World Wide Web (2022) 25:2265–2295
https://doi.org/10.1007/s11280-022-01082-7

HWOA: an intelligent hybrid whale optimization algorithm for multi-objective task selection strategy in edge cloud computing system

Yan Kang[1] · Xuekun Yang[1] · Bin Pu[2] · Xiaokang Wang[3] · Haining Wang[1] · Yulong Xu[1] · Puming Wang[1]

Received: 3 November 2021 / Revised: 31 May 2022 / Accepted: 27 June 2022 /
Published online: 14 September 2022

Abstract
Edge computing is a popular computing modality that works by placing computing resources as close as possible to the sensor data to relieve the burden of network bandwidth and data centers in cloud computing. However, as the volume of data and the scale of tasks processed by edge terminals continue to increase, the problem of how to optimize task selection based on execution time with limited computing resources becomes a pressing one. To this end, a hybrid whale optimization algorithm (HWOA) is proposed for multi-objective edge computing task selection. In addition to the execution time of the task, economic profits are also considered to optimize task selection. Specifically, a fuzzy function is designed to address the uncertainty of task's economic profits and execution time. Five interactive constraints among tasks are presented and formulated to improve the performance of task selection. Furthermore, some improved strategies are designed to solve the problem that the whale optimization algorithm (WOA) is subject to local optima entrapment. Finally, an extensive experimental assessment of synthetic datasets is implemented to evaluate the multi-objective optimization performance. Compared with the traditional WOA, the diversity metric (Δ-spread), the hypervolume (HV) and other evaluation metrics are significantly improved. The experiment results also indicate the proposed approach achieves remarkable performance compared with other competitive methods.

Keywords Edge cloud computing system · Multi-objective optimization · Scheduling · Task selection strategy · Whale optimization algorithm

Yan Kang and Bin Pu equal contributed to this work.

This article belongs to the Topical Collection: *Special Issue on Resource Management at the Edge for Future Web, Mobile and IoT Applications*
Guest Editors: Qiang He, Fang Dong, Chenshu Wu, and Yun Yang

✉ Xiaokang Wang
xkwang@hainanu.edu.cn

Extended author information available on the last page of the article

1 Introduction

The emerging concept of edge computing pushes computation, service, and data from the cloud to the edges of networks [1, 2]. It improves the computing performance of cloud computing and brings new opportunities for the development of the Internet [3–5]. By adopting edge computing, internet applications can offload tasks to edge devices and local edge servers [6–8]. The benefits of using edge computing to run applications are manifold: (i) it saves the overall network bandwidth for data transmission and reduces the risk of overloading the remote central server; (ii) it can respond quickly to time-sensitive applications, thereby providing services to end users more effectively; and (iii) it makes better use of the computing power of edge devices, and end users can also get rewards by performing computing tasks on their devices. However, due to the limitations of computation resources, edge nodes will not have servers with the same capacity as those in remote clouds [9, 10]. In this case, a well-designed task scheduling strategy is needed to optimize the performance of edge computing. In recent years, several resource allocation and task scheduling works have been proposed for edge computing with different system configurations [11–13].

In recent years, the task scheduling problem in edge cloud computing system has attracted widespread attention [14]. Addressing the problem of resource optimization in edge cloud computing system, Chen et al. [15] expressed the computing task scheduling problem as a mixed-integer nonlinear computing process, and transformed the problem of reducing time delay into two sub-problems of task offloading and resource allocation. Liu et al. [16] proposed an optimal multi-resource allocation strategy based on linear programming, which improved the performance of edge computing devices in terms of throughput and delay. Due to the search effectiveness of the heuristic algorithm, Yang et al. [17] used a new heuristic algorithm to divide the computing tasks for the problem, which reduced the average completion time of edge computing tasks to the greatest extent. We consider a natural scheduling problem which arises in many distributed computing frameworks. Tasks with diverse computation requirements (e.g. computation cost requirements) arrive over time and must be served by servers with a finite resource capacity. To improve throughput and delay, the scheduler can pack as many jobs as possible in the servers subject to their capacity constraints. Motivated by the ever-increasing complexity of workloads in servers, we explore how to select a group of tasks reasonably under the current situation of edge device computing resource. It is noted that this task division should not only minimize the total execution time of the task, but it also ensure that the task is under the constraint of a certain execution time. How to reasonably select a group of tasks under the current situation of edge device computing resources has become a challenging edge computing priority task [18].

Major Internet Service Providers (ISPs) want to improve and enrich the current mobile services (e.g., fast Internet access, edge computing, and local caching) for additional profit [19–22]. Since the economic profits brought by task calculation have great potential, it is important to determine how to maximize them [23]. The economic issues of edge computing are largely overlooked in the literature, hindering the successful development of edge computing in the long run. This profit-maximization problem is not easy as edge computing servers have limited computation resources, requiring an optimal allocation for balancing both profit and computing capacity. In this paper, this problem is solved as a multi-objective optimization problem and the Pareto optimal solution set is found. Tirkolaee et al. [24] solved the multi-objective multi-mode

resource-constrained project scheduling problem using Pareto-based algorithms to minimize the completion time. To measure the spread of representative solutions over the Pareto front, Petchrompo et al. [25] presented two novel indicators based on average euclidean distance and cosine similarity between original Pareto solutions and representative solutions, and solved the problem of multi-objective budget allocation. Elsisy et al. [26] proposed a new algorithm for generating the Pareto front for a bi-level multi-objective rough nonlinear programming problem, and discussed the process of generating the Pareto front.

It is challenging with increasing edge computing requirements occurrence rates and incomplete information about the economic issues of edge computing tasks [22, 27]. Due to the uncertainty of the economic profits and the execution time of the task, this paper models the problem as a fuzzy function to be optimized. It is noted that there are different relationships among tasks. For example, some tasks cannot run altogether, since the input of one task depends on the output of another task. And some tasks cannot run altogether since they have unique resource requirements. The profits of these tasks will increase or decrease when a set of tasks are scheduled altogether. Five interactive constraints are given in this paper as: dependence, combination, mutual exclusion, mutual exclusion of economic profits, and dependence on economic profits.

An intelligent optimization algorithm called the hybrid whale optimization algorithm (HWOA) is proposed for edge computing tasks based on multi-objective groups. HWOA can adjust the parameters according to the scale of the computing task so as to better adapt to the problem of edge computing tasks of different scales. This is the first time that the whale algorithm has been applied to solve the optimization problem of multi-objective edge computing tasks. In the multi-objective task selection problem, the occurrence of unforeseen events will always dynamically affect the execution time and economic profits of tasks, so fuzzy functions are used to fuzz the data in this study. As a kind of meta-heuristic algorithm, the whale algorithm is simple and easy to implement and has high flexibility. However, when it comes to solving multi-objective complex engineering problems, the whale algorithm is likely to fall into a local optimal solution, resulting in low convergence accuracy. To solve this problem, the excellent strategies of the dragonfly algorithm and genetic algorithm are integrated into the whale algorithm. The experimental results based on two fuzzy datasets reveal the potential efficiency and practical applicability of the method. The main contributions of this work can be summarized as follows:

- We propose an intelligent hybrid whale optimization algorithm (HWOA) for the topic of task selection strategy, which considers the execution time and economic profits of edge computing multi-objective tasks. Moreover, the increasing task requirement under limited computation resources is formulated as the task selection problem, and it is considered as a 0-1 knapsack problem with five interactive constraints.
- The uncertainty of the task selection problem is investigated. And a triangular fuzzy function is introduced to resolve the uncertainty of the time and profits of the task under a dynamic environment.
- The search strategies of the dragonfly algorithm and genetic algorithm are integrated to improve HWOA's search performance, and the greedy repair strategy is employed to repair the solutions that did not meet the constraints.
- The performance of HWOA in reliably addressing edge computing task selection problems is demonstrated in solving edge computing task datasets and knapsack cases. Compared with other competitive methods, the diversity metric (Δ-spread), the hypervolume (HV) and other evaluation metrics are significantly improved.

The rest of this work is arranged as follows. Section 2 summarises the relevant work. Section 3 describes the problem analysis and the details of the proposed method. Section 4 and Section 5 contain the experiments and the conclusion respectively.

2 Related work

In this section, relevant work involving task selection in edge cloud computing system environments is reviewed, mainly covering task scheduling, multi-objective optimization, and the whale optimization algorithm.

2.1 Task scheduling

Task scheduling is always a hot research area of edge computing since it is proposed. The edge terminal needs to offload the tasks to the edge computing server, and after the server completes the calculation, the calculation result is returned to the edge terminal [28, 29]. The goals of task scheduling can be generally classified into three categories: (i) reducing energy consumption; (ii) improving system throughput; (iii) reducing task completion time.

Aiming to achieve these goals, Li [30] completed the optimal configuration of communication and computing resources in edge computing, and took the shortest delay and the smallest computational cost as the optimization goals. He et al. [31] constructed a heuristic algorithm based on the density of residual tasks to determine the execution time of each task, which improves the efficiency of the scheduling process and reduces the response time. Wang et al. [32] proposed a mobile edge computing task allocation and offloading algorithm based on deep reinforcement learning in response to high-performance computing requirements, and they solved the problems of low terminal storage capacity and diversified network services by optimizing task scheduling. Due to the non-deterministic polynomial hardness (NP-hard) property of task scheduling problems, and heuristic algorithms are often used to solve NP problems. Meta-heuristic algorithms inevitably need to produce near-optimal solutions in a reasonable time [33]. Huang et al. [34] solved the task scheduling problem in an NP-hard way and obtained the best income from edge service providers. Feng et al. [35] improved the task scheduling strategy by normalizing each object in the model to eliminate the influence of dimensionality. Guo [36] proposed a method for cloud computing multi-objective task scheduling based on a fuzzy self-defense algorithm. By selecting the shortest time, resource load balancing degree, and task completion cost as the goals of edge computing task scheduling, an edge computing multi-objective task was constructed. Alrezaamiri et al. [37] put the optimization problem facing two conflicting goals and five interactive constraints into the NP-hard problem category and used artificial chemical reactions to solve this problem. Based on previous research on task scheduling, the problem of edge computing task selection is formalized as a multi-objective knapsack problem in the present study.

2.2 Multi-objective optimization

In recent years, many researchers have modeled problems in the field of edge computing and selected appropriate objectives to form complex multi-objective optimization problems. Wang et al. [38] presented a dynamic way to allocate optimal

energy to unmanned aerial vehicles to balance their hover time and service capability. Dai et al. [39] considered the constraints between high vehicle mobility and dynamic wireless channel conditions in vehicle edge computing and designed an optimal content caching strategy. Wang et al. [40] presented a two-layer optimization method for jointly optimizing the deployment of unmanned aerial vehicles and task scheduling, with the aim of minimizing system energy consumption. Thus, there are many multi-objective optimization problems in the field of edge computing, which need to be modeled specifically for the selected target, and suitable methods must be combined to solve these problems.

Currently, there are three kinds of methods to solve multi-objective optimization problems. The first method is to use traditional algorithms, such as dynamic programming and greedy algorithms [41]. However, these algorithms cannot effectively solve high-complexity problems. The second method is to use machine learning or deep learning training models [42–45]. This method cannot produce satisfactory results for unknown data, so it cannot be applied in many fields. The third method is to use the meta-heuristic algorithms. Compared with the traditional algorithms, the meta-heuristic algorithm is widely used in multi-objective optimization because of its strong adaptability, versatility, implicit parallelism, and expansibility. It is worth mentioning that in recent years, the application of the meta-heuristic algorithm in high-complexity problems has become the mainstream of research. Some researchers have used a single evolutionary algorithm to address relevant problems. Zhao et al. [46] proposed a multi-objective gray wolf evolutionary algorithm combined with an adaptive differential evolution mechanism, which introduced the selection mechanism of the external population and the head wolf, balanced the local development and global detection ability of the algorithm according to the weighted objective function values of alternative solutions, and improved the coverage of finding multi-objective solutions. Akay [47] learned from the honey-gathering behavior of artificial bees to realize the sharing and communication of colony information to find the optimal solution to the selection problem. Zhou et al. [48] developed an ant colony system (ACS) algorithm based on the multiple populations for multiple objectives (MPMO) framework, which used two ant colonies to optimize fairness and satisfaction objectives. Fang et al. [49] proposed an improved multi-objective evolutionary algorithm for HQPM (MOEA-PM), which designed two kinds of population initialization strategies to ensure the population is effectively distributed in the feasible solution space, and an auxiliary tool is proposed to accelerate the convergence of the algorithm. Sun et al. [50] developed an effective scheduling algorithm to solve the multi-objective optimization problem, providing the tradeoff between computational efficiency and energy efficiency. Additionally, more and more researchers are using hybrid algorithms to balance the characteristics of algorithms, which can more efficiently solve problems composed of different constraints. In [51, 52], on the basis of intelligent optimization algorithms, researchers studied the next release problem (NRP) from a multi-objective point of view, which provides a relative optimal solution for the optimization problem between interacting objectives and implementation constraints. However, except for in [53] in which whale optimization algorithm (WOA) was applied to priority ranking, there has been no application of mixing WOA and NSGA-II [54] methods. In this study, HWOA and NSGA-II are mixed and applied to the task selection problem of multi-objective edge computing, which promotes the development of the multi-objective edge computing task selection problem to a certain extent.

2.3 Whale optimization algorithm

The key idea of the WOA [55] algorithm is inspired by the social behavior of humpback whales, particularly their bubble-net hunting strategy. Evaluations on twenty-nine mathematical optimization tasks and six structural design issues demonstrated the superior performance of the WOA, so it gained much attention from the research community and has been successfully applied in many fields, such as feature selection [56], weights optimization [57], and resource allocation [58]. Aljarah et al. [57] proposed a novel training algorithm based on WOA, which can optimize the connection weights in artificial neural networks. Agrawal et al. [56] introduced the Quantum concepts to WOA, abbreviated as QWOA, which can enhance the exploratory and exploitation ability in the feature selection task. Pham et al. [58] first explored the applicability of WOA to resource allocation tasks in wireless networks. To address the problems of falling into local optima and achieving low accuracy when solving high-dimensional optimization problems, a multiple group improvement WOA (MIWOA) was proposed to improve the performance of WOA in handling high-dimensional optimization [59]. Chakraborty et al. [60] produced a novel enhanced WOA for solving the low exploration of the search space of the original WOA. In this paper, a novel HWOA is proposed for solving the disadvantages of falling into local optima, slow convergence speed, and low accuracy of solution. Then the proposed algorithm is applied to task selection in edge cloud computing system.

3 Methodology

First, some relevant notations and definitions are presented, which are mainly related to constraint rules, fuzzy function, and task selection. Second, we analyze this problem. Finally, the details of the proposed method are presented.

3.1 Constraint rules

Constraint rules refer to the conditions that need to be met when tasks are executed. The first is that the execution time cannot exceed the time threshold, and the other is that the relationship between tasks will lead to changes in profits. Considering the complex relationship between tasks in edge cloud computing system and optimization objectives, the problem constraints are chosen to be execution time and five interactive constraints between tasks (dependence, combination, mutual exclusion, mutual exclusion of economic profits, and dependence on economic profits). The execution time constraint is defined in (1).

$$\prod_{i\geq 1}^{i\leq n} f_{\text{time}}\left(\overline{T_i}\right) \leq Threshold, \tag{1}$$

where *Threshold* is the upper limit of the time threshold, and the function f_{time} is used to calculate the execution time of the current task T_i. The total execution time of the selected task subset is less than the upper limit of time.

The current scheduling generally assumes parallelism among tasks; it does not consider the existence of sequential dependencies and tight coupling among actual tasks, mutual exclusivity, and changes in economic profits of tasks when executing multiple tasks. Thus,

the combined scheduling of these tasks can improve the performance of task computing. In this paper, we define five interactive constraints for task selection (see Figure 1). The description is as follows:

- Combination: $C1 = \{(T_j,T_k)|T_j \oplus T_k\}$. For a pair of tightly coupled tasks (T_j,T_k), tasks T_j and T_k must be either both selected or both not selected.
- Mutual exclusion: $C2 = \{(T_j,T_k)|T_j \otimes T_k\}$. For a pair of tasks (T_j,T_k), task T_j and task T_k are mutually exclusive because of their monopoly on computing and storage resources. Either task T_j or task T_k can be selected.
- Dependence: $C3 = \{(T_j,T_k)|T_j \Rightarrow T_k\}$. For a pair of tasks with sequential data or calculation requirements (T_j,T_k), task T_j can be selected only if task T_k has been selected.
- Mutual exclusion of economic profits: $C4 = \{(T_j,T_k)|T_j x\% \Theta T_k\}$. For a pair of tasks (T_j,T_k), in the case that both task T_j and task T_k are selected, task T_j reduces the economic profit of task T_k by $x\%$
- Dependence on economic profits: $C5 = \{(T_j,T_k)|T_j x\% \Rightarrow T_k\}$. For a pair of tasks (T_j,T_k), if task T_k is not selected, the economic profit of task T_j is reduced by $x\%$.

The computing tasks are selected within the specified execution time threshold and five interactive constraints of the task, so the execution time is the least, and the economic profits of the task are the highest.

3.2 Profits and execution time under uncertainty

At present, the time of edge computing is considered to be the worst execution time and is given directly by the equipment. Zhang and Wang [61]. Due to unforeseeable problems such as changes in the economic profits of the tasks and the state of

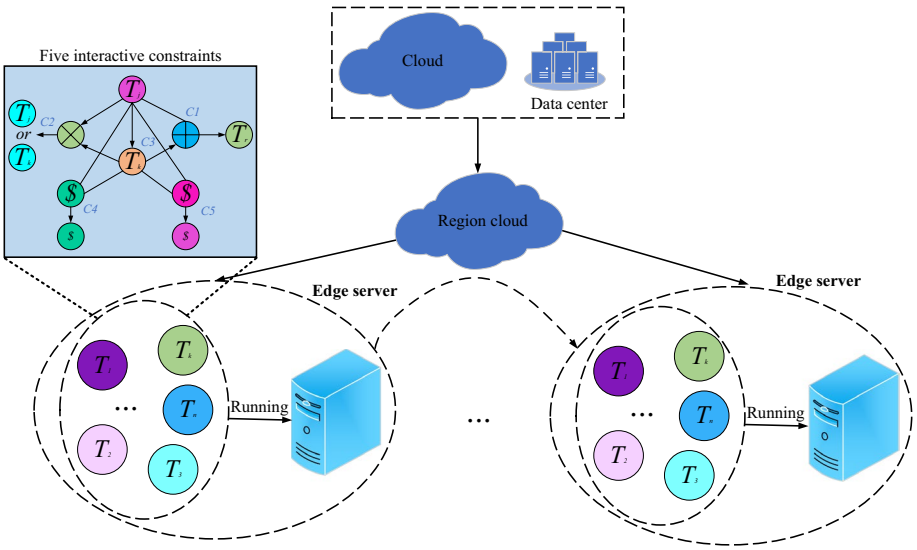


Fig. 1 Five interactive constraints in edge cloud computing tasks. $T_1,T_2,...,T_n$ denote the tasks to be executed on the edge server

operation, it is more reasonable and practical to assign fuzzy values to the economic profit and execution time of each task than to use a certain value. In this study, a triangular fuzzy function is used to solve this kind of problem.

A triangular fuzzy number $\tilde{A}$=(a_1,a_2,a_3) represents the execution time or the economic profit of a task in the process of task selection. Its membership function is shown in (2).

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1}, & x \in [a_1, a_2] \\ \frac{a_3-x}{a_3-a_2}, & x \in [a_2, a_3] \\ 0, & other, \end{cases} \tag{2}$$

where a_1,a_2,a_3 respectively represents the minimum fuzzy number, the most probable fuzzy number and maximum fuzzy number obtained by task T in the edge device. And difference between a_3 and a_1 indicate the degree of fuzziness. The larger the value of a_3-a_1, the stronger the fuzziness.

In fuzzy task selection, the summation, quadrature and distance calculation of triangular fuzzy number are the most important criteria. Let $\tilde{B}$=(b_1,b_2,b_3) and $\tilde{D}$=(d_1,d_2,d_3) be two non-negative triangular fuzzy execution time.

– The addition operation is shown in (3).

$$\widetilde{B} + \widetilde{D} = \left(b_1 + d_1, b_2 + d_2, b_3 + d_3\right), \tag{3}$$

– The multiplication operation is shown in (4).

$$\widetilde{B} \times \widetilde{D} = \left(b_1 \times d_1, b_2 \times d_2, b_3 \times d_3\right), \tag{4}$$

– The distance calculation is shown in (5)

$$D(\widetilde{A}_\alpha, \widetilde{B}_\alpha) = \sqrt{\left[\frac{1}{2}\sum_{i=1}^{n}\left|B^-_{\alpha_i} - D^-_{\alpha_i}\right|^2 + \frac{1}{2}\sum_{i=1}^{n}\left|B^+_{\alpha_i} - D^+_{\alpha_i}\right|^2\right]}, \tag{5}$$

the α-cut of the triangular fuzzy number $\widetilde{A}$ =(a_1,a_2,a_3) is given by $[\widetilde{A}]_\alpha$ (see (6)), which is used to compare two fuzzy numbers.

$$\widetilde{A}_\alpha = \left[\left(a_2 - a_1\right)\alpha + a_1, a_3 - \left(a_3 - a_2\right)\alpha\right], \tag{6}$$

in the calculation of execution time using (5), $\widetilde{B}$ and $\widetilde{D}$ are compared with fuzzy number $\widetilde{0}$=(0,0,0) respectively. If $D_{2,\frac{1}{2}}(\widetilde{B}, \widetilde{0}) \leq D_{2,\frac{1}{2}}(\widetilde{D}, \widetilde{0})$, then $\widetilde{B}$ dominate $\widetilde{D}$. At this time, the calculated fuzzy number will be integrated with the time value of the fuzzy operation. If the execution time is less than the time threshold of the task execution time, it is acceptable; otherwise, the solution is invalid.

In the HWOA algorithm, each solution is encoded into a binary vector such as **s** = [0,0,1,...,0], the value in the vector represents a task, and '1' means that the task will be selected in the next schedule, and '0' means that the task will not be selected.

3.3 Task selection

When an edge device processes a set of tasks, each task consumes a certain execution time and generates a certain economic profit. If the execution time is infinite, the sequential execution of a group of tasks will produce fixed economic profits. The task selection in our research requires reasonable selection and execution of tasks with interactive constraints within the time threshold to achieve the greatest economic profits. Thus, task selection can be defined as the consideration of a variety of constraints between tasks when a limited edge device is processing a set of tasks. When making a reasonable choice of tasks, resources should be allocated to achieve the optimal objective. The goal of task selection is to reduce task execution time in edge computing and increase the economic profits brought about by tasks.

Because the task selection process of edge computing needs to take into account multiple objectives at the same time, the problem is formalized as a multi-objective knapsack problem [62] in this study. For the task selection joint optimization problem, with the goal of maximizing the economic profit and minimizing the execution time of the task, a hybrid fuzzy programming model that satisfies the execution time threshold and task interaction is established. Each edge device is limited to only being able to select 0 or 1 time for each task at a time and record it with x_j, which is called the multi-objective 0-1 knapsack problem. The details are given in (7).

$$\begin{cases} \min \sum_{j=1}^{n} \widetilde{\mu_{\tilde{A}}(\tilde{e}_j)} \times x_j \\ \max \sum_{j=1}^{n} \mu_{\tilde{A}}(\tilde{s}_j) \times x_j \\ \text{subject to } \sum_{j=1}^{n} \tilde{e}_j \times x_j \leq Threshold, \end{cases} \tag{7}$$

where n is the number of tasks, $\mu_{\tilde{A}(\tilde{e}_j)}$ is the fuzzy execution time, and $\mu_{\tilde{A}(\tilde{s}_j)}$ is the economic profit of the fuzzy task. In the edge computing optimization problem, the economic profit and execution time of the task are the optimization objectives, and the solution process involves searching for a set of solutions in the feasible design space to optimize the two objective functions at the same time.

3.4 WOA description

The WOA uses a set of random candidates to update and improve the position of the candidate solution by encircling prey, bubble-net attacking, and searching for prey randomly in each step. Assuming that the size of the whale population is N and the dimension of the problem-solving space is V, then the corresponding position of the j th whale in the r-dimensional space is $S_j = (s_j^1, s_j^2, ..., s_j^r, j = 1, 2, 3..., N)$, and the position of the optimal whale corresponds to the global optimal solution.

3.4.1 Encircling prey

During the hunting process, the whales can identify the location of the prey and surround it. Assuming that the task selection solution to be optimized and the problem variable are the position of the optimal whale closest to the prey, after the position of the optimal whale

is defined, the remaining whales calculate the distance between the individual and the optimal whale (see (8)) and then try to update their respective positions toward the optimal whale. The specific update method is shown in (9).

$$\vec{\mathbf{D}} = \left|\vec{\mathbf{C}} \times \vec{X^*}(t) - \vec{\mathbf{X}}(t)\right|, \tag{8}$$

$$\vec{\mathbf{X}}(t+1) = \vec{X^*}(t) - \vec{\mathbf{A}} \times \vec{\mathbf{D}}, \tag{9}$$

$$\vec{\mathbf{C}} = 2 \times \mathbf{r}, \tag{10}$$

$$\vec{\mathbf{A}} = 2 \times \vec{\mathbf{a}} \times \vec{\mathbf{r}} - \vec{\mathbf{a}}, \tag{11}$$

where $\vec{\mathbf{C}}$ is the swing factor, calculated by (10); $\vec{\mathbf{A}}$ is the convergence factor, calculated by (11); $\mathbf{r}$ is a random vector between [0,1]; $\mathbf{a}$ linearly decreases from 2 to 0 in the iterative process; t is the current number of iterations; $\vec{X^*}(t)$ is the position of the optimal whale at the t th iteration, which is updated in each iteration; and $\mathbf{X}(t)$ is the current individual whale's position at the t th iteration.

3.4.2 Bubble-net attacking method

During bubble-net attacking, a whale has two feeding mechanisms, the contraction encircling mechanism and the spiral renewal mechanism, which occur in a random manner with a probability of 50% each.

- Shrinking encircling mechanism. This behavior is achieved by lowering the value of $\mathbf{a}$ in (11). The value of the convergence factor $\mathbf{A}$ decreases with the decrease of $\mathbf{a}$, so $\mathbf{A}$ is the random vector of [-a, a]. When $\mathbf{A} \in [-1,1]$, the new position of the whale can be defined at any position between the original position of the whale and the current optimal position of the whale.
- Spiral updating position. This behavior is achieved by first calculating the distance from the position of the j th whale (X,Y) to the optimal position (X^*,Y^*) (see (12)). Then a spiral equation is established between (X,Y) and (X^*,Y^*) to update the position (see (13)).

$$\vec{D^*} = \left|\vec{X^*}(t) - \vec{X}(t)\right|, \tag{12}$$

$$\mathbf{X}(t+1) = e^{bl} \times \cos(2\pi l) \times \vec{D^*} + \vec{X^*}(t), \tag{13}$$

where b is the spiral constant, set to 0.618, and l is a random value between [-1, 1].

3.4.3 Searching for prey

In addition to updating its position based on the optimal whale position, each whale in the population will also update its position by randomly searching for prey globally based on the positions of the remaining whales under a certain random ratio when the convergence factor $|\mathbf{A}| > 1$. The specific update method is shown in (14) and (15).

$$\mathbf{D} = \left|\mathbf{C} \times \overrightarrow{X_{\text{rand}}} - \mathbf{X}\right|, \tag{14}$$

$$\mathbf{X}(t+1) = \overrightarrow{X_{\text{rand}}} - \mathbf{A} \times \vec{D}, \tag{15}$$

where $\overrightarrow{X_{rand}}$ represents the location of a random whale in the current whale population.

3.5 HWOA architecture

The HWOA algorithm proposed in this paper is a hybrid algorithm, which is based on the whale algorithm as the overall framework. It incorporates the strategies of avoiding natural enemies from the dragonfly algorithm as well as the mutation strategy and greed strategy from the genetic algorithm, and it is also combined with NSGA-II. HWOA is designed to solve the multi-objective edge computing task selection knapsack problem, and the framework of the algorithm is shown in Figure 2.

Different from the traditional WOA, after initializing the whale population, it uses the NSGA-II algorithm to find the optimal whale individual in HWOA. Although natural enemies are not involved in WOA, the principle of dragonfly algorithm is used in HWOA, and the position of the worst solution is regarded as the position that needs to avoid natural enemies. In addition to avoiding the worst individual when updating the position of whales, HWOA also uses the idea of mutation to obtain the best whale individual and invert the worst whale individual to enrich the diversity of individuals. In the WOA algorithm, with each iteration, the differences between whale individuals decrease. In order to make up for WOA's poor local search ability and its tendency to fall into local optimization, HWOA uses the greedy strategy to modify the feasible task solution and infeasible task solution in the current iteration.

3.5.1 Initialization

Binary code is used to randomly initialize the positions of the whales in the whale population. The initial position of the j th whale is represented by a one-dimensional vector containing r elements $S_j = (s_j^1, s_j^2, ..., s_j^r)$, where j =1,2,3,...,N, $s_j^t \in (0, 1)$ and $1 \leq t \leq r$. According to the property of the 0-1 knapsack problem, the initialization rule is shown in (16). When s_j^t is 1, the task is selected to be put into the knapsack. On the contrary, the task is not selected in the next scheduling when s_j^t is 0.

$$s_j^t = \begin{cases} 0, \text{ rand } < 0.5 \\ 1, \text{ rand } \geq 0.5 \end{cases} \tag{16}$$

3.5.2 Optimal whale selection based on multi-objective task

After the whale population is initialized, the fitness of each individual whale is calculated by NSGA-II to identify the optimal individual whale. NSGA-II is a multi-objective optimization algorithm with fast calculation speed, especially for two-objective optimization problems. Its fast non-dominated ordering of individuals divides the population into different non-inferior frontiers, which can improve the convergence of feasible solutions.

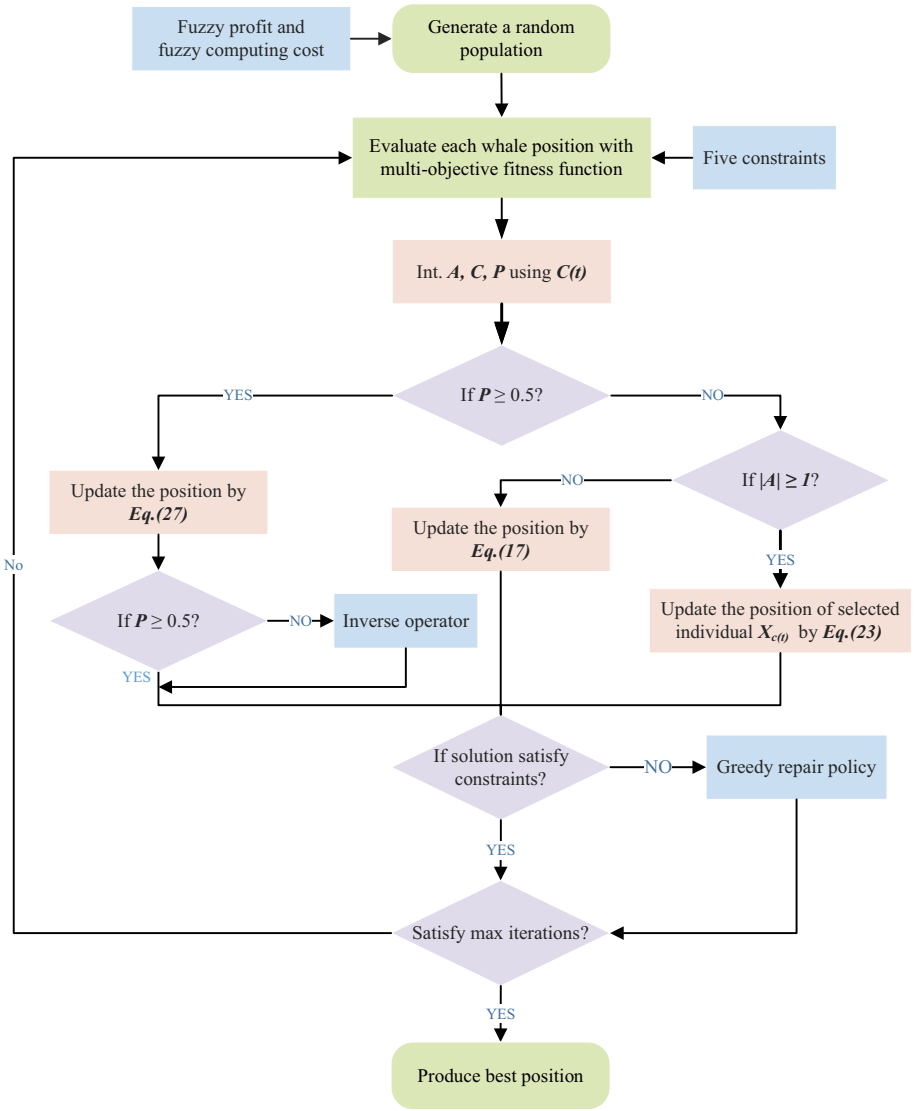


Fig. 2 Overview of HWOA. HWOA is presented for multi-objective task selection problem in edge computing. Some improvement search and multi-objective optimization strategies are presented

It then finds the optimal individuals based on the crowding distance so that the retained solutions have the lowest similarity and maintain the diversity of the solution space.

Fast non-dominated sorting classifies the whale population and guides the search toward the Pareto optimal solution set according to the hierarchical order from best to worst. Assuming that the size of the whale population is N, and j represents an individual in the whale population, the number of individuals n_j in the population that dominates the whale j and the number of individuals S_j in the population that is dominated by the whale j are calculated. The fast non-dominance sorting procedure is as follows:

- First, store all individuals of $n_j = 0$ in the population in F_1, where F_1 is the current set and is defined as the Pareto front of the optimal layer.
- For each individual j in F_1, traverse the individuals dominated by j, that is, each individual l in the set S_j. For $n_l = n_l$-1, save it in the set H when n_l=0.
- With H as the current set, repeat the previous step until the entire whale population is stratified.

After the superiority and inferiority levels are arranged, the whale individuals j in each level need to be ranked by calculating the crowding distance. The calculation steps are as follows:

- Initialize the crowding distance of individuals in the same front layer as 0 (i.e., $L[j]_d = 0$).
- Rank the individuals of the same level in ascending order of the value of the M objective function.
- Set the crowding distance of individuals at the boundary so that they are always selected.
- Calculate the crowding distance of other individuals except the individuals at the boundary, as shown in (17).

$$L[j]_d = \frac{L[j]_d + \left(L[j+1]_M - L[j-1]_M\right)}{f_M^{\max} - f_M^{\min}}, \tag{17}$$

  where $L[j+1]_M$ is the function value of the individual $j+1$ on the M th target; $f_M^{\max}$ is the maximum achieved by the whale population on the M th target; and $f_M^{\min}$ is the minimum value achieved by the whale population on the M th target.
- Repeat the above four steps until each objective function is calculated. Then the crowded distances of individual whales are obtained, and a better solution corresponds to a larger distance value.

3.5.3 Strategy for avoiding natural enemies

In the partial meta-heuristics optimization algorithm, individuals need to avoid natural enemies to survive. Although WOA does not involve natural enemies, this paper uses the principle of the dragonfly algorithm for reference and regards the position of the worst solution as the position that needs to avoid natural enemies. The specific position updating principle is shown in (18) and (19).

$$\mathbf{X}(t+1) = \Delta\overrightarrow{X_{t+1}} + \overrightarrow{X^*}(t), \tag{18}$$

$$\Delta\overrightarrow{X_{t+1}} = \left(e^{bl} \times \cos(2\pi l) \times \overrightarrow{D^*}\right) + \left(\varepsilon\left(\overrightarrow{X^-}(t) + \mathbf{X}(t)\right)\right) + w^\circ \Delta\overrightarrow{X_t}, \tag{19}$$

where $\Delta \boldsymbol{X}_{t+l}$ is the updating mode of the whale step vector, which consists of the optimal individual attractiveness (see (13)) and the worst individual avoidance and inertia updating; ε is the weight factor of the worst individual; w° is the inertia weight, which adjusts itself in the process of optimization; $\mathbf{X}^-(t)$ represents the position of the worst whale in the t th iteration; and $\mathbf{X}(t)$ is the position of the current whale individual during the t th iteration.

3.5.4 Inverse operator mechanism

In each evolution, genetic algorithms try to keep the best individuals in the population, eliminate the worst individuals, and carry out crossover or variation among the best individuals. In addition to avoiding the worst individuals when updating the position of the whale, HWOA also uses the mutation idea of the genetic algorithm. While obtaining the best whale individual, the worst whale individual is subjected to the inverse operator operation, as shown in Figure 3. A value of '1' indicates that the task is put into the knapsack in the next schedule. This operation can enrich the diversity of individuals to a certain extent, but it cannot guarantee the emergence of a relatively superior individual.

3.5.5 Greedy restoration strategy

With each iteration, the differences between individual whales decrease, causing the algorithm to fall into a local solution. In order to make up for WOA's poor local search ability and its tendency to fall into local optimization as much as possible, the greedy strategy is used to repair the feasible task solution and the infeasible task solution in the current iterative scheduling, which promotes better and faster convergence to the global optimal solution.

In most previous studies, only the repair of infeasible task solutions is considered. However, for the feasible task solution, the total execution time of the tasks put into the knapsack is less than the execution time threshold, so there may be cases in which the capacity of the knapsack is not fully used. Therefore, the greedy strategy is used, which consists of two steps. The first step is to traverse the tasks that are not put into the knapsack. If there is a task that is not in the knapsack at the time of initialization, and its total execution time is still less than the execution time threshold after being put into the knapsack, the task is added to the knapsack. The second step is when the traversal is over, if the total execution time of the task put in the knapsack is still less than the execution time threshold, the economic profit of unit execution time is calculated according to (20), and the value is judged in descending order whether the task can be put into the knapsack to correct the current solution set.

$$u = \frac{\text{profit}}{\text{time}}. \tag{20}$$

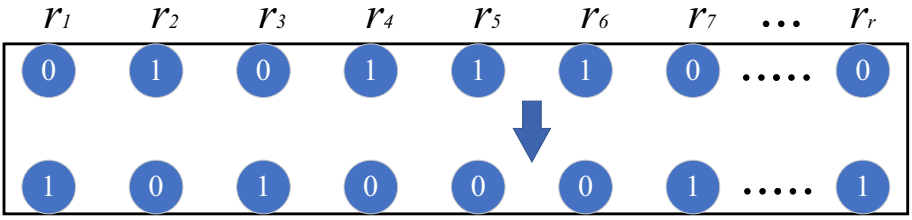


Fig. 3 Inverse operator operation

Algorithm 1 Pseudo-code of the HWOA algorith.

```
Input: N, M.   /*N is the population size. M is the maximum of iterations*/
Output: X*.   /*the best solution*/
Initialize S_j = (s_j^1, s_j^2, ..., s_j^r), j=1,2,3,...,N.   /*the whale population */;
while t < M do
    for X^- do
        Inverted the individual.
    for each solution do
        Update a, l, p, C, |A|, ε, w°;
        if p < 0.5 then
            if |A| < 1 then
                update the position by (9).
            if |A| ≥ 1 then
                select a search position X_c(t); update the position by (15).
        if p ≥ 0.5 then
            update the position by (19).;
            if p < 0.5 then
                Inverse operator.
    Check and amend the solution by modified greedy repair policy if it doesn't
    satisfy the constraints.;
    Fast non-dominant sort and crowding-distance sort.;
    Update X* if there is a better solution.;
    t = t + 1
return X*
```

Time complexity analysis The number of repetitions of basic operations in HWOA is a function f_n of the problem size n. We analyze the variation of f_n with n and determine the order of magnitude of time complexity T_n. Assume that the size of the whale population is NP, the dimension is D, and the number of iterations is $Maxiter$. Referring to the principle and flow chart of HWOA for analysis, the time complexity of HWOA can be calculated in four steps. In the first step, the time complexity of the initialization method based on the whale population is $O(NP \times D)$. In the second step, the time complexity of inverting the individual whale is $O(NP \times D)$; the time complexity of updating the population position is $O(NP \times D \times Maxiter)$; In the third step, the time complexity of checking and amending the solution by modified greedy repair policy is $O(NP \times D)$. In the fourth step, the time complexity of fast non-dominated sorting and crowded distance sorting based on NSGA-II is $O(NP \times \log^{h-1} NP)$, where h is the number of objective functions, and the number of objective functions in HWOA is 2. To sum up, the time complexity of HWOA after iterating $Maxiter$ times is $O(NP \times D \times \log^{h-1} NP)$.

4 Experiment

To verify the performance of the proposed algorithm, two types of experiments are implemented with different datasets. In the first type of experiment, fuzzy type values are used to verify the performance of the algorithm for the uncertainty of economic profits and

execution time. In the other type of experiment, deterministic task problem arithmetics are used to perform experimental analysis and prove the performance of the proposed method. The programming language is Python, and the experimental platform is an Intel Core i7 processor, 16 GB RAM, and a 64-bit Windows server. Table 1 shows the parameters of the experiment. The time limit t is the percentage of the total execution time set in the experiment as the time threshold; weight w is the weight of each dimension of the whale population; whale number *num* is the number of whale population; whale dimension *dim* is the dimension of initialized whale population, and *iter* is the number of iterations set in the experiment.

4.1 Edge computing task selection experiments

For the edge computing task selection problem, in order to verify the performance of the proposed algorithm with fuzzy and constrained relationships, experiments are conducted on two datasets of small to medium size.

4.1.1 Datasets

The two edge computing task datasets were composed of 24 and 72 tasks and were denoted as ECTD-1 [7] and ECTD-2 [8], respectively. The data included the maximum, minimum, and the most probable value of economic profits and execution times under the computation task, which were converted into fuzzy numerical representations by the triangular fuzzy function. The economic profit of any task like T_j is calculated by the triangular fuzzy function and is considered as (p_{1j},p_{2j},p_{3j}), in the same way, the execution time of any task like T_j are calculated by the triangular fuzzy function and is considered as (t_{1j},t_{2j},t_{3j}). The resulting data were the fuzzy economic profits and execution times for the different tasks in the edge device.

The first edge computing task dataset contained 24 tasks and 12 interaction constraints between tasks. The economic profits and execution times for all of the values were normalized to the interval 1 to 5, and the specific data statistics and constraint descriptions are listed in Table 7. The second dataset contained 72 tasks and 20 interaction constraints between tasks. The economic profits and execution times were normalized to the interval 1 to 10, and the specific data statistics and constraint descriptions are listed in Table 8.

In the edge computing task ECTD-1, there were 12 constraints between edge computing tasks. Take tasks T2 and T9 as an example. The combined constraint for these tasks indicates that either both tasks or neither task can be scheduled by one edge

Table 1 Parameters of the experiment

Parameters	Values
Time limits (t)	20%, 40%, 60%, 70%, 80%, 100%
Weight (w)	1, 2.8, 4
α	10
Whale number (*num*)	300
Whale dimension (*dim*)	72, 24
Iterations (*iter*)	200

device at any given time. As another example, the economic profits between tasks T4 and T7 are mutually exclusive, and the selection of task T7 reduces the economic profit of task T4 by 30%.

4.1.2 Evaluation metrics

In this study, four metrics are selected to evaluate the performance of HWOA in edge computing task selection experiments: the algorithm execution time (ET), the number of non-dominant solutions (NDSs), the diversity metric (Δ-spread), and the hypervolume (HV).

Δ-spread is defined in (21).

$$\Delta - spread = \frac{d_f + d_l + \sum_{i=1}^{Num-1} |d_i - \bar{d}|}{d_f + d_l + (Num - 1)\bar{d}}, \tag{21}$$

where d_f is the euclidean distance between the best solution and the first solution in the non-dominated solution set; d_l is the euclidean distance between the best solution and the last solution in the non-dominated solution set; *Num* is the number of non-dominated solutions found so far; d_i is the euclidean distance between two neighboring points in the non-dominated solution set; and $\bar{d}$ is the mean distance between each pair of neighboring solutions.

HV is defined in (22).

$$HV = \text{ volume } \left(U_{i=1}^{|NDS\_\text{archive}|} V_i \right), \tag{22}$$

where *volume* is the Lebesgue measure for volume; |*NDS_archive*| is the number of non-dominated solutions; and V_i is the volume of the hypercube formed by the reference point and the *i* th solution in the set of non-dominated solutions. Taking the two-dimensional case as an example, the geometric representation of HV is shown in Figure 4. From the figure, the non-dominated solutions in the solution set are P_1,P_2,P_3. P_1 dominates the P_1QCR region, and, similarly, the dominated regions of P_2 and P_3 can be found. Then HV in the data dimension is calculated as shown in (23).

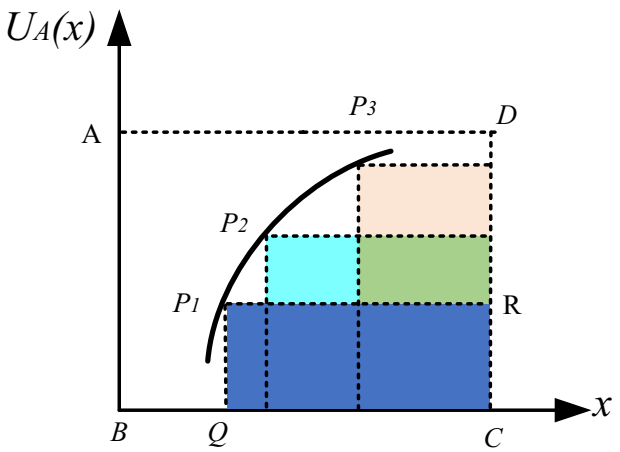


Fig. 4 Geometric interpretation of HV

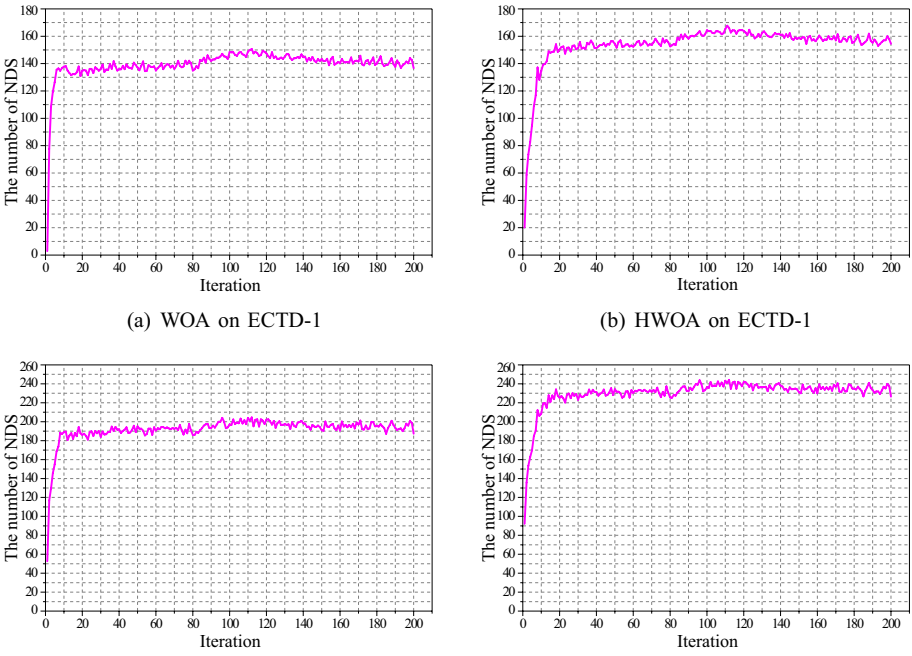


Fig. 5 The number of NDSs using WOA and HWOA on ECTD-1 and ECTD-2. (**a**) the number of NDSs using WOA on ECTD-1. (**b**) the number of NDSs using HWOA on ECTD-1. (**c**) the number of NDSs using WOA on ECTD-1. (**d**) the number of NDSs using HWOA on ECTD-2

$$HV = \frac{V_{NDS}}{V_{ABCD}}, \tag{23}$$

where V_{NDS} is the concatenation of the dominating spaces of P_1,P_2,P_3, and V_{ABCD} is the total area of the solution space.

4.1.3 Results and analysis

For both datasets, six execution time limits (20%, 40%, 60%, 70%, 80%, and 100% of the total execution time) are set for the experiments. Figure 5 shows a graphical representation of the number of NDSs calculated per iteration for both datasets using WOA and HWOA. It shows that the number of NDSs increases rapidly with iteration, indicating that the algorithm detects almost all of the NDSs. In some iterations, the number of NDSs fluctuates slightly. The reason is that the discovery of a new NDS in the current iteration will cause some NDSs in the previous iteration to fail to be discovered. However, as the number of iterations increases, the number of NDSs eventually continues to increase. Figure 5(a) and Figure 5(c) show that the number of non-dominated solutions becomes stable when the number of iterations is beyond 10 using the WOA. Figure 5(b) and Figure 5(d) shows that the number of non-dominated solutions becomes stable when the number of iterations is beyond 20 using the HWOA. The reason is that the proposed method has good performance and can find the optimal solution in a small number of iterations. We integrate the genetic

algorithm to solve the premature of HWOA. However, due to the property of the problem and the defects of the whale algorithm, we have improved the premature of HWOA to a certain extent, but we cannot avoid it completely. By comparing the experiment results of HWOA and WOA, we believe that HWOA improves on the premature convergence problem and it indicates the proposed approach achieves remarkable performance.

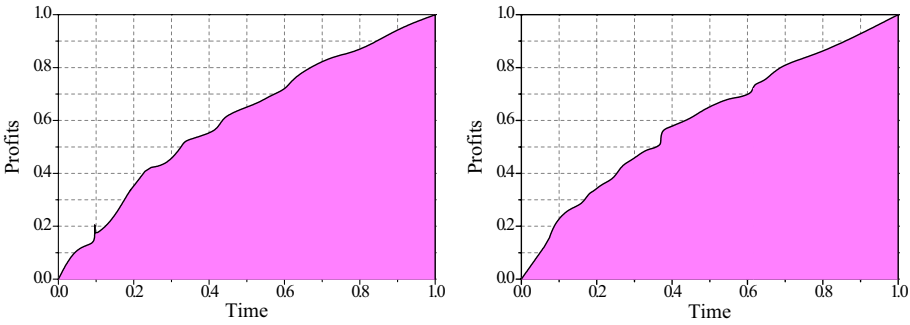
Table 2 illustrates the results of HWOA solving the multi-objective task selection problem. In order to further analyze the performance of the algorithm, six different execution time thresholds are used during the experiments. For both datasets, the greater the constraint on time, the smaller the number of NDSs. The reason is that in the case of limited execution time, the number of higher-profit solutions for the task will be reduced. Without the limitation of time boundary, the value of HV is greater than the five cases of time limitation, showing the diversity and convergence of the obtained Pareto front. The solution in the Pareto front is better than the solution with a time boundary. The Δ-spread shows the uniformity of the distribution of the NDSs solved by the proposed method. The execution time is longer due to the larger size of the ECTD-2 dataset and the more constraints. For both datasets, the computation with a time constraint of 20% takes the longest to execute due to the fact that a large number of infeasible solutions need to be turned into feasible solutions by the repair strategy in the time-constrained computation, resulting in a longer execution time for the case with a larger time constraint.

Figures 6 and 7 show the HV diagrams of the ECTD-1 and ECTD-2 datasets and Pareto front graphs of the ECTD-1 and ECTD-2 datasets with different time boundary.

To further evaluate the performance of HWOA in solving the edge cloud task selection problem, the results of HWOA on datasets ECTD1 and ECTD2 are compared with the WOA, the adaptive cellular particle swarm optimization (ACPSO), the hybrid bat algorithm (HBA), the ant colony optimization algorithm (ACO), multi-objective evolutionary algorithm (MOEA) and multi-objective immune algorithm (MOIA) in terms of the algorithm execution time (ET), the number of non-dominant solutions (NDSs), the diversity metric (Δ-spread), and the hypervolume (HV). Table 3 shows the comparison results. It indicates the proposed method has a significant improvement in performance compared with other competitive methods.

Table 2 Performance of HWOA on ECTD-1 and ECTD-2 datasets with different time boundary

Dataset	NDS	HV	Δ-spread	ET(s)
ECTD-1				
Time boundary 20%	32	68.2%	0.55	6.89
Time boundary 40%	42	69.6%	0.53	6.48
Time boundary 60%	78	69.8%	0.52	6.46
Time boundary 70%	94	69.9%	0.50	6.42
Time boundary 80%	143	70.8%	0.49	6.40
Without time boundary	152	71.5%	0.48	6.02
ECTD-2				
Time boundary 20%	89	68.3%	0.54	58.92
Time boundary 40%	126	69.8%	0.52	56.78
Time boundary 60%	142	70.1%	0.50	54.23
Time boundary 70%	148	71.3%	0.49	51.32
Time boundary 80%	198	72.1%	0.47	50.42
Without time boundary	227	72.4%	0.46	49.45



(a) HV diagram of ECTD-1 (b) HV diagram of ECTD-2

Fig. 6 HV diagrams without a time boundary on the ECTD-1 and ECTD-2 datasets

4.2 The case of knapsack problem

In edge computing, the task selection problem can be modeled as a knapsack problem. In order to further verify the performance of the proposed algorithm, nine 0-1 knapsack problem cases [63] are used to conduct experiments.

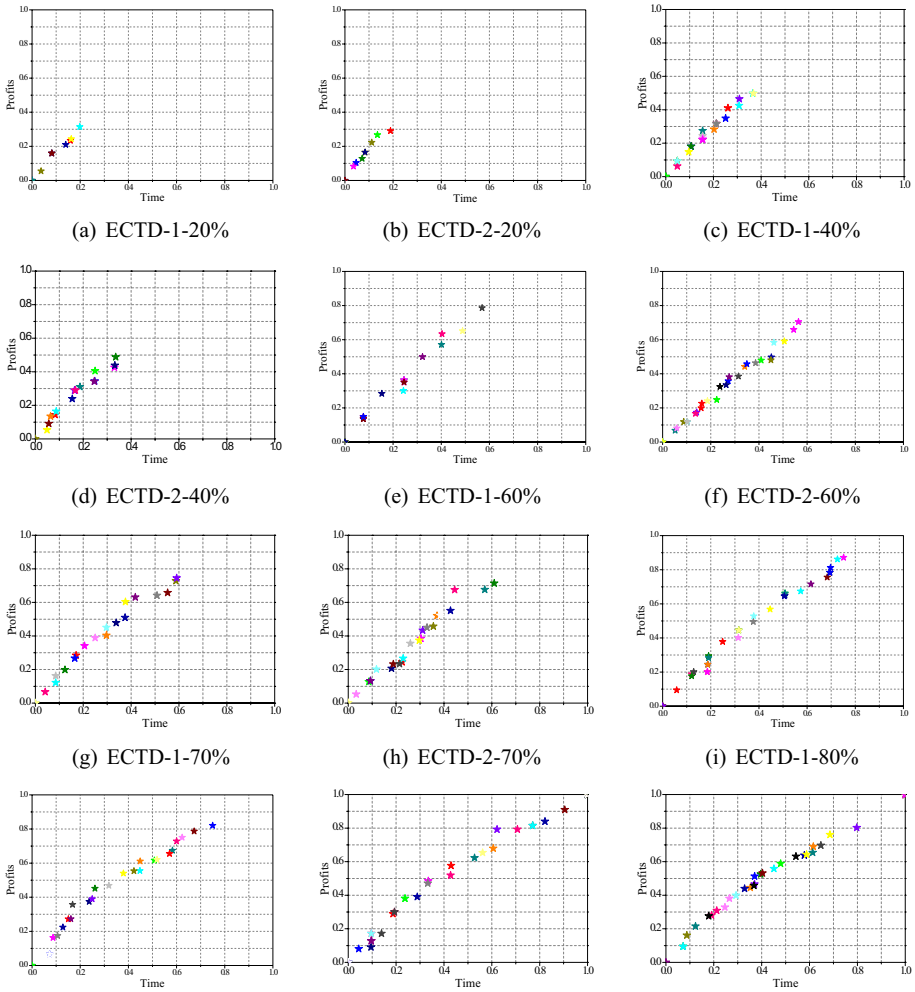
4.2.1 Case description

Each of the nine 0-1 knapsack cases (i.e., KP1-KP9) consists of the dimension of the arithmetic, the set of task execution time, the set of task profits, the time of the knapsack, and the known optimal values. And each task corresponds to an execution time. The total knapsack time limits the total time for all tasks to be executed. The optimal value is the maximum profit that can be obtained from the task given the total duration constraint. Thus, the cases we are trying to solve can be briefly described as follows: given a set of tasks with two attributes of execution time and profits, in the case of limiting the total execution time, find the subset of tasks that maximizes the total profits.

4.2.2 Algorithm performance

The results of HWOA are compared with the adaptive cellular particle swarm optimization (ACPSO) [64] and hybrid bat algorithm (HBA) [65] in Table 4. The results of HWOA are better than ACPSO in all of the cases except for case KP9, indicating that the global search capability of HWOA is better than that of ACPSO. Except for cases KP2, KP4, and KP6, the results of HWOA are comparable to HBA. In terms of the number of successes, HWOA reaches the optimal value in 30 tests for all cases except KP6 and KP9. And the number of times HWOA reaches the optimal value is higher than that of ACPSO for all of the cases, which indicates that HWOA has a stronger robustness in searching. The number of iterations of HWOA is lower than that of ACPSO and HBA for all of the cases except KP2 and KP4, which indicates that the convergence speed of HWOA is better than that of ACPSO and HBA. It also shows that HWOA offers better solution performance for high-dimensional data.

For the comparison with WOA, four cases with different dimensions (i.e., KP3, KP4, KP6, and KP9) are randomly selected from the nine cases. The results are shown in



(a) ECTD-1-20% (b) ECTD-2-20% (c) ECTD-1-40%

(d) ECTD-2-40% (e) ECTD-1-60% (f) ECTD-2-60%

(g) ECTD-1-70% (h) ECTD-2-70% (i) ECTD-1-80%

(j) ECTD-2-80% (k) ECTD-1-NO (l) ECTD-2-NO

Fig. 7 Pareto front graphs of the ECTD-1 and ECTD-2 datasets with different time boundary. (**a**), (**c**), (**e**), (**g**), (**i**) show the Pareto front graphs of the ECTD-1 with 20%, 40%, 60%, 70% and 80% time boundary respectively. (**b**), (**d**), (**f**), (**h**), (**j**) show the Pareto front graphs of the ECTD-2 with 20%, 40%, 60%, 70% and 80% time boundary respectively. (**k**) and (**l**) show the Pareto front graphs of ECTD-1 and ECTD-2 without time boundary respectively

Tables 5 and 6. Table 5 presents the worst value, average value, and best value solved by HWOA and WOA algorithms in four cases when the maximum number of iterations is reached. The optimal values show that HWOA can find the optimal solution for all four cases, whereas WOA can only find the optimal solution for the low-dimensional cases KP1 and KP2. This indicates that the HWOA algorithm has, to a certain extent, overcome the shortcomings of WOA, particularly the shortcoming of falling into local optima. Therefore, the ability of HWOA to find the optimal solution is better than that of WOA. The success ratio in Table 6 shows that HWOA can achieve 100% optimal

Table 3 The comparison of meta-heuristic algorithms on ECTD-1 and ECTD-2 datasets

Dataset	Algorithm	NDS	HV	Δ-spread	ET(s)
ECTD-1	WOA	136	68.5%	0.57	6.55
	ACPSO	142	67.3%	0.58	7.43
	HBA	133	67.6%	0.55	6.53
	ACO	136	66.6%	0.59	7.12
	MOEA	147	68.8%	0.49	8.42
	MOIA	139	65.4%	0.59	7.25
	HWOA	152	70.5%	0.48	6.02
ECTD-2	WOA	188	69.8%	0.53	51.82
	ACPSO	195	67.5%	0.56	56.55
	HBA	184	68.2%	0.57	53.24
	ACO	178	67.8%	0.59	57.12
	MOEA	198	69.6%	0.53	58.59
	MOIA	182	67.4%	0.51	47.63
	HWOA	227	71.3%	0.46	49.45

solutions for both KP1 and KP5, and the success ratio of solving all four cases is higher than the WOA. Combining the two experimental metrics, it can be seen that the success ratio and solution robustness of HWOA are better than those of WOA.

In order to show the performance of HWOA more intuitively, the iterative process of solving four cases by WOA and HWOA is shown in Figure 8. As shown in the figure, HWOA can find the optimal value within a small number of iterations, whereas WOA falls into a local optimum after a certain number of iterations. HWOA overcomes the shortcomings of WOA, that is, being prone to premature maturity and slow convergence, and its global search ability and convergence speed are much better than those of WOA.

5 Conclusion

In this study, the limited resources of the edge computing are considered and formulated as a multi-objective task selection problem. To the best of our knowledge, this is the first time uncertain economic profits, and time costs have been investigated as the optimization objectives. A meta-heuristic algorithm called HWOA is proposed for task selection in the edge cloud computing system which is a multi-objective optimization task with multiple interaction constraints among tasks. The uncertainty of economic profits and computation times of edge computing tasks over time is modeled by a triangular fuzzy function. Based on the WOA, a strategy for avoiding natural enemies and an inverse replacement operator are designed to improve convergence speed and convergence accuracy. In addition, a greedy restoration strategy is leveraged to correct the infeasible task solutions. The performance of the proposed HWOA is evaluated by two datasets of different scales of edge computing tasks with fuzzy constraints, and nine deterministic cases. However, HWOA introduces several improved strategies that make the solution for large sets of tasks expensive in computational time, and we do not pay attention to the security and privacy in our work. In the future, this research will explore methods to reduce the time overhead of solving large task sets in edge cloud computing systems. Multi-objective optimization for allocating high-dimensional tasks to multiple devices and the extension to the edge network with dynamic constraints is also valuable to explore.

Table 4 The comparison of HWOA, ACPSO and HBA on case KP1-9

Example	Dimension	Known optimal solution	Methods	Best	Average	Worst	Number of successes	Min-iterations	Avg-iterations	Max-iterations
KP1	10	295	HWOA	295	295	295	30	1	1.00	1
			ACPSO	295	295	295	30	1	1.80	6
			HBA	295	295	295	30	1	1.00	1
KP2	20	1024	HWOA	1024	1024	1024	30	1	4.73	22
			ACPSO	1024	1021	1018	16	1	194.19	441
			HBA	1024	1024	1024	30	1	1.43	4
KP3	20	1042	HWOA	1042	1042	1042	30	1	24.73	156
			ACPSO	1042	1039.3	1037	14	49	235.00	474
			HBA	1042	1042	1042	30	1	27.23	142
KP4	50	4882	HWOA	4882	4881	4878	30	1	20.53	59
			ACPSO	4882	4857.1	4834	5	91	133.93	421
			HBA	4882	4882	4882	30	1	6.73	39
KP5	100	15170	HWOA	15170	15170	15170	30	1	1.00	1
			ACPSO	15170	15170	15170	30	1	1.00	1
			HBA	15170	15170	15170	30	1	1.83	2
KP6	100	26559	HWOA	26559	26543	26534	7	2	6.70	48
			ACPSO	26559	26527	26525	1	45	45.00	45
			HBA	26559	26551	26534	14	2	10.43	103
KP7	100	2660	HWOA	2660	2660	2660	30	1	1.00	1
			ACPSO	2660	2660	2660	13	20	190.38	417
			HBA	2660	2660	2660	30	1	2.47	5
KP8	100	4143	HWOA	4143	4143	4143	30	1	1.00	1
			ACPSO	4143	4143	4143	30	4	58.83	162
			HBA	4143	4143	4143	30	3	4.57	9
KP9	100	4987	HWOA	4987	4986	4981	10	1	7.52	50
			ACPSO	4986	4986	4986	30	1	1.00	1
			HBA	4987	4986.5	4986	15	2	4.73	33

Table 5 Comparison of the worst value, average value, and best value solved by HWOA and WOA algorithms in four cases

Case	Dimension	Known optimal solution	Methods	Worst value	Average value	Best value
KP3	20	1042	HWOA	1042	1042	1042
			WOA	879	1021	1042
KP4	50	4882	HWOA	4878	4881	4882
			WOA	3969	4564	4876
KP6	100	26559	HWOA	22433	26322	26559
			WOA	18568	23125	26230
KP9	100	4987	HWOA	4880	4966	4987
			WOA	4023	4698	4890

Table 6 Comparison of convergence between WOA and HWOA in four cases

Case	Dimension	Methods	Min-iteration	Avg-iteration	Success ratio
KP3	20	HWOA	1	1	30/30
		WOA	1	3.2	21/30
KP4	50	HWOA	1	2.2	30/30
		WOA	1	8.4	12/30
KP6	100	HWOA	1	20.53	22/30
		WOA	0	0	0/30
KP9	100	HWOA	1	8.6	28/30
		WOA	1	18.6	8/30

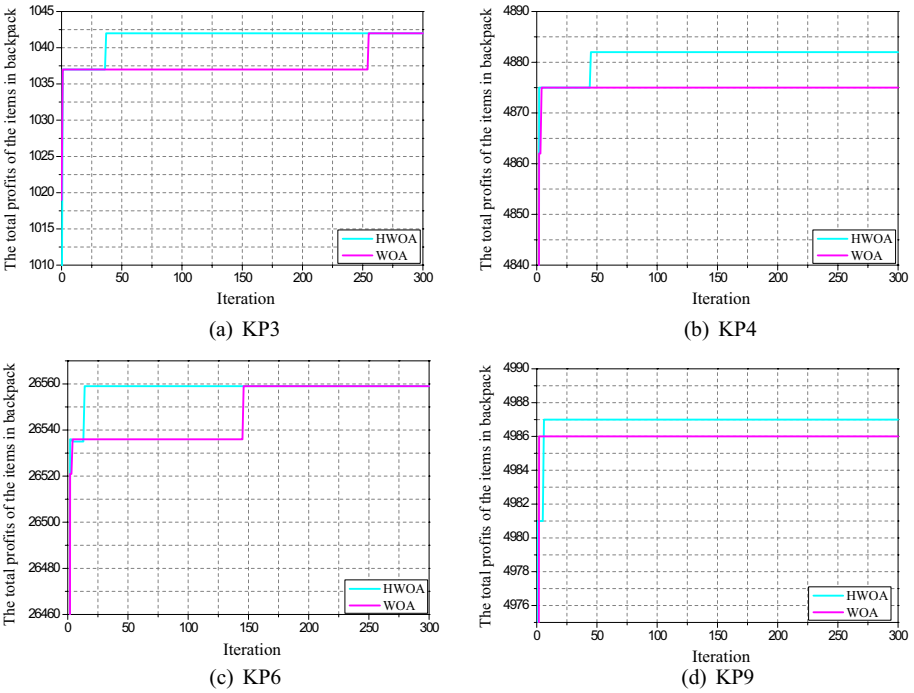


Fig. 8 The iterative process on case KP3, KP4, KP6 and KP9. (**a**), (**b**), (**c**), (**d**) show the total profits of the items in backpack with the iterative process on KP3, KP4, KP6 and KP9 respectively

Appendix: A

Details of data ECTD1 and ECTD2 are shown in Tables 7 and 8.

Table 7 The details of ECTD1

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
p_{1j}	1	2	2	1	2	2	1	3	2	1	2	1
p_{2j}	4	3	4	2	3	3	3	4	4	2	3	2
p_{3j}	2	1	2	1	3	2	1	2	1	1	2	1
t_{1j}	2	1	2	1	3	2	1	2	1	1	2	1
t_{2j}	3	3	3	2	4	3	2	3	3	3	3	3
t_{3j}	5	4	4	3	5	5	3	4	4	4	5	5
	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
p_{1j}	1	2	1	1	1	1	2	3	1	1	2	2
p_{2j}	2	3	2	2	3	3	3	4	4	4	4	3
p_{3j}	3	5	3	5	5	4	5	5	5	5	5	4
t_{1j}	2	1	2	1	3	1	3	1	1	2	2	2
t_{2j}	3	2	3	2	4	2	4	3	2	3	4	3
t_{3j}	4	3	5	4	5	4	5	5	4	4	5	4

The constraints are as follows: T4⇒T10 T11⇒T17 T14⇒T17 T12⨁T15 T22⨁T19 T2⨁T9 T5⨁T24 T3*60%ΘT10 T4*70%ΘT7 T4*50%ΘT15 T11*40% ⊙T6 T21*25% ⊙T22

Table 8 The details of ECTD2

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
p_{1j}	4	2	1	5	7	1	2	3	2	2	3	6
p_{2j}	7	6	4	7	8	3	5	6	7	3	5	8
p_{3j}	8	8	6	9	9	8	7	9	9	4	8	9
t_{1j}	5	2	1	1	5	3	2	3	6	6	3	4
t_{2j}	7	3	2	4	6	4	4	6	7	7	6	5
t_{3j}	9	4	3	6	8	6	6	7	9	8	9	6
	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
p_{1j}	3	4	2	3	1	3	2	4	6	7	1	1
p_{2j}	7	6	3	5	2	6	7	5	7	8	3	3
p_{3j}	8	8	4	8	3	9	9	6	9	9	8	5
t_{1j}	4	3	6	3	1	3	6	4	3	5	3	3
t_{2j}	5	5	7	6	2	6	7	5	4	6	4	5
t_{3j}	7	6	8	9	3	7	9	6	6	8	6	7
	T25	T26	T27	T28	T29	T30	T31	T32	T33	T34	T35	T36
p_{1j}	2	1	7	2	4	2	3	7	1	2	3	2
p_{2j}	6	4	8	5	5	3	5	8	3	4	6	7
p_{3j}	8	6	9	6	9	4	8	9	8	7	9	9
t_{1j}	2	1	6	4	6	6	3	5	3	2	3	6
t_{2j}	3	2	7	7	8	7	6	6	4	3	6	7
t_{3j}	4	3	8	9	9	8	9	8	6	4	7	9
	T37	T38	T39	T40	T41	T42	T43	T44	T45	T46	T47	T48
p_{1j}	2	1	5	3	2	2	1	4	2	3	7	6
p_{2j}	4	4	7	6	7	6	4	5	3	5	8	7
p_{3j}	6	6	9	9	9	8	6	8	4	8	9	9
t_{1j}	3	1	1	3	6	2	1	3	6	3	7	1
t_{2j}	5	2	4	6	7	3	2	4	7	6	8	2
t_{3j}	7	3	6	7	9	4	3	6	8	9	9	4
	T49	T50	T51	T52	T53	T54	T55	T56	T57	T58	T59	T60
p_{1j}	4	3	5	3	7	1	1	3	2	1	3	2
p_{2j}	5	4	6	5	8	3	2	5	6	4	6	7
p_{3j}	9	6	8	8	9	8	4	7	8	6	9	9
t_{1j}	3	2	6	3	5	3	1	1	2	1	3	6
t_{2j}	5	5	7	6	6	4	2	3	3	2	6	7
t_{3j}	6	7	8	9	8	6	4	5	4	3	7	9
	T61	T62	T63	T64	T65	T66	T67	T68	T69	T70	T71	T72
p_{1j}	1	7	1	1	5	2	3	7	3	2	2	1
p_{2j}	4	8	3	4	7	3	5	8	6	7	6	4
p_{3j}	8	9	8	6	9	4	8	9	9	9	8	6
t_{1j}	3	5	3	1	1	6	3	4	3	6	2	1
t_{2j}	5	6	4	2	4	7	6	5	6	7	3	2
t_{3j}	7	8	6	3	6	8	9	6	7	9	3	3

The constraints are as follows: T34⇒T20 T21⇒T47 T34⇒T37 T40⇒T57 T12⨁T35 T32⨁T29 T43⨁T39 T52% ⨁T69 T11⨂T30 T23⨂T24 T55⨂T51 T56⨂T72 T33*50%ΘT38 T61*60%ΘT37 T20*55%ΘT51 T72*65%ΘT64 T17*30% ⊙T38 T30*40% ⊙T27 T45*15% ⊙T66 T42*20% ⊙T64

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant No. 61762092); the Open Foundation of the Key Laboratory in Software Engineering of Yunnan Province (Grant No. 2020SE303); the Major Science and Technology Project of Precious Metal Materials Genome Engineering in Yunnan Province (Grant No. 2019ZE001-1 and 202002AB080001-6); Yunnan provincial major science and technology: Research and Application of key Technologies for Resource Sharing and Collaboration Toward Smart Tourism (Grant No. 202002AD080047).

Declarations

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
2. Shi, W., Jie, C., Quan, Z., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
3. Kong, L., Wang, L., Gong, W., Yan, C., Duan, Y., Qi, L.: Lsh-aware multitype health data prediction with privacy preservation in edge environment. World Wide Web:1–16 (2021)
4. Cen, C., Li, K., Ouyang, A., Zeng, Z., Li, K.: Gflink: an in-memory computing architecture on heterogeneous cpu-gpu clusters for big data. IEEE Trans. Parallel Distrib. Syst. **29**(6), 1275–1288 (2018)
5. Wang, X., Yang, L. T., Wang, Y., Ren, L., Deen, M.J.: Adtt: a highly efficient distributed tensor-train decomposition method for iiot big data. IEEE Trans. Industr. Inform. **17**(3), 1573–1582 (2021)
6. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
7. Alves, M.P., Delicato, F.C., Santos, I.L., Pires, P.F.: Lw-coedge: a lightweight virtualization model and collaboration process for edge computing. World Wide Web **23**(2), 1127–1175 (2020)
8. Zhou, X., Delicato, F.C., Wang, I.K., Huang, R.: Smart computing and cyber technology for cyberization. World Wide Web **23**(2), 1089–1100 (2020)
9. Ren, L., Liu, Y., Wang, X., Lu, J., Deen, M.J.: Cloud–edge-based lightweight temporal convolutional networks for remaining useful life prediction in iiot. IEEE Internet of Things J. **8**(16), 12578–12587 (2020)
10. Yan, C., Zhang, Y., Zhong, W., Zhang, C., Xin, B.: A truncated svd-based arima model for multiple qos prediction in mobile edge computing. Tsinghua Sci. Technol. **27**(2), 315–324 (2022)
11. Kwak, J., Kim, Y., Lee, J., Chong, S.: Dream: dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J. Sel. Areas Commun. **33**(12), 2510–2523 (2015)
12. Sardellitti, S., Scutari, G., Barbarossa, S.: Joint optimization of radio and computational resources for multicell mobile-edge computing. IEEE Trans. Signal Inf. Process. Netw. **1**(2), 89–103 (2015)
13. You, C., Huang, K., Chae, H., Kim, B.H.: Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16** (3), 1397–1411 (2016)
14. Jiang, Y., Ge, H., Wan, C., Fan, B., Yan, J.: Pricing-based edge caching resource allocation in fog radio access networks. Intell. Converged Netw. **1**(3), 221–233 (2020)
15. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun.:587–597 (2018)
16. Liu, Y., Lee, M.J., Zheng, Y.: Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. IEEE Trans. Mob. Comput. **15**(10), 1536–1233 (2016)
17. Yang, B., Chai, W.K., Pavlou, G., Katsaros, K.V.: Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In: 2016 5th IEEE international on cloud networking, pp. 136–141 (2016)
18. Psychas, K., Ghaderi, J.: Scheduling jobs with random resource requirements in computing clusters. In: Proceedings of the IEEE INFOCOM conference on computer communications, pp. 2269–2277 (2019)
19. Chen, C., Li, K., Ouyang, A., Tang, Z., Li, K.: Gpu-accelerated parallel hierarchical extreme learning machine on flink for big data. IEEE Trans. Syst. Man Cybern. Syst.:1–14 (2017)
20. Wang, X., Yang, L. T., Song, L., Wang, H., Deen, J.: A tensor-based multi-attributes visual feature recognition method for industrial intelligence. IEEE Trans. Industr. Inf. **17**(3), 2231–2241 (2020)

21. Xu, X., Li, H., Xu, W., Liu, Z., Yao, L., Dai, F.: Artificial intelligence for edge service optimization in internet of vehicles: a survey. Tsinghua Sci. Technol. **27**(2), 270–287 (2022)
22. Zhang, Y., Zhang, H., Cosmas, J., Jawad, N., Ali, K., Meunier, B., Kapovits, A., Huang, L. K., Li, W., Shi, L., Zhang, X., Wang, J., Koffman, I., Robert, M., Zarakovitis, C.C.: Internet of radio and light: 5g building network radio and edge architecture. Intell. Converged Netw. **1**(1), 37–57 (2020)
23. Yuan, L., He, Q., Tan, S., Li, B., Yu, J., Chen, F., Jin, H., Yang, Y.: Coopedge: a decentralized blockchain-based platform for cooperative edge computing. In: Proceedings of the Web 2021, pp. 2245–2257 (2021)
24. Tirkolaee, E.B., Goli, A., Hematian, M., Sangaiah, A.K., Han, T.: Multi-objective multi-mode resource constrained project scheduling problem using pareto-based algorithms. Computing **101**(6), 547–570 (2019)
25. Petchrompo, S., Wannakrairot, A., Parlikad, A.K.: Pruning pareto optimal solutions for multi-objective portfolio asset management. Eur. J. Oper. Res.:203–220 (2021)
26. Elsisy, M.A., El Sayed, M.A., Abo-Elnaga, Y.: A novel algorithm for generating pareto frontier of bi-level multi-objective rough nonlinear programming problem. Ain Shams Eng. J. **12**(2), 2125–2133 (2021)
27. Nath, S., Wu, J.: Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems. Intell. Converged Netw. **1**(2), 181–198 (2020)
28. Xu, X., Zhang, X., Gao, H., Xue, Y., Dou, W.: Become: blockchain-enabled computation offloading for iot in mobile edge computing. IEEE Trans. Industr. Inf. **16**(6), 4187–4195 (2019)
29. Wang, F., Wang, L., Li, G., Wang, Y., Lv, C., Qi, L.: Edge-cloud-enabled matrix factorization for diversified apis recommendation in mashup creation. World Wide Web:1–21 (2021)
30. Li, X.: A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems. J. Grid Comput. **19**(3), 1–12 (2021)
31. He, Y., Chen, Y., Lu, J., Chen, C., Wu, G.: Scheduling multiple agile earth observation satellites with an edge computing framework and a constructive heuristic algorithm. J. Syst. Archit. **95**, 55–66 (2019)
32. Wang, J., Wang, L.: Mobile edge computing task distribution and offloading algorithm based on deep reinforcement learning in internet of vehicles. J. Ambient. Intell. Humaniz. Comput.:1–11 (2021)
33. Celik, E., Dal, D.: A novel simulated annealing-based optimization approach for cluster-based task scheduling. Clust. Comput. **24**, 2927–2956 (2021)
34. Huang, J., Li, S., Chen, Y.: Revenue-optimal task scheduling and resource management for iot batch jobs in mobile edge computing. Peer-to-Peer Netw. Appl. **13**, 1776–1787 (2020)
35. Feng, S., Chen, Y., Zhai, Q., Huang, M., Shu, F.: Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms. EURASIP J. Adv. Signal Process. **2021**(1), 1–15 (2021)
36. Guo, X.: Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. AEJ - Alexandria Eng. J. **60**(6), 5603–5609 (2021)
37. Alrezaamiri, H., Ebrahimnejad, A., Motameni, H.: Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm (2018)
38. Wang, X., Duan, L.: Dynamic pricing and capacity allocation of uav-provided mobile services. In: Proceedings of the IEEE INFOCOM Conference on Computer Communications, pp. 1855–1863. IEEE (2019)
39. Dai, Y., Xu, D., Zhang, K., Maharjan, S., Zhang, Y.: Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks. IEEE Trans. Veh. Technol. **69**(4), 4312–4324 (2020)
40. Wang, Y., Ru, Z.Y., Wang, K., Huang, P.Q.: Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing. IEEE Trans. Cybern. **50**(9), 3984–3997 (2019)
41. Toth, P.: Dynamic programming algorithms for the zero-one knapsack problem. Computing **25**(1), 29–45 (1980)
42. Jackson, D., Belakaria, S., Cao, Y., Doppa, J.R., Lu, X.: Machine learning enabled fast multi-objective optimization for electrified aviation power system design. In: IEEE Energy Conversion Congress and Exposition (ECCE), pp. 6385–6390. IEEE (2020)
43. Chen, C., Li, K., Teo, S.G., Zou, X., Li, K., Zeng, Z.: Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. ACM Trans. Knowl Discov. Data **14**(4), 1–23 (2020)
44. Chen, C., Li, K., Wei, W., Zhou, J.T., Zeng, Z.: Hierarchical graph neural networks for few-shot learning. IEEE Trans. Circuits Syst. Video Technol. **32**(1), 240–252 (2021)
45. Pu, B., Li, K., Li, S., Zhu, N.: Automatic fetal ultrasound standard plane recognition based on deep learning and iiot. IEEE Trans. Industr. Inf. **17** (11), 7771–7780 (2021)

46. Zhao, Y.T., Chen, J.C., Wei-Gang, L.I.: Multi-objective grey wolf optimization hybrid adaptive differential evolution mechanism. Comput. Sci. (2019)
47. Akay, B.: Artificial bee colony – modifications and an application to software requirements selection swarm intelligence algorithms (2020)
48. Zhou, S.Z., Zhan, Z.H., Chen, Z.G., Kwong, S., Zhang, J.: A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction. IEEE Trans. Intell. Transp. Syst. **22**(11), 6784–6798 (2021)
49. Fang, W., Zhang, Q., Sun, J., Wu, X.J.: Mining high quality patterns using multi-objective evolutionary algorithm. IEEE Trans. Knowl. Data Eng. (2020)
50. Sun, J., Li, H., Zhang, Y., Xu, Y., Wei, Z.: Multi-objective task scheduling for energy-efficient cloud implementation of hyperspectral image classification. IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens. **14**, 587–600 (2020)
51. Pitangueira, A.M., Tonella, P., Susi, A., Maciel, R., Barros, M.: Risk-aware multi-stakeholder next release planning using multi-objective optimization. In: Proceedings of the international working conference on requirements engineering: foundation for software quality, pp. 3–18 (2016)
52. Zhang, Y., Li, H., Bu, R., Song, C., Chen, T.: Fuzzy multi-objective requirements for nrp based on particle swarm optimization. In: Proceedings of the international conference on artificial intelligence and security, pp. 143–155. Springer (2020)
53. Hudaib, A., Masadeh, R., Alzaqebah, A.I.: Wgw: A hybrid approach based on whale and grey wolf optimization algorithms for requirements prioritization. Adv. Syst. Sci. Appl **2**(576), 63–83 (2018)
54. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput. **6** (2), 182–197 (2002)
55. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
56. Agrawal, R.K., Kaur, B., Sharma, S.: Quantum based whale optimization algorithm for wrapper feature selection. Appl. Soft Comput. **106092**, 89 (2020)
57. Aljarah, I., Faris, H., Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. Soft. Comput. **22**(1), 1–15 (2018)
58. Pham, Q.V., Mirjalili, S., Kumar, N., Alazab, M., Hwang, W.J.: Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Trans. Veh. Technol. **69**(4), 4285–4297 (2020)
59. Sun, Y., Chen, Y.: Multi-population improved whale optimization algorithm for high dimensional optimization. Appl. Soft Comput.:107854 (2021)
60. Chakraborty, S., Saha, A.K., Sharma, S., Mirjalili, S., Chakraborty, R.: A novel enhanced whale optimization algorithm for global optimization. Comput. Ind. Eng **153**, 107086 (2021)
61. Zhang, D.Y., Wang, D.: An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems. In: Proceedings of the IEEE INFOCOM Conf. Comput. Com., pp. 766–774 (2019)
62. Luo, R.J., Ji, S.F., Zhu, B.L.: A pareto evolutionary algorithm based on incremental learning for a kind of multi-objective multidimensional knapsack problem. Comput. Ind. Eng **135**(SEP.), 537–559 (2019)
63. Nouioua, M., Li, Z.: New Binary Artificial Bee Colony for the 0-1 Knapsack Problem, pp 153–165. Springer, Cham (2018)
64. Zhi-Yong, L.I., Liang, M.A., Zhang, H.Z., Management, S.O.: Adaptive cellular particle swarm algorithm for solving 0/1 knapsack problem. Comput. Eng.:198–203 (2014)
65. Fister, I., Fister, D., Yang. S.: A hybrid bat algorithm. Elektrotehniski Vestnik/electrotechnical Rev. vol. 80(1) (2013)

Authors and Affiliations

Yan Kang[1] · Xuekun Yang[1] · Bin Pu[2] · Xiaokang Wang[3] · Haining Wang[1] · Yulong Xu[1] · Puming Wang[1]

[1] Key Laboratory in Software Engineering of Yunnan Province, National Pilot School of Software, Yunnan University, Kunming 650500, China

[2] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410006, China

[3] School of Computer Science and Technology, Hainan University, Haikou 570228, China