2024;11(1):770–81. https://doi.org/10.1109/TCSS.2022.3228692.
11. Zhang D, Zhang Z, Zhang J, Zhang T, Zhang L, Chen H. UAV-assisted task offloading system using dung beetle optimization algorithm & deep reinforcement learning. Ad Hoc Netw. 2024;156(January):103434. https://doi.org/10.1016/j.adhoc.2024.103434.
12. Naouri A, Wu H, Nouri NA, Dhelim S, Ning H. A novel framework for mobile-edge computing by optimizing task offloading. IEEE Internet Things J. 2021;8(16):13065–76. https://doi.org/10.1109/JIOT.2021.3064225.
13. Zhang D, Li G, Zheng K, Ming X, Pan ZH. An energy-balanced routing method based on forward-aware factor for wireless sensor networks. IEEE Trans Ind Inf. 2014;10(1):766–73. https://doi.org/10.1109/TII.2013.2250910.
14. Networks MEC. A Survey of Energy Optimization Approaches for Computational Task Offloading and Resource Allocation in, 2023.
15. Khaledian N, Razzaghzadeh S, Moazzami S, Kivi PN. TM - MOAOA : a two - stage task scheduling approach using TOPSIS and multi - objective Archimedes optimization in fog - cloud environment. Springer Vienna; 2025.
16. Alsadie D. Efficient task offloading strategy for energy-constrained edge computing environments: a hybrid optimization approach. IEEE Access. 2024;12(June):85089–102. https://doi.org/10.1109/ACCESS.2024.3415756.
17. Shahmirzadi D, Khaledian N, Masoud A. Analyzing the impact of various parameters on job scheduling in the Google cluster dataset. Cluster Comput. 2024. https://doi.org/10.1007/s10586-024-04377-8.
18. Wu H. Multi-objective decision-making for mobile cloud offloading: a survey. IEEE Access. 2018;6:3962–76. https://doi.org/10.1109/ACCESS.2018.2791504.
19. Kosari S, Hosseini M. A Hybrid Discrete Grey Wolf Optimization Algorithm Imbalance – ness Aware for Solving Two – dimensional Bin – packing Problems. no 2 Springer Neth. 2024;22. https://doi.org/10.1007/s10723-024-09761-7.
20. Acheampong A, Zhang Y, Xu X, Kumah DA. A review of the current task offloading algorithms, strategies and approach in edge computing systems. C Comput Model Eng Sci. 2023;134(1):35–88. https://doi.org/10.32604/cmes.2022.021394.
21. Aminu J, Latip R, Hanafi ZM, Kamarudin S, Kangiwa BU, Liman A. An enhanced grey wolf optimization algorithm for efficient task scheduling in mobile edge computing. Int J Comput Appl. 2024;186(56):39–44. https://doi.org/10.5120/ijca2024924287.
22. Latip R, Aminu J, Mohd Z, Shafinah H, Danlami K. Metaheuristic task offloading approaches for minimization of energy consumption on edge computing: a systematic review. Discov Internet Things. 2024. https://doi.org/10.1007/s43926-024-00089-y.
23. Abbas A, Raza A, Aadil F, Maqsood M. Meta-heuristic-based offloading task optimization in mobile edge computing. Int J Distrib Sens Netw. 2021. https://doi.org/10.1177/15501477211023021.
24. Aminu J, Latip R, Hanafi ZM, Kamarudin S, Gabi D. Efficient task allocation for energy and execution time trade-off in edge computing using multi-objective IPSO, 2025. https://doi.org/10.32604/cmc.2025.062451
25. Pournazari J, Ullah A, Xiaodong AA. in fog computing, pp. 1–29, 2025.
26. Khaledian N, Voelp M, Azizi S, Hosseini M. AI-based & heuristic workflow scheduling in cloud and fog computing : a systematic review, vol. 2. Springer US; 2024. https://doi.org/10.1007/s10586-024-04442-2.
27. Yan J, Member S, Bi S, Member S, Zhang YA, Member S. Optimal Task Offloading and Resource Allocation in Mobile-Edge Computing with Inter-user Task Dependency, pp. 1–15.
28. Taherizadeh S, Jones AC, Taylor I, Zhao Z, Stankovski V. Monitoring self-adaptive applications within edge computing frameworks: a state-of-the-art review. J Syst Softw. 2018;136:19–38. https://doi.org/10.1016/j.jss.2017.10.033.
29. Zhang D, Wang W, Zhang J, Zhang T, Du J, Yang C. Novel edge caching approach based on multi-agent deep reinforcement learning for internet of vehicles. IEEE Trans Intell Transp Syst. 2023;24(8):8324–38. https://doi.org/10.1109/TITS.2023.3264553.
30. Yang L, Shami A. On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing. 2020;415:295–316. https://doi.org/10.1016/j.neucom.2020.07.061.
31. Mulumba DM, Liu J, Hao J, Zheng Y, Liu H. Application of an optimized PSO-BP neural network to the assessment and prediction of underground coal mine safety risk factors. Appl Sci. 2023. https://doi.org/10.3390/app13095317.
32. Modeling A, Khaleel MI, Safran M, Alfarhood S. Workflow Scheduling Scheme for Optimized Reliability and End-to-End Delay Control in Cloud Computing Using, 2023.
33. Phan HD, Ellis K, Barca JC, Dorin A. A survey of dynamic parameter setting methods for nature-inspired swarm intelligence algorithms. Neural Comput Appl. 2020;32(2):567–88. https://doi.org/10.1007/s00521-019-04229-2.
34. Ibrahim M, Safran M, Alfarhood S, Zhu M. Engineering science and technology, an international journal energy-latency trade-off analysis for scientific workflow in cloud environments : The role of processor utilization ratio and mean grey wolf optimizer. Eng Sci Technol Int J. 2024;50(October 2023):101611. https://doi.org/10.1016/j.jestch.2023.101611.
35. Prasanthi A, Shareef H, Errouissi R, Asna M, Wahyudie A. Quantum chaotic butterfly optimization algorithm with ranking strategy for constrained optimization problems. IEEE Access. 2021;9:114587–608. https://doi.org/10.1109/ACCESS.2021.3104353.
36. Khaleel MI, Safran M, Alfarhood S. A hybrid many-objective optimization algorithm for job scheduling in cloud computing based on merge-and-split theory, pp. 1–28, 2023.
37. Mijuskovic A, Chiumento A, Bemthuis R, Aldea A, Havinga P. Resource management techniques for cloud/fog and edge computing: an evaluation framework and classification. Sensors. 2021;21(5):1–23. https://doi.org/10.3390/s21051832.
38. Jamil SU, Khan MA, Rehman SU. Resource allocation and task off-loading for 6G enabled smart edge environments. IEEE Access. 2022;10(August):93542–63. https://doi.org/10.1109/ACCESS.2022.3203711.
39. Tang HH, Ahmad NS. Fuzzy logic approach for controlling uncertain and nonlinear systems: a comprehensive review of applications and advances. Syst Sci Control Eng. 2024. https://doi.org/10.1080/21642583.2024.2394429.

40. Gao T, Tang Q, Li J, Zhang YI, Li Y, Zhang J. A particle swarm optimization with Lévy flight for service caching and task offloading in edge-cloud computing. IEEE Access. 2022;10(June):76636–47. https://doi.org/10.1109/ACCESS.2022.3192846.
41. Alqarni MA, Mousa MH, Hussein MK. Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing. J King Saud Univ-Comput Inf Sci. 2022;34(10):10356–64. https://doi.org/10.1016/j.jksuci.2022.10.026.
42. El Menbawy N, Ali HA, Saraya MS, Ali-Eldin AMT, Abdelsalam MM. Energy-efficient computation offloading using hybrid GA with PSO in internet of robotic things environment. J Supercomput. 2023;79(17):20076–115. https://doi.org/10.1007/s11227-023-05387-w.
43. You Q, Tang B. Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. 2021.
44. Alali S, Assalem A. Metaheuristics method for computation offloading in mobile edge computing: survey. J Adv Res Appl Sci Eng Technol. 2024;36(1):43–73. https://doi.org/10.37934/araset.36.1.4373.
45. Wang B, Wang C, Huang W, Song Y, Qin X. A survey and taxonomy on task offloading for edge-cloud computing. IEEE Access. 2020;8:186080–101. https://doi.org/10.1109/ACCESS.2020.3029649.
46. Keshavarznejad M, Rezvani MH, Adabi S. Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Comput. 2021;24(3):1825–53. https://doi.org/10.1007/s10586-020-03230-y.
47. Zakaryia SA, Ahmed SA, Hussein MK. Evolutionary offloading in an edge environment. Egypt Inform J. 2021;22(3):257–67. https://doi.org/10.1016/j.eij.2020.09.003.
48. Chakraborty S, Mazumdar K. Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing. J King Saud Univ-Comput Inf Sci. 2022;34(4):1552–68. https://doi.org/10.1016/j.jksuci.2022.02.014.
49. Wang X, Zhou Z, Chen H, Zhang Y. Task Offloading and Power Assignment Optimization for Energy-Constrained Mobile Edge Computing. Proc - 2021 9th Int Conf Adv Cloud Big Data CBD 2021. 2022;pp 302–307. https://doi.org/10.1109/CBD54617.2021.00058.
50. Nujhat N. Task offloading exploiting grey wolf optimization in collaborative edge computing. J Cloud Comput. 2024. https://doi.org/10.1186/s13677-023-00570-z.
51. Panigrahy SK, Emany H. A survey and tutorial on network optimization for intelligent transport system using the internet of vehicles. Sensors. 2023;23(1):1–30. https://doi.org/10.3390/s23010555.
52. Tang J, Liu G, Pan Q. A review on representative swarm intelligence algorithms for solving optimization problems: applications and trends. IEEE CAA J Autom Sin. 2021;8(10):1627–43. https://doi.org/10.1109/JAS.2021.1004129.
53. Computing MME. Energy Criticality Avoidance-Based Delay Minimization Ant, 2023.
54. De Souza AB, Rego PAL, Chamola V, Carneiro T, Rocha PHG, De Souza JN. A bee colony-based algorithm for task offloading in vehicular edge computing. IEEE Syst J. 2023;17(3):4165–76. https://doi.org/10.1109/JSYST.2023.3237363.
55. Babar M, Khan MS, Din A, Ali F, Habib U, Kwak KS. Intelligent computation offloading for IoT applications in scalable edge computing using artificial bee colony optimization. Complexity. 2021. https://doi.org/10.1155/2021/5563531.
56. Zhang D, Ge H, Zhang T, Cui YY, Liu X, Mao G. New Multi-Hop Clustering Algorithm for Vehicular Ad Hoc Networks. IEEE Trans Intell Transp Syst. 2019;20(4):1517–30. https://doi.org/10.1109/TITS.2018.2853165.
57. Dong S, Xia Y, Kamruzzaman J. Quantum Particle Swarm Optimization for Task Offloading in Mobile Edge Computing. IEEE Trans Ind Inf. 2022;19(8):9113–22. https://doi.org/10.1109/TII.2022.3225313.
58. Alhaizaey Y, Singer J, Michala AL. Optimizing Heterogeneous Task Allocation for Edge Compute Micro Clusters Using PSO Metaheuristic. 2022 7th Int Conf Fog Mob Edge Comput FMEC 2022. 2022;pp 1–8. https://doi.org/10.1109/FMEC57183.2022.10062755.
59. Yin G, Chen R, Zhang Y. Effective task offloading heuristics for minimizing energy consumption in edge computing, Proc. - IEEE Congr. Cybermatics 2022 IEEE Int. Conf. Internet Things, iThings 2022, IEEE Green Comput. Commun. GreenCom 2022, IEEE Cyber, Phys. Soc. Comput. CPSCom 2022 IEEE Smart Data, SmartD, pp. 243–249, 2022. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00069
60. Zhang Y, Liu Y, Zhou J, Sun J, Li K. Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing. Future Gener Comput Syst. 2020;112:148–61. https://doi.org/10.1016/j.future.2020.05.025.
61. Alfakih T, Hassan MM, Al-Razgan M. Multi-Objective Accelerated Particle Swarm Optimization with Dynamic Programing Technique for Resource Allocation in Mobile Edge Computing. IEEE Access. 2021;9:167503–20. https://doi.org/10.1109/ACCESS.2021.3134941.
62. Li S, Ge H, Gong H, Chen X, Tang R, Liu L. Computation Offloading Strategy for Improved Particle Swarm Optimization in Mobile Edge Computing, 2021.
63. Chen Y. MEC network resource allocation strategy based on improved PSO in 5G communication network. Int J Semant Web Inf Syst. 2023. https://doi.org/10.4018/IJSWIS.328526.
64. Li R, Lim CS, Rana ME, Zhou X. A trade-off task-offloading scheme in multi-user multi-task mobile edge computing. IEEE Access. 2022;10:129884–98. https://doi.org/10.1109/ACCESS.2022.3228403.
65. Yuan H. ScienceDirect Optimization with Optimization with Swarm Optimization Optimization with Hybrid Optimization with Optimization with in Smart Edge, 2021.
66. Huynh LNT, Pham QV, Pham XQ, Nguyen TDT, Hossain MD, Huh EN. Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach. Appl Sci. 2020;10(1):1–17. https://doi.org/10.3390/app10010203.
67. Song H, Gu B, Son K, Choi W. Joint Optimization of Edge Computing Server Deployment and User Offloading Associations in Wireless Edge Network via a Genetic Algorithm. IEEE Trans Netw Sci Eng. 2022;9(4):2535–48. https://doi.org/10.1109/TNSE.2022.3165372.
68. Al-Habob AA, Dobre OA, Garcia Armada A. Sequential task scheduling for mobile edge computing using genetic algorithm. 2019 IEEE Globecom Work GC Wkshps 2019 - Proc. 2019. https://doi.org/10.1109/GCWkshps45667.2019.9024374.
69. Wang H. Collaborative task offloading strategy of UAV cluster using improved genetic algorithm in mobile edge computing. J Robot. 2021. https://doi.org/10.1155/2021/3965689.

70. Mashhadi F, Monroy SAS, Bozorgchenani A, Tarchi D. Optimal auction for delay and energy constrained task offloading in mobile edge computing. Comput Netw. 2020;183(June):107527. https://doi.org/10.1016/j.comnet.2020.107527.
71. Jain M, Saihjpal V, Singh N, Singh SB. An overview of variants and advancements of PSO algorithm. Appl Sci. 2022;12(17):1–21. https://doi.org/10.3390/app12178392.
72. Li J, Majeed APA, Lefevre P. Halfway Escape Optimization: A Quantum-Inspired Solution for General Optimization Problems, pp. 1–30, 2024, [Online]. Available: http://arxiv.org/abs/2405.02850
73. Vink JC. Particle Trajectories for Quantum Field Theory, pp. 1–25, 2017.
74. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput. 2002;6(1):58–73. https://doi.org/10.1109/4235.985692.
75. Yang Z, Wu A. A non-revisiting quantum-behaved particle swarm optimization based multilevel thresholding for image segmentation. Neural Comput Appl. 2020;32(16):12011–31. https://doi.org/10.1007/s00521-019-04210-z.
76. Chen S. Quantum-behaved particle swarm optimization with weighted mean personal best position and adaptive local attractor. Information. 2019. https://doi.org/10.3390/info10010022.
77. Huang H, Ye Q, Zhou Y. Deadline-aware task offloading with partially-observable deep reinforcement learning for multi-access edge computing. IEEE Trans Netw Sci Eng. 2022;9(6):3870–85. https://doi.org/10.1109/TNSE.2021.3115054.
78. Li A, Li L, Yi S. Computation offloading strategy for IoT using improved particle swarm algorithm in edge computing. Wirel Commun Mob Comput. 2022. https://doi.org/10.1155/2022/9319136.

Publisher's note