Many-objective joint optimization for dependency-aware task offloading and service caching in mobile edge computing

Xiangyu Shi[1], Zhixia Zhang[1], Zhihua Cui[1] and Xingjuan Cai[1*]
[1] Shanxi Key Laboratory of Big Data Analysis and Parallel Computing, Taiyuan University of Science and Technology, Taiyuan, 030024, Shanxi, China
[e-mail: xingjuancai@163.com]
[*]Corresponding author: Xingjuan Cai

Received December 26, 2023; revised April 13, 2024; accepted April 23, 2024; published May 31, 2024

Abstract

Previous studies on joint optimization of computation offloading and service caching policies in Mobile Edge Computing (MEC) have often neglected the impact of dependency-aware subtasks, edge server resource constraints, and multiple users on policy formulation. To remedy this deficiency, this paper proposes a many-objective joint optimization dependency-aware task offloading and service caching model (MaJDTOSC). MaJDTOSC considers the impact of dependencies between subtasks on the joint optimization problem of task offloading and service caching in multi-user, resource-constrained MEC scenarios, and takes the task completion time, energy consumption, subtask hit rate, load variability, and storage resource utilization as optimization objectives. Meanwhile, in order to better solve MaJDTOSC, a many-objective evolutionary algorithm TSMSNSGAIII based on a three-stage mating selection strategy is proposed. Simulation results show that TSMSNSGAIII exhibits an excellent and stable performance in solving MaJDTOSC with different number of users setting and can converge faster. Therefore, it is believed that TSMSNSGAIII can provide appropriate sub-task offloading and service caching strategies in multi-user and resource-constrained MEC scenarios, which can greatly improve the system offloading efficiency and enhance the user experience.

Keywords: Mobile Edge Computing (MEC), Dependency Aware Task Offloading Service Caching, Multi-user, Resource Constraint, Many-objective Evolutionary Algorithm.

1. Introduction

Applications that are more sensitive to latency and resource needs are currently proliferating in fields including data stream processing, face recognition, and virtual/augmented reality, because mobile devices and the Internet of Things are so widely used [1]. But as of right now, cloud computing platforms or mobile devices are mostly used to process and execute these apps. It cannot be ignored that, on the one hand, many applications require computing resources that are not met on mobile devices. On the other hand, deploying resource-intensive applications on cloud platforms often requires large amounts on data to be transferred to and from mobile devices to remote servers in the cloud, leading to unpredictable communication latency problems [2]. It is with these considerations in mind that MEC was born [3] and has emerged as a much-anticipated solution to remedy many of the shortcomings brought about by the aforementioned problems. Meanwhile, the application of multi-objective evolutionary algorithms in various fields [4-10] also provides new ideas to solve the problems in MEC.

In practical applications, MEC still has a lot of obstacles to overcome. Based on 4 million apps, Alibaba's statistics indicates that over 75% of applications (tasks) are made up of dependent subtasks. Nevertheless, contemporary mobile apps frequently have several interdependent subtasks. These dependence subtasks must be carried out in a certain order and must be supported by particular services [11]. For example, the user must first input image loading in a face recognition application; before face detection and feature extraction, the image usually needs to be preprocessed; once the image loading and preprocessing are completed, the next step is to detect the face in the image; after the face has been acquired After obtaining the face's feature representation, it can be recognized or its identity verified by comparing it to the known features stored in the database. The following factors must be carefully considered when offloading these dependency-aware subtasks to edge nodes:

•**Dependency perception between subtasks:** the execution order of subtasks is constrained by the dependencies between them. A subtask can only start processing if it receives the output from all predecessor subtasks, and it needs to forward the obtained results to the successor subtasks for subsequent processing. For example, in the face recognition application described above, "Face Detection" subtasks' outputs are inputs to "Face Alignment" subtasks. Therefore, the "Face alignment" subtask can only begin when the "Face detection" subtask is finished.

•**Dependency-aware correlation between subtask offloading and service caching:** subtask execution depends on the support from the corresponding service, this means that the corresponding services supporting subtask processing must be cached on edge servers where subtasks are offloaded to. For example, the "positioning" sub-task of a map navigation task should be offloaded to edge server which already caches GPS and BeiDou satellite services.

•**Multi-user, resource-constrained scenarios pose challenges for dependency-aware subtask offloading:** a task is usually partitioned into multiple dependency subtasks for offloading, the number of dependency subtasks to be offloaded (decision-variable dimensions) will explode with the growth in the number of users, and edge servers typically with finite compute and store resources. Therefore, how to assign these huge number of dependent subtasks to appropriate edge servers (resource constrained) will greatly affect the offloading efficiency in edge computing environments.

The dependency-aware subtask offloading and service caching strategies formulated after comprehensive consideration of the above factors will be more applicable to realistic MEC scenarios. It will greatly improve the subtask hit service rate and greatly improve the offloading efficiency of the MEC system. It can also fully utilize the computing and storage resources of edge servers in the MEC system to avoid the waste of resources. The main thing is that it improves the

user experience and can provide users with low latency and low energy consumption.

The primary contributions of this paper can be summed up as follows:

1) To solve the joint optimization problem of dependency-aware subtask offloading and service caching under multi-user, resource-constrained MEC environments, a many-objective model for joint optimization of dependency-aware task offloading and service caching (MaJDTOSC) is constructed, taking into account the dependency-awareness of subtasks, the correlation between dependency-aware subtask offloading and service caching, as well as the existence of the challenges of such environments.
2) In order to develop efficient, suitable and adaptable dependency subtask offloading, service caching strategies for multi-user, resource constrained edge computing environments. Five optimization objectives are set in MaJDTOSC: task completion time, task processing energy consumption, subtask hit rate, load variability, and storage resource utilization. Inter-subtask dependencies and resource-constrained edge servers are also considered as their constraints.
3) Because the constructed MaJDTOSC belongs to many-objective optimization problem and "dimension explosion" of decision variables occurs when the task is partitioned into a series of perceptually dependent subtasks. A based on Three-Stages Mating Selection (TSMS) strategy's many-objective evolutionary algorithm TSMSNSGAIII is presented to better solve MaJDTOSC to obtain a suitable and efficient dependency subtask offloading and service caching decision.

2. Related Works

Due to its urgency and criticality, task offloading has steadily emerged as one of the main research topics in the framework of MEC in recent years. Tang et al. [12] investigated the problem of non-separable and latency-sensitive task offloading in dynamic edge-loaded environments. They proposed a distributed algorithm based on model-free deep reinforcement learning to solve the problem, aiming at minimizing the long-term cost. Zhou et al. [13] investigated joint optimization issues for resource allocation and computational offloading for dynamic multiuser MEC systems. Their objective was to minimize the energy consumption of the entire MEC system. A reinforcement learning approach based on value iteration was proposed to determine the resource allocation and computational offloading strategies. Xu et al. [14] investigated how to minimize the task processing latency for Internet of Vehicles (IoV) users in the presence of limited edge server resources. And they designed a fuzzy task offloading and resource allocation scheme: using game theory to determine the optimal task offloading strategy for IoV users, and using Q-learning to determine the resource allocation strategy. Nguyen et al. [15] proposed a new collaborative block mining and task offloading scheme for blockchain based MEC systems. In order to solve the latency issue created by blockchain operations in MEC, a consensus mechanism was developed to maximize the utility of the system. Yang et al. [16] studied a MEC system made up of mobile devices supporting various radio access technologies and heterogeneous edge servers. They constructed the process of determining the optimal offloading location as a Markov Decision Process (MDP) which is solved using the Value Iteration Algorithm (VIA).

All of the above work focuses on how tasks are completely offloaded. Mobile applications may have numerous interdependent tasks as modern MEC applications become more complicated. Offloading dependent tasks is therefore required in many real-world MEC applications. Zhao et al. [17] studied ways to offload dependency tasks to edge nodes with service caches and designed a convex planning based algorithm to solve the problem. They

also studied a special case of the problem and presented an approximation algorithm with bounded approximation factors to solve this case. Shen et al. [18] investigated the service caching and dependency-aware task offloading problem of VEC. Their aim was to maximize the offloading efficiency. To address the problem, they developed a semi-distributed algorithm based on dynamic planning. A dependency-aware offloading approach based on edge cloud collaboration was developed by Chen et al. [19]. They separated the offloading problem into two subproblems: minimizing the application completion time in two different collaboration models. The two subproblems were solved, respectively, by a greedy algorithm and an efficient greedy method. The dependency task offloading problem was examined by Nguyen et al. [20] in a cooperative UAV-assisted MEC scenario, and divides the problem into two subproblems: communication resource allocation and offloading decision. A suboptimal solution to the former problem is found using a meta-heuristic, meanwhile convex optimization is used to address the second problem. An et al. [21] jointly optimized the dependency task offloading strategy and the allocation strategy of communication and computation resources under fast fading and slow fading channels. The aim is to minimize the energy consumption of each IoT device. Yan et al. [22] explored methods to acquire the best dependency task offloading and resource allocation policies. Their objective was to reduce the weighted sum of user's task execution time and energy consumption in two-user MEC networks.

Actually, the influence of service caching on task offloading efficiency should be taken into account in addition to dependent task offloading. For example, in **Fig. 1**, assume that four dependency-aware subtasks need to be offloaded in a certain edge computing scenario. Among them, subtask 3 depends on subtask 2 and subtask 4 depends on subtask 1. In the left figure, assuming that all types of services are stored on all edge servers without considering the service caching status, the optimization yields an offloading policy where subtasks 1 and 4 are offloaded to edge server 1, and subtasks 2 and 3 are offloaded to edge server 2. While in the right figure the actual edge caching situation is considered: services of types 2 and 4 are cached on edge server 1, services of types 1 and 3 are cached on edge server 2 (i.e., there is only a limited number of types of services that each edge server can cache.), and the optimization yields an offloading policy as follows: subtasks 2 and sub-subtasks 4 are offloaded to edge server 1, subtasks 1 and sub-subtasks 3 are offloaded to edge server 2. It is obvious that the dependencies between subtasks and the service caching policy simultaneously affect the development of offloading policies for dependent subtasks. Moreover, single-objective optimization is often performed in previous work. In real world scenarios, the optimization factors considered by users are often comprehensive and complex. Thus, in MEC scenarios, this paper focuses on building a many-objective joint optimization dependency-aware task offloading and service caching model and suggests an appropriate way to obtain subtask offloading and service caching policies that greatly increase the offloading efficiency.

3. Proposed MaJDTOSC

3.1 Task Model

Each user has a task (consisting of multiple dependent subtasks) to process, a directed acyclic graph $G_n = (V_n, E_n)$ can be used to represent each user task T_n, where $V_n = \{T_{n,start}, T_{n,1}, ..., T_{n,exit}\}$ stands for the set of subtasks in task T_n and $E_n = \{R_n^{start,i}, R_n^{start,j}, ..., R_n^{k,exit}\}$ stands for the set of subtask dependencies in task T_n, where $i, j, k \in [1, exit-1]$ *and* $i \neq j \neq k$. **Fig. 2** illustrates the dependency graph between subtasks in a task, where purple, green, red and blue lines represent

four different types of subtasks that require support from the corresponding services in order to be processed. $T_{n,start}$ and $T_{n,exit}$ represent the start subtask and exit subtask (indicated by black lines) of task T_n. Presume that Start Subtasks and Exit Subtasks can only be processed locally because they typically require local data collection and local display of processing results, respectively. $T_{n,i}$ stands for the ith intermediate subtask of task T_n. In **Fig. 2**, we call subtask $T_{n,2}$ is the predecessor subtask of subtask $T_{n,4}$, and subtask $T_{n,4}$ is the successor subtask of subtask $T_{n,2}$. $D_{T_{n,i}T_{n,j}}$ represents the data forwarded by Subtask $T_{n,i}$ to Subtask $T_{n,j}$. The same implies that the input data of Subtask $T_{n,j}$ is dependent on the output data of Subtask $T_{n,i}$. The dependency between them is denoted as $R_n^{i,j}$. Therefore, Subtask $T_{n,j}$ cannot start its execution without Subtask $T_{n,i}'s$ completion.

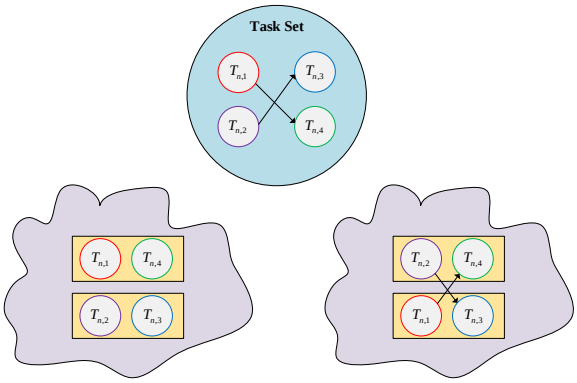

Fig. 1. Research motivation figure.

3.2 System Model

As indicated in **Fig. 3**, in this paper, the system model considers a three-layer MEC architecture: 1) Cloud layer, 2) Edge layer, and 3) Device layer. The cloud layer is located at the topmost layer in the three-tier architecture and consists of cloud servers, which have unlimited computational and storage resources in which all types of services required by the user are cached, and it is responsible for distributing the services to the various edge servers based on the service caching policy. The edge layer belongs to the core layer in the three-tier architecture, which mainly consists of multiple edge servers and base stations, each edge server has only limited computing and storage resources, the edge servers communicate with the users wirelessly through the base station, and the users can offload their tasks to the edge servers for processing. The edge servers communicate with each other through wired means and they can collaborate with each other to process tasks. The device layer, also known as the user layer, consists of multiple users and is located at the lowest level of the three-tier architecture. Different users have different tasks to be processed, and a task in turn consists of multiple dependent subtasks, and different types of subtasks require the support of corresponding services to be processed, and the dependencies between subtasks need to be considered. Users offload these dependency-aware subtasks to the target edge server according to the subtask offload policy.

3.3 Data Rate Model

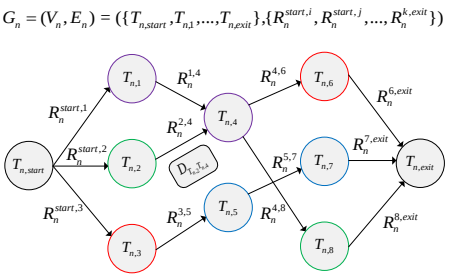

Fig. 2. Subtask dependency graph.

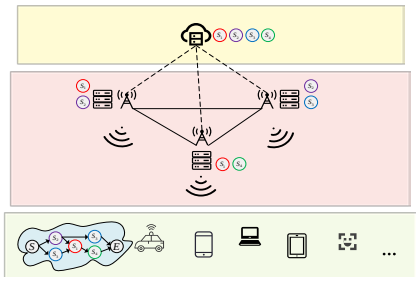

Fig. 3. System model.

Three types of data rates are included in MaJDTOSC: 1) The data transfer rate when a user uploads a subtask. 2) The data propagation rate of the subtask between edge servers. 3) The data forwarding rate of the predecessor subtask. The data transfer rate between user n and edge server k when offloading the ith subtask and the data propagation rate of the subtask between edge server k' and edge server k are defined as:

$$R_{n,i}^{k} = B_{n,i}^{k} \cdot \log_2(1 + \frac{p_{n,i}^{k} \cdot h_{n,i}^{k}}{\delta^2}) \tag{1}$$

$$R_{n,i}^{k'k} = B_{n,i}^{k'k} \cdot \log_2(1 + \frac{p_{n,i}^{k'k} \cdot h_{n,i}^{k'k}}{\delta^2 + \Lambda_{k'k}}) \tag{2}$$

where $B_{n,i}^{k}$ represents the channel bandwidth when user n uploads the ith subtask to edge server k. $p_{n,i}^{k}$ represents the transmission power when user n uploads the ith subtask to the edge server k, and $h_{n,i}^{k}$ represents the channel gain between user n and the edge server k when offloading the ith subtask. δ^2 represents the additive Gaussian white noise power. $B_{n,i}^{k'k}$ represents the channel bandwidth when the edge server k' propagates user $n's$ ith subtask to the edge server k. $p_{n,i}^{k'k}$ stands for the propagation power of the edge server k' to propagate the ith subtask of user n to the edge server k, and $h_{n,i}^{k'k}$ stands for the channel gain of the edge server k' when it propagates the ith subtask of user n to the edge server k. $\Lambda_{k'k}$ represents the noise power between edge server k' and edge server k.

The data forwarding rate of the predecessor subtask i' of the ith subtask of user n is defined as:

$$R_{n,i'i} = \begin{cases} B_{n,i'i} \cdot \log_2(1 + \dfrac{p_{n,i'i} \cdot h_{n,i'i}}{\delta^2}) & if\ off(n,i') = 0\ \|\ off(n,i) = 0 \\ B_{n,i'i}^{k'k} \cdot \log_2(1 + \dfrac{p_{n,i'i}^{k'k} \cdot h_{n,i'i}^{k'k}}{\delta^2 + \Lambda_{k'k}}) & if\ off(n,i') = k'\ \&\ off(n,i) = k \end{cases} \tag{3}$$

where $off(n,i)$ represents the offloading location of the ith subtask of user n. $off(n,i') = 0\ \|\ off(n,i) = 0$ represents the case where at least one of the ith subtask of user n and its predecessor subtask i' is processed locally, denoted as case one. $off(n,i') = k'\ \&\ off(n,i) = k$ stands for the case where the ith subtask of user n and its predecessor subtask i' are processed on different edge servers, denoted as case two. $B_{n,i'i}$ represents the channel bandwidth when the predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case one, $p_{n,i'i}$ represents the forwarding power when the

predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case one, and $h_{n,i'i}$ represents the channel gain when the predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case one. $B_{n,i'i}^{k'k}$ represents the channel bandwidth when the predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case two, $p_{n,i'i}^{k'k}$ represents the forwarding power when the predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case two, and $h_{n,i'i}^{k'k}$ represents the channel gain when the predecessor subtask i' of the ith subtask of user n forwards data to subtask i in case two.

3.4 Objective Model

3.4.1 Task Completion Time Objective

Task completion time is defined as the time taken from the start subtask execution to complete the last subtask. The ith subtask of user $n's$ actual completion time is as follows:

$$STFT_A(n,i,k) = STST_A(n,i,k) + ET_{n,i}^k, \quad \forall k \in \{K \bigcup \{0\}\} \tag{4}$$

where $ET_{n,i}^k$ represents the time consumed by the ith subtask of user n to offload to the edge server k (or local device) to perform the task. $STST_A(n,i,k)$ represents the time when the ith subtask of user n is offloaded to the edge server k (or local device) to actually start execution. Obviously a subtask cannot start execution without the output of its predecessor subtasks, and it must also be checked if the edge server (or local device) is currently idle. This leads to the definition of $STST_A(n,i,k)$:

$$STST_A(n,i,k) = \max\{avail\{n,i,k\}, STST_T(n,i,k)\} \tag{5}$$

where $STST_T(n,i,k)$ represents the time when the ith subtask of user n is offloaded to edge server k (or local device) to theoretically start execution, i.e., the moment when edge server k (or local device) has received and generated all of the input data required for the ith subtask. $avail\{n,i,k\}$ indicates the moment when the local device or edge server is prepared to carry out the task, i.e., the time when the edge server k and the local device are idle. In this paper, we assume all edge servers as well as user devices are equipped with multi-core processors that can handle multiple subtasks simultaneously, hence $avail\{n,i,k\} = 0$. $STST_T(n,i,k)$ is defined recursively:

$$STST_T(n,i,k) = \max_{i' \in p(n,i)} (STFT_A(n,i') + DFT_{n,i'i}) \tag{6}$$

where $p(n,i)$ represents the set of direct predecessor subtasks of the ith subtask of user n. $DFT_{n,i'i}$ represents the data forwarding time of the output of the predecessor subtask i' in user n to the edge server (or local device) where subtask i is offloaded. It is defined as:

$$DFT_{n,i'i} = \frac{d_{n,i'i}}{R_{n,i'i}} \tag{7}$$

where $d_{n,i'i}$ represents the amount of data forwarded to subtask i by subtask i', the predecessor of user $n's$ ith subtask, and $R_{n,i'i}$ represents the data forwarding rate. If $off(n,i') = off(n,i) = k$ represents that user $n's$ subtask i performs tasks on the same edge server as its predecessor subtask i', no data forwarding time is incurred, i.e., $DFT_{n,i'i} = 0$. Obviously, $STFT_A(n,exit,k)$ stands for the actual completion time of the exit subtask of user n, and $STST_A(n,start,k)$ stands for the actual start of the execution time of the start subtask of

user n. Therefore, the task processing time of user n is $STFT_A(n,exit,k)-STST_A(n,start,k)$.

This paper defines task completion time objective as the sum of all user task completion times:

$$TCT=\sum_{n=1}^{N}[STFT_A(n,exit,k)-STST_A(n,start,k)] \tag{8}$$

3.4.2 Energy Consumption Objective

There are four main components to the energy consumption generated by mobile users: computational energy consumption, transmission energy consumption, propagation energy consumption, and forwarding energy consumption. Subtasks only generate computational energy consumption when they are processed locally. If offloaded to an edge server, there are transmission energy consumptions when uploading subtasks. Propagation energy consumption is generated when subtasks are propagated between edge servers. Forwarding energy consumption is incurred during forwarding of the predecessor subtask to the subtask. The computational energy consumption of the ith subtask of user n is defined as:

$$EC_{n,i}^{c}=\kappa\cdot f_{n,i}^{2}\cdot d_{n,i} \tag{9}$$

where κ stands for power efficiency, depending on chip architecture. $f_{n,i}$ represents the CPU frequency for processing the ith subtask of user n. $d_{n,i}$ represents the amount of data to be processed for the ith subtask of user n.

The transmission energy consumed by user n when offloading the ith subtask, the propagation energy consumption of the subtask between edge servers, and the forwarding energy consumption of the predecessor subtask are defined as:

$$EC_{n,i}^{t}=P_{n,i}^{t}\cdot\frac{d_{n,i}}{R_{n,i}^{k}} \tag{10}$$

$$EC_{n,i}^{s}=P_{n,i}^{s}\cdot\frac{d_{n,i}}{R_{n,i}^{k'k}} \tag{11}$$

$$EC_{n,i'i}^{f}=P_{n,i'i}^{f}\cdot\frac{d_{n,i'i}}{R_{n,i'i}} \tag{12}$$

where $EC_{n,i}^{t}$ represents the transmission energy consumption of the ith subtask of user n, $EC_{n,i}^{s}$ stands for the propagation energy consumption of the ith subtask of user n, and $EC_{n,i'i}^{f}$ stands for the forwarding energy consumption of the predecessor subtasks of the ith subtask of user n. $P_{n,i}^{t}$, $P_{n,i}^{s}$, and $P_{n,i}^{f}$ represent transmission, propagation, and forwarding power, respectively. Use $\chi_{n,i}=\{x_{1,1},x_{1,2},...,x_{N,exit-1},x_{N,exit}\}$ to indicate the offloading decision for subtasks in each task, if $x_{n,i}=1$ means the subtask will be offloaded to the edge server for processing, otherwise $x_{n,i}=0$ means the subtask is processed locally. Use $\gamma_{n,i}=\{y_{1,1},y_{1,2},...,y_{N,exit-1},y_{N,exit}\}$ to indicate whether the corresponding service is cached on the edge server where the subtask is offloaded. If $y_{n,i}=1$ represents that the corresponding service is cached, $y_{n,i}=0$ represents that the corresponding service is not cached. Therefore, the energy consumption objective is defined as:

$$EC=\sum_{n=1}^{N}\sum_{i=1}^{|exit|}[x_{n,i}\cdot y_{n,i}\cdot(EC_{n,i}^{t}+EC_{n,i}^{s}+EC_{i'i}^{f}+EC_{n,i}^{c})+(1-x_{n,i})\cdot(EC_{i'i}^{f}+EC_{n,i}^{c})] \tag{13}$$

where N represents the number of users (and also the number of tasks) and $|exit|$ represents the number of subtasks contained in the task.

3.4.3 Subtask Hit Rate Objective

The subtask hit rate objective is the number of subtasks successfully offloaded (i.e., subtasks offloaded with corresponding services cached on the edge server) as a percentage of the number of subtasks offloaded. $H_{n,i}^{score}$ represents the hit score situation of the *ith* subtask of user n. If the subtask directly hits a service on the associated server score 1, i.e., $H_{n,i}^{score}=1$; if the subtask hits a service on the Collaboration Edge Server score 0.5, i.e., $H_{n,i}^{score}=0.5$; otherwise it means that the subtask does not hit the corresponding service and will be processed locally, score 0, i.e., $H_{n,i}^{score}=0$. Therefore, the subtask hit rate objective is defined as:

$$STHR=\frac{\sum_{n=1}^{N}\sum_{i=1}^{|exit|}(x_{n,i}\cdot y_{n,i}\cdot H_{n,i}^{score})}{\sum_{n=1}^{N}\sum_{i=1}^{|exit|}x_{n,i}}\times 100\% \tag{14}$$

where $x_{n,i}$ is used to determine whether the *ith* subtask of user n is offloaded or not, and $y_{n,i}$ represents the service cache on the target edge server where the *ith* subtask of user n is offloaded.

3.4.4 Load Variability Objective

Load variability can be measured by the standard deviation of the server load. A larger standard deviation value indicates a higher volatility of the load, i.e., the load is more variable. A smaller standard deviation value indicates a less volatile load, i.e., less load variability. Less load variability represents that all ESs in the MEC scenario fully utilize their computational resources and avoid wastage of computational resources. By calculating load standard deviation, the load variability can be quantified and used for the calculation of the load variability objective. The formula for calculating load variability is defined as:

$$L_{var}=\sqrt{\frac{\sum_{k=1}^{K}(Load_k-Load_{Avg})^2}{K}} \tag{15}$$

where K stands for the number of edge servers. $Load_k$ stands for the load of the edge server k. $Load_{Avg}$ stands for the average load of the edge server.

3.4.5 Storage Resource Utilization Objective

Setting storage resource utilization as an optimization goal can provide multiple benefits, including increased resource utilization efficiency, reduced energy consumption, improved user experience, and support for large-scale deployments. This enhances overall system performance and effectiveness by optimizing resource management and quality of service in MEC environments. Its formula is defined as:

$$SRU=\frac{\sum_{k=1}^{K}\sum_{l=1}^{L}(y_k^l\cdot SDV_l)}{\sum_{k=1}^{K}ESCC_k}\times 100\% \tag{16}$$

where y_k^l represents whether edge server k caches a service of type l or not. SDV_l represents the storage space occupied by the $l-type$ services. $ESCC_k$ represents edge server $k's$ storage capacity.

In summary, the constructed MaJDTOSC is:

$$MaJDTOSC = \begin{cases} min\ \ TCT \\ min\ \ EC \\ max\ STHR \\ max\ L_{var} \\ max\ SRU \end{cases} \quad \begin{array}{ll} s.t. & x_{n,i}^k \in \{0,1\}, \forall n \in N, i \in I, k \in K \\ & y_k^l \in \{0,1\}, \forall k \in K, l \in L \\ & O_k \le ESOC_k, \forall k \in K \\ & SC_k \le ESCC_k, \forall k \in K \\ & i_k^{start} = finish(p(i)) \,\&\, avail(k), \forall i \in I, k \in K \end{array} \tag{17}$$

From formula 17, the decision variables for MaJDTOSC are composed of $x = \{x_{1,1}^k, x_{1,2}^k, ..., x_{n,i}^k\}$ and $y = \{y_1^l, y_2^l, ..., y_k^l\}$, and x and y represent the task offloading and service caching policies, respectively. MaJDTOSC needs to minimize objectives TFT and EC and maximize objectives $STHR$, L_{var} and SRU. Its constraints are described on the right-hand side of formula 17. where the first line $x_{n,i}^k$ represents whether the ith subtask of user n is offloaded to edge server k, with 1 meaning it has been offloaded and 0 meaning it is processed locally. The second line represents whether the $l-type$ service is cached on the kth edge server, where 1 means it is cached and 0 means it is not cached. O_k represents the amount of data offloaded to edge server k, and $ESOC_k$ represents the offload capacity of edge server k, therefore, the third line constraint states that no edge server's offload capacity can be exceeded by the volume of data offloaded. SC_k represents the amount of storage resources consumed by the caching service of edge server k, and $ESCC_k$ stands for the cache capacity of edge server k, therefore, the constraint in the fourth line implies that the quantity of data stored in each edge server cannot exceed their cache capacity. In the fifth constraint line, the set of subtask $i's$ previous subtasks is denoted by $p(i)$. $finish()$ is used to determine whether the predecessor subtasks have all completed their outputs, and $avail()$ is used to determine whether the edge server (or local device) is ready. Subtask i can be executed only if both conditions are satisfied.

4. Proposed TSMSNSGAIII

4.1 TSMSNSGAIII Framework

Researchers have thoroughly studied multi-objective evolutionary algorithms and designed various optimization ideas [23-29]. In order to better solve MaJDTOSC in light of the challenges that it faces, a many-objective evolutionary algorithm called TSMSNSGAIII based on the Three-Stages Mating Selection (TSMS) strategy is presented in this study. TSMSNSGAIII changes the mating selection strategy of NSGAIII and retains its environmental selection strategy. In TSMSNSGAIII, the mating selection process is split into three stages to balance convergence and diversity in the evolutionary process, focusing on convergence, convergence & diversity, and diversity, respectively. The three-phase division rule is defined as:

$$MS_{stage}=\begin{cases} convergence & if\ \ t \le r \cdot t_{max} \\ convergence\ \ \&\ \ diversity & if\ \ r \cdot t_{max} < t \le (1-r) \cdot t_{max} \\ diversity & if\ \ (1-r) \cdot t_{max} < t \le t_{max} \end{cases} \tag{18}$$

where r is a parameter controlling the proportion of each stage, and r should be less than 0.5; in this paper, r is set to 0.2.

Algorithm 1 shows the framework of TSMSNSGAIII. As shown in Algorithm 1, TSMSNSGAIII requires inputs of population size, number of optimization objectives, maximum number of iterations, stage division ratio, cross-variance probability, and cross-variance distribution index, and finally outputs the population obtained after optimization is completed. Each individual in the population represents one of the given dependency subtask offloading, service caching policies. TSMSNSGAIII is initialized by randomly generating an initial population of size and a set of uniformly distributed reference points (lines 1~2), and then obtaining the ideal points of the initialized population (line 3). In the evolutionary process, a three-stage mating selection strategy was first used to select the best individuals to join the mating pool to generate the offspring population after cross mutation (lines 5~12). After generating the offspring population, combine it with the current population and update the ideal point information (line 13), and then use the environment selection strategy of NSGAIII to select the combined population to generate the next generation population (line 14). The optimized population will be returned as soon as the maximum number of evaluations of the function has been reached.

Algorithm 1 Framework of TSMSNSGAIII

Input: Population size: N , Number of objective: M , Maximum number of function evaluations: t_{max} , Stage division ratio: r , Crossover probability: p_c , Mutation probability: p_m , Crossover distribution index λ_c , Mutation distribution index λ_m .

Output: Final population: $P^{t_{max}}$.

1: Initialize the Population: $P_0 \leftarrow \{P_1, P_2, ..., P_N\}$;
2: $Z \leftarrow Uniformpoint(N, M)$;
3: $Z^* \leftarrow min\_obj(P_0)$;
4: **while** $t < t_{max}$ **do**
5: **if** $t \le r \cdot t_{max}$ **then**
6: $P_t' \leftarrow Phase1\_MatingSelection(N, P_t, Z^*)$; ***(Reference algorithm 2)***
7: **elseif** $r \cdot t_{max} < t \le (1-r) \cdot t_{max}$ **then**
8: $P_t' \leftarrow Phase2\_MatingSelection(N, P_t, Z^*, t, t_{max})$; ***(Reference algorithm 3)***
9: **else**
10: $P_t' \leftarrow Phase3\_MatingSelection(N, P_t)$; ***(Reference algorithm 4)***
11: **endif**
12: $Q_t \leftarrow crossover\ \&\ mutation(P_t', p_c, p_m, \lambda_c, \lambda_m)$;
13: $Z^* \leftarrow Update\_min\_obj(P_t \cup Q_t)$;
14: $P_{t+1} \leftarrow NSGAIII\_EnvironmentSelection(P_t \cup Q_t, N, Z, Z^*)$;
15: **endwhile**

4.2 Three-Stage Mating Selection Strategy

4.2.1 First Phase

The algorithm needs to obtain better subtask offloading and service caching strategies as soon as possible when it first starts solving MaJDTOSC, and the algorithm needs better convergence

performance. Therefore, algorithms in the mating selection of a stage ($t \le r \cdot t_{max}$) is mainly concerned with the performance of individual convergence performance, how to carry out the judgment of convergence performance is a problem worth thinking about. This paper evaluates the individual convergence performance using the Achievement Scalar Function (ASF) and the Pareto dominance (PD) relationship. Pareto dominance relationship means that all the objective values of one individual p_1 are less than or equal to all the objective values of another individual p_2 and there is a less-than relationship on at least one objective value, if this condition is satisfied, we say that the individual p_1 dominates the individual p_2, denoted as $p_1 \prec p_2$. The ASF belongs to the convergence discriminant, which guarantees the convergence performance of an individual. Calculating an individual $p_1's$ ASF value is defined as follows:

$$ASF(p_1) = \max_{i=1:M} \left\{ (f_i^{p_1} - z_i^*) / \lambda_i^{p_1} \right\} \tag{19}$$

$$\lambda_i^{p_1} = \frac{f_i^{p_1}}{\sum_{j=1:M} f_j^{p_1}} \tag{20}$$

where z_i^* represents the best value of objective i found among all individuals so far, $f_i^{p_1}$ represents the *ith* objective value of individual p_1, and $\lambda_i^{p_1}$ represents the weight vector of an individual p_1 on the *ith* objective, which is set to 10^{-6} if $\lambda_i^{p_1}$ equals to zero. The process of calculating the ASF value of an individual p_1 is as follows: first, the objective value is transformed based on the ideal point information, then divided by the weight vector for normalization, and finally the maximum value of it is taken. The ASF value reflects each individual's level of convergence. The smaller the ASF value of individual p_1, the better the convergence performance. ASF values were used to the first and second stages of the mating selection process.

Algorithm 2 shows the flow of mating selection one phase. As shown in Algorithm 2, the mating Selection phase 1 requires inputs of population size, current population, and ideal points, and finally outputs the parent population consisting of the individuals selected into the mating pool. First, an empty set is created for storing the parent population (line 1), and then the ASF values of all individuals in the current population are calculated according to formula 19 (lines 2~4). Randomly select two individuals p_1, p_2 from the current population , and check the dominance relationship between them. If individual p_1 dominates individual p_2, then individual p_1 is added to the mating pool, and if individual p_1 is dominated by individual p_2, then individual p_2 is added to the mating pool. If the individuals do not dominate each other, compare their ASF values and select the individual with the smaller ASF value to join the mating pool. If two individuals have the same ASF value, one of these two individuals is randomly selected to join the mating pool (lines 6~19). Until the number of individuals in the mating pool exceeds the size of the population, this mating selection process is stopped.

Algorithm 2 Phase 1 of MatingSelection

Input: Population size: N, Current population: P_t, Ideal point: Z^*.
Output: Mating pool: P_t'.

1: $P_t' \leftarrow \varnothing$;
2: **while** $i \le N$ **do**
3: $ASF(i) \leftarrow$ Calculate the ASF value for each individual according to formula 19;
4: **endwhile**

while $\left|P_t'\right| \le N$ **do**
  Two individuals p_1 and p_2 are randomly selected from P_t;
  if $p_1 \prec p_2$ **then**
    $P_t' \leftarrow P_t' \cup \{p_1\}$;
  elseif $p_1 \succ p_2$ **then**
    $P_t' \leftarrow P_t' \cup \{p_2\}$;
  else
    if $ASF(p_1) < ASF(p_2)$ **then**
      $P_t' \leftarrow P_t' \cup \{p_1\}$;
    elseif $ASF(p_1) > ASF(p_2)$ **then**
      $P_t' \leftarrow P_t' \cup \{p_2\}$;
    else
      $P_t' \leftarrow P_t' \cup \{random(p_1, p_2)\}$;
    endif
  endif
endwhile
return P_t';

4.2.2 Second Phase

The algorithms need to formulate sub-task offloading and service caching strategies that are both appropriate and diverse in solving the MaJDTOSC mid-term, requiring both good convergence and diversity of the algorithms. Therefore, the algorithm focuses on the overall performance of individual convergence and diversity in the second stage of mating selection ($r \cdot t_{max} < t \le (1-r) \cdot t_{max}$). The ASF, minimum clip angle, and Adaptive Comprehensive Performance Score (ACPS) are used to judge individual convergence and diversity in this stage. The calculation of the ASF has been given in the previous subsection. The formula for calculating the minimum angle between an individual $p1$ and other individuals ($p_2 \in P, p_2 \ne p_1$) in the population is defined as:

$$\theta_{p_1}^{min} = \min_{p_2 \in P, p_2 \ne p_1} \theta_{p_1}^{p_2} \tag{21}$$

$$\theta_{p_1}^{p_2} = arccos \frac{\sum_{i=1}^{M} [(f_i^{p_1} - z_i^*) * (f_i^{p_2} - z_i^*)]}{\sqrt{\sum_{i=1}^{M} (f_i^{p_1} - z_i^*)^2} * \sqrt{\sum_{i=1}^{M} (f_i^{p_2} - z_i^*)^2}} \tag{22}$$

where P represents the current population, p_2 represents other individuals in population P, and $\theta_{p_1}^{p_2}$ represents the angle between individual p_1 and individual p_2. M represents the number of objectives, $f_i^{p_1}$ stands for the objective value of individual p_1 on objective i, and z_i^* represents the ideal objective value on the ith objective. If the $\theta_{p_1}^{min} - value$ of individual p_1 is large, it represents a better diversity of individual p_1. In this paper, an adaptive composite performance scoring index is proposed for further judging individual performance, which is defined as:

$$ACPS(p_1) = \frac{(1 - \frac{t}{t_{max}}) \cdot ASF(p_1)}{\frac{t}{t_{max}} \cdot \theta_{p_1}^{min}} \tag{23}$$

where t is the current number of objective function evaluations and t_{max} is the maximum number of objective function evaluations. As the number of evaluations rises, ACPS progressively modifies the weights given to convergence and diversity. A larger ACPS represents a better overall performance of an individual under the current evolutionary stage.

The flow of the mating selection phase 2 is given in Algorithm 3. As shown in Algorithm 3, the mating Selection phase 2 requires inputs of population size, current population, ideal point, current number of function evaluations, and maximum number of function evaluations, and finally outputs the parent population consisting of the individuals selected into the mating pool. First, an empty set is created to store the parent population (line 1), and then calculate the ASF value, the minimum pinch angle, and the ACPS value for all individuals in the current population according to formula 19, formula 21, and formula 23 (lines 2~6). Randomly select two individuals p_1, p_2 from the current population p_t, if the ASF value of an individual p_1 is smaller than the ASF value of an individual p_2 and the minimum clip angle of an individual p_1 is larger than the minimum clip angle of an individual p_2, then add the individual p_1 to the mating pool. If the ASF value of individual p_1 is larger than the ASF value of individual p_2 and the minimum clip angle of individual p_1 is smaller than the minimum clip angle of individual p_2, then individual p_2 is added to the mating pool. If the above conditions are not met, the ACPS values of the two individuals are compared, and the individual whose ACPS value is higher is let into the mating pool; otherwise, one of the two individuals is randomly selected to be added to the mating pool (lines 8~20). Until the number of individuals in the mating pool exceeds the size of the population, this mating selection process is stopped.

Algorithm 3 Phase 2 of MatingSelection

Input: Population size: N, Current population: P_t, Ideal point: Z^*, Current function evaluation times: t, Maximum number of function evaluations: t_{max}.
Output: Mating pool: P_t'.

1: $P_t' \leftarrow \varnothing$;
2: **while** $i \le N$ **do**
3: $ASF(i) \leftarrow$ Calculate the ASF value for each individual according to formula 19;
4: $\theta^{min}(i) \leftarrow$ Calculate the θ^{min} value for each individual according to formula 21;
5: $ACPS(i) \leftarrow$ Calculate the ACPS value for each individual according to formula 23;
6: **endwhile**
7: **while** $|P_t'| \le N$ **do**
8: Two individuals p_1 and p_2 are randomly selected from P_t;
9: **if** $ASF(p_1) < ASF(p_2) \,\&\, \theta^{min}(p_1) > \theta^{min}(p_2)$ **then**
10: $P_t' \leftarrow P_t' \cup \{p_1\}$;
11: **elseif** $ASF(p_1) > ASF(p_2) \,\&\, \theta^{min}(p_1) < \theta^{min}(p_2)$ **then**
12: $P_t' \leftarrow P_t' \cup \{p_2\}$;
13: **else**
14: **if** $ACPS(p_1) > ACPS(p_2)$ **then**
15: $P_t' \leftarrow P_t' \cup \{p_1\}$;
16: **elseif** $ACPS(p_1) < ACPS(p_2)$ **then**
17: $P_t' \leftarrow P_t' \cup \{p_2\}$;
18: **else**
19: $P_t' \leftarrow P_t' \cup \{random(p_1, p_2)\}$;
20: **endif**

21: **endif**
22: **endwhile**
23: **return** P_t';

4.2.3 Third Phase

Algorithms in solving MaJDTOSC late, the formulated sub-task offloading and service caching strategies already have good convergence, need to improve the diversity of the formulated strategies, that requires the algorithms to have good diversity. Therefore, the algorithm focuses on the performance of individual diversity in the phase 3 of mating selection ($(1-r)\cdot t_{max} < t \le t_{max}$). The minimum clip angle is used in this stage to judge the individual diversity. The formula for calculating the minimum clip angle has been given in the previous subsection. The flow of the phase 3 of mating selection is given in Algorithm 4. As shown in Algorithm 4, the phase 3 of mating selection require the input of the population size, the current population, and finally the output of the parent population consisting of the individuals selected into the mating pool. First, an empty set is created for storing the parent population (line 1), and then the minimum clip angle of all individuals in the current population is calculated according to formula 21 (lines 2~4). Two individuals are randomly selected from the current population, and the individual with the larger minimum clip angle is selected to join the mating pool. Until the number of individuals in the mating pool exceeds the size of the population, this mating selection process is stopped.

Algorithm 4 Phase 3 of MatingSelection

Input: Population size: N, Current population: P_t.
Output: Mating pool: P_t'.

1: $P_t' \leftarrow \varnothing$;
2: **while** $i \le N$ **do**
3: $\theta^{min}(i) \leftarrow$ Calculate the θ^{min} value for each individual according to formula 21;
4: **endwhile**
5: **while** $|P_t'| \le N$ **do**
6: Two individuals p_1 and p_2 are randomly selected from P_t;
7: **if** $\theta^{min}(p_1) > \theta^{min}(p_2)$ **then**
8: $P_t' \leftarrow P_t' \cup \{p_1\}$;
9: **elseif** $\theta^{min}(p_1) < \theta^{min}(p_2)$ **then**
10: $P_t' \leftarrow P_t' \cup \{p_2\}$;
11: **else**
12: $P_t' \leftarrow P_t' \cup \{random(p_1, p_2)\}$;
13: **endif**
14: **endwhile**
15: **return** P_t';

4.2.4 TSMSNSGAIII Time Complexity Analysis

In the TSMSNSGAIII calculation process, the time complexity of the algorithm mainly comes from:1) population initialization; 2) three-stage mating selection process; 3) environment selection process. The algorithm starts to execute by first randomly generating N individuals, so the time complexity of population initialization is $O(N)$. The first stage of mating selection requires picking N individuals based on the Pareto dominance relation and ASF values, with

time complexity $O(MN)$; the second stage picks based on the ASF values and the minimum clip angle, with time complexity $O(MN^2)$; and the third stage picks based on the minimum clip angle only, with time complexity $O(MN^2)$. Therefore, the time complexity of the whole mating selection process is $O(MN^2)$. Since the environment selection is based on the NSGAIII algorithm, the time complexity of the environment selection process is $O(MNH)$, where H represents the number of reference points. In summary, the time complexity of TSMSNSGAIII is $O(MN^2)$.

5. Experiment

5.1 TSMSNSGAIII Performance Test

In this paper, five excellent many-objective evolutionary algorithms were used with TSMSNSGAIII on the DTLZ test set using Inverted Generational Distance (IGD) metrics, they are NSGAIII [30], hpaEA [31], MaOEADDFC [32], SPEAR [33], KnEA [34]. **Table 1** demonstrates the parameter settings in TSMSNSGAIII, where the first four parameters follow the default settings of the NSGAIII algorithm, and the fifth parameter is used to control the ratio of the three phases, which is recommended to be set to: $0 < r \le 0.2$ if the algorithm is required to have a strong comprehensive performance, and $0.2 < r < 0.5$ if convergence or diversity is sought for a single performance. **Table 2** demonstrates the test results. It uses "+", "-", and "=" to represent that the comparison algorithms outperform, underperform, and equal the TSMSNSGAIII performance, respectively, and also makes use of the blue color to highlight the best results on the various test problems.

Table 1. TSMSNSGAIII Parameter Settings

Parameters	Description	Value
p_c	Probability of simulated binary crossover	1
λ_c	The distribution index of simulated binary crossover	20
p_m	Probability of polynomial mutation	$1/N$
λ_m	The distribution index of polynomial mutation	20
r	Stage division ratio	0.2

As can be seen from **Table 2**, TSMSNSGAIII shows a definite performance advantage over other algorithms for the benchmarking problems DTLZ1, DTLZ3, DTLZ5, DTLZ6 with the number of targets set to 5, 7, 9, 11, and 13.

Slightly worse than KnEA for most target number settings of DTLZ2, DTLZ4, but better than KnEA when the target number of DTLZ2 is set to 5. The number of advantages and disadvantages is more balanced when comparing with other algorithms.

On the DTLZ7 there is still a definite performance advantage at all other target number settings, except for a slight inferiority to KnEA at target number setting of 9.

Overall, TSMSNSGAIII achieves excellent performance on the DTLZ test set.

Table 2. IGD performance test of TSMSNSGAIII algorithm on DTLZ

Problem	M	D	NSGAIII	hpaEA	MaOEADDFC	SPEAR	KnEA	TSMSNSGAIII
DTLZ1	5	9	5.0035e-1 (3.06e-1) =	7.1033e-1 (5.08e-1) -	8.1223e-1 (3.87e-1) -	1.2821e+0 (5.33e-1) -	8.3796e-1 (5.81e-1) -	3.8848e-1 (2.54e-1)
	7	11	1.0629e+0 (7.22e-1) -	2.1369e+0 (9.78e-1) -	3.1891e+0 (1.93e+0) -	2.8427e+0 (1.29e+0) -	4.8318e+0 (2.93e+0) -	4.9308e-1 (2.80e-1)
	9	13	1.3658e+0 (6.14e-1) -	2.4477e+0 (9.83e-1) -	8.5617e+0 (6.22e+0) -	2.4563e+0 (1.51e+0) -	8.0522e+0 (8.45e+0) -	5.3335e-1 (2.84e-1)
	11	15	1.5202e+0 (1.03e+0) -	3.2323e+0 (1.57e+0) -	1.7165e+1 (1.15e+1) -	4.1597e+0 (2.45e+0) -	9.0753e+0 (1.27e+1) -	4.8441e-1 (2.20e-1)
	13	17	8.6966e-1 (4.70e-1) =	3.3741e+0 (1.24e+0) -	1.7291e+1 (7.75e+0) -	6.7981e+0 (2.54e+0) -	1.2900e+1 (1.10e+1) -	6.8286e-1 (3.12e-1)
DTLZ2	5	14	2.1622e-1 (1.14e-3) =	2.1554e-1 (2.65e-3) =	2.2398e-1 (2.31e-3) -	2.3191e-1 (5.02e-3) -	2.3329e-1 (5.23e-3) -	2.1580e-1 (7.66e-4)
	7	16	3.6128e-1 (3.34e-2) +	3.7946e-1 (1.44e-2) +	3.8967e-1 (1.83e-2) -	3.8631e-1 (7.27e-3) -	3.6981e-1 (1.37e-2) +	3.8364e-1 (7.01e-2)
	9	18	4.8942e-1 (7.96e-2) +	5.0174e-1 (2.67e-2) =	6.1715e-1 (4.94e-2) =	4.4188e-1 (2.17e-2) +	4.9396e-1 (1.56e-2) =	5.5912e-1 (9.35e-2)
	11	20	5.8860e-1 (3.50e-2) +	6.4769e-1 (3.99e-2) =	7.5893e-1 (6.98e-2) -	6.1438e-1 (3.04e-2) +	5.3831e-1 (1.50e-2) +	6.3303e-1 (2.99e-2)
	13	22	6.4542e-1 (3.45e-2) =	7.6038e-1 (3.76e-2) -	8.5505e-1 (7.83e-2) -	7.2117e-1 (4.54e-2) -	5.9542e-1 (3.80e-2) +	6.5899e-1 (2.42e-2)
DTLZ3	5	14	2.0587e+1 (7.71e+0) -	2.7698e+1 (1.23e+1) -	3.4353e+1 (1.21e+1) -	3.5204e+1 (1.07e+1) -	2.2615e+1 (6.79e+0) -	1.0808e+1 (4.61e+0)
	7	16	4.0786e+1 (1.23e+1) -	9.1856e+1 (2.30e+1) -	1.3671e+2 (3.93e+1) -	1.0163e+2 (2.59e+1) -	1.0565e+2 (3.10e+1) -	1.8853e+1 (1.07e+1)
	9	18	4.9367e+1 (2.10e+1) -	9.8379e+1 (2.19e+1) -	3.0093e+2 (6.95e+1) -	7.4604e+1 (2.36e+1) -	2.2582e+2 (8.38e+1) -	2.4212e+1 (1.00e+1)
	11	20	4.8544e+1 (2.32e+1) -	1.0337e+2 (2.41e+1) -	4.2610e+2 (7.57e+1) -	1.6707e+2 (4.47e+1) -	3.0078e+2 (1.48e+2) -	2.4741e+1 (1.09e+1)
	13	22	4.6073e+1 (2.10e+1) -	1.1606e+2 (2.91e+1) -	4.4387e+2 (9.68e+1) -	2.1248e+2 (4.59e+1) -	3.4328e+2 (1.72e+2) -	2.7671e+1 (1.01e+1)
DTLZ4	5	14	3.1130e-1 (1.17e-1) +	4.8496e-1 (1.76e-1) +	3.2078e-1 (1.19e-1) +	2.4681e-1 (1.09e-2) +	3.2818e-1 (1.14e-1) +	6.9640e-1 (2.67e-1)
	7	16	4.3175e-1 (9.06e-2) +	5.4596e-1 (1.45e-1) =	5.2451e-1 (5.68e-2) =	4.1398e-1 (2.10e-2) +	3.9058e-1 (4.76e-2) +	5.1378e-1 (8.10e-2)
	9	18	5.4026e-1 (8.12e-2) +	6.1650e-1 (7.90e-2) =	7.3844e-1 (5.81e-2) -	5.6967e-1 (2.34e-2) +	5.0050e-1 (9.67e-3) +	5.7693e-1 (7.65e-2)
	11	20	5.9578e-1 (5.59e-2) =	7.1343e-1 (5.64e-2) -	8.6763e-1 (1.02e-1) -	6.6392e-1 (3.06e-2) -	5.3710e-1 (5.90e-3) +	6.2001e-1 (4.09e-2)
	13	22	6.5087e-1 (3.01e-2) =	8.0061e-1 (5.48e-2) -	9.3719e-1 (9.61e-2) -	7.4588e-1 (3.51e-2) -	6.0056e-1 (6.13e-3) +	6.6089e-1 (2.22e-2)
DTLZ5	5	14	1.6175e-1 (4.25e-2) -	1.6439e-1 (2.74e-2) -	3.6231e-1 (1.96e-1) -	2.4221e-1 (9.91e-2) -	2.0484e-1 (4.75e-2) -	7.4630e-2 (2.30e-2)
	7	16	2.0970e-1 (3.24e-2) -	3.0911e-1 (5.22e-2) -	1.7155e+0 (2.96e-1) -	5.6717e-1 (1.32e-1) -	2.7604e-1 (6.31e-2) -	1.3038e-1 (2.05e-2)
	9	18	2.7134e-1 (6.10e-2) -	4.0732e-1 (6.82e-2) -	2.1849e+0 (9.07e-2) -	4.8644e-1 (8.45e-2) -	3.6999e-1 (1.03e-1) -	1.1816e-1 (2.07e-2)
	11	20	2.9177e-1 (4.96e-2) -	5.3135e-1 (9.33e-2) -	2.2906e+0 (1.19e-1) -	6.7444e-1 (2.77e-1) -	3.9327e-1 (8.70e-2) -	1.9758e-1 (4.73e-2)
	13	22	3.2331e-1 (4.45e-2) -	5.8488e-1 (6.83e-2) -	2.3064e+0 (1.20e-1) -	1.1950e+0 (1.60e-1) -	5.1186e-1 (1.52e-1) -	2.0130e-1 (2.86e-2)
DTLZ6	5	14	1.5844e+0 (6.64e-1) -	2.7003e+0 (8.27e-1) -	4.0085e+0 (1.08e+0) -	2.7553e+0 (7.48e-1) -	1.2107e+0 (7.44e-1) -	7.2267e-1 (6.38e-1)
	7	16	3.6969e+0 (9.85e-1) -	5.6699e+0 (8.76e-1) -	8.7117e+0 (4.17e-1) -	7.2730e+0 (6.68e-1) -	2.4519e+0 (5.14e-1) -	2.0669e+0 (9.87e-1)
	9	18	4.7020e+0 (8.09e-1) -	5.1617e+0 (6.35e-1) -	9.1436e+0 (2.30e-1) -	5.1582e+0 (1.10e+0) -	2.9678e+0 (4.99e-1) -	1.9307e+0 (9.65e-1)
	11	20	4.9322e+0 (9.53e-1) -	6.0111e+0 (5.45e-1) -	9.0743e+0 (3.03e-1) -	8.3884e+0 (5.39e-1) -	3.1118e+0 (4.54e-1) -	2.4958e+0 (1.10e+0)
	13	22	4.7919e+0 (9.95e-1) -	6.5463e+0 (5.88e-1) -	9.1985e+0 (2.77e-1) -	9.0648e+0 (3.87e-1) -	3.3198e+0 (5.62e-1) -	2.6468e+0 (1.15e+0)
DTLZ7	5	24	4.8570e-1 (7.08e-2) -	6.8643e-1 (2.70e-1) -	6.6143e-1 (2.59e-1) -	5.6439e-1 (9.35e-2) -	4.5482e-1 (1.72e-1) =	3.7654e-1 (2.76e-2)
	7	26	1.2673e+0 (2.72e-1) -	1.3526e+0 (3.15e-1) -	1.1221e+0 (1.20e-1) -	2.1009e+0 (3.22e-1) -	9.1244e-1 (2.73e-1) -	7.3020e-1 (4.25e-2)
	9	28	5.6108e+0 (1.62e+0) -	6.6116e+0 (1.77e+0) -	3.5192e+0 (7.76e-1) -	1.0478e+1 (4.01e+0) -	1.3702e+0 (3.11e-1) =	1.6223e+0 (6.40e-1)
	11	30	6.0230e+0 (9.21e-1) -	7.5748e+0 (1.77e+0) -	6.8515e+0 (1.35e+0) -	7.6702e+0 (1.18e+0) -	5.4859e+0 (2.12e+0) -	2.7465e+0 (5.53e-1)
	13	32	8.0454e+0 (1.24e+0) -	1.2266e+1 (1.45e+0) -	1.0404e+1 (2.07e+0) -	1.0567e+1 (1.04e+0) -	1.1249e+1 (2.22e+0) -	4.9619e+0 (7.37e-1)
+/-/=			6/23/6	2/28/5	1/32/2	5/30/0	8/24/3	

5.2 TSMSNSGAIII Solving for MaJDTOSC

5.2.1 Simulation Experiment Parameter Setting

Table 3 shows the parameters of the multi-user MEC scenario.

Table 3. Multi-user MEC scenario parameter settings

Parameters	Description	Value
	Communication range	2.2×2.2 km^2
	Subtask type	1 ~ 5
	Number of subtasks contained in the task	8 ~ 12
ESCC	Edge server cache capacity	12 ~ 14 GB
ESOC	Edge server offload capacity	130 ~ 190 GB
σ^2	Noise power	15 ~ 25 -dBm
K	Energy conversion efficiency	10^{-6}
N	Number of users	40 ~ 60
M	Number of edge servers	10
B	Channel bandwidth	80 ~ 200 Mbps
h	Channel gain	1.08 ~ 1.50
p	Signal transmission power	14 ~ 49 W
f	Computing capability	100 ~ 1000 cycle/s
ds	Task size	1 ~ 5 GB

5.2.2 Analysis of Simulation Experiment Results

In **Fig. 5 to Fig. 9**, the performance of TSMSNSGAIII solving MaJDTOSC is evaluated under different number of users settings. The results show that as the number of users increases, the task completion time, energy consumption, and load variability objectives become larger in the MEC environment. Whereas, the subtask hit rate and storage resource utilization objectives show stable performance under different number of users settings. Compared to the other five many-objective evolutionary algorithms, TSMSNSGAIII demonstrates a performance advantage in solving MaJDTOSC: in the task completion time objective, TSMSNSGAIII always provides the optimal objective value of the task completion time, irrespective of the number of users anticipated in the MEC setting; in the energy consumption objective, with the

number of users set to be 40, the For the energy consumption objective, TSMSNSGAIII achieves the optimal performance when the number of users is set to 40. When the number of users is set to 50, it is slightly worse than KnEA, and when the number of users is set to 60, it is slightly worse than hpaEA and KnEA. The reason for the above phenomenon is that because the energy consumption of subtasks when offloading is inevitably higher than the energy consumption of the local computation, the lower energy consumption objective value represents the fact that most of the subtasks fail in hitting the objective edge servers and choose to perform the computation locally instead of offloading it to the appropriate edge servers. Alternatively, the algorithm does not make a good trade-off between multiple objectives, which is achieved by the energy objective value obtained when solving MaJDTOSC in TSMSNSGAIII. Therefore, it is also concluded that TSMSNSGAIII performs excellently on the energy consumption objective value; on the subtask hit rate objective, TSMSNSGAIII also performs the best under all the user number setting conditions, indicating that the subtasks in the MEC environment are offloaded to the appropriate edge servers; on the load variability objective, TSMSNSGAIII always enables all the edge servers in the mobile In the load variability objective, TSMSNSGAIII always makes full use of the computing resources of all edge servers in the MEC environment; in the storage resource utilization objective, TSMSNSGAIII always allocates all types of services to the appropriate edge servers to support the computation of the corresponding subtasks.

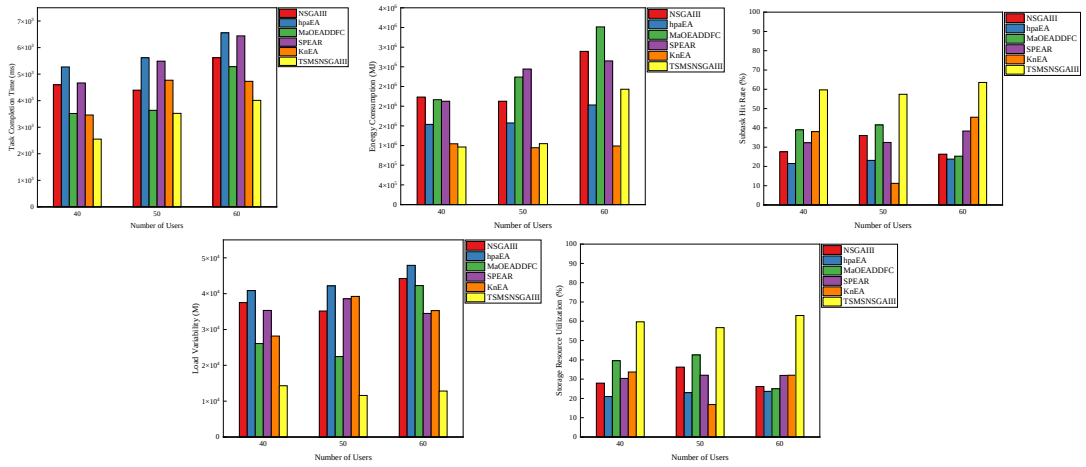

Fig. 5. ~ Fig. 9. Performance of six many-objective evolutionary algorithms on five objectives with different numbers of users.

In **Fig. 10 to Fig. 14**, the convergence curves of six many-objective evolutionary algorithms for solving MaJDTOSC with the number of users set to 50 are tested. The number of iterations is set to 1000 in this paper. In terms of the energy consumption objective, TSMSNSGAIII can reach the excellent objective value and remain stable with only 200 iterations compared to the other many-objective evolutionary algorithms. For task completion time, subtask hit rate, and cache resource utilization objectives, TSMSNSGAIII needs only 300 iterations to obtain the optimal objective values and maintain stability. For the load variability objective, TSMSNSGAIII obtains the optimal objective value with less fluctuation compared to other algorithms, showing good convergence and stability.

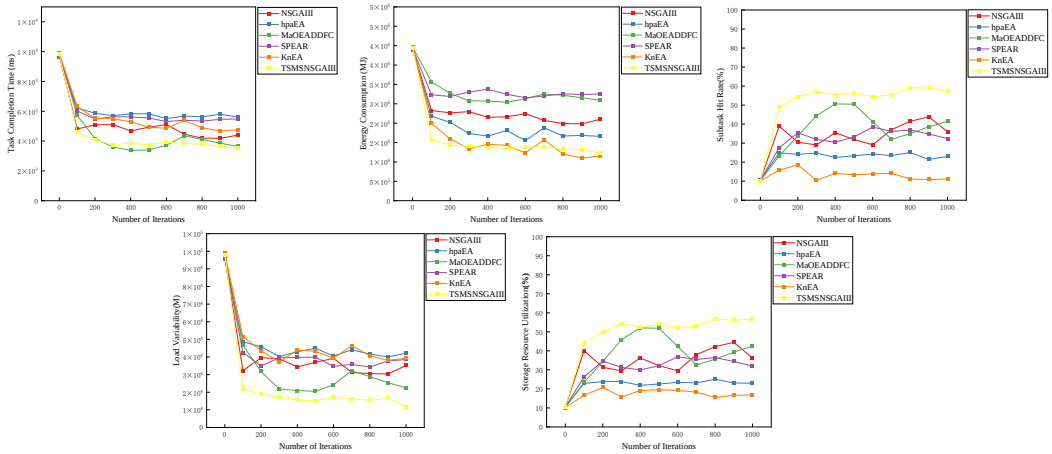

Fig. 10. ~ Fig. 14. Iterations of six many-objective evolutionary algorithms on five objectives.

6. Conclusion

In this paper, we study the problem of joint optimization of dependency-aware subtask offloading and service caching in a multi-user, resource-constrained MEC scenario. After comprehensively considering the impact of dependency-aware subtasks, resource constraints of edge servers, and multi-users on the formulation of task offloading and service caching policies, we construct a many-objective joint optimization model for dependency-aware task offloading and service caching (MaJDTOSC), in which five optimization objectives are considered: task completion time, energy consumption, subtask hit rate, load variability, and storage resource utilization. In order to solve MaJDTOSC better, a many-objective evolutionary algorithm TSMSNSGAIII based on the Three-Stages Mating Selection (TSMS) strategy is proposed. Simulation results show that TSMSNSGAIII exhibits excellent performance compared to the other five many-objective evolutionary algorithms in solving models set with different numbers of users. Therefore, it is concluded that TSMSNSGAIII can provide suitable sub-task offloading and service caching strategies in multi-user, resource-constrained MEC scenarios.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No.61806138, No.61772478, No.U1636220, Science and Technology Development Foundation of the Central Guiding Local under Grant No. YDZJSX2021A038, Australian Research Council (ARC) projects DP190101893, DP170100136, and LP180100758. China University Industry-University-Research Collaborative Innovation Fund (Future Network Innovation Research and Application Project), No.2021FNA04014. Taiyuan University of Science and Technology Scientific Research Initial Funding(TYUST SRIF), No.20232087, and the Shanxi University Science and Technology Innovation Funding, No.2023L177.

References

[1] Li, S. C., Xu, L. D. and Zhao, S. S., "5G Internet of Things: A survey," *Journal of Industrial Information Integration*, 10, 1-9, Jun 2018. Article (CrossRef Link)

[2] Kong, L. H., Tan, J. L., Huang, J. Q., Chen, G. H., Wang, S. T., Jin, X., Zeng, P., Khan, M. and Das, S. K., "Edge-computing-driven Internet of Things: A Survey," *Acm Computing Surveys*, 55(8), 1-41, Aug 2023. Article (CrossRef Link)

[3] Mach, P. and Becvar, Z., "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys and Tutorials*, 19(3), 1628-1656, 2017. Article (CrossRef Link)

[4] Aishwaryaprajna, Kirubarajan, T., Tharmarasa, R. and Rowe, J. E., "UAV path planning in presence of occlusions as noisy combinatorial multi-objective optimisation," *International Journal of Bio-Inspired Computation*, 21(4), 209-217, 2023. Article (CrossRef Link)

[5] Das, G., "Techno-economic analysis of novel multi-objective soft computing technique," *International Journal of Bio-Inspired Computation*, 20(3), 172-182, 2022. Article (CrossRef Link)

[6] Wang, H., "Research on VLSI layout method based on spatial evolutionary algorithm,"*International Journal of Computing Science and Mathematics*, 18(2), 176-188, 2023. Article (CrossRef Link)

[7] Ma, X., Fu, Y., Gao, K., Zhu, L. and Sadollah, A., "A multi-objective scheduling and routing problem for home health care services via brain storm optimization," *Complex System Modeling and Simulation*, 3(1), 32-46, 2023. Article (CrossRef Link)

[8] Semujju, S. D., Huang, H., Liu, F., Xiang, Y. and Hao, Z., "Search-Based Software Test Data Generation for Path Coverage Based on a Feedback-Directed Mechanism," *Complex System Modeling and Simulation*, 3(1), 12-31, 2023. Article (CrossRef Link)

[9] Shen, X., Lu, J., You, X., Song, L. and Ge, Z., "A region enhanced discrete multi-objective fireworks algorithm for low-carbon vehicle routing problem," *Complex System Modeling and Simulation*, 2(2), 142-155, 2022. Article (CrossRef Link)

[10] Shu, Z., Song, A., Wu, G. and Pedrycz, W., "Variable reduction strategy integrated variable neighborhood search and nsga-ii hybrid algorithm for emergency material scheduling," *Complex System Modeling and Simulation*, 3(2), 83-101, 2023. Article (CrossRef Link)

[11] Shu, C., Zhao, Z. W., Han, Y. P., Min, G. Y. and Duan, H. C., "Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach," *IEEE Internet of Things Journal*, 7(3), 1678-1689, Mar 2020. Article (CrossRef Link)

[12] Tang, M. and Wong, V. W. S., "Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems," *IEEE Transactions on Mobile Computing*, 21(6), 1985-1997, Jun 2022. Article (CrossRef Link)

[13] Zhou, H., Jiang, K., Liu, X. X., Li, X. H. and Leung, V. C. M., "Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing," *IEEE Internet of Things Journal*, 9(2), 1517-1530, Jan 2022. Article (CrossRef Link)

[14] Xu, X. L., Jiang, Q. T., Zhang, P. M., Cao, X. F., Khosravi, M. R., Alex, L. T., Qi, L. Y. and Dou, W. C., "Game Theory for Distributed IoV Task Offloading With Fuzzy Neural Network in Edge Computing," *IEEE Transactions on Fuzzy Systems*, 30(11), 4593-4604, Nov 2022. Article (CrossRef Link)

[15] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J. and Poor, H. V., "Cooperative Task Offloading and Block Mining in Blockchain-Based Edge Computing With Multi-Agent Deep Reinforcement Learning," *IEEE Transactions on Mobile Computing*, 22(4), 2021-2037, Apr 2023. Article (CrossRef Link)

[16] Yang, G. S., Hou, L., He, X. Y., He, D. J., Chan, S. and Guizani, M., "Offloading Time Optimization via Markov Decision Process in Mobile-Edge Computing," *IEEE Internet of Things Journal*, 8(4), 2483-2493, Feb 2021. Article (CrossRef Link)

[17] Zhao, G. M., Xu, H. L., Zhao, Y. M., Qiao, C. M. and Huang, L. S., "Offloading Tasks With Dependency and Service Caching in Mobile Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, 32(11), 2777-2792, Nov 2021. Article (CrossRef Link)

[18] Shen, Q. Q., Hu, B. J. and Xia, E. J., "Dependency-Aware Task Offloading and Service Caching in Vehicular Edge Computing," *IEEE Transactions on Vehicular Technology*, 71(12), 13182-13197, Dec 2022. Article (CrossRef Link)
[19] Chen, L., Wu, J. G., Zhang, J., Dai, H. N., Long, X. and Yao, M. Y., "Dependency-Aware Computation Offloading for Mobile Edge Computing With Edge-Cloud Cooperation," *IEEE Transactions on Cloud Computing*, 10(4), 2451-2468, Oct 2022. Article (CrossRef Link)
[20] Nguyen, L. X., Tun, Y. K., Dang, T. N., Park, Y. M., Han, Z. and Hong, C. S., "Dependency Tasks Offloading and Communication Resource Allocation in Collaborative UAV Networks: A Metaheuristic Approach," *IEEE Internet of Things Journal*, 10(10), 9062-9076, May 2023. Article (CrossRef Link)
[21] An, X. M., Fan, R. F., Hu, H., Zhang, N., Atapattu, S. and Tsiftsis, T. A., "Joint Task Offloading and Resource Allocation for IoT Edge Computing With Sequential Task Dependency," *IEEE Internet of Things Journal*, 9(17), 16546-16561, Sept 2022. Article (CrossRef Link)
[22] Yan, J., Bi, S. Z., Zhang, Y. J. and Tao, M. X., "Optimal Task Offloading and Resource Allocation in Mobile-Edge Computing With Inter-User Task Dependency," *IEEE Transactions on Wireless Communications*, 19(1), 235-250, Jan 2020. Article (CrossRef Link)
[23] Xiao, S., Wang, W., Wang, H. and Huang, Z., "A new multi-objective artificial bee colony algorithm based on reference point and opposition," *International Journal of Bio-Inspired Computation*, 19(1), 18-28, 2022. Article (CrossRef Link)
[24] Wang, S., Ma, D., Ren, Z., Qu, Y. and Wu, M., "An adaptive multi-objective particle swarm optimisation algorithm based on fitness distance to streamline repository," *International Journal of Bio-Inspired Computation*, 20(4), 209-219, 2022. Article (CrossRef Link)
[25] Usman, A. M., Yusof, U. K. and Naim, S., "Filter-based feature selection: a comparison among binary and continuous Cuckoo optimisation algorithms along with multi-objective optimisation algorithms using gain ratio-based entropy," *International Journal of Bio-Inspired Computation*, 20(3), 183-192, 2022. Article (CrossRef Link)
[26] Lan, H.-y., Xu, G. and Yang, Y.-q., "An enhanced multi-objective particle swarm optimisation with Levy flight," *International Journal of Computing Science and Mathematics*, 17(1), 79-94, 2023. Article (CrossRef Link)
[27] Pan, N., Lv, L., Fan, T. and Kang, P., "A multi-objective firefly algorithm combining logistic mapping and cross-variation," *International Journal of Computing Science and Mathematics*, 18(3), 255-265, 2023. Article (CrossRef Link)
[28] Lin, X., Ren, T., Yang, J. and Wang, X., "Multi-objective cellular memetic algorithm," *International Journal of Computing Science and Mathematics*, 15(3), 213-223, 2022. Article (CrossRef Link)
[29] Gu, F., Liu, H. and Liu, H., "A coevolutionary algorithm for many-objective optimization problems with independent and harmonious objectives," *Complex System Modeling and Simulation*, 3(1), 59-70, 2023. Article (CrossRef Link)
[30] Deb, K. and Jain, 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(4), 577-601, 2014. Article (CrossRef Link)
[31] Chen, H., Tian, Y., Pedrycz, W., Wu, G., Wang, R. and Wang, L., "Hyperplane assisted evolutionary algorithm for many-objective optimization problems," *IEEE Transactions on Cybernetics*, 50(7), 3367-3380, 2020. Article (CrossRef Link)
[32] Cheng, J., Yen, G. G. and Zhang, G., "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Transactions on Evolutionary Computation*, 19(4), 592-605, 2015. Article (CrossRef Link)
[33] Jiang, S. and Yang, S., "A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, 21(3), 329-346, 2017. Article (CrossRef Link)
[34] Zhang, X., Tian, Y. and Jin, Y., "A knee point-driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, 19(6), 761-776, 2015. Article (CrossRef Link)

Xiangyu Shi is currently working toward M.S. degree at computer science and technology, Taiyuan University of Science and Technology, Taiyuan, China. His research interests include computational intelligence and edge computing.

Zhixia Zhang received the Ph.D. degree from Taiyuan University of Science and Technology, Taiyuan, China, in July 2023. Her research interests include computational intelligence, network security, and machine learning.

Zhihua Cui received the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2008. He is a Professor with Taiyuan University of Science and Technology, Taiyuan, China. His research interests include evolutionary optimization, big data modeling, and networking security.

Xingjuan Cai received the Ph.D. degree in control theory and engineering from Tongji University, Shanghai, China, in 2017. She is a Professor with the School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan, China. Her interesting includes cloud computing, bio-inspired computation and applications.