Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 1, 2023

An Adaptive Multitasking Evolutionary Computation Offloading Algorithm for Mobile Edge Computing

Yingjie Hou [1], Zhangang Wang [2, *], Xu Liu [1]

[1] School of Software, Tiangong University, Tianjin 300387, China
[2] School of Computer Science and Technology, Tiangong University, Tianjin 300387, China
[*] **Corresponding author**: Zhangang Wang (Email: wangzhangang@tiangong.edu.cn)

Abstract: In the mobile edge computing scenario, intelligent terminal devices can reduce the waiting delay by offloading the computing task to a server. The offloading scheme's optimization has been proven an NP-hard problem. The heuristic algorithms, including evolutionary algorithms, are widely used to search for the optimal scheme. User experience is mainly limited by energy consumption and time delay. Most existing research results combine it linearly into a single objective or focus on the optimal solution in a specific area. Based on this, this paper proposes an adaptive multitasking evolutionary optimization algorithm, which considers multiple independent areas to be optimized. It abstracts the task offloading system model in each area as a multi-objective programming problem, aiming at minimizing the average energy consumption and delay of intelligent devices. By learning the user distribution and the similarity of tasks to be processed in different areas dynamically to adjust the degree of population communication, the convergence has been sped up. The performance of the proposed algorithm is verified by a set of instances.

Keywords: Multitasking Optimization; Multi-objective Optimization; Edge Computing; Resource Allocation.

1. Introduction

The introduction should provide background information (including relevant references) and should indicate the purpose of the manuscript. Cite relevant work by others, including research outside your company. Place your work in perspective by referring to other research papers. Inclusion of statements at the end of the introduction regarding the organization of the manuscript can be helpful to the reader.

With the development of the Internet of Things, mobile devices are no longer limited to small and microelectronic devices such as smartphones and personal computers. Application scenarios, such as augmented virtual reality, automatic driving, and digital medical treatment, put forward higher requirements for response delay and computing speed. The central cloud architecture that centralizes computing resources at the core is difficult to meet the needs of the large number of users. Based on the relevant research on the marginalization of computing resources, the European Telecommunications Standards Association proposed the concept of mobile edge computing. It aims to distribute computing resources to provide computing services with lower delay and higher bandwidth, creating an open and collaborative network environment.

Mobile Edge network mainly comprises mobile edge servers (MESs) and mobile devices (MDs). Compared with the terminal device, the edge server of- ten configures many computing resources, which can significantly improve the processing speed. In the mobile edge computing scenario, MD can decompose the applications to be processed into subtasks and offload them to MESs to use computing resources fully. By deploying servers with solid computing power n- ear the intelligent device terminal, the processing delay is reduced by shortening the straight-line distance.

Due to the complexity of the MES environment, many factors affect the offloading decision. Designing the optimal offloading decision strategy to fully tap the performance gain of MES in time delay and energy consumption is a very challenging scientific problem. As an NP-hard problem, heuristic algorithms are often used to solve the optimal scheme under different mode architectures. User experience is mainly restricted by time delay and energy consumption. Most of the existing research results combine them into a single objective function, which is difficult to reflect on the restrictive relationship between them. When it is solved as a multi-objective optimal problem, the amount of calculation also increases. Based on this, this paper proposes an adaptive multitasking evolutionary optimization algorithm, which considers multiple independent areas to be optimized. It abstracts the task offloading system model in each area as a multi-objective programming problem. The target is to minimize the average energy consumption and delay of intelligent devices, dynamically learns the distribution of users and the similarity of tasks to be processed in different areas, and adjust the degree of population communication. The task offloading schemes in different regions are planned and solved based on the same population.

The main contributions of this paper are summarized as follows:

(1) A scheme for coding and decoding the individuals is designed. It can be resolved into an offloading strategy that meets the constraints of the region. In the iterative process, only the source code is used as the individual for mating, non-dominated sorting, and other operations. Finally, the optimal individual is decoded into the task offloading scheme corresponding to different regions.

(2) An adaptive parameter learning method based on distribution is used to dynamically adjust the communication degree of the population to reduce the negative feedback caused by excessive communication on the convergence speed.

(3) The individuals in the population are ranked non-dominated based on a set of pre-generated reference points, which provides additional information for individuals as a

reference from the dispersed area to improve the uniformity of non-dominated solutions effectively.

This paper is organized as follows. Section II briefly introduces the research results of offloading optimization scheme in the mobile edge computing scenario; Section III introduces and explains the mobile edge computing system model used in this paper in detail; Section IV describes the algorithm proposed in this paper in detail in combination with the application scenario; Section V compares with other algorithms on multiple instances to illustrate the performance of the proposed algorithm.

2. Related Work

Besides processing computing tasks with their servers, mobile devices can also offload tasks to edge servers. To advance the experience of users, reducing waiting time is an important goal. Liu et al. [1] transformed the computing offload problem into a Markov decision process. By analyzing the average delay time of tasks and the average energy consumption of mobile devices, they aimed to minimize the delay time under power limitation and find the best computing offload strategy by a one-dimensional search algorithm. Mao et al. [2,3] proposed a computational offloading strategy based on MES system with green energy collection. Taking the delay time and the cost of task failure as performance indicators, the author proposes a dynamic online computing offloading algorithm based on Lyapunov optimization.

Due to the need for data transmission in the offloading process, the energy consumption caused by this process cannot be ignored. You et al. [4] considered the resource allocation problem of offloading multiple terminal devices to a single edge server in MES and transformed the problem into a convex optimization problem to minimize the energy consumption of mobile devices under the constraint of delay time. Zhao et al. [5] optimized the offloading selection, radio resource allocation, and computing resource allocation in the MES system of multi-mobile terminal devices to minimize the energy consumption of intelligent mobile devices. The classic three-node MES system consists of a terminal node, an auxiliary node, and an edge node. Cao et al. [6] proposed the joint computing and communication cooperation method in the classical three-node MES system and obtained the optimal solution through the convex optimization method. Lyu et al. [7] proposed a new architecture of the cloud, edge cloud, and Internet of things devices and proposed a framework that allows lightweight requests to solve the scalability problem. A selective offload scheme is designed to minimize the energy consumption of the device. Allowing the device to specify or refuse offload can further reduce the signaling overhead.

To reduce the delay, it is practical to offload most of the pending tasks to the server. However, the power storage capacity of mobile devices is limited, which obviously cannot complete the transmission task of too large an order of magnitude. Taking time delay and energy consumption as optimization objectives, respectively, will ignore the potential constraints between them, which is far from practical application. Therefore, some studies abstract task offloading as a multi-objective optimization problem to find the optimal solution in the trade-off. Chen et al. [8] proved that solving the optimal solution of computing offloading scheme is NP-hard and designed a distributed computing offloading algorithm. Dinh et al. [9] proposed an optimization framework for offloading tasks from one mobile device to multiple MES servers. The goal is to minimize the total task execution delay and mobile device energy consumption by jointly calculating the offloading and CPU frequency of mobile devices. Using the idea of a software-defined network (SDN), Chen et al. [10] studied the offloading calculation problem of MES in the ultra-dense network (UDN). They transformed it into two sub-problems, namely task offloading and resource allocation, and proposed an effective calculation offloading scheme of software-defined task offloading (SDTO). Ning et al. [11] considered the resource competition among mobile terminal devices, transformed the multi-terminal device computing offloading problem into a mixed integer linear programming (MILP) problem, and designed an iterative heuristic MES resource allocation (IHRA).

The above researches are mostly based on single-user and single-task scenarios. However, in real-task scenarios, multi-user and multitasking scenarios are more common and meaningful. Li et al. [12] proposed a deep reinforcement learning (DRL) algorithm to solve the problem of computing offload in heterogeneous edge servers. The algorithm considers multitasking edge subnet heterogeneity and edge device mobility. It studies the network environment, generates computing offload decisions, uses the Actor Critical algorithm and deep deterministic policy gradient, and uses evolutionary computing offload decisions to minimize tasks. Lei et al. [13] proposed a collaborative computing offload and multi-user scheduling algorithm in the edge computing system of the Internet of Things, which minimizes the long-term weighted sum of delay and power consumption when random traffic arrives. The dynamic optimization problem is transformed into a Continuous Time Markov decision process (CTMDP) model. The semi-gradient descent method and time difference algorithm are used to linearly approximate the value function, and a simple algorithm for solving the CTMDP model is derived. Rui et al. [14], based on the edge unloading system of an intelligent wearable device communication network, proposed a sort and segmentation algorithm for multi-purpose computing, which divides some tasks to be unloaded before unloading. Once the unloading decision is made, the relevant model with the minimum transmission cost as the goal will be established, and the dynamic unloading strategy will be proposed. Literature [15] proposed an efficient computing offload scheme for the distributed load-balancing MEC network combined with cloud computing. The execution delay and cost of edge and cloud services can be reduced by using queue theory to model the nonlinear multi-objective optimization problem. Literature [16] established a multi-objective optimization strategy to balance the energy consumption and delay performance of the server and solved the optimization problem using the interior point method and queuing theory. An unloading scheme based on multitasking unloading game theory is proposed in [17]. It is close to the optimal solution of the theoretical optimal policy and significantly reduces the system overhead. The problem of task offloading in multi-user mobile edge computing scenarios is described as a convex optimization problem in [18], which minimizes the weighted sum of delay and energy consumption under limited resource conditions. It uses the computational offloading mechanism to minimize mobile device energy consumption and task execution delay.

The classical multi-objective evolutionary algorithms, such as NSGA-II [19] and MOEA/D [20], have been applied to

task offloading optimization to re- duce energy consumption and time delay simultaneously. However, as these algorithms do not support information exchange of subpopulations, it can only optimize the objectives in a single region at a time. In this paper, a multitasking evolutionary named MOEA-RP is proposed. By adaptively learning the mating parameter rmp, individuals in subpopulations can do knowledge transmission in different extent dynamically. Convergence of subtasks can be accelerated through using the potential similarities based on the single population simultaneously.

3. System Model

The intelligent terminal device has powerful functions, and the amount of data to be processed has increased. It is challenging to only meet the needs by relying on the terminal's computing power.

With the improvement of the functions of personal terminals and the further popularization of other intelligent devices, the demand for edge servers is becoming increasingly urgent. The edge network system model of multi-application service deployment based on multitasking evolution is shown in figure 1 below. The system comprises three parts: central cloud, edge server, and mobile devices. All services users require can be provided on the cloud, and it is assumed that there are almost unlimited computing resources on the cloud. Edge servers often provide higher computing speed than wired terminal devices of computing resources. In a specific area, the mobile device can decompose part of the application data and offload it to the edge computing server for processing to reduce the response delay.

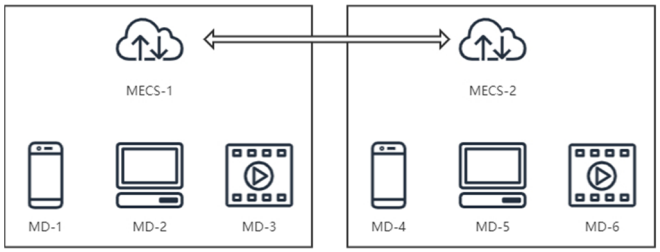


Figure 1. System Model

As shown in figure 1, the data transmission between the mobile device and the edge server is realized through the wireless mode. The wired connection between edge servers dramatically improves the transmission speed while reducing the packet loss rate.

Table 1. Symbol Description

R_i	Area containing optimization task of unloading scheme
MD_i	Mobile devices proposed in each region
Q_i	Pending tasks proposed by mobile devices
c_i	Number of decomposable copies of tasks to be processed
D_i	Total amount of data to be processed by mobile devices
σ_{iw}	Data volume of each subtask after decomposition
v_{self_i}	Processing speed of mobile devices
T_{iw}^{self}	Processing time of the mobile device itself
γ	Ratio of data uploaded and downloaded by subtasks
Dis	Distance between mobiles device and edge computing servers
μ	Channel interference factor

There are n edge computing servers and m mobile terminals in a specific region (region R). Users complete S tasks through intelligent terminals, such as data processing, image restoration, etc. The task can be decomposed into data packets of different sizes for distributed processing to speed up the processing speed. There are differences in the total amount of data packets and the corresponding data volume, code volume, complexity, and other characteristics. Formulate an appropriate resource allocation strategy, select the most appropriate processing mode for each intelligent terminal in the area, and shorten the delay and energy consumption caused by task processing as far as possible, which is the optimization goal.

The resource allocation optimization problems in multiple regions $(R_1, R_2, \ldots, r_k)$ have potential similarities. The pending tasks Q_i of user u_i can be decomposed into ci subtasks. The data volume of each subtask is δ_{iw}. The total amount of data to be processed D_i of User u_i is represented as follows.

$$D_i = \sum_{w=1}^{c_i} \delta_w \tag{1}$$

There are two main processing methods for subtasks: using their processorand offloading to the edge computing server in the region.

When the subtask Q_{iw} is processed by mobile device MD_i, the time required is mainly affected by the computing power of the processor.

$$T_{iw}^{self} = \delta_{iw} / v_{self_i} \tag{2}$$

In which V_{self_i} is the processing speed of mobile device MD_i processor.

When subtask Q_{iw} is offloaded to edge server ECS_j to be processed, the corresponding mobile device MD_I uploads data packets to the edge server in advance, After the task processing is completed, the data generated by subtask Q_{iw} is downloaded to the terminal memory.

$$\begin{aligned} T_{iw}^{core} &= T_{iw}^{up} + T_{iw}^{core} + T_{iw}^{down} \\ &= D_{iw}/v_{up} + D_{up}/v_{core} + D_{down}/v_{down} \end{aligned} \tag{3}$$

The amount of data to be downloaded after the execution is D_{down}, mainly related to the complexity of operations and the amount of original data. Set the ratio of D_{up} and D_{down} as γ.

$$\gamma = D_{up}/D_{down} \tag{4}$$

The wireless transmission mode is mostly used when the MDs offloads the subtask to the MESs. It divides the frequency band into channels, which are more vulnerable to interference than the wired mode.

$$T_{iw}^{up} = \frac{(DATA_{upload} + DATA_{original}) \cdot Dis}{speed\_upload \cdot \mu} \tag{5}$$

μ is the channel interference factor, which is negatively correlated with the density of devices that send requests simultaneously in the area. The up- load speed will slow when the surrounding mobile devices exceed the threshold. When MESs completes the subtask, the calculated data will be downloaded by the MDs. This process is less affected by electromagnetic interference. The transmission rate is relatively stable due to the strong transmission signal.

After the task is successfully uploaded to the receiving end, the receiving end will call the computing resources for calculation. The computing time is mainly determined by the

amount of data, the complexity of the task, and the computing speed of the server itself.

$$T_{solve} = \frac{DATA_{solve}}{speed\_core \cdot \omega} \tag{6}$$

The operation speed is expressed by the clock frequency of the processing core.

The energy consumption of mobile terminal equipment in edge computing networks is divided into static and dynamic energy consumption. When the mobile device has no computing task, the stored power will also be consumed in the vacant state. In data transmission and processing, there will be corresponding dynamic energy consumption. For mobile device subtask Q_{iw}, when the equipment itself processes it, the corresponding equipment energy consumption E^{self} is:

$$E^{self} = P^{solve} \cdot T^{solve} \tag{7}$$

Where T^{solve} is the time of processing subtask Q_{iw} by MD_i in (6). P^{solve} is the rated power of mobile device MD_i when performing calculation tasks. It is related to the performance of the MDs.

Node EC_i offload task to node EC_j when processed, EC_i will generate energy consumption in uploading and downloading. Meanwhile, EC_j will also generates corresponding energy consumption due to downloading and uploading data and sending operation results.

$$E^{core} = P^{up}T^{up} + P^{down}T^{down} + P^{static} \tag{8}$$

The mobile edge computing system model in area R_i can be summarized as follows:

Where ϕ^{self} and ϕ^{core} are the 0-1 variables, indicating the location of processing subtask Q_{iw}

$$min \quad \overline{T} = \frac{1}{m}\sum_{i=1}^{m}\sum_{w=1}^{c_i}(\varphi_w^{self}\frac{D_{iw}}{v^{self}} + \varphi_w^{core}(\frac{D^{up}}{v^{up}} + \frac{D}{v^{core}} + \frac{D^{down}}{v^{down}})),$$

$$min \quad \overline{E} = \frac{1}{m}\sum_{i=1}^{m}\sum_{w=1}^{c_i}(\varphi_w^{self}P^{self}T^{self} + \varphi_w^{core}(P^{up}T^{up} + P^{down}T^{down} + P^{static})) \tag{9}$$

Through utilizing the potential similarity of resource allocation in edge computing scenarios in different regions, the populations of different optimization problems are exchanged to speed up the convergence of the solution process.

The problems to be optimized under the condition of parallel processing are:

$$\begin{aligned} min \quad & \{\overline{T}_1, \overline{T}_2, \ldots, \overline{T}_K\}, \\ min \quad & \{\overline{E}_1, \overline{E}_2, \ldots, \overline{E}_K\} \end{aligned} \tag{10}$$

The average delay and energy consumption of mobile terminal devices in K edge computing networks are simultaneously minimized.

For each area R_k, the mobile device can decompose more complex tasks and offload them to the edge server for processing. The computing speed is greatly accelerated because the edge server has sufficient computing resources. However, the mobile device needs additional energy to upload and download data. When the amount of data is too large, the energy consumption generated by data transmission may exceed that processed by the device itself in this process. The storage capacity of mobile devices is limited, which is challenging to support too frequent task offloading. Therefore, minimizing the average energy consumption $\overline{T}$ and average delay $\overline{E}$ simultaneously in the region is challenging. The reduction of one target must be at the cost of increasing another target.

Although the optimal offloading schemes of mobile devices in different regions have no correlation and no numerical relationship, their edge computing networks have potential similarities, so they can provide reference to each other and speed up the optimization process. In the following sections, this similarity and its use are further discussed.

4. Proposed Algorithm

This paper combines the idea of multitasking with the traditional multi-objective evolutionary algorithm. Based on the classical evolutionary algorithm NSGA-III [21], a multi-objective evolutionary optimization algorithm is proposed to optimize the task offloading scheme of intelligent terminals in the target area in the edge computing scene. Compared with NSGA-III, the simplified reference point generation method is used to replace the straight-line distance with the angular relationship, while using the additional information provided by the reference point to improve the uniformity of the solution, the computational resources consumed in this link are reduced.

4.1. The Main Structure of MOEA-RP

The main steps of the algorithm are summarized as follows:

Step 1: Code the individual through the algorithm 1 to get the initial population P_0, randomly allocate skill factor $tau_k(k = 1 \ldots K)$. The reference point set Z is obtained from the reference point generation strategy;

Step 2: Decode the initial population through the algorithm 2, and calculate the initial population P by the formula (9). The objective function values of individuals in $\{\overline{E_1}, \overline{E_2}, \ldots, \overline{E_K}\}$ and $\{\overline{T_1}, \overline{T_2}, \ldots, \overline{T_K}\}$, initialization ideal point $Z^* = [\overline{E}_{min}, \overline{T}_{min}]$;

Step3: Start iteration. Adaptive learning communication parameter rmp, dynamically control of different skill factors τ_k; Randomly select the parent individuals, further expand the solution space through cross mutation, and obtain the offspring population S_t, children retain the skill factor of the parent through inheritance strategy; Decode the offspring individuals, calculate their fitness values $\overline{E}$ and $\overline{T}$, and update the ideal point Z^*; The parent and child populations are merged, and the non-dominated sorting is carried out with the help of the information of the reference point set Z, and the optimal individual solution with the same number as the initial population is retained;

Step 4: Reach the upper limit of iteration times. Output the non-dominated solution corresponding to each skill factor, decode for different regions, and obtain the optimal task unloading scheme for each region.

4.2. Encode and Initialization

For k areas with edge computing networks $R_1, R_2, \ldots, R_K$, where the numbers of edge computing servers and mobile devices is $m_1, m_2, \ldots, m_K, n_1, n_2, \ldots, n_K$, respectively. The task to be processed is $Q_1, Q_2, \ldots, Q_S$, whose corresponding number of decomposable subtasks is $c_1, c_2, \ldots, c_S$. The number of individuals included in the initial population P_0 is N. Each individual can be transformed into a solution for the sub-task allocation scheme of mobile terminal equipment in different regions through unique decoding methods. Each individual in the population is represented in the form of a matrix of the

same size. The number of rows of the matrix is the maximum number of mobile terminal devices ineach area. The number of columns is the maximum number of decomposable copies of the task to be processed. Namely:

$$\alpha = max\{m_1, m_2, \ldots, m_K\}$$
$$\beta = max\{c_1, c_2, \ldots, c_K\} \quad (11)$$

Take one of the individuals p_{t_i} in t-generation as an example:

$$P_{t_i} = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1\beta} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{\alpha 1} & \phi_{\alpha 2} & \cdots & \phi_{\alpha\beta} \end{bmatrix} \quad (12)$$

Where $\phi_{ij}(i = 1 \ldots \alpha, j = 1 \ldots \beta)$ The value range of is a floating-point number between [0, 1], which is used to represent the task processing position of each user after decomposition. $\phi_{ij} = 0$ means that the device itself processes the corresponding subtask, otherwise, it is offloaded to the edge computing server for processing. The probability of 0 is constant ξ. At the same time, the skill factor $\tau_k(k = 1 \ldots K)$ is randomly assigned to the initialized population.

4.3. Mating

Two individuals p_i and p_j are selected each time randomly. If they have the same skill factors, means that the two candidate solutions are for task allocation in the same area, the crossover and mutation process can be carried out directly. The newly generated two offspring individuals inherit the skill factors of their parents.

If the two skill factors differ, they are candidate solutions for task allocation in different regions. The extent of mating is limited by the parameter rmp. If the generated random number is less than rmp, p_i and p_j can exchange information according to the established crossover rules, and the generated offspring s_i and s_j randomly inherit the skill factor of the parent τ_i, τ_j. Otherwise, the parent individual p_i, pjcrosses the crossover step and mutates respectively to generate offspring s_i, s_j directly inherits the skill factors of their parents. An online learning method with parameter rmp has been introduced, quantifying its similarity based on the population distribution with different skill factors in the iterative process. Unlike the static method of specifying specific values in advance, this method can dynamically change the similarity evaluation with the population update and has good adaptability. This method is briefly described below. See the original literature for detailed derivation and details. Set the candidate solution population p_t^k for the k region matches the distribution q_{ct}^k, it can be expressed as:

$$q_{ct}^k = (1 - \frac{0.5}{K}\sum_{k \neq j} rmp_{kj}) \cdot q_t^k + \frac{0.5}{K}\sum_{j \neq k} rmp_{kj} \cdot q_t^j \quad (13)$$

There are many methods to measure distribution differences, such as KL divergence, JS divergence, cross-entropy, Wasserstein distance, and others.

Considering the problem characteristics and subsequent operation steps, KL divergence is selected to measure the distribution q_{ct}^k of population p_t^k:

$$D_{KL}(p_t^k \| q_{ct}^k) = \sum_{i=1}^{N} P_t^k \cdot (\log P_t^k - \log q_{ct}^k) \quad (14)$$

In (14), p_t^k is a constant. According to (13), rmp is the only variable affecting the distribution and the difference between populations. From (13) and (14), q_{ct}^k and $D_{KL}(p_t^k \| q_{ct}^k)$ are equivalent:

$$\arg\min_{rmp_{kj}} D_{KL}(p_t^k \| q_{ct}^k) \Leftrightarrow \arg\min_{rmp_{kj}} \sum_{k=1}^{K}\sum_{i=1}^{N} q_c^k(x_{ik}, t) \quad (15)$$

The minimization problem is a convex optimization problem, which can be solved generally. The optimal rmp_{kj} is the similarity measure between the region k and the region j user task offloading candidate solution. Figure 2 shows the parent individual p_i, p_j. Generate the sites $pos_1, pos_2 \in [1 \ldots \beta]$ randomly, where $pos_1 < pos_2$. Individuals p_i, p_j are in matrix form. Exchange the columns between $[pos_1, pos_2]$ to generate offspring s_i, s_j, respectively. Generate site $pos_0 \in [1 \ldots \beta]$ randomly to generate a new matrix ptemp as shown in the figure replaces the pos_0 of the individual column. Where matrix ptemp the value of is [0, 1], and the probability of 0 is constant ξ.

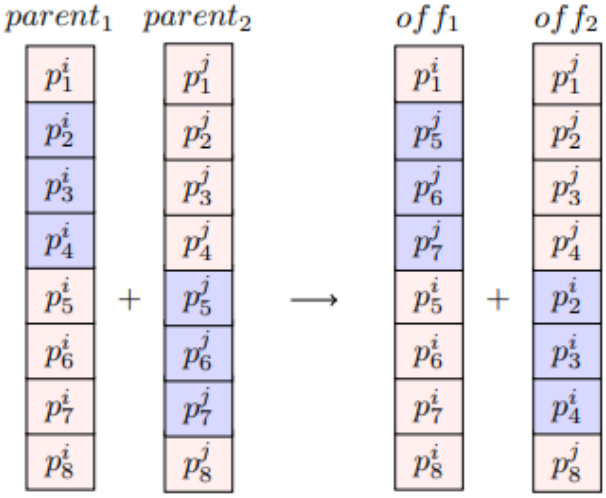


Figure 2. Crossover

Algorithm 1 Population Initialization

Input: The number of tasks and the MDs and MESs in all the areas, ξ
Output: The initial population: P_0

```
Search for the area with most MDs : M = max{m_1, m_2, ..., m_K}
Search for the task with most parts : C = max{c_1, c_2, ..., c_K}
Set the size of each individual as M × C
For each individual p^i do
    For each decomposed tasks do
        If rand < ξ then
            p^i_cm = 0  // solved by MDs locally
        else
            p^i_cm = rand  // solved by the MESs
        End if
        Assign skill factor τ_k(k = 1...K)  // specific for one area
    End for update the population : P_0 ← P_0 ∪ p^i
End for
```

4.4. Population Decode

In the framework of the multitasking algorithm, the individuals in the population are coded and no longer target the task allocation scheme in a specific region, which has practical significance. Therefore, it is necessary to decode so

that the same individual can represent multiple allocation schemes simultaneously. Take the individual pi in iteration t as an example, the decoding method for two regions. The number of edge computing servers in area R_1, R_2 is 4 and 5, respectively. Segments round the values in the individual with the node set {0.25,0.5,0.75,1.0} and {0.2,0.4,0.6,0.8,1.0} respectively, to realize the one-to-one correspondence from the floating-point value to the server number. Individual codes are redundant for regions with fewer users and users with few- er total tasks. The redundant part is truncated and discarded at this time, and only the required length is used for calculation. Thus, individual decoding is realized according to its skill factor τ_i, which calculates the corresponding objective function value through (9).

Algorithm 2 Population Decode

Input: The encoded population P_t, the position, and the tasks of MDs and MESs in all the areas
Output: The Decoded population P_t^*

Calculate the distance between MDs and MESs specifically : $\{D_1, D_2, \dots, D_K\}$
Order the MESs for all the MDs according to $D_k(K = 1 \dots K)$
For each individual p^i **do**
  If $p_{cm}^{i*} == 0$ **then**
    Continue // solved by MDs locally
  else
    Calculate pos : $p_{cm} \in [pos/D_k, (pos+1)/D_k]$
    $p_{cm}^{i*} \leftarrow pos$ // Encode the number of MES
  End if
  Update the population : $P_t^* \leftarrow P_t^* \cup p^{i*}$
End for

4.5. Non-dominated Sorting

Algorithm 1 Population Initialization

Input: The encoded population Pt, the set of reference points Z, the ideal point Z^*
Output: The population for the next iteration: P_{t+1}

Calculate $\overline{E}$ and $\overline{T}$ for each individuals by (9)
Calculate the rank F of all the individuals by $\{\overline{E}, \overline{T}\}$
Update the ideal point Z^* by min $\overline{E}$ and min $\overline{T}$
Repeat
  $P_t = P_t \cup F_i$
Until $|P_t| \geq N$
Update the last nondominated front: $F_l \leftarrow F_i$
If $|P_t| = N$ then
  $P_{t+1} = P_t$
  break
else
  $P_{t+1} = \cup_{j=1}^{l-1} F_j$ // maintain individuals in the last $l-1$ front
  Calculate the number of individuals F_l: $K = N - |P_{t+1}|$
  Associate each individual in P_t with a single reference point
  Compute the number of associated individuals of the reference points
  Select K individuals from F_l to construct P_{t+1}
End for

They are sorted according to their objective function values by defining the dominating relationship between individuals to retain the optimal individuals. After cross and mutation, parent Pt and offspring St are merged to form a population with a size of $2N$. For individual p_i and p_j, if $obj1_i < obj1_J$ and $obj2_i < obj2_j$, then it is called individual p_j individual p_i control. The set of individuals with no dominant relationship with each other constitutes the same level of the non-dominant plane. Generate K reference points in $[0, \frac{\pi}{2}]$, which is the same as the number of candidate solutions to be selected in the llayer. The generated reference points are evenly distributed on the curve formed by the two objective functions. The angle between the kreference point and the x-axis is $\frac{K_\pi}{2K}$.

Standardization keeps the values of two objective functions in the same order of magnitude. The normalized values of the two objective functions fall within the range of [0,1], and the angle with the x-axis is θ. Reserved: select k individuals in the last layer. Firstly, the cosine function value of the angle formed by the standardization of two objective functions of each individual is calculated, the absolute value of the cosine function value of the set of reference points is taken as the distance between the individual and each reference point. The one with the smallest distance is the reference point corresponding to the individual. If multiple individuals correspond to the same reference point, only the one with the smallest distance is retained. Select k individuals with the smallest distance to keep. Calculate the value of the objective function corresponding to the individual of layer F_l and limit it to the interval of [0,1] through maximum- minimum standardization. Calculate the ratio of the two objective functions, and obtain the angle θ between the individual and the origin coordinate through the formula (16).

$$\theta = \arctan\left(\frac{\overline{obj_1}}{\overline{obj_2}}\right) \tag{16}$$

Determine the range of individuals $[\theta_a, \theta_b]$ in combination with the pregenerated reference point angle and Z, and calculate the difference between them and the left and proper boundaries, respectively. Take the smaller one as its corresponding distance:

$$\theta_{dis} = \arg\min_{\theta}\{\theta - \theta_a, \theta_b - \theta\} \tag{17}$$

The reference points in the set Zare sorted monotonically and incrementally according to the angle with the x-axis, i.e., $theta_i < \theta_j, i < j$. Calculate the angle between each individual in the last layer, the x-axis, and the nearest reservation to the reference point.

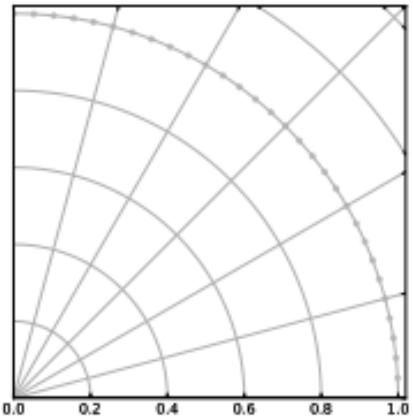


Figure 3. Reference Points

5. Result

5.1. Test Instances

The user task in area R1 and area R2 is divided into two optimization problems to be processed simultaneously. The area size is $200 \times 200\ m^2$. The number of edge servers in area R_1 is 4, and the coordinates are [50,50], [50,150], [150,50], [150,150]. The number of edge servers in area R_2 is 5. Based on the above location, add servers [100,100]. The number of users in the area is 200-1200, and the user location is randomly generated. Set the population size to 100, and the number of reference points is the same as the population size. Cross factor $\mu_c = 0.9$, variation factor $\mu_m = 0.1$. Each algorithm involved in the comparison runs independently 20 times. The parameters of MESs and MDs are shown in table 2.

Table 2. Parameter setting

Parameter	Value
Area size	200mx200m
c_i	$[3,7]MB$
σ_{iw}	$[100,200]KB$
υ_{self_i}	$[8,10]MB$ FLOP per second
υ_{core}	$[5,10]GB$ FLOP per second
γ	$[0.2,0.5]$
μ	$[0.05,0.10]$

5.2. Performance Metric

IGD and GD [22] are selected to evaluate the optimization effect of each algorithm.

$$IGD(PF,PF^*) = \frac{\sum_{i=1}^{|PF|} \min_{j=1}^{|PF^*|} \|p_i - p_j^*\|}{|PF|} \quad (18)$$

In which PF is composed of the nondominated solutions obtained by the iterative solution of an algorithm. PF^* is the set of nondominated solutions obtained by multiple algorithms during independent operations. The average delay $\overline{T}$ and average energy consumption $\overline{T}$ are less than the set of candidate solutions of other schemes.

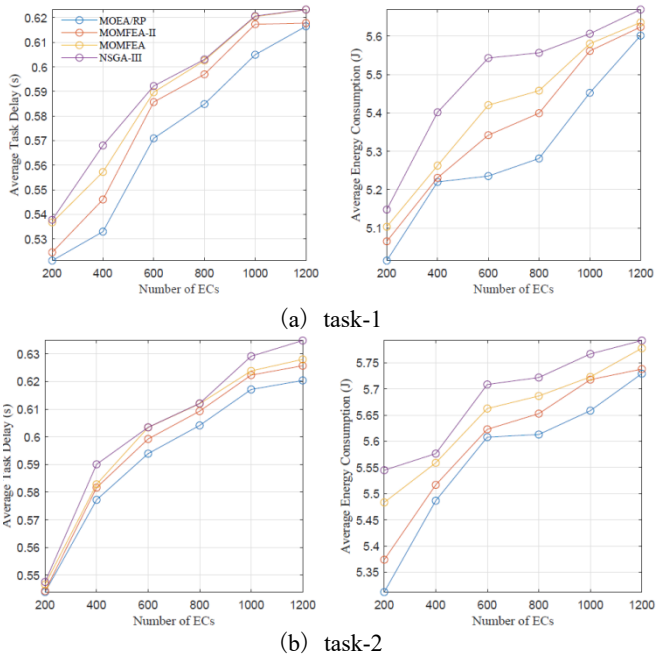
$$GD(PF,PF^*) = \sqrt{\frac{\sum_{i=1}^{|PF|} d(PF,PF^*)}{|PF|}} \quad (19)$$

Where $d(PF,PF^*)$ is the Euclidean distance between the non-dominated solutions obtained by the algorithm and the nondominated plane.

5.3. Comparison Results

This paper selects the classic multitasking evolutionary algorithms MOM- FEA [23], MOMFEA-II [24], and NSGA-III [21] based on reference points as the comparison algorithm. Among them, NSGA-III, a single-task

Figure 4 shows the minimum values of average waiting delay and average equipment energy consumption when the number of users in the area increases step by step. It can be seen from the figure that the number of users increases with the increase in transmission speed, and the main reason is that the number of users decreases with the increase in transmission speed. Algorithm MOEA-RP in area R_1 and area R_2 It minimizes the realization of the two objectives 420 within and has obvious advantages over other comparison algorithms.



(a) task-1

(b) task-2

Figure 4. Two independent regions R_1 and R_2 Average delay and energy consumption of users

Table 3. IGD of instances

Instance	NSGA-III	MOMFEA	MOMFEA-II	MOEA-RP
1	7.86(1.06)	7.64(0.15)	6.45(0.31)	**0.19(0.21)**
	6.03(0.91)	7.24(0.21)	6.23(0.20)	**0.15(0.07)**
2	4.88(0.25)	4.19(0.11)	3.24(0.35)	**0.47(0.14)**
	3.25(0.18)	3.76(0.72)	3.12(0.31)	**0.31(0.21)**
3	3.14(0.16)	3.11(0.24)	2.51(0.25)	**0.81(0.29)**
	2.74(0.23)	2.45(0.15)	1.82(0.13)	**0.72(0.23)**
4	5.74(0.22)	5.48(0.16)	4.32(0.14)	**0.55(0.21)**
	4.12(0.13)	4.23(0.34)	2.53(0.26)	**0.24(0.18)**
5	8.91(0.39)	7.06(0.19)	5.21(0.25)	**1.14(0.30)**
	7.65(0.25)	5.92(0.17)	4.22(0.17)	**0.73(0.14)**
6	9.60(0.39)	8.23(0.25)	3.25(0.13)	**1.68(0.55)**
	8.72(0.42)	7.25(0.43)	2.51(0.34)	**1.21(0.23)**

Table 3 lists the IGD values of the algorithm in different simulation examples, and the minimum value in each subtask is marked in bold. Because IGD can reflect the convergence and uniformity of the optimal solution to a certain extent, the smaller the value is, the closer it is to the real non-dominated plane. MOEA-RP obtains the minimum IGD value in many examples compared to the comparison algorithm, which shows that the algorithm has good convergence and stable performance.

Figure 5 and figure 6 respectively draw the area R_1 during independent operations and area R_2, MOEA-RP and comparison algorithm are the nondominated plane closest to the average IGD value. MOEA-RP algorithm reduces the average delay and energy consumption in the two regions to a better value and has obvious advantages over other algorithms. Compared with the single-task algorithm NSGA-III, MOEA-RP makes full use of the population similarity of different subtasks with the help of multitasking ideas and adaptive learning parameters. It makes individuals with different skill factors promote convergence to effectively expand the search solution space and accelerate the convergence speed. Compared with the multitasking algorithms MOMFEA and MOMFEA- II, the algorithm MOEA-RP uses the pre-generated reference points to provide additional information when performing non-dominated

sorting. It provides direction guidance for individuals in the population together with the constantly updated ideal points to obtain a more representative non-dominated solution on the premise of ensuring the convergence of the solution and taking into account the uniformity.

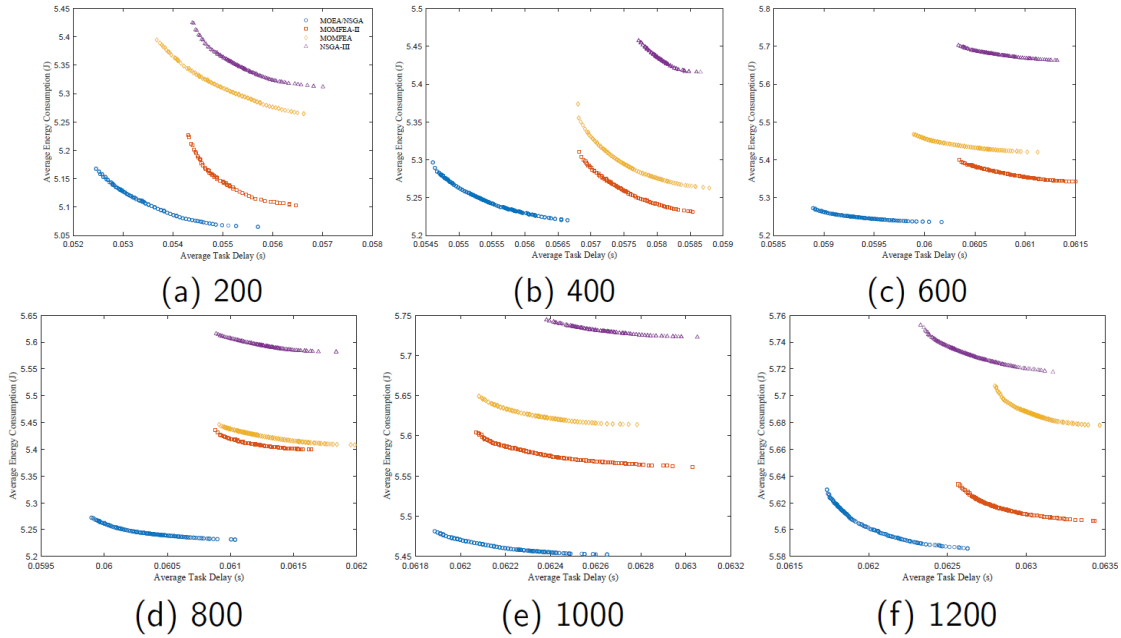


Figure 5. Average energy consumption delay PF surface obtained by each algorithm within area R_1

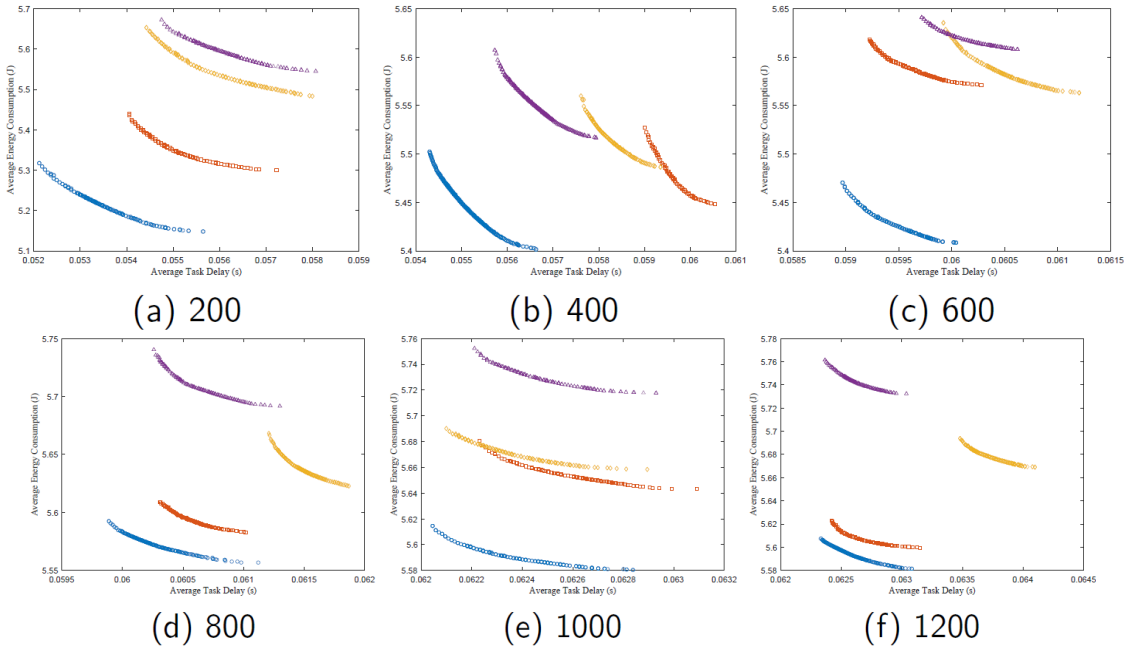


Figure 6. Average energy consumption delay PF surface obtained by each algorithm within area R_2

6. Conclusion

As the task complexity and data volume of personal terminal processing rise, the edge computing server is set in the area near the user. By decomposing some tasks and offloading them to the server for processing, the user waiting delay and equipment energy consumption can be reduced to improve the user experience. For optimizing the task offloading schemes of users in the region, most of the existing research results choose to integrate it into single objective planning, which is challenging to cover the containment relationship between delay and energy consumption comprehensively. Based on this, this paper establishes the system model of server user equipment, proposes an adaptive multitasking evolutionary algorithm, designs the corresponding encoding and decoding strategy, and realizes the solution of the objective function. The performance of the algorithm is verified on several test examples. Compared with the comparison algorithm, it achieves better convergence speed, convergence effect, and optimal solution distribution uniformity. Although the algorithm has obvious advantages in the final effect, it still has room for improvement. The unique coding method used in this paper is highly related to the number of users in the region and the decomposition method of processing tasks. With the increase of users, the amount of data processed in the iterative process increases significantly, resulting in the slowdown of optimization speed. In the follow-up research, we will explore more streamlined coding methods to reduce the consumption of computing resources

caused by information redundancy.

Acknowledgments

This work was supported by the Research project of undergraduate teaching reform and quality construction in Tianjin colleges and universities in 2023 (B231005806). Tianjin Research Innovation Project for Postgraduate Students (2022SKY145).

Conflicts of Interests

There are no conflicts of interest.

References

[1] J. Liu, Y. Mao, J. Zhang, K. B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, IEEE International Symposium on Information Theory (ISIT) (2016) 1451–1455doi:10.1109/isit. 2016.7541 539.

[2] Y. Mao, J. Zhang, K. B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, IEEE Journal 480 on Selected Areas in Communications 34(12) (2016) 3590—3605. doi: 10.1109/ jsac. 2016.2611964.

[3] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, K. Huang, Energy harvesting wireless communications: A review of recent advances, IEEE Journal on Selected Areas in Communications 33(3) (2015) 360—381. 485 doi:10.1109 /jsac.2015.2391531.

[4] C. You, K. Huang, Multiuser resource allocation for mobile-edge computation offloading, IEEE Global Communications Conference (GLOBECOM) (2016) 1–6doi:10.1109/glocom. 2016.7842016..

[5] P. Zhao, H. Tian, C. Qin, G. Nie, Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing, IEEE Access 5 (2017) 11255—11268. doi:10.1109/access.2017.2710056.J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” IEEE J. Quantum Electron., submitted for publication.

[6] X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint computation and communication cooperation for mobile edge computing, 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) (2018) 1–6doi:10.23919/wiopt.2018.8362865.

[7] X. Lyu, H. Tian, L. Jiang, A. V. S. Maharjan, S. Gjessing, Y. Zhang, Selective offloading in mobile edge computing for the green internet of things, IEEE Network 32(1) (2018) 54–60. doi:10.1109/mnet.2018.1700101.

[8] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Transactions on Networking 24(5) (2016) 2795—2808. doi:10.1109/tnet.2015.2487344.

[9] T. Q. Dinh, J. Tang, Q. D. La, T. Q. Quek, Offloading in mobile edge computing: task allocation and computational frequency scaling, IEEE Transaction on Communications 65(8) (2017) 3571–3584. doi:10.1109/tcomm.2017.2699660.

[10] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE Journal on Selected Areas in Communications 36(3) (2018) 587–597. doi:10.1109/jsac.2018.281536.

[11] Z. Ning, P. Dong, X. Kong, F. Xia, A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things, IEEE Internet of Things Journal 6(3) (2018) 4804–4814. doi:10.1109/jiot. 2018.2868616..

[12] Y. Li, F. Qi, Z. Wang, X. Yu, S. Shao, Distributed edge computing offloading algorithm based on deep reinforcement learning, IEEE Access 8 (2020) 85204–85215. doi:10. 1109/ access.2020.2991773..

[13] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, Joint computation offloading and multi-user scheduling using approximate dynamic programming in nbiot edge computing system, IEEE Internet of Things Journal 6(3) (2019) 5345–5362. doi:10.1109/jiot.2019.2900550.

[14] L. Rui, Y. Yang, X. Qiu, Computation offloading in a mobile edge communication network: a joint transmission delay and energy consumption dynamic awareness mechanism, IEEE Internet of Things Journal 6(6) (2019) 10546–10559. doi:10. 1109/ jiot.2019.2939874.

[15] F. Sufyan, A. Banerjee, Computation offloading for distributed mobile edge computing network: A multiobjective approach, IEEE Access 8 (2020) 149915–149930. doi:10.1109/ access. 2020. 3016046.

[16] L. Liu, Z. Chang, X. Guo, T. Ristaniemi, Multi-objective optimization for computation offloading in mobile-edge computing, IEEE Symposium on Computers and Communications (ISCC) (2017) 832–837doi:10.1109/ iscc. 2017. 8024630..

[17] Z. Ning, P. Dong, X. Wang, Deep reinforcement learning for vehicular edge computing: an intelligent offloading system, ACM Transactions on Intelligent Systems and Technology (TIST) 10(6) (2019) 1–24. doi: 10.1145/3317572.

[18] J. Yan, S. Bi, L. Huang, Y.-J. A. Zhang, Deep reinforcement learning based offloading for mobile edge computing with general task graph, IEEE International Conference on Communications (ICC) (2020) 1–7doi:10.1109/ icc40277. 2020.9148846.

[19] D. K, P. A, A. S, M. T, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2002) 182– 197. doi:10.1109/4235.996017.

[20] Z. Qingfu, L. Hui, Moea/d: a multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (2007) 712–731. doi:10. 1109/ TEVC. 2007.892759..

[21] D. K, J. H, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (2014) 577–601. doi:10.1109/TEVC.2013.228153.

[22] O. Schuetze, X. Esquivel, Adriana, C. A. C. Coello, Some comments on gd and igd and relations to the hausdorff distance, Proceedings of the 12th Annual Conference Comp on Genetic and Evolutionary Computationdoi: 10.1145/1830761.1830837.

[23] G. Abhishek, O. Yew-Soon, F. Liang, T. K. Chen, Multiobjective multifactorial optimization in evolutionary multitasking, IEEE Transactions on 555 Cybernetics 47 (2017) 1652–1665. doi:10.1109/TCYB.2016.2554622.

[24] B. K. Kumar, G. Abhishek, O. Yew-Son, T. P. Siew, Cognizant multitasking in multiobjective multifactorial evolution: Mo-mfea-ii, IEEE Transactions on Cybernetics 51 (2020) 1–13. doi: 10. 1109/TCYB.2020.2981733.