An Algorithm of Mobile Edge Computing Offloading Based on Improved SPEA2 Algorithm

Dan Ye
Sichuan University of Science and Engineering
https://orcid.org/0000-0002-2795-1637

Xiaogang Wang (wxg_zf@163.com)
Sichuan University of Science and Engineering

Jin Hou
Sichuan University of Science and Engineering

Research Article

Keywords: edge computing, computing offloading, multi-objective optimization, Internet of things

Posted Date: November 3rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-872980/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
An algorithm of mobile edge computing offloading based on improved SPEA2 algorithm

Dan Ye 1,2, Xiaogang Wang 1,2, * and Jin Hou 1,2

1 School of Automation and Information Engineering, Sichuan University of Science and Engineering, Yibin 644000, China; e-mail: Yedan_ydd@163.com
2 Artificial Intelligence Key Laboratory of Sichuan Province, China; e-mail:wxg_zf@163.com
* Correspondence: e-mail:wxg_zf@163.com

Abstract: Internet of things devices can offload some tasks to the edge servers through the wireless network, thus the computing pressure and energy consumption are reduced. But this will increase the cost of communication. Therefore, it is necessary to maintain the balance between task execution energy and experiment when designing the offloading strategy for the edge computing scenario of the Internet of things. This paper proposes an offloading strategy which can optimize the energy consumption and time delay of task execution at the same time. This strategy satisfies different preferences of users. First, the above task is modeled as a multi-objective optimization problem, and the Pareto solution set is found by improving the strength Pareto evolutionary algorithm (SPEA2). Based on the Pareto set, the offloading strategy satisfying the requires of users with different preferences by offloading cost estimation. Second, a simulation experiment is carried out to verify the robustness of the improved SPEA2 algorithm under the influence of different main parameters. By comparing with other algorithms. It is proved that the improved SPEA2 algorithm can minimize the balance between task execution delay and energy consumption.

Keywords: edge computing; computing offloading; multi-objective optimization; Internet of things

1. Introduction

With the growth of the number of smart mobile devices and the popularization of 5G network technology, the scale of the Internet of things (IoT) is expanding rapidly [1]. The application requirement of the Internet of things is also improved [2]. IoT devices is facing two problems: limited computing power and small battery capacity, which urgently needs to find a solution [3-4]. Mobile edge computing (MEC) technology is hopeful to become a solution. IoT devices can solve the problems of insufficient processing capacity and small battery capacity by offloading intensive tasks to the edge server [5-6]. Before MEC offloading technology appears, the intensive tasks of IoT devices are usually offloaded to remote cloud processing. However, cloud servers are usually deployed in remote locations, which greatly increases the transmission delay of tasks. Edge servers are usually deployed closer to the device, which can reduce the task transmission delay [7-9].

This paper focuses on the problem of computing offloading in the MEC scenario of IoT. Generally, an extensive number of IoT devices are connected to IoT. Simply deploying one edge server in IoT is not enough to handle the intensive tasks generated by a great quantity IoT devices. Therefore, in the Internet of things MEC system studied in this paper, multiple edge servers with different performance are deployed, which is a multi-user multi MEC server system. IoT devices can reduce the energy con-
sumption of the device by offloading the intensive tasks to the edge server. However, with the increase of accessed devices, the cost of mission transmission will also increase. Most IoT applications are limited by time delay and energy consumption when designing offloading strategies [10]. Task execution delay or device execution energy consumption is often considered as an optimization index in the MEC offloading problem, while another condition is used as a constraint [11-13]. In IoT scenario, some users require low latency, some users require low energy consumption, and even a small number of users require low latency and low energy consumption. If the offloading strategy is designed in the above way, it may not satisfy the requires of all users. Consequently, when designing the offloading strategy, this paper expects to achieve a balance between the delay and energy consumption of the task execution. The purpose of this paper is to design an offloading strategy that can balance and minimize the execution delay and energy consumption of tasks in the Internet of things MEC system. In this paper, task execution delay and device energy consumption will be optimized simultaneously as a performance index. This is a multi-objective optimization problem [14]. By constructing a multi-user and multi MEC server scenario, this problem is modeled as a multi-objective optimization problem. By improving the operator coding mode and operator crossover and mutation mode of the strength Pareto evolutionary algorithm (SPEA2) [15], the Pareto optimal set the requirements is found. Pareto optimal set is a compromise optimal solution set for the two optimization indexes proposed in this paper. Through the evaluation of the performance indexes proposed in this paper, the offloading strategies that satisfy the requirements of different users can be quickly found from this set.

The main contributions of this paper are as follows:

- This paper considers the multi device and multi MEC server system in IoT. Combined with the system scenario constructed in this paper, the local execution process, task transfer process and MEC server execution process are modeled.;
- The offloading problem in this paper is modeled as a multi-objective optimization problem. The Pareto optimal set of the multi-objective optimization problem is obtained by the improved SPEA2 algorithm.;
- The effectiveness of the improved SPEA2 algorithm is verified by simulation experiments, The performance of the algorithm under the influence of different parameters is analyzed and evaluated. By comparing with other baseline algorithms, the algorithm finds that which can minimize the energy consumption and the time delay of task execution.

The following parts of the article are as follows. The second section introduces recent work related to MEC computing offloading. The third section introduces the MEC system studied in this paper and modeling it. The fourth section introduces the details of the improved SPEA2 algorithm. In the fifth section, the performance of the improved algorithm is verified by simulation experiments. Finally, the sixth section summarizes the work of this paper.

2. Related Work
Computational offloading is an important technology in MEC, which determines whether the tasks produced by the device are offloaded and which MEC servers are offloaded. In this section, the recent research work and process on this optic will be introduced.

In [16], Mach et al. Introduced the MEC system. They divided the study of offloading into three key areas. The two most commonly used performance evaluation indexes, time delay and energy consumption, are proposed. In [17], Mao et al. Used task execution delay and execution cost of task failure as performance indicators. A dynamic offloading algorithm based on Lyapunov optimization (LODCO) is proposed. In [18], Umber et al. Studied the moving edge computing unloading scheme of D2D. The total delay of task execution is taken as a performance index. A novel scheme named Joint Partial Offloading and Resource Allocation (JPORA) is proposed. In [19], Shu et al. Considered the dependency between IoT applications and took the earliest completion time (AFT) of the task as the performance index. An offloading algorithm DEFO combined with game theory and suitable for Distributed Internet of things MEC system is proposed. In [20], Huang et al. Considered a time-varying channel model and took the transmission rate of tasks in a single time slot as the performance index. An online offloading framework DROO based on deep reinforcement learning is proposed. These studies only focus on one performance index to find the offloading strategy. However, the application of IoT is often constrained by two indicators of task execution delay and energy consumption. When designing the MEC offloading strategy of IoT, the optimization of multiple indicators should be considered.

Some scholars also studied the offloading strategy with multiple optimization indexes. In [21], Liu et al. Proposed a multi-objective optimization problem with energy consumption, delay and payment cost as performance indicators. They use scalar method to transform multi-objective optimization problems into single objective optimization problems. In [22], Tang et al. Weighted the time and energy consumption in the user experience as the mixed overhead, and specified the mixed overhead minimization problem of time and energy consumption. The block coordinate descent method is used to solve this problem. This method of transforming multiple objectives into single objective requires the allocation of weights ahead of time. When the user preferences change, it is necessary to find the optimal strategy again. In [23], de et al. Proposed a multi-objective optimization problem with user satisfaction and provider cost as performance indicators, and proposed an offloading algorithm combined with NSGAII algorithm. In [24], Cui et al. Proposed a multi-objective optimization problem with task execution delay and energy consumption as performance indicators, and found the Pareto optimal solution set to solve the problem by improving NSGAII algorithm. The NSGAII algorithm they use is a typical multi-objective evolutionary algorithm (MOEA) [25]. Such algorithms will eventually get a set of optimal solutions and become Pareto sets. When user preferences change, the offloading strategy satisfying user requires can be found quickly from the Pareto set, which is exactly the same as the purpose of this article. In [23], the performance indicators selected are user satisfaction and provider cost, which are different from the main performance in-
dicators considered in this paper. In [24], the scenario of unit centralized cell is mainly considered as a single MEC server sce-
nario, which is different from the MEC scenario of multi device and multi MEC server in this paper.

This article considers a distributed Internet of things MEC scenario with multiple edge devices and multiple MEC servers. We focus on finding an offloading strategy that can minimize the delay and minimize the energy consumption. This problem is modeled as a multi-objective optimization problem. Through the improved SPEA2 algorithm proposed in this paper, the Pareto optimal solution satisfying the optimization problem is found. The offloading strategy satisfying different user requires can be found quickly from this Pareto optimal solution set.

3. System Model

In this section, the details of the system model will be introduced in detail. This paper considers an $L \times L$ ($km^2$) MEC system. As shown in Figure 1, Multiple IoT devices are being accessed in the scene, record as $N = \{1,2,\ldots,n\}$ . There are multiple MEC servers with different performance in the scenario, record as $M = \{1,2,\ldots,m\}$ . The task of each edge device is regarded as an integral whole. Tasks are independent of each other. Each task can be executed locally or offloaded to MEC server. This paper mainly considers the energy consumption and task delay of edge devices in processing tasks. Next, these two problems will be modeled separately.

![Figure 1. Network model of the MEC system](image)

3.1. Communication Model

The tasks of edge devices can be executed locally or unloaded to the MEC server. The energy consumption of the edge device execution task is mainly composed of two parts: the energy consumption of the local execution task and the energy consumption of the transmission task to the MEC server. The delay of task execution is mainly composed of three parts: the delay of local execution task and the delay of MEC execution task and the delay of task transmission [26]. Two tuples $D_i = (d, c)$ is used to represent tasks on device $i$. Here $d_i$ denotes the size of computation input data of task $i$, $c_i$ denotes the required total CPU cycles to accomplish the task $i$. Some parameters involved in this paper are listed in Table 1.
Table 1. Some parameters and their implications

| Notation | Description |
|----------|-------------|
| $N$      | The set of IoT devices |
| $M$      | The set of MEC servers |
| $d_i$    | The size of computation input data of task $i$ |
| $c_i$    | The required total CPU cycles to accomplish the task $i$ |
| $f_i^e$  | The computation capability of the edge device $i$ |
| $f_j^s$  | The computation capability of the MEC server $j$ |
| $\sigma_i$ | The power consumption coefficient when executing tasks locally |
| $T_{i,l}^l$ | The time of the task $i$ executes locally |
| $E_{i,l}^l$ | The energy consumption of task $i$ executed locally |
| $T_{i,j}^s$ | The execution time of task $i$ offloaded to MEC server $j$ |
| $E_{i,j}^s$ | The energy consumption of task $i$ offloaded to MEC server $j$ |
| $a_k$    | Offloading policy |
| $T(a_k)$ | The average time that takes offloading decision $a_k$ to process all tasks |
| $E(a_k)$ | The average energy consumption of edge devices that take offloading decision $a_k$ to process all tasks |

3.1.1. Local Computing Model

Calculates the time that the task was executed locally. The computation capability of the edge device $i$ is represented by $f_i^e$.

Then, the time of the task $i$ executes locally $T_{i,l}^l$ is given by:

$$T_{i,l}^l = \frac{c_i}{f_i^e} \quad (1)$$

Next, the power consumption coefficient when executing tasks locally is expressed in $\sigma_i$. Then, the energy consumption of task $i$ executed locally $E_{i,l}^l$ is given by:

$$E_{i,l}^l = \sigma_i T_{i,l}^l = \frac{\sigma_i c_i}{f_i^e} \quad (2)$$

3.1.2. Edge Computing Model

In general, the performance of the server is better than that of the edge device. The execution time of task offloading to MEC server is mainly composed of three parts: task uploading time, task execution time in MEC server and result feedback time. Generally, the feedback time is far less than the uploading time and task processing time [27]. Therefore, this paper calculates the time of task processing on the edge server, considering only the uploading time and the execution time of the task.

The transmission rate $r_{i,j}$ of task $i$ to server $j$ can be given by [28]:

$$r_{i,j} = W \log_2\left(1 + \frac{p_{i}^{tu}H_{i,j}}{W N_0}\right) \quad (3)$$

Here, $W$ is the channel bandwidth, $p_{i}^{tu}$ is the transmission power of the device $i$, $H_{i,j}$ is the link channel gain between device $i$ and MEC server $j$. $N_0$ is the receiver's noise power density.
The execution time of task $i$ offloaded to MEC server $j$ includes two parts: task transfer time and task execution time in MEC. The computation capability of the MEC server $j$ is represented by $f_j^s$. The execution time of task $i$ offloaded to MEC server $j$ is given by:

$$T_{ij}^s = d_{ij}/r_{ij} + c_i/f_j^s$$

(4)

Here, $d_{ij}/r_{ij}$ is the task transmission time, $c_i/f_j^s$ is the execution time of the task on the MEC server.

Next, calculate the energy consumption of the task offloaded to the MEC server, consider only the energy consumption of the device, in other words, only consider the energy consumption when the device sends tasks. The energy consumption of task $i$ offloaded to MEC server $j$ is given by:

$$E_{ij}^s = p_i^{tu}d_{ij}/r_{ij}$$

(5)

3.2. Problem Formulation

Use a set of $A$ to represent the set of offloading policies, record as $A = \{a_1, a_2, ..., a_s\}$. An element in the collection represents one offloading decision. Specifically, $a_{k,kes}$ represents an offloading policy, expressed as:

$$a_{k,kes} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

(6)

As in equation (6), the size of the decision matrix is $n \times m$. Each row vector in the matrix represents the decision vector of the corresponding task. The value of the row vector should be 0 or 1. The sum of the elements of a row vector should not exceed 1. This is because a task can only be offloaded to one MEC server. The $i-th$ line in the offloading decision matrix $a_{k,kes}$ is the decision made by the decision against the task of device $i$. If all elements in line $i$ are 0, the task $i$ is executed locally. If line $i$ has and only one element has a value of 1, the task $i$ is offloaded to the server $j$’ execution. Here, $j'$ is the corresponding column element of element 1 in line $i$.

For $a_k$, combined with computational modeling, the completion time $T_i(a_k)$ of offloading decision $a_k$ processing task $i$ is given by:

$$T_i(a_k) = \left(1 - \sum_{j=1}^{m} a_{k,i,j}\right) \cdot T_i^l + \sum_{j=1}^{m} a_{k,i,j} \cdot T_{ij}'^s$$

$$= \left(1 - \sum_{j=1}^{m} a_{k,i,j}\right) \cdot \left(\frac{c_i}{f_j^l}\right) + \sum_{j=1}^{m} a_{k,i,j} \cdot \left(\frac{d_{ij}}{r_{ij}'} + \frac{c_i}{f_j'}\right)$$

(7)

Next, the energy consumption $E_i(a_k)$ of offloading decision $a_k$ processing task $i$ is given by:
\[
E_i(a_k) = \left( 1 - \sum_{j=1}^{m} a_{k,i,j} \right) \cdot E_i^l + \sum_{j=1}^{m} a_{k,i,j} \cdot E_{i,j}'
\]

\[
= \left( 1 - \sum_{j=1}^{m} a_{k,i,j} \right) \cdot (\sigma_i c_i) + \sum_{j=1}^{m} a_{k,i,j} \cdot \left( \frac{d_{p_u}^t}{r_{i,j'}} \right)
\]

In combination with the above, the average time \( T(a_k) \) required to take offloading decision \( a_k \) to process all tasks can be given by:

\[
T(a_k) = \frac{1}{N} \sum_{i=1}^{n} T_i(a_k)
\]

\[
= \frac{1}{N} \sum_{i=1}^{n} \left( \left( 1 - \sum_{j=1}^{m} a_{k,i,j} \right) \cdot \frac{c_i}{f_i^l} + \sum_{j=1}^{m} a_{k,i,j} \cdot \left( \frac{d_{p_u}^t}{r_{i,j'}} + \frac{c_i}{f_{j'}} \right) \right)
\]

Next, the average energy consumption \( E(a_k) \) of edge devices that take offloading decision \( a_k \) to process all tasks can be given by:

\[
E(a_k) = \frac{1}{N} \sum_{i=1}^{n} E_i(a_k)
\]

\[
= \frac{1}{N} \sum_{i=1}^{n} \left( \left( 1 - \sum_{j=1}^{m} a_{k,i,j} \right) \cdot (\sigma_i c_i) + \sum_{j=1}^{m} a_{k,i,j} \cdot \left( \frac{d_{p_u}^t}{r_{i,j'}} \right) \right)
\]

Finally, the decision problem is described as a multi-objective optimization problem with two objective functions. This problem is represented by \( P1 \) and given by:

\[
P1: \min_{a_k \in A} \{T, E\}
\]

\[
s.t. \quad 0 < f_i^l \leq f_{loc}^{max}
\]

\[
0 < f_j^s \leq f_{ser}^{max}
\]

\[
0 < p_{t,u}^{l} \leq p_{tu}^{max}
\]

\[
a_{k,i,j} \in \{0\} \cup \{1\}
\]

\[
\sum_{j=1}^{m} a_{k,i,j} \leq 1
\]

Here, constrained (12) guarantees the computation capability of edge devices. Constrained (13) guarantees the computation capability of the MEC server. Constrained (14) guarantees the transmitting frequency of edge devices. Constrained (15) and (16), as described above, guarantee the rationality of \( a_k \).

4. Problem Solving Based on Improved SPEA2
The optimization problem $P_1$ proposed in this paper is a multi-objective optimization problem. SPEA2 is a classical multi-objective optimization algorithm. It distributes and selects fitness based on the concept of Pareto dominance, and uses the idea of "k-nearest neighbor" to maintain diversity. Finally, an optimal Pareto solution set will be obtained. The offloading strategy satisfying different user requires can be quickly found from this optimal solution set. In addition, the network size of the MEC system is smaller than that of the traditional network [24]. This feature also reduces the solution space of the problem, so it is possible to use SPEA2 algorithm to solve the offloading problem. At the same time, the SPEA2 algorithm also has the disadvantage of insufficient local search capability. In this paper, an improved SPEA2 algorithm is proposed, which strengthens the local search ability of the algorithm, and uses the improved SPEA2 algorithm to solve the $P_1$ problem.

4.1. Initialization Parameter

First, initialize some parameters required for SPEA2 i.e., Initialize a population $P$ (size $NP$) and an empty external reserve set $P'$ (size $NP'$), maximum iteration number $Max$, crossover probability $CR$ and mutation probability $PM$. And some MEC related data i.e., the number of edge devices is $N$, and the number of MEC servers is $M$.

4.2. Chromosome Encoding

In general, the chromosome of traditional SPEA2 algorithm can be an arbitrary number of intervals, and the chromosome genes in this paper need to satisfy the constraints (15) and (16). The number of genes in chromosomes should be equal to the number of edge devices, in other words, the chromosomal genes are one-to-one correspondence with the tasks of edge devices. A chromosome is an offloading strategy. Theoretically, the gene of chromosomes should be a binary vector of $n \times 1$. In order to facilitate the subsequent crossover and mutation operations on chromosome genes, the binary vectors are encoding in integer form. The specific encoding method is shown in Figure 2.

![Figure 2. Chromosome encoding methods of population.](image)

As shown in Figure 2, this is a population $P$ initialized with a size of $NP$. It is a MEC system for $M$ MEC servers and $N$ edge devices. The gene on each chromosome is the offloading strategy of its corresponding edge device. As shown in Figure 2,
the first gene of the first chromosome is 3, which means that the task of the edge device numbered 1 is offloaded to the MEC server numbered 3. The second gene indicates that the task of the edge device numbered 2 will be executed locally, etc.

4.3. Fitness Assignment

In the SPEA2 algorithm iteration, the fitness value $F$ corresponding to chromosomes in $P$ and $P'$ should be calculated. To avoid the situation that individuals dominated by the same archive members have identical fitness values, with SPEA2 for each individual both dominating and dominated solutions are taken into account [15]. Before that, it is necessary to explain the definition of Pareto [14].

**Theorem 1.** Assuming that any two solutions $x_1$ and $x_2$ are all superior to $x_2$ for all targets, then we call $x_1$ dominance $x_2$. If $x_1$ is not governed by other solutions, then $x_1$ is called the non-dominated solution (not governed by the solution), also known as the Pareto solution.

Assign an intensity value $S(i)$ to each individual (chromosome) in $P$ and $P'$:

$$S(i) = |\{j | j \in P + P' \land i > j\}|$$  \hspace{1cm} (17)

In fact, $S(i)$ is to calculate the number of dominant points of point $i$. Based on the value of $S(i)$, the initial fitness $R(i)$ of individual $i$ is calculated:

$$R(i) = \sum_{j \in P + P', j > i} S(j)$$  \hspace{1cm} (18)

In fact, $R(i)$ is the sum of the value of $S(i)$ that dominates the point of $i$. In other words, the point of $R(i) = 0$ is a non-dominated point. The purpose of the algorithm is to find individuals with smaller $R(i)$ values and save them. However, most of the individuals do not dominate each other during the actual operation of the algorithm. Therefore, additional density information is introduced to distinguish individuals with the same original fitness value [15]. The density $D(i)$ of individual $i$ is given by:

$$D(i) = \frac{1}{\sigma_k^2 + 2}$$  \hspace{1cm} (19)

Here, $k = \sqrt{NP + NP'}$. The fitness $F(i)$ of $i$ is obtained by adding the density of individual $D(i)$ to the original fitness value $R(i)$.

$$F(i) = D(i) + R(i)$$  \hspace{1cm} (20)

4.4. Environment Selection

First, the non-dominated individuals in $P_t$ and $P'_t$ (individuals whose fitness $F$ is less than 1) is copied to $P'_{t+1}$. There will be three situations:

1) The number of individuals in $P'_{t+1}$ is equal to $NP'$;

2) The number of individuals in $P'_{t+1}$ is less than $NP'$;
3) The number of individuals in $P'_{t+1}$ is greater than $NP'$.

If it is the case 1, then the environment selection phase ends. In the case of 2, the best dominant individuals in $P_t$ and $P'_t$ are copied into $P'_{t+1}$ to fill the number of individuals equal to $NP'$. In the case of 3, $P'_{t+1}$ needs to be truncated. During each interception iteration, the individual that nearest to the $k$th individual is deleted until the number of $P'_{t+1}$ individuals is equal to $NP'$.

4.5. Chromosome Rrossover and Mutation

$P'_{t+1}$ preserves the outstanding individuals of the previous generation. $P_{t+1}$ needs to increase the diversity of the population to prevent the algorithm from falling into local optimization. Therefore, $P_{t+1}$ individuals need to be obtained by crossover and mutation from $P'_{t+1}$ individuals.

4.5.1. Chromosome Crossover

The traditional crossover operator only partially reconstructs two individuals according to a certain cross probability. If the similarity between two individuals is very high, the significance of this cross evolution to population diversity is not large, and the evolution of population will become stagnant and increase the possibility of falling into local optimum [29]. In this paper, the improved SPEA2 algorithm will determine the difference of chromosomes before crossing. Only when the difference is greater than a certain value will crossover and add the result to $P_{t+1}$. The chromosome genes in this paper are all integers with the same chromosome length. The difference between two chromosomes can be judged by calculating Hamming distance [30].

$$Crossover = \begin{cases} 
1, & HD_{ij} < Sim \\
0, & HD_{ij} \geq Sim 
\end{cases} \quad (21)$$

Here, $HD_{ij}$ represents Hamming distance between two individuals. The result of Hamming distance is the number of identical genes between two individuals. The smaller the value of $HD_{ij}$, the better. $Sim$ is a similarity threshold. The value of $Crossover$ is 1, indicating that these two individuals can crossover. The value of $Crossover$ is 0, indicating that the similarity between these two individuals is too high to crossover. It is necessary to find new individuals for crossover operation.

When chromosome crossover, two positions are randomly selected from the individual $p_1$ as crossover points. The position at the front is called crossover point 1. Another point is called crossover point 2. Copying the genes before crossover point 1 and the genes after crossover point 2 to the new individual $p1_\prime$. The remaining part of $p1_\prime$ is filled by sequence of genes corresponding to $p2$. The remaining genes of individuals $p1$ and $p2$ are filled into new individuals $p2_\prime$ in sequence. The chromosome crossover process is shows in Figure 3.
4.5.2. Chromosome Mutation

The traditional SPEA2 algorithm has strong randomness in genetic operation. Although randomness can increase the scope of search, avoid falling into local convergence. But it also weakens the local search ability of SPEA2 algorithm [28]. In order to maintain the global search ability of SPEA2 algorithm and enhance the ability of local search, the mutation individuals are divided into two parts, and two ways are used for mutation.

The individuals in the mutation pool \( M_u' \) are randomly selected from the \( P_{t+1}' \). The individuals in the \( M_u' \) are arranged from small to large according to their fitness \( F \). The first half of \( M_u' \) is considered to be an excellent individual in the mutant pool, and needs to retain their advantages. Formula (7) and (8) are used to calculate the time delay and energy consumption of gene corresponding tasks on chromosome. The two genes corresponding to the minimum time delay and energy consumption are considered to be the dominant genes. In addition to these two dominant genes, one gene is randomly selected to mutate on this chromosome. In this way, the mutation maintains the advantage of the chromosome itself and fine-tuning the chromosome.

For the remaining individuals in \( M_u' \), two locations are randomly selected from chromosomes to exchange their corresponding genes. In the exchange process, a certain probability mutation coefficient is introduced. In other words, a gene may mutate in the process of gene exchange, which also ensures the global search ability of the algorithm.

4.6. Improved SPEA2 Algorithm Steps

In conclusion, the improved SPEA2 algorithm proposed in this paper solves P1 steps as follows:

![Figure 3. Chromosome crossover process.](image-url)
3. Calculate the fitness value $F$ of the individual in $P$ and $P'$
4. Select the environment according to the fitness $F$ to get $P'_{t+1}$
5. Select individuals from $P'_{t+1}$ to perform crossover and mutation operations to obtain $P_{t+1}$
6. $t = t + 1$
7. end

5. Simulation and Result Analysis

In this paper, experimental comparison is carried out to verify the performance of the improved SPEA2 algorithm in solving the MEC computing offloading problem. The sensitivity of the algorithm under several important parameters is analyzed by experiments. The performance of the algorithm is verified by comparison with some baseline algorithms offloaded by MEC.

This paper mainly uses MATLAB to verify the MEC scenario and algorithm proposed in this paper. The data collected in this paper are as follows:

The bandwidth of the wireless channel is set to 20MHz. The size of the task to be processed and the CPU number needed to execute the task are randomly generated between $[1000, 3000]$Kb and $[0.2, 0.3]$GHz. The processing capacity of the MEC server is generally stronger than that of the edge devices. The MEC server computing capability of this article is $[20, 40]$GHz. The marginal capability of this paper is $[0.5, 1]$GHz. The transmitting power of the edge device is usually $[0.5, 1]$W. This paper will test the influence of the number of tasks to be processed and the number of different MEC servers on the result of the algorithm. During testing, all variables except the test amount are tested by the median value of the change interval.

For the SPEA2 algorithm, we set the maximum number of iterations $I_{Max}$ to 100, the number of population $NP$ to 50, and the number of external storage sets $NP'$ to 50. The crossover probability is 0.7, the mutation probability is 0.3, and the mutation probability is 0.001.

5.1. The Impact of Number of Edge Devices

The impact of the number of edge devices on the algorithm results will be verified in this section. In the MEC network of this paper, the task to be handled by an edge device is an indivisible and independent whole. When the number of edge devices in the IoT changes, the number of tasks to be processed will also change. In this group of experiments, there are ten different MEC servers in the MEC scenario, and the Pareto solution set of formula (9) and (10) can be satisfied by the improved SPEA2 algorithm. The number of edge devices $n = \{20, 30, 40, 50\}$ is verified respectively. Pareto front is shown in Figure 4.
In this group of experiments, in order to reduce the influence of irrelevant variables, the performance of the MEC server is controlled at $[30, 35] \text{GHz}$. In this group of experiments, the number of MEC servers $M = 10$. Through the principle of greedy algorithm, the formula (9) and (10) are optimized respectively, and the greedy limit values $GE$ and $GT$ for single target are obtained. It's shown in Figure 4. The improved SPEA2 algorithm can achieve a balance between $GE$ and $GT$. This shows that the improved SPEA2 algorithm can find a better offloading strategy for the system. With the increase of the number of access devices, the optimal set is getting bigger. This is because with the increase of the number of edge devices, the task of decision making is also increasing, which increases the dimension of algorithm decision. And the number of access devices will increase, which will also increase the overall communication cost and execution cost of the task.

5.2. The Impact of Number of MEC Server

The impact of the number of MEC server on the algorithm results will be verified in this section. In this group of tests, the number of edge devices $N$ is set to 30. All variables except MEC server number $M$ remain unchanged to reduce the effect of other variables on the experimental results. In this group of experiments, the CPU performance of the MEC server ranges from $[20, 40] \text{GHz}$. The results of $M = \{5, 10, 15, 20\}$ are verified respectively, Pareto front is shown in Figure 5.

![Figure 4](image-url). Pareto front for different number of edge devices.
It's shown in Figure 5, the improved SPEA2 algorithm can achieve a balance between GE and GT. This shows that the improved SPEA2 algorithm can find a better offloading strategy for the system. With the increasing number of MEC servers in the access scenario, the algorithm may find new decisions to enable them to find non-dominated points with smaller fitness values. However, as the number of MEC servers increases, the range of values of chromosome genes increases, which also increases the difficulty of algorithm optimization. Therefore, as the number of MEC servers increases, the algorithm may find a more suitable set of solutions, but its search space will also increase.

5.3. Offloading Cost Comparison Verification

The improved SPEA2 algorithm proposed in this paper will eventually output a set of Pareto optimal solutions. The offloading strategy that meets different user requirements can be quickly identified from this set. In order to find out the unloading strategy satisfying the needs of different users from the collection, the concept of offloading cost $\text{Cost}$ is proposed.

The time delay coefficient $\alpha$ and the energy consumption coefficient $\beta$ represent the degree of users requires for delay and energy consumption respectively. Here, $\alpha + \beta = 1$. When users prefer to perform tasks with low latency, a larger $\alpha$ value can be obtained. In extreme cases, when $\alpha$ takes 1, it means that users only care about the average delay of task execution. Offloading cost $\text{Cost}$ is given by:

$$\text{Cost} = \alpha e^T + \beta e^E$$  \hspace{1cm} (22)

In order to verify the performance of the improved SPEA2 algorithm, we use formula (22) to calculate and find out the minimum offloading cost $\text{Cost}$ of each generation External storage set $P_{t+1}'$ in the algorithm iteration process. When handling different numbers of tasks, the offloading cost $\text{Cost}$ varies with the number of iterations, as shown in Figure 6.
As shown in Figure 6, the value of Cost decreases as the number of iterations increases. This shows that the improved SPEA2 algorithm continuously finds better solutions when the number of iterations increases.

Finally, we use the offloading cost formula (22) to find out the minimum offloading cost from the set of Pareto optimal solutions exported from the improved SPEA2 algorithm. This minimum value is compared with the results of other baseline offloading algorithms to verify the optimization ability of the improved SPEA2 algorithm. The contrast algorithm includes local algorithm, MEC random offloading algorithm, and greedy based MEC offloading algorithm. Here, the local algorithm is that tasks are executed locally only. MEC random offloading algorithm, namely, tasks will be randomly offloading to the MEC server and cannot be executed locally. Greedy based MEC offloading algorithm is based on greedy algorithm to find the optimal solution for the current task's current goal. Three algorithms are used to calculate the formula (9) and (10) respectively, and the unloading cost Cost corresponding to the result is calculated by formula (22). This paper simulates the situation of different algorithms dealing with different number of edge device tasks when the number of edge servers is $M = 10$. The comparison is shown in Figure 7.

From Figure 7, it is easy to see that when considering the time delay and energy consumption of task execution, the offloading cost of LOACL algorithms is the highest. MEC random offloading algorithm can reduce the offloading cost of users to a certain extent, but because of its randomness, it is impossible to ensure that the optimal unloading strategy is obtained. The im-
proved SPEA2 algorithm and the greedy based MEC offloading algorithm have little difference in the offloading cost. This shows that the improved SPEA2 algorithm can achieve the requirement of user execution delay and energy consumption minimization. Greedy based MEC offloading algorithm can only calculate the optimal solution of a single target alone, and finally evaluate its offloading cost by weighting. The algorithm does not consider the influence between targets in computation. Moreover, when user preferences change, it is necessary to re-run the algorithm to find a user satisfaction offloading strategy. The Pareto set obtained by the improved SPEA2 algorithm is aimed at the equilibrium optimal solution of the two objectives. When user preferences change, the offloading cost formula (22) can be used to quickly find an offloading strategy to meet user needs from the Pareto set.

6. Conclusions

This paper studies the computing offloading problem in the MEC system of the Internet of things. Based on the requirements of IoT scenario, we hope to design an offloading strategy which can optimize the execution time and energy consumption of devices simultaneously. The Pareto optimal solution set of this multi-objective optimization problem is found by the improved SPEA2 algorithm proposed in this paper. Combined with the offloading cost formula proposed in this paper, the offloading strategy satisfying different user needs is quickly found from Pareto solution set. In this paper, the performance of the improved SPEA2 algorithm is verified by simulation experiments. The simulation results show that the improved SPEA2 algorithm can get the Pareto optimal set between the extremes of the two targets. In addition, by studying the performance of the improved SPEA2 algorithm under different parameters, the robustness of the improved SPEA2 algorithm in MEC offloading problem is verified.

Acknowledgements

We sincerely appreciate the valuable comments given by the judges and editors.

Authors' contributions

Dan Ye conceived and designed this study. Dan Ye conducted simulation experiments. Dan Ye and Xiaogang Wang wrote this paper. All authors reviewed and edited the manuscript. All authors read and approve the final manuscript.

Funding

This work was supported in part by National Natural Science Foundation of China (grant no. 61902268), Sichuan Science and Technology Program (grant no. 2018JY0197, 19ZDZX0037, 2019YFSY0045, 20ZDYF0919, 21ZDYF4052, 2020YFH0124, 2021YFSY0060).

Availability of data and materials

The details of experimental parameters are given in Table1 and Section 5.

Declarations

Competing interests
The authors declare that they have no competing interests.

**Author details**

1 School of Automation and Information Engineering, Sichuan University of Science and Engineering, Yibin 644000, China.
2 Artificial Intelligence Key Laboratory of Sichuan Province, China.

**References**

1. Wu, J., Zhang, G., Nie, J., Peng, Y., & Zhang, Y (2021) Deep Reinforcement Learning for Scheduling in an Edge Computing-Based Industrial Internet of Things. Wire Comm Mob Comp. https://doi.org/10.1155/2021/8017334.
2. S. Wijethilaka and M. Liyanage (2021) Survey on Network Slicing for Internet of Things Realization in 5G Networks. IEEE Commu Surv Tutor 23(2):957-994.
3. Abbasi, M., Mohammadi-Pasand, E., & Khosravi, M. R (2021) Intelligent workload allocation in IoT–Fog–cloud architecture towards mobile edge computing. Compu Comm 169: 71-80.
4. Ometov, A., Shubina, V., Klus, L., Skibińska, J., Saafi, S., Pascacio, P., Lohan, E. S (2021) A survey on wearable technology: History, state-of-the-art and current challenges. Comp Net 193: 108074.
5. Bilal, K., Khalid, O., Erbad, A., & Khan, S. U (2018) Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. Comp Net 130: 94-120.
6. D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach and F. Giust (2016) Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things. IEEE Cons Elec Mag 5(4):84-91.
7. Xu, Z., Zhang, Y., Li, H (2020) Dynamic resource provisioning for cyber-physical systems in cloud-fog-edge computing. J Cloud Comp 9(32). https://doi.org/10.1186/s13677-020-00181-y.
8. Sun, W., Liu, J., & Yue, Y (2019) AI-enhanced offloading in edge computing: When machine learning meets industrial IoT. IEEE Network 33(5): 68-74.
9. Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., & Yang, X (2017) A survey on the edge computing for the Internet of Things. IEEE access 6: 6900-6919.
10. Adhikari, M., & Gianey, H (2019) Energy efficient offloading strategy in fog-cloud environment for IoT applications. Inter Thing 6: 100053.
11. Xiao, S., Liu, C., Li, K., & Li, K (2020) System delay optimization for mobile edge computing. Future Generation Comp Sys 109: 17-28.
12. Li, H., Xu, H., Zhou, C., Lü, X., & Han, Z (2020) Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. IEEE Tran Veh Tech 69(9): 10214-10226.
13. Gu, X., Jin, L., Zhao, N., & Zhang, G (2019) Energy-efficient computation offloading and transmit power allocation scheme for mobile edge computing. Mob Infor Sys 2019. https://doi.org/10.1155/2019/3613250.

14. Giagkiozis, I., & Fleming, P. J (2015) Methods for multi-objective optimization: An analysis. Infor Sci 293: 338-350.

15. Zitzler, E., Laumanns, M., & Thiele, L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. Paper presented at the Technical Report, 103 2001.

16. Mach, P., & Becvar, Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Comm Surv Tutor 19(3): 1628-1656.

17. Mao, Y., Zhang, J., & Letaief, K. B (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sele Areas Comm 34(12): 3590-3605.

18. U. Saleem, Y. Liu, S. Jangsher, X. Tao and Y. Li (2020) Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing. IEEE Tran Veh Tech 69(4): 4472-4486.

19. Shu, C., Zhao, Z., Han, Y., & Min, G. (2019). Dependency-aware and latency-optimal computation offloading for multi-user edge computing networks. Paper presented at the 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 1-9 June 2019.

20. Huang, L., Bi, S., & Zhang, Y. J. A (2019) Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Tran Mob Comp 19(11): 2581-2593.

21. Liu, L., Chang, Z., Guo, X., Mao, S., & Ristaniemi, T (2017) Multiobjective optimization for computation offloading in fog computing. IEEE Inter Things J 5(1): 283-294.

22. Tang, Q., Lyu, H., Han, G (2020) Partial offloading strategy for mobile edge computing considering mixed overhead of time and energy. Neural Comput Applic 32:15383–15397.

23. De Maio, V., & Brandic, I (2019, April) Multi-objective mobile edge provisioning in small cell clouds. Paper presented at the 2019 ACM/SPEC International Conference on Performance Engineering, 127-138 April 2019.

24. Cui, L., Xu, C., Yang, S., Huang, J. Z., Li, J., Wang, X., Lu, N (2018) Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things. IEEE Inter Things J 6(3): 4791-4803.

25. Zhang, Q., & Li, H (2007) MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans evolu compu 11(6): 712-731.

26. Kuang, Z., Li, L., Gao, J., Zhao, L., & Liu, A (2019) Partial offloading scheduling and power allocation for mobile edge computing systems. IEEE Inter Things J 6(4): 6774-6785.

27. V. Hassel, D. Gesbert, M. Alouini and G. E. Oien (2007) A Threshold-Based Channel State Feedback Algorithm for Modern Cellular Systems. IEEE Tran Wire Comm 6(7): 2422-2426.
28. Ranji, R., Mansoor, A. M., & Sani, A. A (2020) EEDOS: an energy-efficient and delay-aware offloading scheme based on device to device collaboration in mobile edge computing. Telecomm Sys 73(2): 171-182.

29. Liu, X., & Zhang, D. (2019). An Improved SPEA2 algorithm with local search for multi-objective investment decision-making. Appli Scie 9(8): 1675.

30. Liu, D., Yao, Z., & Chen, L (2021) Emergency Scheduling Optimization Simulation of Cloud Computing Platform Network Public Resources. Complexity, 2021.