Journal of Grid Computing (2025) 23:4
https://doi.org/10.1007/s10723-024-09790-2

RESEARCH

Efficient Resource Allocation Algorithm for Maximizing Operator Profit in 5G Edge Computing Network

Jing Liu · Yuting Huang · Chunhua Deng · Longxin Zhang · Cen Chen · Keqin Li

Received: 1 August 2024 / Accepted: 12 December 2024

Abstract Resource allocation in edge computing is a research hotspot and difficulty in academia and Industry. The nature like urgency and the priority of tasks are not taken into account, which is adverse to obtain a good solution. Meanwhile, 5G has the characteristics of higher network speed, high reliability, low latency, and low-power massive connections. In this article, we present a novel algorithm to solve the multi-objective resource allocation problem in 5G edge computing (EC) network, the objective is to maximize the operator profit and minimize the total completion time of tasks with priorities from the perspective of service operators under time and workload constraints. The algorithm is based on the beluga whale optimization algorithm, and it utilizes three methods to update the positions of beluga whales by swimming, predating, and migrating. In addition, to enhance the ability to escape from local optima during searching the best beluga whale position, it uses centroid information to improve the process of searching the optimal position, and adds the mutation operation in the process of position updating. Simulated results show that the proposed algorithm is high efficient in terms of reducing total task completion time and improving the revenue for operators, compared with existing strategies. For example, our algorithm reduces the time by 1.87% and improves the profit by 10.47% for 80 tasks in comparison with MOPSO.

Keywords Edge computing · Multi-objective optimization · Revenue maximization

Jing Liu and Yuting Huang contributed equally to this work.

J. Liu · Y. Huang · C. Deng
College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430081, China
e-mail: luijing_cs@wust.edu.cn

Y. Huang
e-mail: huangyuting@wust.edu.cn

J. Liu · Y. Huang · C. Deng (✉)
Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430081, China
e-mail: dchzx@wust.edu.cn

L. Zhang
College of Computer Science, Hunan University of Technology, Zhuzhou, China

C. Chen
School of Future Technology, South China University of Technology, Guangzhou 510641, China

Shenzhen Research Institute of Hunan University, Guangzhou 518052, China

K. Li
Department of Computer Science, State University of New York, New Paltz, NY, USA

1 Introduction and Background

With the rapid development of 5G communication technology, various new applications emerge one after another. Meanwhile, users have an increasing demand for computationally intensive and latency sensitive applications. According to a Cisco report, global mobile device usage is expected to reach 12.3 billion units by 2022. The global smart technology industry is developing rapidly, and the number of devices such as smartwatches, mobile phones, and smart cars is growing exponentially. The tasks generated by mobile devices require a large amount of computing resources to process, especially for applications such as virtual reality (VR) and smart healthcare that have strict time requirements. The traditional cloud computing model often struggles to meet these real-time requirements, as large-scale data transmission consumes a significant amount of bandwidth resources, resulting in longer transmission times and increased network bandwidth pressure. As a computing paradigm with broad application prospects, edge computing pushes the processing and storage of computing tasks from the traditional cloud computing center to the edge of the network to respond to user requests faster and reduce network latency.

A classic EC system usually includes edge servers, base stations and user devices. Tasks generated on user devices can be offloaded to edge servers for execution or executed locally. Although edge servers have more computing and storage resources than user devices, they are still limited. When different resources are allocated to computing tasks, it will lead to different delays, energy consumption, costs, profits and so on. In order to meet different needs, it is very important to efficiently allocate resources in EC. Edge computing resource allocation refers to the allocation of appropriate computing resources for a group of computing tasks. It is a research hotspot and difficulty in academia and industry.

There are plenty of researches on resource allocation in edge computing. Most of researches on resource allocation in edge computing are from users' perspective while a few of work are from service operators' perspective. Huang et al. [1] propose approaches to deal with the communication and computation resource allocation problem in mobile edge computing (MEC) having multiple mobile users and edge servers, aiming to to maximize the user profit in terms of utility. Zhang et al. [2] propose algorithms to solve the resource allocation and task offloading problem considering resource constraints in mobile edge computing with multiple servers. The problem has four objectives: maximize trust metrics, minimize task time delay, maximize user experience utility, and minimize server energy consumption. Both [1] and [2] do not consider the time constraint. However, for latency-sensitive applications such as autonomous driving, it is significant to take deadline into consideration. If the deadline can not be satisfied, it will cause traffic accidents. Peng et al. [3] design an intelligent approach to solve the single-objective resource allocation and computation offloading problem in MEC, and the goal is to maximize the resource utilization and minimize the total time, total energy consumption, as well as load balancing under multiple constraints like time and energy. All above work are from the perspective of users. In fact, operators hopes to attain as much profit as possible by providing services for users with ensuring the quality of service (QoS). Moreover, the nature like urgency and the priority of tasks are not taken into account, which is adverse to obtain a good solution. From service operators' perspective, Chen et al. [4] present an online control method to figure out the resource allocation problem in EC considering time and capacity constraints with the objective of maximizing the profit of operators. All above work do not consider task priority. However, different tasks have different nature and emergency. Taking task priority into account will obtain better resource allocation scheme and optimization objective values. Sharif et al. [5] present an efficient adaptive resource allocation algorithm to maximize the utilization of resource and the number of user requests in EC considering task priority, time and resource constraints. Although lots of researches on resource allocation in EC have been done so far, there exist some import problems that are not be solved.

Moreover, 5G has the characteristics of higher network speed, high reliability, low latency, and low-power massive connections. The 5G network adopts a completely new network architecture, separating control plane functions from user plane functions and introducing UPF. As a key user plane function in 5G, UPF controls the forwarding and routing of data. Sinking UPF to the network edge can decrease transmission latency, achieve local data flow diversion, and alleviate data transmission pressure on the core network. Currently, there are only a few studies on UPF. Qian et al. [6] study the computation and storage resource allocation of UPF based on isolation performance in dedicated 5G networks, aiming to minimize costs and maximize resource utilization. Different resource allocation schemes in 5G EC network will generate different results, such as different task completion time, energy consumption and revenue. Thus, it is considerably important to allocate resources efficiently. What is more, minimizing task completion time and maximizing the operator revenue are two conflict objectives so that it is challenging to find a high efficient approach to simultaneously optimize them.

Different from previous work, in this article, we study the problem of resource allocation in EC for 5G network considering task priorities, the time and workload constraints, and the objective is to minimize total task completion time and maximize the operator profit from the perspective of service operators. Meta-heuristic algorithms are widely used to solve optimization problems due to their many advantages such as efficiency, flexibility, scalability, and robustness [7–9]. Thus, we propose a meta-heuristic method, the improved multi-objective beluga whale optimization (IMOBWO) algorithm to work out the problem. The algorithm uses four means to change the positions of beluga whales: whale swimming, whale predating, whale migrating, and whale mutation. In addition, we devise three sets of experiments to test the algorithm, and the simulation results show that our proposed algorithm is effective and efficient.

The main contributions of this article include:

- We formulate the edge computing resource allocation problem as a multi-objective problem, aiming to minimize the total task completion time and maximize the profit of operators under time and workload constraints from the perspective of service providers. Each task has a priority.
- We present a novel meta-heuristic algorithm named IMOBWO to solve the formulated problem based on the beluga whale optimization (BWO) algorithm. In response to the problem of being easily trapped in local optima during the searching process of the best beluga whale position, this paper uses centroid information to improve the process of searching the optimal position, and adds the mutation operation in the process of updating the beluga whale position to enhance the ability to escape from local optima.
- We design simulation experiments to verify the effectiveness and efficiency of our proposed algorithm. Simulation results demonstrate IMOBWO can achieve good total task completion time, operator profit, and load balancing.

The remaining part of this article is arranged as follows. Section 2 outlines the relative work. Section 3 addresses the models and the studied optimization problem. Section 4 introduces the proposed IMOBWO algorithm. Section 5 conducts experiments to test the effectiveness and efficiency of our presented algorithm. Section 6 concludes this article by briefly summarizing the whole work.

2 Related Work

In recent decades, researchers have conducted a great many of work on resource allocation in edge computing. These work either have different objectives or constraints, and are briefly divided into two categories. one is from users' perspective, and the majority work belong to this category; the other is from service providers' perspective.

2.1 From the Perspective of Users

Xue and Guan [10] propose methods to solve the resource allocation problem in MEC having multiple mobile users and an edge server, and the goal is to minimize the cost that users buy computing resources. Li et al. [11] design a resource allocation method based on pricing via repeated bidding to maximize the social welfare considering capacity and resource constraints in a three-tier and heterogeneous EC system. Kumar et al. [12] present a game-theory based resource allocation method to minimize the total user cost in an edge computing environment with satisfying the QoS requirement under resource constraints. All these work do not consider the time constraint, and are not suitable for latency-sensitive applications.

Researches [13–15] design efficient algorithms to tackle the resource allocation and task offloading problem in MEC with a single edge server, and the goal is to minimize the total energy consumption under multiple constraints like time and resource. Cao et al. [16] propose a two-level alternating iterative method to tackle the resource allocation and task partial offloading problem in a multiple vehicle users and servers EC scenario, and the objecitve is to minimize the sum of the weight sum of energy consumption and system latency of all tasks under multiple constraints such as time, energy, cost, capacity and so on. Wang et al. [17] present an efficient resource allocation approach based on digital twin assisted end-edge-cloud collaborative computing to minimize the system cost which are the the sum of the weight sum of energy consumption and latency of all tasks under time constraint. An et al. [18] present an efficient resource allocation approach to minimize the sum of the weight sum of energy consumption and latency of all tasks under time and resource constraints in MEC with multiple edge servers. Šlapak et al. [19] present a novel method to maximize the ratio of the failed tasks to the total generated tasks of the resource allocation problem in MEC considering time and budget constraints. Wang et al. [20] present task offload and resource allocation methods to solve two optimization problems in a collaborative edge-cloud computing environment with multiple edge servers. One is to maximize the total number of served mobile devices under time and resource constraints; the other is to minimize the total energy consumption under time and resource constraints. Saleem et al. [21] propose efficient algorithms to minimize the total latency of tasks for the resource allocation and task assignment problem in cooperative MEC under time and energy constraints. Ansere et al. [22] present an efficient quantum Deep Reinforcement Learning method to solve the resource allocation problem in a Internet-of-Things system based on MEC, and the goal is to maximize the system efficiency under time, energy as well as resource constraints. Huang et al. [23] design a deep reinforcement learning method to deal with the resource allocation and task offloading problem in an EC-based vehicular network to maximize the utility (i.e., the weight sum of energy cost and revenue for vehicles processing tasks) under time and resource constraints.

All above work do not consider task priority, which may not make full use of the nature and emergency of tasks to achieve better optimization objective values. Sharif et al. [24] propose a method to solve the resource allocation and task scheduling problem in EC considering task priority with the objective of minimizing the bandwidth cost and the total task processing time. They do not consider time constraint. Wang et al. [25] present a meta-heuristic method to solve the resource allocation and task offloading problem in MEC. The objective is to minimize the weight sum of energy consumption and latency of all vehicle tasks with priorities under time and resource constraints. Different from our work, their studied problem is single-objective and with distinct objective.

2.2 From the Perspective of Service Providers

Habiba et al. [26] propose methods to work out the resource allocation and task offloading problem in an MEC system considering resource constraints, aiming to maximize the sum valuation finished by the computing resources of servers. Fan et al. [27] present an efficient algorithm to solve the pricing of computing resources and resource allocation problem for mobile blockchain, aiming to maximize the revenue of edge server operators under resource constraint. Both these two work do not consider time constraint. Nguyen et al. [28] present an efficient algorithm to maximize the resource utility in terms of revenue of services by reasonably pricing edge nodes for the resource allocation problem in EC under time and budget constraints. Yuan and Zhou [29] present a meta-heuristic method to solve the resource allocation and computation offloading problem in a cloud and edge collaborative computing environment, and the goal is to maximize the revenue of the system provider under multiple constraints like time, energy and resource. Falsafain et al. [30] present a branch-and-price method to solve the allocation of spectrum holes, and the aim is to maiximize the utilization of total spectrum. Schieber et al. [31] propose three algorithms to solve the job schedule problem with taking the theoretical aspects into account, and the goal is to maximize throughput under time constraint. Jamil et al. [32] design a reinforce learning-based eco-driving approach to deal with traffic flow consisting of autonomous vehicles and human-driven vehicles, and the objective is to minimize the total vehicle delay and fuel energy consumption. All these work do not consider task priority.

Comparing with above work, we study the problem of resource allocation in EC for 5G network, and the goal is to minimize total task completion time and maximize the operator profit considering task priorities, the time and workload constraints.

3 Models and Problem Definition

In this section, we will present the models and studied problem in our paper.

3.1 System Model

The targeted 5G edge computing system considered in this paper consists of three layers: industrial park, edge server, and cloud center. The industrial park layer contains multiple unmanned factories, a base station, and a user plane function (UPF) device. The edge server layer contains multiple edge servers, and the cloud center includes a cloud server. Tasks generated by industrial parks are first sent to UPF devices through base stations, and then offloaded to edge servers for execution. Communication between an industrial park and a base station is wireless communication; communication between a base station and a UPF device is wired communication; communication between a UPF device and an edge server is wired; communication between an edge server and a cloud server is wired; communication between UPF devices is wired. Figure 1 gives a system model example of 5G edge computing.

Denote S and U to be the set of all edge servers and the set of UPF devices, respectively. That is, $S = \{s_1, s_2, ..., s_m\}$ and $U = \{u_1, u_2, ..., u_n\}$, where m is the total number of edge servers and n is the total number of UPF devices. We use the notation $a_{i,j}$ to represent the connection between edge servers and UPF devices, $1 \le i \le n, 1 \le j \le m$. If there is a path between UPF u_i and edge server s_j , then $a_{i,j} = 1$; on the contrary, $a_{i,j} = 0$. $W = (w_{i,j})_{n\times m}$ is a distance matrix that represents the hop distance between nodes, and defined as follows:

$$w_{i,j} = \begin{cases} k, & \text{if } a_{i,j} = 1 \\ \infty, & \text{if } a_{i,j} = 0 \end{cases} \tag{1}$$

where the integer k is positive.

Different industrial parks generate multiple computing tasks. To meet the computing requirements of different businesses, each computing task is assigned a

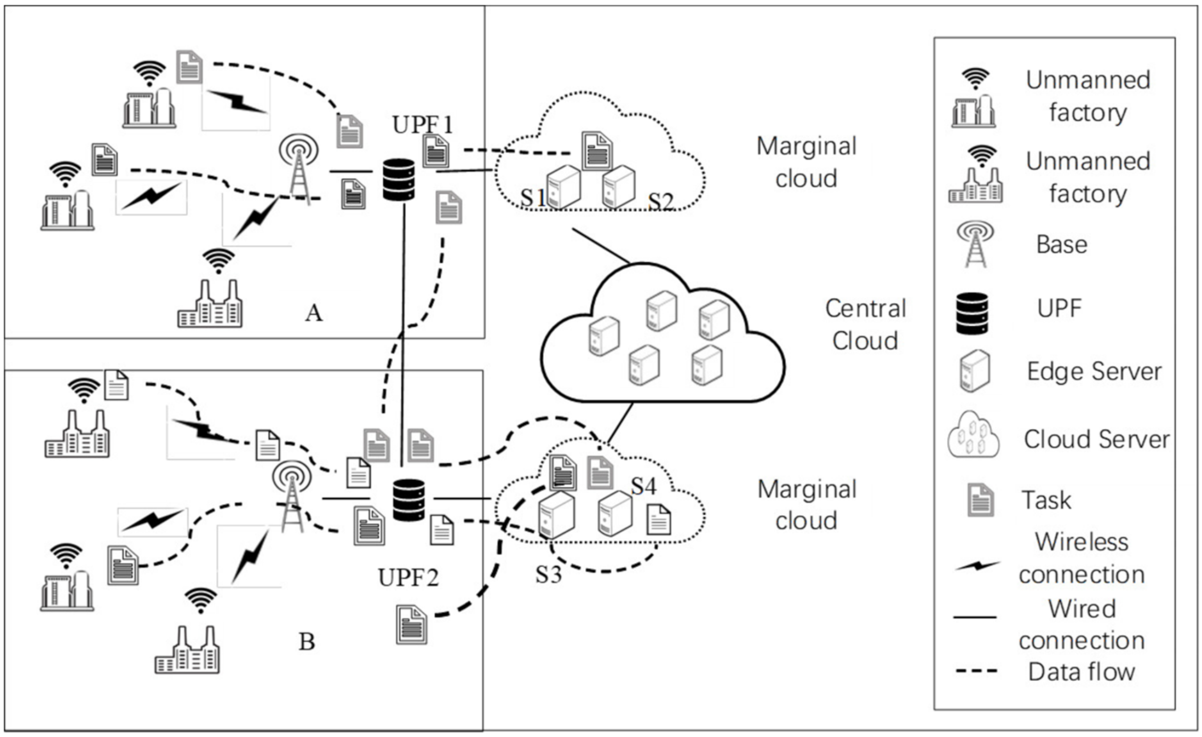


Fig. 1 An example of 5G edge computing system model

corresponding attribute, and represented by a quadruple $t = \left(t^{size}, t^{ide}, t^{max}, t^{val}\right)$, where t^{size} represents the size of the task, in MB as the basic unit, t^{ide} represents the ideal completion time of the task, t^{max} represents the maximum completion time of the task, and t^{ide} represents the reward given for completing the task.

3.2 Edge Computing Resource Allocation Representation

As it is shown in Fig. 1, different industrial parks generate many computing tasks in each time slot. These tasks are transmitted through UPF devices to various edge servers for processing. Let q_i be the number of tasks transferred from industrial park i, and $x_{i,j}$ be the edge server which task $t_{i,j}$ is offloaded to for processing, where $t_{i,j}$ is the jth task generated from park i, $1 \le i \le n$, $1 \le j \le q_i$, $1 \le k \le m$. We have

$$x_{i,j} = \begin{cases} k, & \text{if the task } t_{i,j} \text{ is offloaded to} \\ & \quad \text{execute on edge server } k, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

3.3 Computing Time

When a task is offloaded to an edge server for processing, the completion time of the task is determined by the transmission delay, network latency, waiting time, and processing delay. The completion time denoted by $d_{i,j}^{fin}$ of task $t_{i,j}$ can be expressed as

$$d_{i,j}^{fin} = d_i^{net} + d_{i,j}^{tra} + d_{i,j}^{wait} + d_{i,j}^{proc}, \tag{3}$$

where d_i^{net} represents the current network delay in the park i, and according to the reference [33], it is calculated by

$$d_i^{net} = \frac{d^{ide}}{1-\theta}, \tag{4}$$

$$\theta = \frac{d_i^{pass}}{\tau}, \tag{5}$$

where d^{ide} represents the delay when the network is idle, and θ is the utilization rate of the current network. τ represents a time slot, and d_i^{pass} indicates the time for data to pass through the network in park i. The ratio of d_i^{pass} to τ reflects the magnitude of utilization. The larger the d_i^{pass} value, the busier the network, which in turn leads to an increase in network latency d_i^{net}.

$d_{i,j}^{wait}$ represents the queue waiting time for task $t_{i,j}$, and is calculated as follows:

$$d_{i,j}^{wait} = \alpha d_{x_{i,j}}^{Ewt}, \tag{6}$$

where α is the priority coefficient of task $t_{i,j}$. According to the quantification method in reference [34], $\alpha = 0.5 \times \left(t_{i,j}^{size}/c + t_{i,j}^{val}/v - 1\right) \times \left(t_{i,j}^{size}/c + t_{i,j}^{val}/v - 2\right) + t_{i,j}^{val}/v$. v represents the value obtained from processing 1MB sized computing task. c represents the size of the basic computational task, and here $c = 1MB$. For tasks with the same amount of computation, the larger the task's value, the higher its ranking. $d_{x_{i,j}}^{Ewt}$ represents the average waiting time of task $t_{i,j}$ on edge server $s_{x_{i,j}}$. According to queuing theory [35], we have

$$d_{x_{i,j}}^{Ewt} = \frac{1}{\mu(s_{x_{i,j}}) - \gamma}. \tag{7}$$

γ represents the task arrival rate, which is the number of tasks that reach the edge server $s_{x_{i,j}}$ per unit of time. $\mu\left(s_{x_{i,j}}\right)$ represents the service rate, which is the reciprocal of the average service time per unit of task volume on edge server $s_{x_{i,j}}$. The stronger the processing power of the edge server $s_{x_{i,j}}$, the larger the value $\mu\left(s_{x_{i,j}}\right)$. As a result, the average waiting time $d_{x_{i,j}}^{Ewt}$ on edge server $s_{x_{i,j}}$ is smaller.

$d_{i,j}^{proc}$ represents the processing time of the task $t_{i,j}$, and it is calculated as follows:

$$d_{i,j}^{proc} = \frac{t_{i,j}^{size}}{f_{x_{i,j}}}, \tag{8}$$

where $f_{x_{i,j}}$ represents the processing capacity of edge server $s_{x_{i,j}}$.

$d_{i,j}^{tra}$ represents the transmission time of task $t_{i,j}$, and it is calculated by

$$d_{i,j}^{tra} = \begin{cases} d_{i,u_i}^{tra} + d_{u_i,s_{x_{i,j}}}^{tra}, & s_{x_{i,j}} \text{ is in current park,} \\ d_{i,u_i}^{tra} + d_{u_i,u_{i_0}}^{tra} + \sum_{s=0}^{l-1} d_{u_{i_s},u_{i_{s+1}}}^{tra} + d_{u_l,s_{x_{i,j}}}^{tra}, & \\ \quad \text{otherwise,} \end{cases} \tag{9}$$

where d^{tra}_{i,u_i} represents the transmission time of task $t_{i,j}$ from the terminal device to UPF device u_i, $d^{tra}_{u_i,s_{x_{i,j}}}$ represents the transmission time of task $t_{i,j}$ from UPF u_i to edge server $s_{x_{i,j}}$, $d^{tra}_{u_i,u_l}$ represents the transmission time of task $t_{i,j}$ from UPF u_i to UPF u_l. The transmission time $d^{tra}_{x,y}$ for task $t_{i,j}$ between different nodes x and y can be expressed as:

$$d^{tra}_{x,y} = \begin{cases} \frac{t^{size}_{i,j}}{Bwan}, t_{i,j} \text{ is transmitted by wires,} \\ \frac{t^{size}_{i,j} w_{x,y}}{Blan}, t_{i,j} \text{ is transmitted wirelessly,} \end{cases} \quad (10)$$

where $Bwan$ and $Blan$ represent the bandwidth sizes for wireless and wired transmission, respectively.

3.4 Quality of Service

The quality of service (QoS) provided by the operator is closely related to the completion time of tasks [36]. This article measures the quality of service of the operator in terms of the total time taken to complete tasks. The total time T^{tol} for completing all tasks can be expressed as

$$T^{tol} = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{q_i} d^{fin}_{i,j}. \quad (11)$$

3.5 Energy Consumption of Edge Server

The energy consumption of servers is affected by multiple factors, such as the status of the memory, CPU, network card, and so on, where the CPU is the most significant energy-consuming device [37]. According to the calculation method in reference [77], the energy consumption e_k of edge server s_k from time t_1 to t_2 is calculated by

$$e_k = \int_{t_1}^{t_2} p_k \, dx \quad (12)$$

$$p_k = p^{ide} + (p^{max} - p^{ide}) \times \varphi_k \quad (13)$$

$$\varphi_k = \frac{\omega(s_k)}{\omega_{max}} \quad (14)$$

$$\omega(s_k) = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{q_i} (x_{i,j} = k) \quad (15)$$

where p_k is the power of server s_k, p^{ide} represents the power when s_k is idle, and p^{max} represents the power when s_k is fully operational. φ_k represents the utilization efficiency of s_k, and its value can be derived from formula (14). $\omega(s_k)$ represents the workload of s_k, and ω_{max} represents the maximum workload of s_k.

The total energy consumption E^{tol} of all edge servers is calculated as follows

$$E^{tol} = \sum\nolimits_{k=1}^{m} e_k. \quad (16)$$

The total cost m^{cost} caused by the energy consumption of edge servers is computed by

$$m^{cost} = p^e \times E^{tol}, \quad (17)$$

where p^e represents the electricity price per kilowatt-hour.

3.6 Revenue for Operators

To better simulate the revenue of operators, we adopts a payment mechanism with penalties. The actual payment $p^{real}_{i,j}$ obtained by the edge node processing task $t_{i,j}$ is calculated by

$$p^{real}_{i,j} = \begin{cases} t^{val}_{i,j}, & d^{fin}_{i,j} \le t^{ide}_{i,j} \\ t^{val}_{i,j} - \sigma(d^{fin}_{i,j} - t^{ide}_{i,j}), & t^{ide}_{i,j} < d^{fin}_{i,j} \le t^{max}_{i,j} \\ 0, & d^{fin}_{i,j} > t^{max}_{i,j} \end{cases} \quad (18)$$

$$\sigma = \frac{t^{val}_{i,j}}{t^{max}_{i,j} - t^{ide}_{i,j}}, \quad (19)$$

where σ is the proportion by which the reward is reduced represents the actual reward obtained by edge servers. When $d^{fin}_{i,j} < t^{ide}_{i,j}$, operators receive full reward; when $t^{ide}_{i,j} < d^{fin}_{i,j} \le t^{max}_{i,j}$, operators' reward will be reduced; when $d_{i,j}^{fin} > t_{i,j}^{max}$, the service deteriorates severely and operators will not receive any reward.

The total revenue ψ from processing tasks can be represented as

$$\psi = m^{in} - m^{cost} \tag{20}$$

$$m^{in} = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{q_i} p_{i,j}^{real} \tag{21}$$

where m^{in} is the total revenue of operators.

3.7 Problem Formulation

Given the 5G EC system previously referred, the problem is described in mathematical language as follows

$$\begin{aligned} P: \ & Minimize\ (-\psi,\ T^{tol}) = (p^e \sum\nolimits_{k=1}^{m} e_k \\ & - \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{q_i} p_{i,j}^{real}, \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{q_i} d_{i,j}^{fin}), \quad (22) \\ \text{s.t.} \ & C_1: \sum\nolimits_{k=1}^{m} (x_{i,j} = k) = 1, 1 \le i \le n, 1 \le j \le q_i, \\ & C_2: d_{i,j}^{fin} \le t_{i,j}^{max}, 1 \le i \le n, 1 \le j \le q_i, \\ & C_3: \omega(s_k) \le \omega_{max}, 1 \le k \le m, \end{aligned}$$

where constraint C_1 ensures that a task can be only assigned to one edge server, C_2 is the completion time constraint of tasks, and constraint C_3 indicates that the workload of an edge server does not outstrip its maximum load. That is, the objective is to minimize the total task completion time and maximize the operator profit under time and workload constraints.

4 The Proposed Algorithm

Inspired by the BWO algorithm which was first proposed by Zhong et al. [38] in 2022, we design the improved multi-objective BWO algorithm to work out the problem above. We will present the details of the IMOBWO algorithm below.

4.1 Constructions of the Beluga Whales and the Fitness Function

For our studied problem, its solution is the edge server assignment of all tasks generated by all industrial parks. Therefore, each beluga whale is set to be the edge server assignment of all tasks. We consider all tasks generated in one time slot. Assume that there are total d tasks, and then the beluga whale i can be expressed as

$$X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d}), \tag{23}$$

where $x_{i,j} \in \{1, 2, ..., m\}$ represents the edge server assignment of task i. Also, X_i is called the position of the beluga whale i.

To evaluate the quality of each solution, we use (11) and (20) to compute the fitness function value. That is, the fitness function value of the solution X_i is expressed as

$$f(X_i) = (f_1(X_i), f_2(X_i)), \tag{24}$$

where $f_1(X_i)$ and $f_2(X_i)$ are computed by (11) and (20), respectively.

4.2 The Method to Compute the Best Beluga Whale Position in a Beluga Whale Population

Suppose that there are h beluga whales in a beluga whale population, and we will introduce how to obtain the best beluga whale in a given beluga whale population. First, apply the non-dominated sorting strategy for the multi-objective problem to get the optimal solution set. Second, calculate the number of beluga whales in the optimal solution set, denoted by k. Third, calculate the centroid $C = (f_1(X_c), f_2(X_c))$ of of the fitness value for the optimal solution set by

$$\begin{aligned} C &= (f_1(X_c), f_2(X_c)) \\ &= \left(\frac{1}{k} \sum\nolimits_{i=1}^{k} f_1(X_i), \frac{1}{k} \sum\nolimits_{i=1}^{k} f_2(X_i) \right), \end{aligned} \tag{25}$$

Finally, compute the best beluga whale position X_{best} by

$$X_{best} = \min\{D(X_1), D(X_2), ..., D(X_k)\}, \tag{26}$$

$$D(X_i) = \sqrt{(f_1(X_i) - f_1(X_c))^2 + (f_2(X_i) - f_2(X_c))^2}, \tag{27}$$

That is, the beluga whale position that is closest to the centroid is the best beluga whale position.

Algorithm 1 shows the strategy for computing the the best beluga whale position in a beluga whale population.

Algorithm 1 The method to compute the best beluga whale position.

```
Require: The size of the beluga whale population $h$, the positions $X_1^t, X_2^t, ..., X_h^t$ of beluga whales in the population for the $t$th iteration.
Ensure: The best beluga whale position $X_{best}^t$.
1: for $i \leftarrow 1$ to $h$ do
2:     Calculate the fitness value $f(X_i^t)$ by (24);
3: end for
4: Calculate the optimal solution set by the non-dominated sorting strategy;
5: Calculate the number of whales in the optimal solution set;
6: Calculate the centroid of the fitness value of the optimal solution set by (25);
7: Calculate the best beluga whale position by (26);
```

4.3 Ways for Updating Positions of Beluga Whales

Beluga whales can change their positions in three ways: swimming, predation, and migration. Which way is taken to update the positions is determined by the balance factor B_f and the fall factor W_f. They are computed by

$$B_f = B_0\left(1 - \frac{t}{2T_{max}}\right) \tag{28}$$

$$W_f = 0.1 - \frac{0.05t}{T_{max}}, \tag{29}$$

where B_0 is a randomly generated value from the interval (0,1), t is the current iteration number, as well as T_{max} represents the maximum iteration number.

4.3.1 Position Updation by Swimming

When beluga whales are swimming, their positions are changed by

$$x_{i,j}^{t+1} = x_{i,p_j}^t + \left(x_{r,p}^t - x_{i,p_j}^t\right)(1 + r_1)\sin(2\pi r_2), \quad \forall j \in [1, d] \text{ and } j\%2 = 1, \tag{30}$$

$$x_{i,j}^{t+1} = x_{i,p_j}^t + \left(x_{r,p}^t - x_{i,p_j}^t\right)(1 + r_1)\cos(2\pi r_2), \quad \forall j \in [1, d] \text{ and } j\%2 = 0, \tag{31}$$

where $x_{i,j}^{t+1}$ is the position of the beluga whale i in the dimension j in the iteration time $t + 1$. p_j, r and p are random integers, $p_j \in [1, d]$, $r \in [1, h]$, $p \in [1, d]$. r_1 and r_2 are both random numbers from the interval (0, 1).

Algorithm 2 is the process of updating the positions of whales during their swimming phase.

Algorithm 2 The method of updating positions by swimming.

```
Require: $h, X_1^t, X_2^t, ..., X_h^t$, the individual dimension $d$.
Ensure: $X_1^{t+1}, X_2^{t+1}, ..., X_h^{t+1}$ after the $t$th iteration.
1: Generate integer $r$ from $[1, h]$;
2: Generate $r_1, r_2$ from $(0, 1)$;
3: for $i \leftarrow 1$ to $h$ do
4:     Produce integers $p$ and $p_j$ from $[1, d]$;
5:     Update the position of beluga whale $i$ by (30) and (31);
6: end for
```

4.3.2 Position Updation by Predation

When beluga whales prey, their positions are changed by

$$X_i^{t+1} = r_3 X_{best}^t - r_4 X_i^t + C_1 L_F\left(X_r^t - X_i^t\right), \tag{32}$$

where X_i^{t+1} is the position of beluga whale i in the iteration time $t + 1$. $r \in [1, h]$, is a random integer. X_{best}^t is the best whale position in the tth iteration. r_3 and r_4 are both random numbers from the interval (0, 1). C_1 is the random jump intensity, used to measure Levy flight intensity, and L_F is the Levy flight function. They are computed by

$$C_1 = 2r_4\left(1 - \frac{t}{T_{max}}\right), \tag{33}$$

$$L_F = 0.05 \times \frac{u \times \sigma}{|v|^{1/\beta}}, \tag{34}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\pi\beta/2)}{\Gamma((1+\beta)/2) \times \beta \times 2^{(\beta-1)/2}}\right)^{1/\beta}, \tag{35}$$

where v and u are random numbers following normal 0-1 distributions. β is a constant.

Algorithm 3 is the process of updating positions of white whales during their hunting phase.

Algorithm 3 The method of updating positions by predation.

```
Require: $h, X_1^t, X_2^t, ..., X_h^t, X_{best}^t$.
Ensure: $X_1^{t+1}, X_2^{t+1}, ..., X_h^{t+1}$.
1: Generate $r_3, r_4$ from $(0, 1)$;
2: Calculate $C_1$ by (33);
3: Calculate $L_F$ by (34) and (35);
4: for $i \leftarrow 1$ to $h$ do
5:     Generate integer $r$ from $[1, h]$;
6:     Calculate the position of beluga whale $i$ by (32);
7: end for
```

4.3.3 Position Updation by Migration

When white whales migrate, the updation of their positions are computed by

$$X_i^{t+1} = r_5 X_i^t - r_6 X_r^t + r_7 C_{step}, \tag{36}$$

$$C_{step} = (u_b - l_b)\exp(-C_2 t/T_{max}), \tag{37}$$

$$C_2 = 2W_f \times h, \tag{38}$$

where r_7, r_6 and r_5 are random values from the interval $(0, 1)$. C_{best} is the stride of a falling whale. C_2 is the step factor associated to the fall probability of whale and population size. l_b and u_b are the lower and upper bounds of the variables in the problems, respectively.

Algorithm 4 describes the process for updating positions of beluga whales during their migration phase.

In response to the problem of falling into local optima during the optimization process of beluga whales, this paper adds a mutation operation during

Algorithm 4 The method of updating positions by migration.

```
Require: $h, X_1^t, X_2^t, ..., X_h^t, l_b, u_b$.
Ensure: $X_1^{t+1}, X_2^{t+1}, ..., X_h^{t+1}$.
1: Generate $r_5$, $r_6$, and $r_7$ from $(0, 1)$;
2: Generate integer $r$ from $[1, h]$;
3: Calculate $C_2$ by (38);
4: for $i \leftarrow 1$ to $h$ do
5:     Calculate $C_{step}$ according to formula (37);
6:     Calculate the position of beluga whale $i$ by (36);
7: end for
```

the search process for the best position to strengthen the ability of algorithm to escape local optima. Specifically, the mutation is operated as follows: (1) calculate the mutation factor p_t; (2) when the randomly generated mutation probability is greater than the mutation factor, calculate the number of beluga whales for mutation and perform mutation operation on the mutated whales.

The mutation factor p_t and the number of mutated whales N_t are computed by

$$p_t = p_0 \times \frac{(T_{max} - t)}{T_{max}}, \tag{39}$$

$$N_t = l_0 \times h \times \left(1 - \frac{t}{T_{max}}\right), \tag{40}$$

where p_0 is the mutation rate and l_0 is the scaling factor.

When the selected individual beluga whale i undergoes a mutation, its position is updated as follows

$$x_{i,q}^{t+1} = l_b + r_5(u_b - l_b), \tag{41}$$

where $x_{i,q}^{t+1}$ represents the mutation of offspring i in a random dimension q, $q \in [1, d]$.

Algorithm 5 shows the process of mutation operation of beluga whales.

4.4 The IMOBWO Algorithm

Algorithm 6 shows how the improved multi-objective beluga whale optimization algorithm works. Since the time complexities of Algorithms 1 to 5 are $O(hnqm + h^2nq)$, $O(hnq)$, $O(hnq)$, $O(hnq)$, and $O(hnqm)$, the

Algorithm 5 The mutation operation.

```
Require: $h$, $X_1^t$, $X_2^t$, ..., $X_h^t$, $u_b$, $l_b$, $p_0$, and $l_0$.
Ensure: $X_1^{t+1}$, $X_2^{t+1}$, ..., $X_h^{t+1}$.
1: Calculate $p_t$ by (39);
2: Generate $e$ from (0, 1);
3: Generate integer $r$ from $[1, h]$;
4: if $e < p_t$ then
5:     for $i \leftarrow 1$ to $h$ do
6:         Calculate $f(X_i)$ by (24);
7:     end for
8: end if
9: Calculate $N_t$ by (40);
10: Update the positions of $N_t$ randomly selected beluga whales by (41);
```

time complexity of Algorithm 6 is $T_{max} \times (O(hnqm + h^2nq) + O(hnq) + O(hnq) + O(hnq) + O(hnqm)) = O(T_{max}hnq(h+m))$, where h, n, m, T_{max}, q are the size of population, the number of industrial parks, the number of edge servers, the maximum number of tasks generated in one time slot in all parks, and the maximum number of iterations, respectively.

Algorithm 6 The IMOBWO algorithm.

```
Require: $h$, $T_{max}$, $d$.
Ensure: The globally best beluga whale position $X_{best}$ and its fitness value $f(X_{best}) = (f_1(X_{best}), f_2(X_{best}))$.
1: Initialize the beluga whale population $X_1^0$, $X_2^0$, ..., $X_h^0$, $t = 0$, $X_{best}$;
2: while $t < T_{\max}$ do
3:     Call Algorithm 1 to compute the fitness values of all beluga whales and the best whale position $X_{best}^t$ in current iteration;
4:     if $X_{best}^t$ is better than $X_{best}$ then
5:         $X_{best} \leftarrow X_{best}^t$;
6:     end if
7:     Calculate $B_f$ by (28);
8:     Calculate $W_f$ by (29);
9:     if $B_f > 0.5$ then
10:        Call Algorithm 2 to renew the beluga whale positions;
11:    end if
12:    if $B_f < 0.5$ then
13:        Call Algorithm 3 to renew the beluga whale positions;
14:    end if
15:    if $B_f < W_f$ then
16:        Call Algorithm 4 to renew the beluga whale positions;
17:    end if
18:    Call Algorithm 5 to renew the beluga whale positions;
19:    $t = t + 1$;
20: end while
```

4.5 Validation of the Effectiveness for the IMOBWO Algorithm

To verify the effectiveness of IMOBWO, we apply the widely used multi-objective test functions ZDT1-ZDT4 and ZDT6 proposed in 2000 for testing, choose the MOGA algorithm [39] as well as the MOPSO algorithm [40] as comparison algorithms, and take the spacing (SP) and error ratio (ER) as evaluation indicators, like many other related work. The maximum number of iterations for each algorithm on each test function is 1000, and the experimental results show that each algorithm runs 20 times on the same test function.

(1) SP: The standard deviation used to measure the distance between each solution and other solutions is an important indicator for evaluating solution differences. It is calculated as follows:

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}, \tag{42}$$

where d_i is the Euclidean distance between the i-th solution and other solutions, $\bar{d}$ is the average of all d_i, and n is the number of elements in the solution set. The smaller the SP value, the more uniform and diverse the solution set.

(2) ER: Used to indicate the percentage of optimal solutions. The smaller the ER value, the higher the proportion of the obtained optimal solution set in the Pareto front. It is calculated as follows:

$$ER = \frac{1}{n}\sum_{i=1}^{n} e_i. \tag{43}$$

When the i-th non dominated solution is in the true Pareto front, $e_i = 0$; otherwise, $e_i = 1$.

Tables 1 and 2 show the SP and ER values of our IMOBWO algorithm compared to other comparative algorithms on five given test functions. The underlined part is the optimal value. It can be seen that IMOBWO achieves 5 optimal mean and standard deviation values on the SP indicator, and obtains 4 optimal mean values and 4 optimal standard deviations on the ER indicator. They indicate that the solutions computed by the IMOBWO algorithm have better diversity and convergence on the whole. This is because IMOBWO improves the search strategy for the best beluga position and adds the mutation operation. At the same time, the

Table 1 Comparison of SP evaluation indicators between our algorithm and other multi-objective algorithms

	MOGA		MOPSO		IMOBWO	
Test function	Average value	Standard deviation	Average value	Standard deviation	Average value	Standard deviation
ZDT1	6.16×10^{-2}	5.94×10^{-4}	7.08×10^{-2}	5.94×10^{-4}	$\underline{9.76 \times 10^{-3}}$	$\underline{9.73 \times 10^{-5}}$
ZDT2	1.83×10^{-2}	2.03×10^{-3}	1.76×10^{-2}	4.96×10^{-3}	$\underline{7.38 \times 10^{-3}}$	$\underline{1.62 \times 10^{-4}}$
ZDT3	8.87×10^{-2}	5.73×10^{-2}	9.47×10^{-2}	5.87×10^{-2}	$\underline{7.07 \times 10^{-3}}$	$\underline{1.91 \times 10^{-4}}$
ZDT4	3.04×10^{-2}	1.67×10^{-3}	3.96×10^{-2}	1.93×10^{-3}	$\underline{8.71 \times 10^{-3}}$	$\underline{1.23 \times 10^{-4}}$
ZDT6	4.06×10^{-2}	1.95×10^{-2}	4.96×10^{-2}	2.43×10^{-2}	$\underline{8.32 \times 10^{-3}}$	$\underline{8.93 \times 10^{-4}}$

probability of mutation occurring in IMOBWO gradually decreases with the increase of iteration times, making IMOBWO has good convergence.

5 Simulation and Evaluation

This section conducts experiments to assess the efficiency and effectiveness of our presented IMOBWO algorithm by comparing with the Random algorithm which randomly allocates a task to an edge server, the Greedy algorithm which allocates a task to the edge server that makes the task have the minimum completion time and energy consumption, the MOGA algorithm [39], and the MOPSO algorithm [40].

5.1 Experimental Parameter Setting

In order to make the experiment more realistic, this article investigated the server rental prices [41–43], commercial electricity prices, and key indicators of 5G networks of different cloud service operators. The rental price of servers is influenced by factors such as CPU memory configuration and server area nodes, usually ranging from several thousand to several hundred thousand RMB per year. For example, the Alibaba Cloud GPU computing type GN6V cloud server (8-core 60GB, equipped with Intel Xeon (Skylake) Platinum 8163 CPU, with a full state power of 150W), priced at 26.50RMB/h, with a maximum public network bandwidth (transmission network bandwidth on the Internet, used to connect different data centers or user devices) of 100Mbps, and an internal network bandwidth (network transmission bandwidth used for communication between servers in the same data center) of 1Gbps; the commercial electricity price is 0.8433 RMB/kWh [44]. Therefore, the input-output ratio of operators is approximately 1:150. After understanding the above situation, the basic experimental parameters are set as shown in Table 3. In addition, the initial population size is set to be 40 [45, 46], and the maximum number of iterations is set to 400.

To verify the performance of our presented algorithm under different task numbers and maximum completion time constraints, this paper conducts three sets of comparative experiments as following:

1. The first set of experiments: Set the number of edge servers to be 10, the task number to be 100, and the maximum task completion time range to be [150,750] ms. The aim is to test the convergence of algorithms MOGA, MOPSA, and IMOBWO at

Table 2 Comparison of ER evaluation indicators between our algorithm and other multi-objective algorithms

	MOGA		MOPSO		IMOBWO	
Test function	Average value	Standard deviation	Average value	Standard deviation	Average value	Standard deviation
ZDT1	9.23×10^{-3}	1.34×10^{-2}	8.14×10^{-3}	2.52×10^{-3}	$\underline{7.93 \times 10^{-4}}$	$\underline{1.37 \times 10^{-4}}$
ZDT2	9.57×10^{-3}	3.66×10^{-3}	9.19×10^{-3}	2.14×10^{-3}	$\underline{5.45 \times 10^{-3}}$	$\underline{1.28 \times 10^{-5}}$
ZDT3	2.69×10^{-2}	8.32×10^{-3}	2.36×10^{-2}	7.56×10^{-3}	$\underline{9.76 \times 10^{-3}}$	$\underline{9.72 \times 10^{-3}}$
ZDT4	$\underline{2.37 \times 10^{-2}}$	4.87×10^{-3}	2.67×10^{-2}	4.13×10^{-3}	2.87×10^{-2}	$\underline{2.47 \times 10^{-3}}$
ZDT6	2.75×10^{-3}	2.65×10^{-3}	2.57×10^{-3}	$\underline{1.87 \times 10^{-3}}$	$\underline{1.98 \times 10^{-3}}$	2.09×10^{-3}

Table 3 Experimental parameter

Parameter	Description	Value
f_i	Server processing capacity	[50, 75]GB
p^{max}	Server maximum power	150W
$Bwan$	wireless network bandwidth	30Mbps
$Blan$	wired network bandwidth	300Mbps
t^{size}	task size	[1, 3]MB
t^{val}	task value	[1, 5]RMB
p^e	energy consumption price	0.15RMB/s
τ	time interval	1s
d^{ide}	Network idle latency	3ms
$w_{i,j}$	distance coefficient	[2, 4]

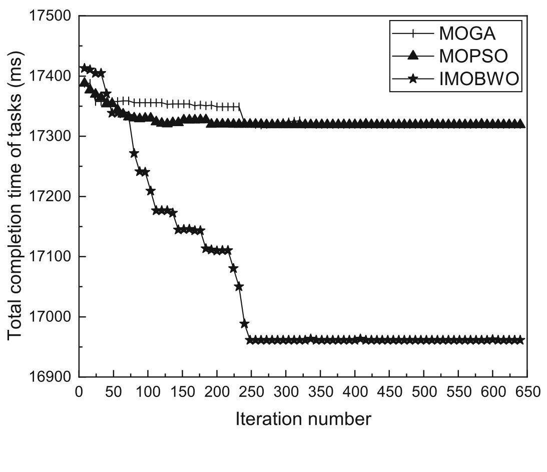
different iterations. The maximum number of iterations is set based on this set of experiments.

2. The second set of experiments: Set the number of edge servers to be 10, the maximum completion time of tasks to be 400ms, and the total number of tasks to change from 20 to 100. The aim is to verify the performance of the presented algorithm under different task quantities.
3. The third set of experiments: Set the total number of tasks to be 100, the number of edge servers to be 10, and the maximum completion time of the task to change from 200ms to 600ms. The aim is to verify the performance of the presented algorithm under different maximum task completion time constraints.

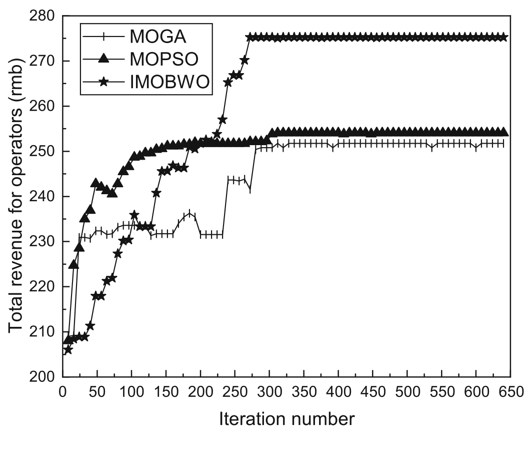
5.2 Simulation Experimental Results

5.2.1 The Result and Discuss for the First Set of Experiments

Figure 2 shows the changes of total completion time of tasks and total revenue for operators computed by algorithms MOGA, MOPSO, and IMOBWO when the iteration times increase. From Fig. 2a, it is observed that when the iteration times larger than 80, IMOBWO can



(a)



(b)

Fig. 2 The total completion time of tasks (a) and the total revenue for operators (b) when the iteration number increases computed by algorithms

get the minimum total task completion time among all algorithms. When the iteration times larger than 350, all three algorithms can get a stable total task completion time. From Fig. 2b, it is observed that when the iteration times larger than 230, IMOBWO can calculate the maximum total revenue for operators among all algorithms. When the iteration times larger than 350, all three algorithms can get a stable total revenue for operators. Thus, we set the maximum number of iterations to be 400 in the following two sets of experiments. Obviously, our proposed IMOBWO algorithm is convergent.

5.2.2 The Result and Discuss for the Second Set of Experiments

Figure 3 displays the changes of total completion time of tasks and total revenue for operators computed by algorithms Random, Greedy, MOGA, MOPSO, and IMOBWO when the total number of tasks increases. It is observed that when the task number increases, total completion time of tasks and total revenue for operators obtained by all five algorithms increase, where IMOBWO can achieve the minimum total completion time of tasks and the maximum total revenue for operators. This is because more tasks will consume more total task completion time, and more tasks will lead to more total revenue for operators. Table 4 lists the reduction percentage of the total completion time of tasks and the improvement percentage of the total revenue for operators by comparing our proposed IMOBWO and four baseline algorithms for the values in Fig. 3.

Figure 4 shows the maximum number of tasks, the minimum number of tasks, and the maximum number difference of tasks on one edge server computed by algorithms Random, Greedy, MOGA, MOPSO, and IMOBWO when the total number of tasks increases. It is observed that when the task number increases, IMOBWO can achieve the minimum maximum number of tasks, the maximum minimum number of tasks, and the minimum maximum number difference of tasks among all five algorithms. Thus, IMOBWO can achieve better load balancing.

5.2.3 The Result and Discuss for the Third Set of Experiments

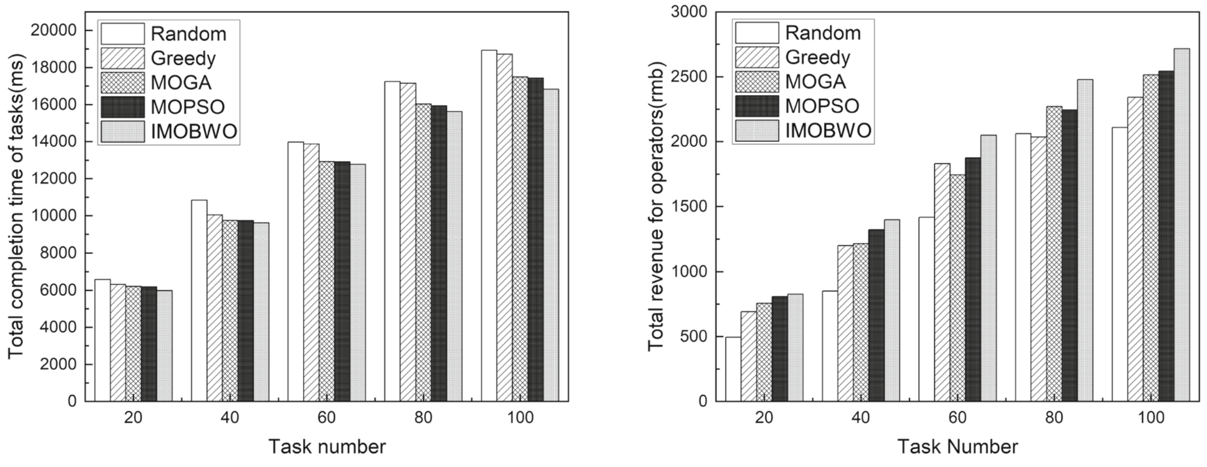
Figure 5 displays the changes of total completion time of tasks and total revenue for operators computed by algorithms Random, Greedy, MOGA, MOPSO, and IMOBWO when the maximum completion time increases. It is observed that when the maximum completion time increases, IMOBWO can achieve the minimum total completion time of tasks and the maximum total revenue for operators among all five algorithms. Table 5 lists the reduction percentage of the total completion time of tasks and the improvement percentage of the total revenue for operators by comparing our proposed IMOBWO and four baseline algorithms for the values in Fig. 5.

Figure 6 shows the maximum number of tasks, the minimum number of tasks, and the maximum number difference of tasks on one edge server computed by algorithms Random, Greedy, MOGA, MOPSO, and IMOBWO when the maximum completion time increases. It is observed that when the task number increases, IMOBWO can achieve the minimum maximum number of tasks, the maximum minimum number of tasks, and the minimum maximum number difference of tasks among all five algorithms. Thus, IMOBWO can achieve better load balancing.

On the whole, our proposed IMOBWO algorithm is high efficient.

6 Conclusion

This article designs a multi-objective optimization algorithm, the improved beluga whale optimization algorithm to deal with the resource allocation problem in EC of 5G network. The objective is to minimize the total completion time of tasks and maximize the total profit for operators. The proposed algorithm takes four operations including whale swimming, whale predating, whale migrating, and whale mutation to update the solutions of the studied problem. Simulation experiments demonstrate that our proposed algorithm can achieve good total task completion time, total operator revenue, and loading balance. Additionally, we have yet to verified the advantages of our proposed algorithm in broader and more realistic networks, different types of tasks (such as latency-sensitive tasks and batch tasks), different edge computing environments, and different load conditions. When these conditions change, new problems may arise. At this point, our proposed algorithm may not be the best suitable to solve these problems. In the future, we will develop more general algorithms to solve resource allocation problems under

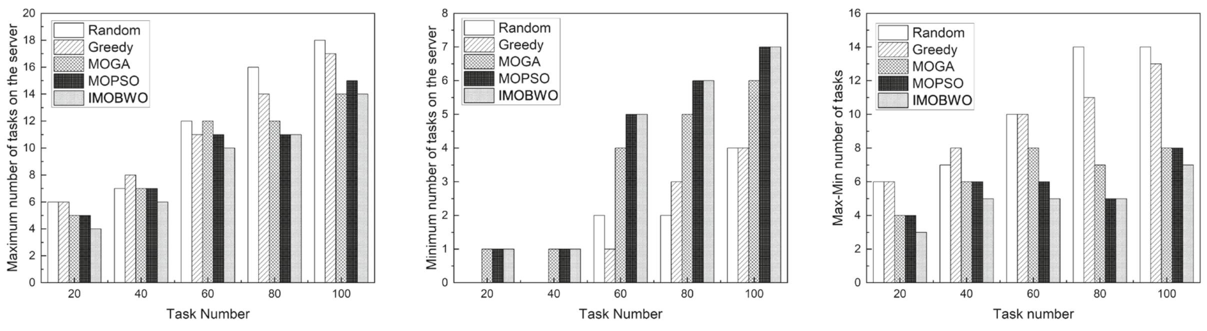


(a) (b)

Fig. 3 The total completion time of tasks (a) and the total revenue for operators (b) when the task number increases computed by algorithms

Table 4 The reduction percentage of the total completion time of tasks and the improvement percentage of the total revenue for operators by comparing our proposed IMOBWO and four baseline algorithms for the values in Fig. 3

Task Number	IMOBWO vs Random		IMOBWO vs Greedy		IMOBWO vs MOGA		IMOBWO vs MOPSO	
	Time	Revenue	Time	Revenue	Time	Revenue	Time	Revenue
20	8.96%	67.27%	5.22%	19.48%	3.59%	9.52%	3.21%	2.22%
40	11.23%	64.29%	4.18%	16.46%	1.29%	15.00%	1.18%	5.75%
60	8.53%	44.62%	7.82%	11.90%	1.04%	17.50%	0.98%	9.30%
80	9.34%	20.25%	8.86%	21.79%	2.48%	9.20%	1.87%	10.47%
100	11.12%	28.77%	10.11%	16.05%	3.80%	8.05%	3.47%	6.82%



(a) (b) (c)

Fig. 4 The maximum tasks (a), the minimum tasks (b), and the maximum number difference of tasks (c) on edge servers when the task number increases computed by algorithms

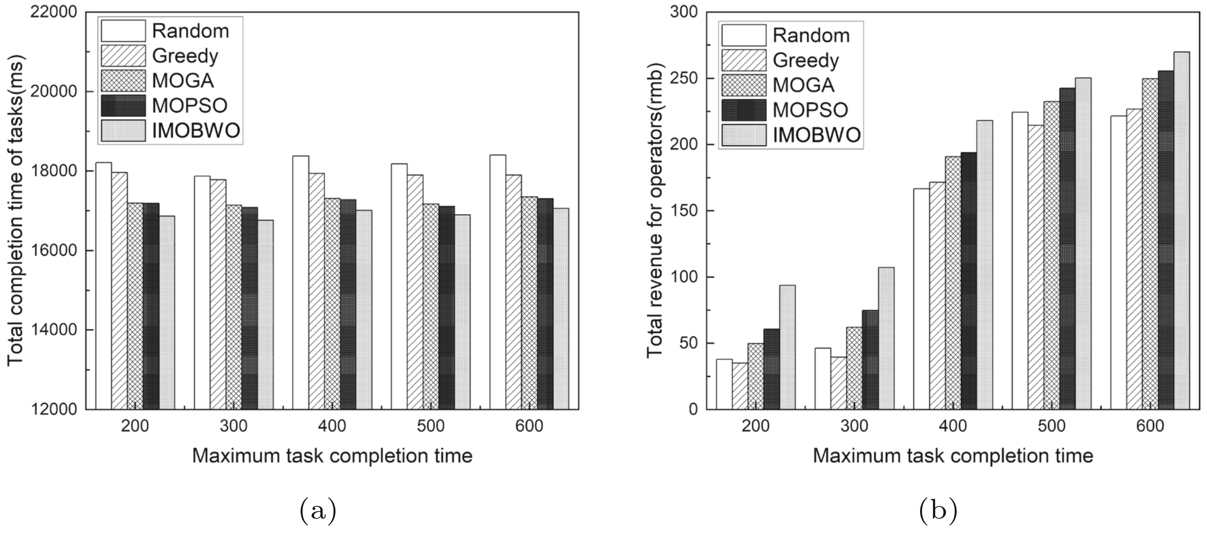


Fig. 5 The total completion time of tasks (a) and the total revenue for operators (b) when the maximum completion time increases computed by algorithms

Table 5 The reduction percentage of the total completion time of tasks and the improvement percentage of the total revenue for operators by comparing our proposed IMOBWO and four baseline algorithms for the values in Fig. 5

Maximum Time	IMOBWO vs Random		IMOBWO vs Greedy		IMOBWO vs MOGA		IMOBWO vs MOPSO	
	Time	Revenue	Time	Revenue	Time	Revenue	Time	Revenue
200	7.38%	146.15%	1.89%	166.67%	1.89%	88.24%	1.86%	54.36%
300	6.19%	131.45%	2.21%	171.43%	2.21%	72.73%	1.89%	43.15%
400	7.43%	30.91%	1.73%	27.12%	1.73%	14.29%	1.52%	12.50%
500	7.06%	11.48%	1.58%	16.67%	1.58%	7.69%	1.26%	3.26%
600	7.28%	21.74%	1.65%	18.99%	1.65%	8.05%	1.39%	5.56%

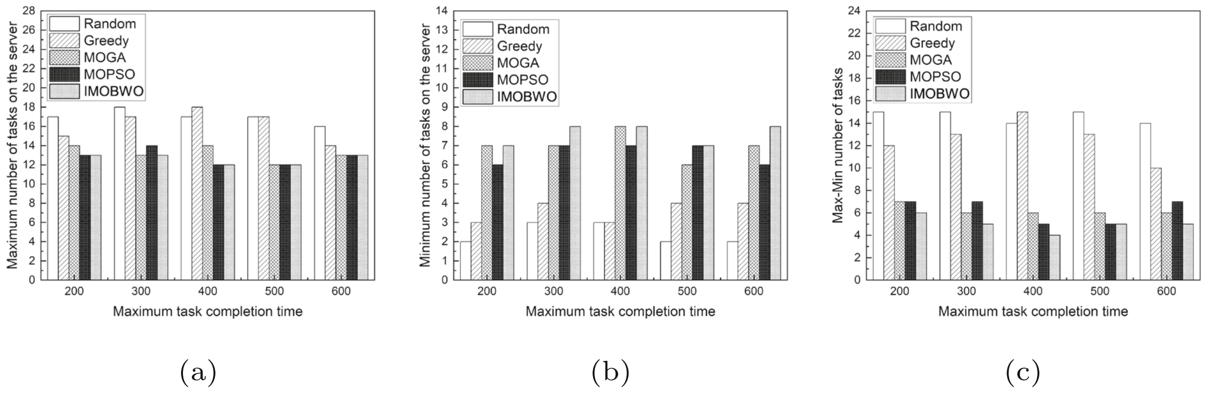


Fig. 6 The maximum tasks (a), the minimum tasks (b), and the maximum number difference of tasks (c) on edge servers when the maximum completion time increases computed by algorithms

different network conditions, types of tasks, edge computing environments, and load conditions. Moreover, there are many other factors that we do not consider in this article, such as edge caching, user experience, energy efficiency, etc., which we will investigate in our next work.

Acknowledgements This work was supported the Natural Science Foundation of Hunan Province (Grant No. 2023JJ50204), and the Scientific Research Foundation of Hunan Provincial Education Department, China (Grant No. 23B0560).

Author contributions Jing Liu , Yuting Huang, Chunhua Deng wrote the main manuscript text and Cen Chen and Keqin Li prepared figures 1-15. All authors reviewed the manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Huang, X., Zhang, W., Yang, J., Yang, L., Yeo, C.K.: Market-based dynamic resource allocation in mobile edge computing systems with multi-server and multi-user. Comput. Commun. **165**, 43–52 (2021)
2. Zhang, J., Gong, B., Waqas, M., Tu, S., Han, Z.: A hybrid many-objective optimization algorithm for task offloading and resource allocation in multi-server mobile edge computing networks. IEEE Trans. Serv. Comput. **16**(5), 3101–3114 (2023)
3. Peng, K., Huang, H., Zhao, B., Jolfaei, A., Xu, X., Bilal, M.: Intelligent computation offloading and resource allocation in iiot with end-edge-cloud computing using nsga-iii. IEEE Trans. Netw. Sci. Eng. **10**(5), 3032–3046 (2022)
4. Chen, S., Chen, B., Tao, X., Xie, X., Li, K.: An online dynamic pricing framework for resource allocation in edge computing. J. Syst. Architect. **133**, 102759 (2022)
5. Sharif, Z., Jung, L.T., Razzak, I., Alazab, M.: Adaptive and priority-based resource allocation for efficient resources utilization in mobile-edge computing. IEEE Internet Things J. **10**(4), 3079–3093 (2023)
6. Qian, S., Chen, F., Ning, H., Lin, T., Yuanyuan, W.: Computing and storage resources allocation of upf based on isolation in private 5g networks. In: 2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring), pp. 1–6 (2022). IEEE
7. Boussaïd, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)
8. Singh, A., Dulal, N.: A survey on metaheuristics for solving large scale optimization problems. Int. J. Comput. Appl. **170**(5), 1–7 (2017)
9. Sharma, S., Jain, R.: Eaco: An enhanced ant colony optimization algorithm for task scheduling in cloud computing. Int. J. Secur. Appl. **13**(4), 91–100 (2019)
10. Xue, J., Guan, X.: Collaborative computation offloading and resource allocation based on dynamic pricing in mobile edge computing. Comput. Commun. **198**, 52–62 (2023)
11. Li, Y., Xia, M., Duan, J., Chen, Y.: Pricing-based resource allocation in three-tier edge computing for social welfare maximization. Comput. Netw. **217**, 109311 (2022)
12. Kumar, S., Sharma, A., Gupta, R.: Qos driven cost-efficient resource allocation in edge computing: A distributed game theoretic approach. J. Comput. Sci. **72**, 102106 (2023)
13. Zhao, M., Yu, J.-J., Li, W.-T., Liu, D., Yao, S., Feng, W., She, C., Quek, T.Q.: Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems. IEEE Trans. Veh. Technol. **70**(10), 10925–10940 (2021)
14. Tan, L., Kuang, Z., Zhao, L., Liu, A.: Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing. IEEE Trans. Wirel. Commun. **21**(3), 1960–1972 (2021)
15. Liu, X., Liu, J., Wu, H.: Energy-efficient task allocation of heterogeneous resources in mobile edge computing. IEEE Access **9**, 119700–119711 (2021)
16. Cao, D., Gu, N., Wu, M., Wang, J.: Cost-effective task partial offloading and resource allocation for multi-vehicle and multi-mec on b5g/6g edge networks. Ad Hoc Netw. **156**, 103438 (2024)
17. Wang, Y., Fang, J., Cheng, Y., She, H., Guo, Y., Zheng, G.: Cooperative end-edge-cloud computing and resource allocation for digital twin enabled 6g industrial iot. IEEE J. Sel. Top. Signal Process. **18**(1), 124–137 (2024)
18. An, L., Wang, Z., Yue, J., Ma, X.: Joint task offloading and resource allocation via proximal policy optimization for mobile edge computing network. In: 2021 International Conference on Networking and Network Applications (NaNA), pp. 466–471 (2021). IEEE
19. Šlapak, E., Gazda, J., Guo, W., Maksymyuk, T., Dohler, M.: Cost-effective resource allocation for multitier mobile edge computing in 5g mobile networks. IEEE Access **9**, 28658–28672 (2021)
20. Wang, S., Li, X., Gong, Y.: Energy-efficient task offloading and resource allocation for delay-constrained edge-cloud computing networks. IEEE Trans. Green Commun. Netw. **8**(1), 514–524 (2024)
21. Saleem, U., Liu, Y., Jangsher, S., Li, Y., Jiang, T.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wirel. Commun. **20**(1), 360–374 (2021)
22. Ansere, J.A., Gyamfi, E., Sharma, V., Shin, H., Dobre, O.A., Duong, T.Q.: Quantum deep reinforcement learning for dynamic resource allocation in mobile edge computing-based iot systems. IEEE Trans. Wirel. Commun. **23**(6), 6221–6233 (2024)
23. Huang, X., He, L., Chen, X., Wang, L., Li, F.: Revenue and energy efficiency-driven delay-constrained computing task offloading and resource allocation in a vehicular edge computing network: A deep reinforcement learning approach. IEEE Internet Things J. **9**(11), 8852–8868 (2022)
24. Sharif, Z., Jung, L.T., Ayaz, M., Yahya, M., Pitafi, S.: Priority-based task scheduling and resource allocation in

edge computing for health monitoring system. J. King Saud Univ.-Comput. Inf. Sci. **35**(2), 544–559 (2023)
25. Wang, Y., Liu, Y., Sun, Z., Liu, L., Li, J., Sun, G.: Priority-aware task offloading and resource allocation in vehicular edge computing networks. In: 2022 18th International Conference on Mobility, Sensing and Networking (MSN), pp. 205–212 (2022). IEEE
26. Habiba, U., Maghsudi, S., Hossain, E.: A repeated auction model for load-aware dynamic resource allocation in multi-access edge computing. IEEE Trans. Mob. Comput. **23**(7), 7801–7817 (2024)
27. Fan, Y., Wang, L., Wu, W., Du, D.: Cloud/edge computing resource allocation and pricing for mobile blockchain: an iterative greedy and search approach. IEEE Trans. Comput. Soc. Syst. **8**(2), 451–463 (2021)
28. Nguyen, D.T., Le, L.B., Bhargava, V.: Price-based resource allocation for edge computing: A market equilibrium approach. IEEE Trans. Cloud Comput. **9**(1), 302–317 (2021)
29. Yuan, H., Zhou, M.: Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems. IEEE Trans. Autom. Sci. Eng. **18**(3), 1277–1287 (2021)
30. Falsafain, H., Heidarpour, M.R., Vahidi, S.: A branch-and-price approach to a variant of the cognitive radio resource allocation problem. Ad Hoc Netw. **132**, 102871 (2022)
31. Schieber, B., Samineni, B., Vahidi, S.: Interweaving real-time jobs with energy harvesting to maximize throughput. In: WALCOM: Algorithms and Computation, pp. 305–316. Springer, Cham (2023)
32. Jamil, U., Malmir, M., Chen, A., Filipovska, M., Xie, M., Ding, C., Jin, Y.-F.: Developing an eco-driving strategy in a hybrid traffic network using reinforcement learning. Sci. Prog. **107**(3), 00368504241263406 (2024)
33. Tanenbaum, A.S.: Computer Networks. Pearson Education India, Upper Saddle River, New Jersey (2003)
34. Wang, Y., Wang, Q., Wang, H., Jing, H., Dai, G.: A real-time scheduling algorithm based on priority table and its implementation. J. Softw. **15**(3), 360–370 (2004)
35. Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., Rius, J.: A queuing theory model for cloud computing. J. Supercomput. **69**, 492–507 (2014)
36. Bardsiri, A.K., Hashemi, S.M.: Qos metrics for cloud computing services evaluation. Int. J. Intell. Syst. Appl. **6**(12), 27 (2014)
37. Xiang, Y., Liu, Z., Qu, X.: Cpu frequency scaling algorithm for energy-saving in cloud data centers. J. Softw. **9**(9), 2283–2290 (2014)
38. Zhong, C., Li, G., Meng, Z.: Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. Knowl.-Based Syst. **251**, 109215 (2022)
39. Afrin, M., Jin, J., Rahman, A., Tian, Y.-C., Kulkarni, A.: Multi-objective resource allocation for edge cloud based robotic workflow in smart factory. Futur. Gener. Comput. Syst. **97**, 119–130 (2019)
40. Luo, Q., Li, C., Luan, T.H., Shi, W.: Minimizing the delay and cost of computation offloading for vehicular edge computing. IEEE Trans. Serv. Comput. **15**(5), 2897–2909 (2021)
41. Huawei Cloud. https://activity.huaweicloud.com/ecs.html
42. Tencent Cloud. https://cloud.tencent.com/
43. Ali Yun. https://www.aliyun.com/product/list
44. State Grid. https://www.95598.cn/osgweb/index
45. Fei, W., Liu, C., Hu, S.: Research on swarm intelligence optimization algorithm. J. China Univ. Posts Telecommun. **27**(3), 1–20 (2020)
46. Wang, D., Tan, D., Liu, L.: Particle swarm optimization algorithm: an overview. Soft Comput. **22**(2), 387–408 (2018)