Multiobjective Optimization for Joint Task Offloading, Power Assignment, and Resource Allocation in Mobile Edge Computing

Peng Wang, Kenli Li, *Senior Member, IEEE*, Bin Xiao, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Mobile edge computing (MEC) is an emerging computational paradigm for providing storage and computing capabilities in network edge, to improve the experience of users, to shorten the delay, and to reduce the energy consumption of mobile devices. In this article, we consider a multiuser and multiserver scenario, where each user has an application composed of multiple independent tasks that need to be executed, and each MEC server is equipped on a base station (BS) for assisting mobile users to execute computation-intensive and time-sensitive tasks. Multiobjective optimization for joint task offloading, power assignment, and resource allocation is studied to maximize the offloading gains of users. A multivariable and multiobjective optimization problem with three objectives is constructed. An efficient multiobjective evolutionary algorithm is developed to solve the problems of minimizing the response time, minimizing the energy consumption, and minimizing the cost. Simulation results verify the effectiveness of our algorithm, and show the method significantly improves the user's offloading benefits. According to the author's knowledge, this is the first paper on the exploration of multiobjective optimization of multiuser with multiple tasks and multiserver MEC system, in which the worst user offloading revenue is regarded as the optimization objectives.**

***Index Terms*—Delay, mobile edge computing (MEC), multiobjective optimization, power assignment, resource allocation, task offloading.**

I. Introduction

With the advancement of society and the rapid development of the communication technology, smart mobile devices (MDs), such as smartphones, wearable devices, and tablets computers are becoming more and more popular and indispensable in our work and life. Due to the convenience and ever-increasing performance of MD, mobile users' demand for various mobile applications (such as image/video processing, voice recognition, interactive games, etc.) is increasing explosively [1], [2]. However, with the size of MD and limited battery life, storage capacity, and computing power, it is very difficult or even impossible to complete these computing-sensitive mobile applications individually. Mobile cloud computing (MCC) is regarded as a possible way to solve such problems. It reduces the burden of computing by offloading tasks to remote public clouds through wireless access to resource-rich cloud centers. However, because the cloud center is deployed in a remote place, offloading requires a long transmission delay. It is difficult to meet the requirements of mobile applications, such as user experience continuity, high reliability, and ultralow latency [3]. These challenges have driven the deployment of services at the edge of mobile networks close to users.

In order to meet these challenges, the concept of mobile edge computing (MEC) has emerged, which can provide storage, computing, and other services at the network edge, that is, very close to the mobile user geometrically. Different from the remote public cloud computing system, the MEC server uses a general computing platform to be directly implemented on the local cellular base station (BS) or wireless access point (AP). MEC allows nearby mobile users to execute applications on it through computational offloading, which greatly reduces the round-trip transmission delay and reduces the burden on the back-haul network [4], [5]. The idea of the MEC computing paradigm is to bring computing resources, storage resources, and radio resources closer to MD, which can enhance the server's scalability in computing, storage, and radio [6], [7]. By empowering ubiquitous wireless access networks (such as 5G BSs) with powerful communication, computing, and storage capabilities, MEC can provide MD users with universal and agile computing enhancement services anytime and anywhere [8], [9]. Because the MEC platform is close to mobile users, it has the advantages of high bandwidth, location awareness, and ultralow latency [10], [11].

Computing offloading from MD to MEC server is an effective method to solve the contradiction between MD resource constraints and users' ultralow latency requirements for mobile applications. By using MEC storage and computing resources,

Manuscript received March 23, 2021; revised July 12, 2021 and September 30, 2021; accepted November 19, 2021. Date of publication December 2, 2021; date of current version July 7, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61876061 and Grant 62006074; in part by the National Key Research and Development Program of China under Grant 2018YFB1003401; and in part by the Postdoctoral Science Foundation of China under Grant 2020M672487. *(Corresponding authors: Kenli Li; Keqin Li.)*

Peng Wang is with the College of Information Science and Engineering and National Supercomputing Center in Changsha, Hunan University, Changsha 410082, Hunan, China, and also with the College of Computer Science and Engineering, Hunan Institute of Technology, Hengyang 412002, China.

Kenli Li is with the College of Information Science and Engineering and National Supercomputing Center in Changsha, Hunan University, Changsha 410082, Hunan, China (e-mail: lkl@hnu.edu.cn).

Bin Xiao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

Keqin Li is with the College of Information Science and Engineering and National Supercomputing Center in Changsha, Hunan University, Changsha 410082, Hunan, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2021.3132080

MD can reduce the time delay of its applications, save the energy consumption, and increase their standby time. However, it needs to occupy the up-link wireless device for communication because the task of MD is to be offloaded to the neighboring MEC server, which requires extra energy and time consumption for communication transmission. In addition, with many mobile users who need to execute computational offloading, the limited storage, and computing resources on the MEC server significantly affect the time delay of offloading tasks for the MEC system [12], [13]. Therefore, offloading decision making, power assignment, and MEC computing resource allocation have become key issues to achieve efficient tasks offloading.

Different from previous research, this article studies the computing offloading strategy of the MEC system with multiusers and multiservers with limited resources, and designs an overall strategy plan for joint task offloading, power assignment, and resource allocation to maximize the overall user task offloading benefit. Specifically, this article considers an MEC system with multiusers and multiservers, where the user MD has an application that can be decomposed into multiple independent computing tasks to be executed. And we have the following work. First, we consider the heterogeneity of MD and MEC servers. Each MD and MEC server have different calculation frequencies. Second, we consider the differences in tasks, they all have different computing workloads and input data sizes. Third, a performance indicators based on time delay, energy consumption, and the cost is proposed to comprehensively measure the performance of the algorithm strategy. Compared with most studies, the studied in this article are more close to the real situation, but also face greater challenges. First, the multiuser and multiserver scenario is more complex; second, heterogeneous servers and tasks make it more difficult to search for offloading strategies; and third, considering multiple indicators, its objectives are more difficult to optimize.

In the multiuser and multiserver MEC system, there are four key issues to be solved. First, how to choose which computing tasks to offload when face to the multiple tasks that mobile users need to be executed? Second, how to decide how much power to provide for offloading tasks when face to limited transmission power? Third, how to choose a suitable server for offloading so that users can offload as high as possible when there are multiple servers. Fourth, due to the complexity of these problems, this makes the optimization problem not a convex problem. In response to these issues, the main contributions of this article are summarized as follows.

1) We considered the difference among computing tasks and the heterogeneity among MEC servers. At the same time, we limit the resources of MD users and MEC servers, and consider paid offloading tasks. In this way, the scenario considered is closer to the real scenario in this article.
2) We combine task offloading, power allocation, and resource allocation to establish the delay, energy consumption, and cost indicators for user's offloading utility and compare them by analyzing the delay, energy consumption, and cost indicators. It is formulated as a constrained multiobjective optimization problem to maximize the system offloading utility.
3) We design a suitable code for the multiobjective optimization problem proposed in this article, and improve the coding form of the multiobjective evolutionary algorithm to apply to specific problem problems. And, we also improve the selection strategy of the multiobjective evolutionary algorithm based on decomposition (MOEA/D) to adapt to multiple constraints. We use SAW (simple additive weighting) and MCDM (multiple criteria decision-making) methods to calculate the offloading utility of each strategy in the population, and select the best offloading strategy.
4) We conduct comprehensive experiments and evaluations to verify the performance and effectiveness of the proposed the MOEAD_MEC algorithm. The influence of various parameters on offloading efficiency is discussed. In addition, we also put forward cost-performance indicators for a comprehensive evaluation of unloading benefits.

The remaining articles are organized as follows. We review the relevant references in Section II. In Section III, we introduce the system model and explain the joint offloading, power assignment, and resource allocation issues. On this basis, the mathematical model is established and formulated as a constrained multiobjective optimization problem. A multiobjective evolutionary algorithm for solving the proposed optimization problem is designed in Section IV. Section V introduces the simulation experiment and evaluates the performance of the MOEAD_MEC algorithm. The conclusions and future work is given in Section VI.

II. Related Work

In recent years, the MEC paradigm has attracted great attention from related researchers and has become a research hotspot in academia and industry, especially for research on computational offloading. Since Nokia launched the first real-world MEC platform in 2013 [14], a lot of work has been proposed to explore the potential benefits of MEC systems, especially for the current 5G hotspot environment [15]. Recently, more and more researches have been devoted to exploring the benefits of computational offloading on MEC systems [11].

Computing offloading research is a very challenging and attractive topic that involves making decisions about how to offload, where to run computing tasks, and how much computing resources to allocate [16]. Wang *et al.* [17] jointly optimized the calculation speed, transmission power, and offload rate of MDs by designing two objectives of delay and energy consumption minimization. Aiming at the situation of multiple tasks for a single user, with the purpose of reducing user delay and equipment energy consumption, Mao *et al.* [18] jointly optimized the power allocation and task offloading of the MEC system under the assumption that the tasks are independent of each other. For the multiuser single-task situation, the optimal resource allocation strategy for a computing offloading in the MEC system is explored by

minimizing the energy consumption weighted sum under delay constraints [19]. Ren *et al.* [20] investigated the computation offloading strategy in a hierarchical network architecture. On the basis of establishing a queuing model for MD and multiple heterogeneous MECs, Li [21] rigorously analyzed all offloadable and nonoffloadable tasks, as well as the average task response time in MD and each MEC server. A distributed algorithm consisting of computing offload selection, transmission power distribution, and clock frequency control strategy was proposed to reduce energy consumption and shorten time delay [22]. Yousefpour *et al.* [23] proposed a collaboration offloading strategy to minimize delay based on the characteristics of fog-capable devices. Based on the consideration of resource competition among mobile users, Ning *et al.* [24] formulated multiuser computing offloading as a mixed-integer linear programming problem, and designed an iterative heuristic algorithm to optimize the problem. Zhang *et al.* [25] investigated sustainable computation offloading in an edge-computing system that consists of energy harvesting-enabled MDs and a dispatcher. Aiming at the multiuser and multitasking situation, the tradeoff between the two conflicting goals (MD power consumption and the execution delay of computing tasks) in the MEC system is studied [26]. The offloading strategy is studied by formulating two offloading games to maximize individuals interest and maximize the overall systems interest [27].

The task offloading is more and more considered as a multiobjective problem in MEC, and it is becoming popular to solve it through evolutionary algorithms. Gedawy *et al.* [28] designed a set of heuristic algorithms to solve a multiobjective, resource-aware task allocation and scheduling problem in the edge system. In order to study the tradeoff between energy consumption and latency, this is a very important requirement for users. Kabir and Masouros [29] proposed a weighted multiobjective optimization problem, which is solved by using the Lagrangian method to design an iterative algorithm. By finding the optimal offloading probability and transmit power of each MD, a joint objective is used to formulate a multiobjective optimization problem to minimize energy consumption, execution delay, and payment costs [30]. An improved multiobjective evolutionary algorithm based on a specific problem is proposed to solve the problem of modeling the tradeoff between delay and energy consumption as a multiobjective computing offloading problem [31]. Bozorgchenani *et al.* [32] modeled the task offloading as a constrained multiobjective optimization problem in MEC, and designs an evolutionary algorithm that can effectively find a representative sample of the best tradeoff between energy consumption and task processing delay.

In this article, the research is a scenario of each MD with multiple tasks in multiple users and MECs environments, which is quite different from all existing studies. Most of the existing literature considers the delay, energy consumption, or resource usage cost independently when designing the offloading scheme. The problem of jointly optimizing these three objectives has not been well resolved in the multiple users and MECs system so far. Moreover, most of the existing studies believe that transmit power is constant, which do not match the actual situation.

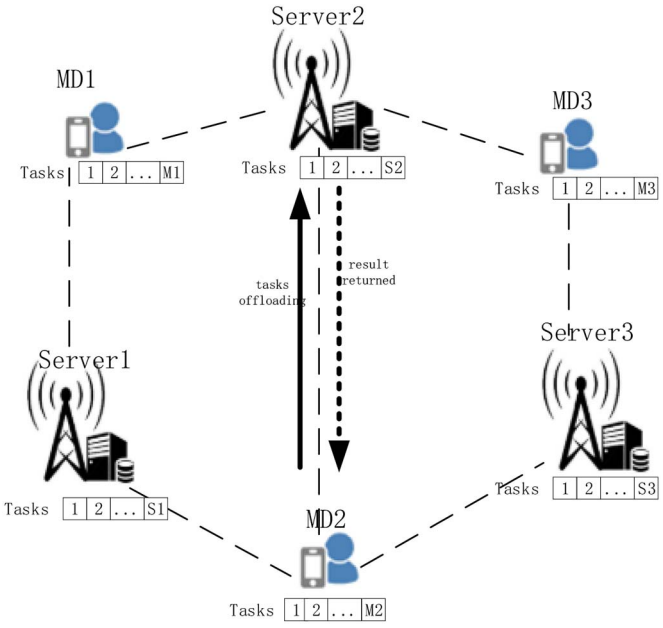

Fig. 1. Example of the MEC system with multiusers and multiservers deployed at the BSs.

III. System Model and Problem Formulation

A. System Model

As shown in Fig. 1, we consider a cell composed of multiple MD users (such as smartphones, wearable devices, and tablets) and multiple MEC servers (which can be physical servers or virtual machines with medium computing capabilities provided by operators), where each MEC server is equipped with a different BS. The MEC server communicates with mobile users through the wireless transmission provided by the corresponding BS and provides them with computing offloading services. Each MD user can execute single or multiple tasks locally, or offload tasks to different MEC servers through wireless transmission, then the MEC server processes tasks and transmits the calculation results back to the MD user. In this article, we use $J = \{1, 2, \ldots, J\}$, $S = \{1, 2, \ldots, S\}$, and $M = \{1, 2, \ldots, M\}$ to denote the task set on the MD user, the MEC server set, and the MD user set. At the same time, we make the following assumptions about the model based on the consideration of reality and comprehensibility. First, each MD has one mobile application needs to be executed, and the application can be decomposed into multiple independent tasks. Second, mobile users only know own the computational workload, input data size, and channel side information of their tasks. Third, mobile users can select different transmission power which transmit tasks to the server under the condition of meeting the transmission energy constraint. Fourth, the wireless links are assumed to be orthogonal channels in the MEC communication environment, so there are no interference among links. Fifth, the tasks offload to the MEC server are executed in parallel by allocating computing resources, and the MD executes tasks serially. Sixth, the MD user sends the necessary information for offloading to the MEC server simultaneously.

TABLE I
SUMMARY OF KEY SYMBOLS

Notations	Meanings
M	Set of MD user
S	Set of MEC server
J	Set of user task
f_s	The computing resources of server
$f^j_{m,s}$	The computing resources allocated to the mth user' jth task of on the MEC server
p^j_m	The transmission power allocated to the mth user' jth task of on the MD
x^j_m	The offloading policy of the mth user' jth task
CS_i	The ith solution in the evolutionary algorithm population
T	The delay of MD
E	The energy consumption of MD
MC	The cost of MD acquiring MEC server resources
k	The energy coefficient of MD
β	The cost coefficient of computing resources
c	The workload of task
d	The input data of task
f_m	The computing resources of MD
W	Uplink system bandwidth
N	The channel white Gaussian noise
h	The channel fading coefficient

The following is a detailed introduction and modeling of the system model through the local user computing model, task uploading model, and MEC server computation model. For ease of reading, Table I summarizes the main symbols.

1) Local User Computation Model: Each MD m has only one mobile application, and it can be broken down into multiple independent computing tasks in this article. And the CPU (cycles/s) of each MD is represented by $f_m > 0$, which shows the computing capability of the local MD. Each computing task denoted as j can be offloaded to one MEC server or executed locally, which can be represented by a tuple of two parameters $<c^j_m, d^j_m>$. The c^j_m (cycles) specifies the workload of the computing task, that is, how much calculation is needed to complete it. And the d^j_m (bits) is the information data that must be known to complete the calculation task, such as system settings, input parameters, and program codes. The tasks offloading reduce the energy consumption and time delay for completing tasks owing to the MEC server has greater computing capability, but it also increases communication energy consumption and time. In this article, x^j_m is used to denote the computing offloading strategy, and its possible value is $x^j_m \in \{0, 1, 2, \ldots, S\}$. When $x^j_m = 0$, it means that the computing task is executed on the local MD, and other values mean that it is offloaded to one MEC server corresponding to the value.

If the computing task j is executed locally, its execution time is $t = (c^j_m/f_m)$. Since the user of the MD m may have need to execute multiple computing tasks locally, the local execution time for completing the mobile application is

$$T^{\text{loc}}_m = \sum_{j \in L} \frac{c^j_m}{f_m}, \quad L = \left\{ j | x^j_m = 0 \right\}. \tag{1}$$

In order to calculate the MD' energy consumption when the tasks are executed locally, we use the energy consumption model of per computing cycle [33], which is widely used in other literature. The energy consumption of computing task j executed locally is $e = \eta_m c^j_m f^2_m$, where η_m is the energy coefficient related to the CPU architecture. The locally energy consumption of mobile user m is

$$E^{\text{loc}}_m = \sum_{j \in L} \eta_m c^j_m f^2_m, \quad L = \left\{ j | x^j_m = 0 \right\}. \tag{2}$$

2) Task Uploading Model: MD users offload tasks to the MEC server, which increase additional communication overhead, including transmission time and energy consumption. The transmission time is divided into transmitting the information data necessary to complete the task on the uplink, and transmitting the result completed back to MD on the downlink. Since the result returned to the user is usually very small compared with the input information data, we omit the overhead of returning the result in the downlink in the model. The transmission energy consumption mainly refers to the transmission energy of MD, that is, required to transmit input data necessary to complete the task to the MEC server. From the Rayleigh fading channel model [34], [35], it can be seen that the rate of sending information data d^j_m from the MD user m to MEC server s is

$$R^j_{m,s} = W_{m,s} \log_2 \left(1 + \frac{p^j_{m,s} h}{D^{\omega}_{m,s} N} \right) \tag{3}$$

where $W_{m,s}$ is the transmission channel bandwidth. $p^j_{m,s}$ is the transmission power assigned to task j by the MD m, which satisfies $0 < p^j_{m,s} < p_{\max}$. N and h are the channel white Gaussian noise and the channel fading coefficient, respectively. $D^w_{m,s}$ is the distance between MD m and MEC server s, and ω is the channel path loss exponent. Therefore, the transmission time of the information data of task j sent from the MD user m to s can be calculated as follows:

$$T\_up^j_{m,s} = \frac{d^j_m}{R^j_{m,s}}. \tag{4}$$

The energy consumption for the transmission of input data of task j sent from the MD user m to s can be calculated as follows [36]:

$$E\_up^j_{m,s} = p^j_{m,s} T\_up^j_{m,s}. \tag{5}$$

3) MEC Server Computation Model: Each MEC server can receive task offloads from different MD users, and allocate computing resources for each task. The resources allocated $f^j_{m,s}$ to task j must meet the constraints $0 < f^j_{m,s} < f^{\max}_s$ which $f^{\max}_s$ is computing power of MEC s, and for each MEC server must meet $\sum_{m=1}^{M} \sum_{j=1}^{J} f^j_{m,s} < f^{\max}_s$. For the offloading task j from the mobile user m, it is the execution time on the MEC server s is

$$T\_exe^j_{m,s} = \frac{c^j_m}{f^j_{m,s}}. \tag{6}$$

In addition, MD users must pay for the resources on the MEC server. In this article, we assume the unit cost of MEC s is β_s, and the cost paid is related to how many resources are used. Therefore, the cost of offloading task j from m to s can be calculated as follows:

$$\text{MC}^j_{m,s} = \beta_s f^j_{m,s}. \tag{7}$$

B. Problem Formulation

In this article, our intention is to reduce the delay of application response, lower the energy consumption of MD, and control as much as possible to complete tasks at a low cost. These three objectives are in conflict with each other.

It can be known from the hypothesis in this article, each MD user has a mobile application composed of several mutually independent tasks to be executed. Since the tasks are no relationship of each other and the MEC servers complete the offloading tasks in parallel, the delay of task offloading depends on the task with the longest delay. For MD users, the latency of executing the entire mobile application includes the latency of task offloading and the latency of locally executed tasks, and the latency of mobile applications depends on the longer one between them. So for the MD user m, we can calculate the delay of its mobile application as follows:

$$T_m = \max\left\{T_m^{\text{loc}}, \max_{j\in K_m}\left\{T\_up_{m,s}^j + T\_exe_{m,s}^j\right\}\right\} \tag{8}$$

where K_m is a collection of tasks that the MD user m offloading to the MEC server.

The energy consumption of MD users comes from the calculation when tasks are executed locally and the transmission when tasks are offloaded. Therefore, for the MD user m, the total energy consumption as follows:

$$E_m = \sum_{j\in K_m} E\_up_{m,s}^j + E_m^{\text{loc}}. \tag{9}$$

MD users want to reduce the time delay and energy consumption of executing mobile applications by offloading tasks. Accordingly, they should pay the MEC server provider. The fees payable should be the sum of the fees payable for offloading tasks on each MEC server. Therefore, for the MD user m, the cost is

$$\text{MC}_m = \sum_{j\in K_m} \text{MC}_{m,s}^j. \tag{10}$$

For a given offloading strategy x, up-link power assignment value p, and MEC server computing resource allocation value f for offloading tasks, the average offloading utility of all MD users can reflect the benefits of the entire MEC system. The formula is as follows:

$$T = \frac{\sum_{m\in M}\{T_m\}}{|M|} \tag{11}$$

$$E = \frac{\sum_{m\in M}\{E_m\}}{|M|} \tag{12}$$

$$\text{MC} = \frac{\sum_{m\in M}\{\text{MC}_m\}}{|M|}. \tag{13}$$

Based on the above analysis of execution delay, energy consumption, and cost, we can formulate a joint minimization problem. Considering the conflict among optimization objectives, one objective value decreases, other target values tend to become larger. Therefore, consider them as a multiobjective optimization problem and search for a solution for the trade-off among objective, which includes minimizing execution delay, minimizing energy consumption, and minimizing cost, as follows:

$$\min_{x_m^j, p_m^j, f_{m,s}^j} \quad \{T, E, \text{MC}\} \tag{14}$$

$$\text{subject to:} \quad x_m^j \in \{0, 1, 2, \ldots, S\} \tag{15}$$

$$0 \le p_m^j \le p_m^{\max} \tag{16}$$

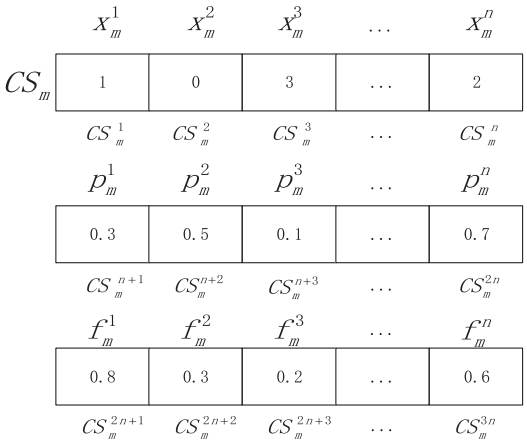

Fig. 2. Encoding instance for the computing tasks.

$$0 \le f_{m,s}^j \le f_s^{\max} \tag{17}$$

$$\sum_{j\in J}\{p_m^j\} \le p_m^{\max} \tag{18}$$

$$\sum_{m=1}^{M}\sum_{j=1}^{J} f_{m,s}^j < f_s^{\max}. \tag{19}$$

IV. Algorithm Design

Compared with traditional methods, evolutionary algorithms require very little domain-specific knowledge and assumptions, and evolutionary algorithms can obtain an approximate solution of the Pareto front only by running it once. In addition, because the optimization problem is NP-hard, the optimization effect of traditional methods is not ideal, and evolutionary algorithm is a possible exploration. Therefore, in this section, the evolutionary algorithm is selected to solve the above-mentioned multiobjective optimization problem. We first encode decision space of the multiobjective optimization problem proposed in the previous section. Then, taking into account the constraints, we modify the MOEA/D algorithm to search for the optimal solution set. Finally, the best solution is selected from the set of solutions based on evaluation of SAW and MCDM methods.

A. Encoding

As mentioned in Section III, each computing task corresponds to a computing offloading strategy, an up-link power assignment strategy, and an MEC server computing resource allocation strategy. In the differential evolution (DE) algorithm, genes represent computational offloading strategies, power assignment strategies, and computing resource allocation strategies of computing tasks. The genes form chromosomes, which represent a solution in the algorithm. Fig. 2 shows an example of the coding of the computing offloading strategy, power assignment strategy, and computing resource allocation strategy for computing task. In this example, one-third of the length of the chromosome is coded with integers $\{0, 1, 2, \ldots, S)\}$ to represent the computational offloading strategy, and the others are coded with real numbers

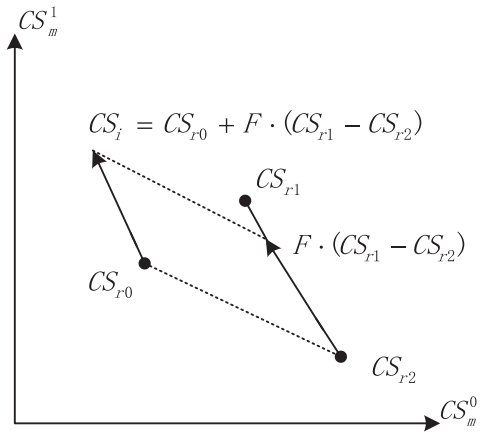

Fig. 3. Operation example of DE.

between 0 and 1 to represent the power assignment strategy and computational resource allocation strategy.

B. Computation Offloading Method MOEAD_MEC

1) Initialization: The initialization of the MOEAD_MEC algorithm includes the initialization of the relevant parameters of the multiobjective evolutionary algorithm. The parameters of DE include population size POP, crossing probability P_c, and mutation probability P_m. The parameters in MOEAD_MEC include the maximum value iter of iteration and the choice of the decomposition method (the Chebyshev decomposition method). Each chromosome represents the task offloading strategy of the computing task, the up-link power assignment strategy, and the server computing resource allocation strategy.

2) Reproduction Operating: In the evolutionary algorithm, the reproduction operation is a very critical step, which is to produce offspring populations. The reproduction operation can be divided into two steps, one is the crossover operation, and the other is the mutation operation. In this article, the reproduction operation is the same as the operation in [37], specifically using the DE operating and polynomial mutation to generate new solutions. Fig. 3 shows example of operation of the DE operator. Due to actual needs, the decoding is different from general real number coding or discrete coding in this article, which is a combination of them. Therefore, we made corresponding modifications during the operation. Specifically, we divide a solution into two parts (discrete and real numbers), reproduce them separately and then assemble them.

3) Evolution Operating: Selection operating is a key step that can be converged in evolutionary algorithms. With reference to the environment's choice of individuals, the selection operating is a choice of the fitness value of the objective function. During this operation, we aim to select better chromosomes as the next-generation solution of the population. After the reproduction operation, a new solution is produced. The fitness values of the three objective functions are evaluated separately. On this basis, a multiobjective evolution strategy is used to select a new solution to replace the solution in the original population and keep the population size unchanged at POP.

In this article, we need to optimize a multiobjective optimization problem with constraints, while the classic MOEA/D algorithm does not consider constraints. For the multiobjective optimization problem with constraints, it is obviously not feasible to use the unconstrained multiobjective evolutionary algorithm. Therefore, in order to solve this kind of constrained multiobjective optimization problem, we have made appropriate modifications [38]. For a solution CS_n, if it satisfies the constraint conditions, it is called a feasible solution, otherwise an infeasible solution.

For infeasible solutions, the degree of constraint violation can reveal it is the fitness value of an infeasible solution. In this article, constraint violation value is used, which is used to quantitatively describe the degree to which a solution violates constraints. For a solution CS_n, its value can be calculated as follows:

$$\mathrm{CV}(\mathrm{CS}_n) = \sum_{ci \in \mathrm{CI}} \langle g_{ci}(\mathrm{CS}_n) \rangle \tag{20}$$

where CI represents the set of constraints and $\langle \alpha \rangle$ means that if $\alpha \leq 0$, then $\langle \alpha \rangle = 0$, otherwise $\langle \alpha \rangle = |\alpha|$. Obviously, for a solution CS_n, the smaller the $\mathrm{CV}(\mathrm{CS}_n)$ value, the better the solution. At the same time, for a feasible solution, its CV value is 0, and for an infeasible solution, its CV value is greater than 0.

The following introduces the modified replacement strategy to deal with constrained multiobjective optimization problems. Assuming that *Cy* is a newly generated new solution, MOEAD_MEC randomly select one solution CS_n in the neighborhood set to determine whether to replace solution CS_n with solution *Cy*. This process until traversing the entire neighborhood set or replacing two existing solutions. It is replaced if any of the following conditions is met: 1) CS_n is an infeasible solution and *Cy* is a feasible solution; 2) CS_n and *Cy* are infeasible solution, but there are $\mathrm{CV}(\mathrm{CS}_n) > \mathrm{CV}(Cy)$; and 3) CS_n and *Cy* are all feasible solutions, but the aggregate function value of the solution *Cy* is smaller.

4) Selection Solution Using SAW and MCDM: In this article, the method proposed aims to optimize the delay, energy consumption, and cost of MD users, and to provide MD users with a compromise among delay, energy consumption, and cost. There are |POP| chromosomes as the solution to the multiobjective optimization problem in each population, which Each chromosome represents a task offloading strategy, the power assignment strategy, and the MEC server computing resource allocation strategy. In this article, we use SAW and MCDM methods to choose a compromise solution for the MEC system [39].

In the MEC system, the longer the delay for MD users, the worse the offloading strategy. Similarly, the energy consumption and cost of MD users are also negative standards. We standardize the delay, energy consumption, and cost as follows:

$$V(T^n) = \begin{cases} \frac{T^{n,\max} - T(\mathrm{CS}_n)}{T^{n,\max} - T^{n,\min}}, & T^{n,\max} \neq T^{n,\min} \\ 1, & T^{n,\max} = T^{n,\min} \end{cases} \tag{21}$$

$$V(E^n) = \begin{cases} \frac{E^{n,\max} - E(\mathrm{CS}_n)}{E^{n,\max} - E^{n,\min}}, & E^{n,\max} \neq E^{n,\min} \\ 1, & E^{n,\max} = E^{n,\min} \end{cases} \tag{22}$$

Algorithm 1 MOEAD_MEC

Input:
MOP: multi-objective optimization problem ; *NN*: the population size; *TN*: The neighborhood size; $\lambda_1, \lambda_2, \ldots, \lambda_{NN}$: A set of *NN* uniformly distributed weight vectors; Max evaluations: total generation or function evaluations.
Output:
solution: CS_n
Initialization:
1: Generate a set of *NN* weight vectors, $\{\lambda_1, \lambda_2, \ldots, \lambda_{NN}\}$;
2: find the *TN* the closest weight vector to each vector λ_i.
For each sub-problem $i = 1, \ldots, NN$, $B(i) = \{i_1, \ldots, i_{TN}\}$ is the neighborhood set.
3: Randomly generate the initial population $\{CS_1, CS_2, \ldots, CS_{NN}\}$;
4: Compute the F-function values $F(x_i)$, $\forall i = 1, \ldots, NN$;
5: Initialize the reference point $z = (z_1, \ldots, z_{mm})$.
Update:
6: **for** $i \leftarrow 1$ to solution size *NN* **do**
7: *Reproduction*: generate offloading strategy, capacity allocation strategy and computing resource allocation strategy section of a child solution *y*.
8: *Repair*: Apply problem-specific repair/improvement heuristic on the *y* to get the new solution CS_{new};
9: *Evaluation*: Compute the F-function values of CS_{new}, $F(CS_{\text{new}})$;
10: *Update of z*: Set $z_j = f_j(CS_{\text{new}})$, for each $j = 1, \ldots, mm$, if $f_j(CS_{\text{new}}) < z_j$;
11: *Replacement*: Set $CS_j = CS_{\text{new}}$ and $F(CS_j) = F(CS_{\text{new}})$.
12: **end for**
Stopping Criteria:
13: If the number of iterations is greater than max evaluations, then obtain approximation solutions: $\{x_1, \ldots, CS_{NN}\}$ and *PF*: $\{F(CS_1), \ldots, F(CS_{NN})\}$ else continue to update.
14: Use the SAW and MCDM method to select the most suitable solution CS_n in the population $\{CS_1, \ldots, CS_N\}$

$$V\left(\mathrm{MC}^n\right) = \begin{cases} \frac{\mathrm{MC}^{n,\max} - \mathrm{MC}(\mathrm{CS}_n)}{\mathrm{MC}^{n,\max} - \mathrm{MC}^{n,\min}}, & \mathrm{MC}^{n,\max} \neq \mathrm{MC}^{n,\min} \\ 1, & \mathrm{MC}^{n,\max} = \mathrm{MC}^{n,\min} \end{cases} \quad (23)$$

where $T^{n,\max}$, $T^{n,\min}$, $E^{n,\max}$, $E^{n,\min}$, $\mathrm{MC}^{n,\max}$, and $\mathrm{MC}^{n,\min}$ represent the two extreme values of delay, energy consumption, and cost in all solutions of the population. $T(\mathrm{CS}_n)$, $E(\mathrm{CS}_n)$, and $\mathrm{MC}(\mathrm{CS}_n)$ represent the delay, energy consumption, and cost of the nth solution, respectively. The weights of the three objective functions are represented by γ, δ, and ζ, respectively. The utility value of the nth solution is

$$V(\mathrm{CS}) = \max_{\mathrm{POP}} \left\{ \gamma V\left(T^n\right) + \delta V\left(E^n\right) + \zeta V\left(\mathrm{MC}^n\right) \right\}. \quad (24)$$

We choose the solution with the greatest benefit in the population as the final solution.

C. MOEAD_MEC Method Overview

This work aims to minimize the delay, energy consumption, and cost of MD users. The problem of optimizing task offloading strategy is defined as a constrained multiobjective optimization problem, and using MOEAD_MEC to search for the optimal offloading strategy, power assignment strategy, and MEC server computing resource allocation strategy. The overall process of algorithm design is as follows. First, the unloading strategy, power allocation strategy, and computing resource allocation strategy are encoded as the input of the algorithm, and the decomposition vector and its neighborhood are calculated. Then, a new chromosome is generated as a new solution through the duplication operation. Aiming at the fitness function and constraint conditions of the multiobjective optimization problem, the selection operation of the MOEA/D algorithm is modified, and the most suitable solution is selected as the next-generation population. Finally, the best strategy is selected through SAW and MCDM methods. For the MOEAD_MEC algorithm details, see Algorithm 1 as follows.

The MOEAD_MEC is summarized in Algorithm 1. We enter the maximum number of iterations, population size, neighborhood size, and multiobjective optimization problem. In lines 1–5 of Algorithm 1, $\lambda_1, \lambda_2, \ldots, \lambda_{NN}$, neighborhood sets, populations, and reference points are initialized. The closest weight vector is obtained by computing the Euclidean distances between any two weight vectors, and the reference point is calculated by $z_j = \min_{1<i<NN} f_j(x_i)\} \ \forall j = 1, \ldots, mm$. Lines 6–12 perform loop iteration. Line 7 is the reproduction operation by using DE operators over parents solution CS_u, CS_l, and CS_r, where they are selected from $B(i)$ randomly. Line 8 is to modify the new solution, line 9 is for the objective function fitness evaluation of the population, and line 11 is for updating the reference point. Line 11 is for the selection operation, for each index $j \in B(i) = \{i_1, \ldots, i_{TN}\}$, if $\mathrm{CV}(\mathrm{CS}_{\text{new}}) < \mathrm{CV}(x_j)$ or if $\mathrm{CV}(\mathrm{CS}_{\text{new}}) = \mathrm{CV}(x_j) = 0$ and $g(\mathrm{CS}_{\text{new}}|\lambda_j, z) \leq g(\mathrm{CS}_j|\lambda_j, z)$.

V. Performance Evaluation and Results Analysis

In the section, for the multiobjective optimization problem of joint task offloading, energy alloction, and resource assignment strategies, extensive simulation experiments are designed to evaluate the effectiveness of MOEAD_MEC. Specifically, we not only analyze the effect of the system, but also study the impact of related parameters, such as the number of users, the number of tasks, the number of servers, the size of data, and the size of the workload. In order to show that our proposed algorithm MOEAD_MEC can reduce time delay, energy consumption and cost, the following three schemes are used as benchmarks to evaluate the effect of the MOEAD_MEC algorithm.

1) *All Tasks Are Executed Locally (LE):* That is, all tasks are not offloaded. Each MD user runs its own mobile application task.
2) *All Tasks Are Executed on MEC(MEC_E):* All computing tasks are offloaded for execution, that is, all MD are not executed task. Each MD uses equal power to send offload tasks. And the offloading tasks are sent to different MEC servers in turn, and the server equally allocates computing resources to the received offloading tasks.
3) *Each Task Is Randomly Executed Locally or on MEC Server(R_E):* Each task is randomly executed locally or on server. Under the premise of ensuring the constraint conditions, all MD users randomly allocate power to different tasks, and the MEC servers randomly allocate resources to the offloading task.

The simulation experiment is simulating a multiuser and multiserver MEC platform. We refer to [21] and [35] for setting the values of the following parameters, see Table II for details. The weights of the three objective functions are $\gamma = 0.7$, $\delta = 0.2$, and $\zeta = 0.1$. In order to reflect the difference of tasks, the workload of the second task and the input data of the second task of each mobile user are multiplied by 100 on this basis. In order for the system to be as close to the real as possible, there are randomly placing MEC servers and MD users within 500 (m).

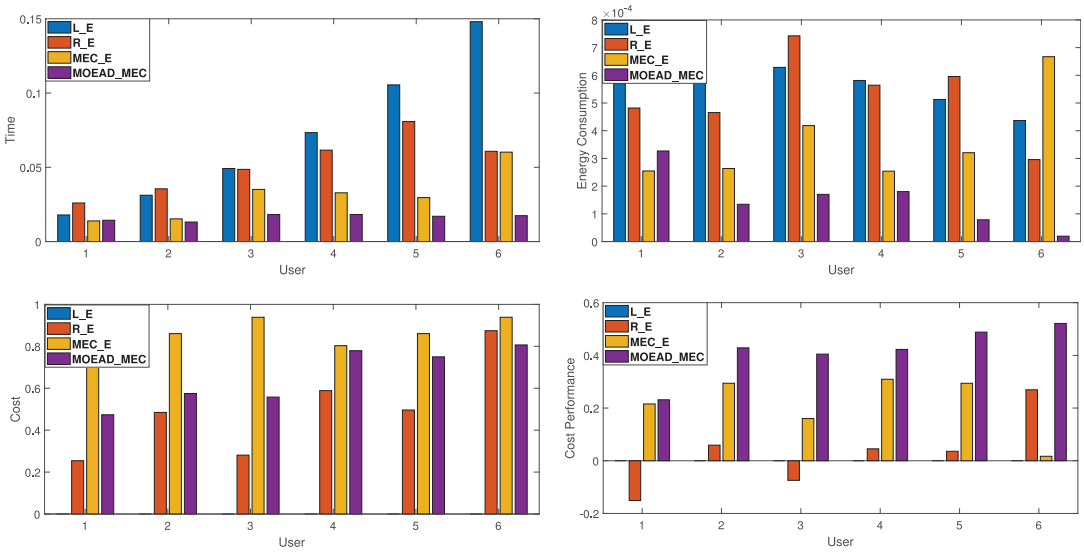

Fig. 4. Analysis of the latency, energy consumption, cost, and cost performance indicators of each MD device.

TABLE II
SUMMARY OF THE SIMULATION PARAMETERS

Parameter	Value
Set of MD user (M)	6
Set of MEC server (S)	3
Set of user task (J)	4
The channel path loss exponent (ω)	2
The computing resources of server (f_s)	$(2.0+0.1i)10^{9.0+0.1i}$ (cycles/s)
The computing resources of MD(f_m)	$(2.0+0.1)10^{8.0-0.1i}$ (cycles/s)
The transmission power on the MD (p)	$20+0.2i$ (W)
The energy coefficient of MD (k)	$(1.0+0.1i)10^{-27}$
The cost coefficient of computing resources (β)	$(5.1+0.2i)10^{-10}$
The workload of task ($c_{i,j}$)	$(5.0+0.5(i-1)+0.1j)10^4$
The input data of task ($d_{i,j}$	$(1.0+0.5(i-1)1.0j)10^2)$
Uplink system bandwidth (W)	10^8
The channel white Gaussian noise (N)	10^{-9}
The channel fading coefficient (h)	1.0^{-3}

A. Effect Analysis

In the section, there is an experimental analysis on the revenue obtained by each MD user through task offloading, as shown in Fig. 4.

In Fig. 4, the four subgraphs represent the analysis of the task offloading benefits of each MD user for delay, energy consumption, cost, and cost performance, respectively. The performance of the strategy obtained by the MOEAD_MEC algorithm is better than other strategy in all performance comparisons. It is worth noting that among these performance indicators. For the delay, energy consumption, and cost, the value is the smaller the better. On the contrary, the value of the cost performance indicator is the higher the better. It can be seen from the figure that the strategy obtained by the MOEAD_MEC algorithm can achieve better performance among the four indicators, and the performance obtained by each MD user is relatively little difference. This verifies the effectiveness of the optimization model and algorithm proposed in this article.

B. Effect of Number of Users

In Fig. 5, the performance of the MEC system with different numbers of user is shown. The four subgraphs show the performance of the MEC platform with different numbers of MD users for performance indicator delay, energy consumption, cost, and cost performance, respectively. In Fig. 5, the performance of the strategy obtained by the MOEAD_MEC algorithm is better than other strategy in all performance comparisons.

As shown in Fig. 5(a), the delay obtained by the strategy of MOEAD_MEC is increasing as MD users increases. Since when users increase, the number of tasks also increase. And, the average delay increase due to the total computing resources remain unchanged in the MEC system. In Fig. 5(b), the MOEAD_MEC algorithm maintains energy consumption at a low level when there are fewer users. This is because there are fewer tasks that need to be completed. Most of the tasks are offloaded, which consumes less energy. When the number of users is large and most tasks are offloaded, it seriously affects the time delay due to limited computing resources. Therefore, energy consumption is increased in exchange for the acceptable time by leaving some tasks are executed local. As users increases, the average cost gradually decreases in Fig. 5(c). This is because the MEC computing resources that can be purchased are certain. With the increase of users, the MEC computing resources used by each user are decreasing, so the average cost decrease. As shown in Fig. 5(d), the cost performance fluctuates less and has maintained a high value. It shows that the MOEAD_MEC algorithm is relatively stable and can maintain a good balance among the three goals.

C. Effect of Number of Tasks

In the section, we explored the impact of different numbers of MD tasks on the performance of the MEC platform, as shown in Fig. 6. The four subgraphs display the MEC system performance of different numbers of user' tasks for indicator delay, energy consumption, cost, and cost performance, respectively. The number of users' tasks is increasing linearly. It can be seen that the performance of the strategy obtained by the MOEAD_MEC algorithm is better than other strategies in all performance comparisons.

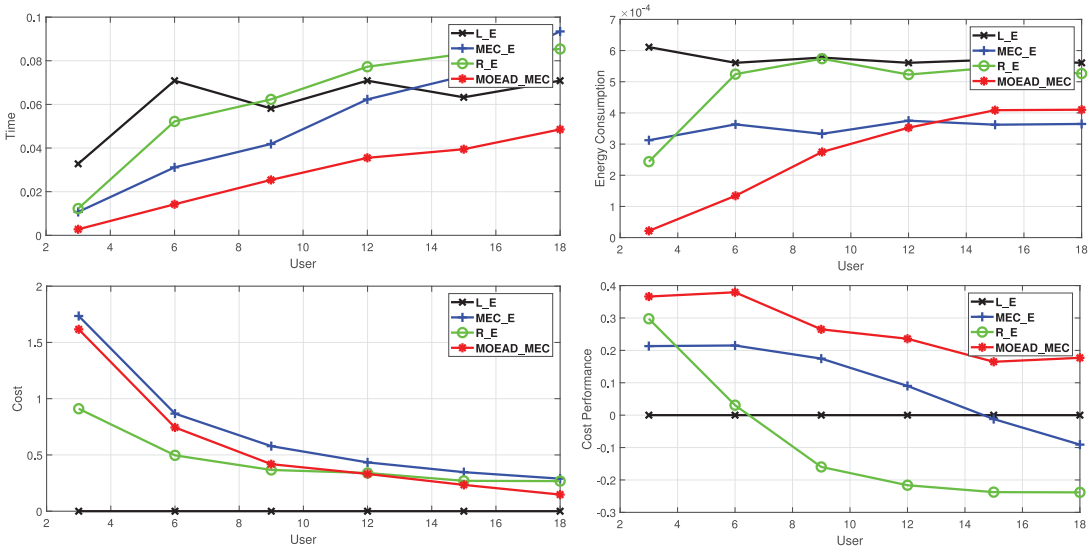

Fig. 5. Performance analysis of the MEC system with different numbers of users.

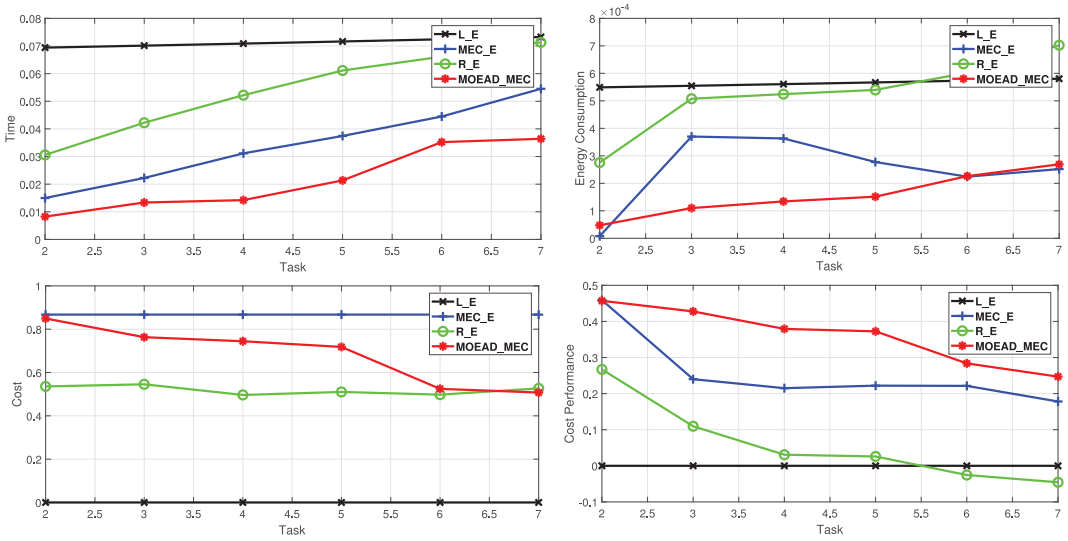

Fig. 6. Performance analysis of the MEC system with different numbers of tasks.

The average user delay of the strategy obtained by MOEAD_MEC increases as tasks for MD users increases in Fig. 6(a). With the fixed computing resources in the MEC system need to handle more tasks when computing tasks increase, so the average delay per user increase. The MOEAD_MEC algorithm maintains its energy consumption at a low level when there are fewer tasks in Fig. 6(b). This is because most tasks can be offloaded to utilize stronger computing resources when there are fewer tasks. When the tasks owned by users increases, excessive offloading of tasks to the limited resources MEC server inevitably increase the application delay. To control the excessive increase in delay, some tasks are executed locally. As shown in Fig. 6(c) and (d), there is no obvious trend in the cost and cost performance as the number of user tasks changes.

D. Effect of Number of Servers

The impact of having different numbers of MEC servers on the performance of the MEC system is evaluated in this section. The four subgraphs represent the performance of task offloading for different numbers of MEC servers in the indicator delay, energy consumption, cost, and cost performance in Fig. 7, respectively. The performance of the strategy obtained by the MOEAD_MEC algorithm is better than other strategies in the four subgraphs.

The average delay of the strategy obtained by MOEAD_MEC decrease as MEC servers increases in Fig. 7(a). It means that the MEC system has more computing resources to computing these tasks when MEC servers increases. MD users can use more computing resources of the MEC server, thereby more tasks are offloaded and the average delay of MD users are reduced accordingly. The energy consumption of MD users under the MOEAD_MEC algorithm decreases with the increase of MEC servers in Fig. 7(b). This is because MD users are more inclined that the computing tasks are offloaded for execution with servers increase. The tasks executed locally are reduced and the energy consumption tends to decrease. As MEC servers

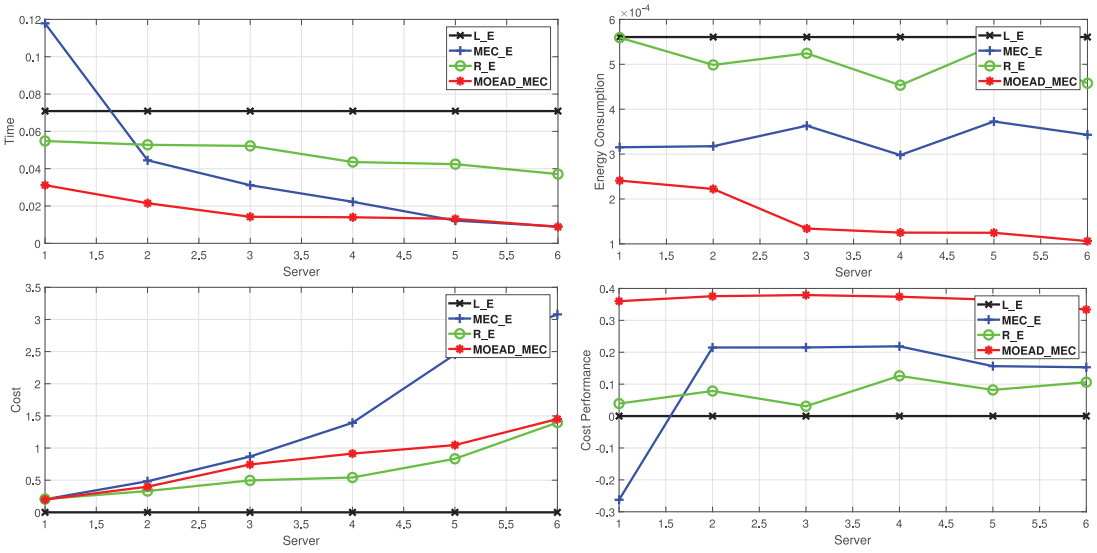

Fig. 7. Performance analysis of the MEC system with different numbers of servers.

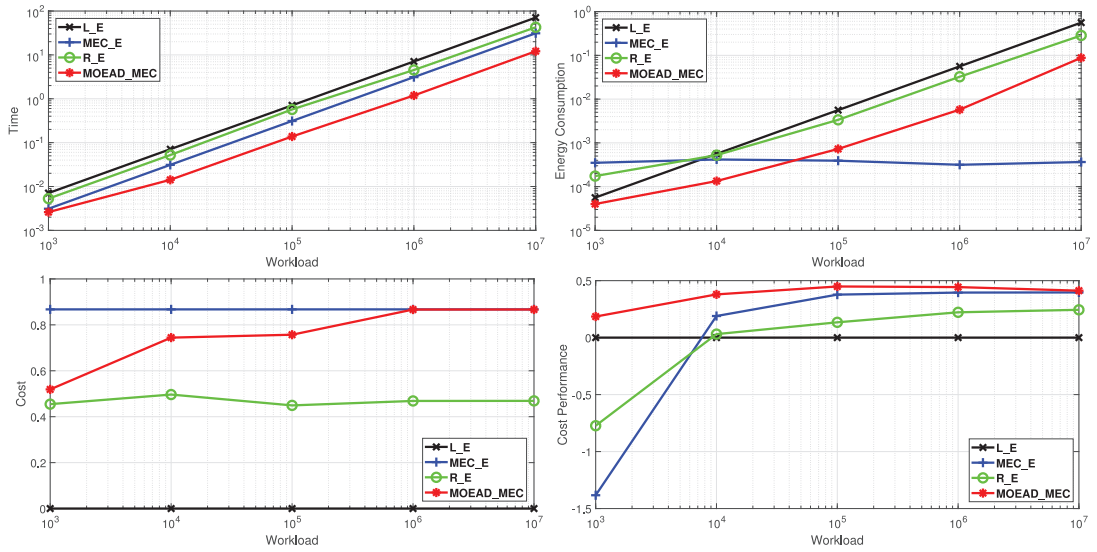

Fig. 8. Performance analysis of MEC system with different size of workload.

increases, the average cost of users gradually increases in Fig. 7(c). This is because the same user and computing tasks choose more MEC server resources to reduce the delay and energy consumption of MD. The MEC computing resources that are purchased increase, and users have to pay more. As shown in Fig. 7(d), the cost performance has been relatively stable with MEC servers increases. It shows from the side that the MOEAD_MEC algorithm is relatively stable and can maintain a good balance among the three objectives.

E. Effect of Size of Workload

In the section, we evaluate the performance impact of offloading strategy for different workloads for the tasks, as shown in Fig. 8. The four subgraphs represent the performance of MD users with different workload tasks for indicator delay, energy consumption, cost, and cost performance, respectively. In Fig. 8, the workload of the offloading task changes exponentially. It can be seen that the performance of the strategy obtained by the MOEAD_MEC algorithm is better than other strategy in all performance comparisons.

In Fig. 8(a), the mean delay of the strategy obtained by MOEAD_MEC keep increasing as the task workloads increases. Its execution time increase accordingly when the task workloads increases. And the total computing resources in the MEC system remain unchanged, so the average delay increase. In Fig. 8(b), similar to the experimental results of delay, energy consumption increases with the workload increase of this task in the MOEAD_MEC algorithm. This is because it seriously affects the delay when the workload increases and most tasks are offloaded. Due to limited computing resources, it tends to increase local execution energy consumption in exchange for the acceptable delay. The average cost of users tends to increase as the workload of tasks

increases in Fig. 8(c). This is because excellent strategies tend to that the tasks are offloaded for shortening delay, as the MEC server has stronger computing and shorter execution time. For Fig. 8(d), the cost performance is relatively stable as the workload changes and has maintained a high value. It shows that the MOEAD_MEC algorithm is relatively stable and can maintain a good balance among the three objectives.

VI. Conclusion

In this article, we consider issue of the offloading policy on multiuser and multiserver MEC platform. Through joint task offloading, power assignment, and resource allocation strategies, we are to optimize the delay, energy consumption, and cost of MD users. In order to make all users have a better offloading benefit, we have designed the objective to focus on optimizing offloading benefit of MD users with the worst performance. The optimization problem involves three conflicting optimization objectives, so it is expressed as multiobjective constrained optimization problem in this article. For this problem, we design a suitable population code, modify the replacement strategy in MOEA/D, and propose the MOEAD_MEC method to solve it. We design a comprehensive simulation experiment to verify the performance of MOEAD_MEC and the feasibility of the multiobjective model. The experimental results indicate the proposed strategy obviously outperforms the baseline method for time delay, energy consumption, cost, and cost performance, respectively. And these show the effectiveness of the MOEAD_MEC algorithm and the model.

In future work, we will continue to follow the work of this article and focus on some issues worthy of research.

1) It assumes that each user's mobile application is composed of mutually independent tasks in this article. In the follow-up research, the dependencies between tasks will be considered.
2) In this article, we are considering channel transmission without mutual interference. In the follow-up research, we will discuss the efficiency of our model under interference channels.
3) We will extend the models and methods in this article to the cloud-side collaboration system in the follow-up work, which is very interesting and valuable.

References

[1] F. Liu *et al.*, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[4] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[5] C. Liu, K. Li, J. Liang, and K. Li, "COOPER-SCHED: A cooperative scheduling framework for mobile edge computing with expected deadline guarantee," *IEEE Trans. Parallel Distrib. Syst.*, early access, Jun. 7, 2019, doi: 10.1109/TPDS.2019.2921761,

[6] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[7] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, pp. 869–904, 2nd Quart., 2020.

[8] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Darmstadt, Germany, 2013, pp. 26–30.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[10] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Trans. Ind. Informat.*, early access, Apr. 4, 2019, doi: 10.1109/TII.2019.2909473.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[12] C. Liu, K. Li, J. Liang, and K. Li, "Cooper-match: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC," *IEEE Trans. Mobile Comput.*, early access, Jun. 11, 2019, doi: 10.1109/TMC.2019.2921713.

[13] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.

[14] Intel and Nokia Solutions and Networks. "Increasing Mobile Operators' Value Proposition With Edge Computing." Technical Brief. 2013. [Online]. Available: http://nsn.com/portfolio/liquid-net/intelligent-broadband-management/liquid-applications

[15] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[16] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[17] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[18] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, 2017, pp. 1–6.

[19] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[20] J. Ren, K. M. Mahfujul, F. Lyu, S. Yue, and Y. Zhang, "Joint channel allocation and resource management for stochastic computation offloading in MEC," *IEEE Trans. Veh. Technol.*, vol. 69, pp. 8900–8913, Aug. 2020.

[21] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, early access, Mar. 12, 2019, doi: 10.1109/TSUSC.2019.2904680.

[22] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, pp. 319–333, Feb. 2019.

[23] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[24] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[25] D. Zhang *et al.*, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 880–893, Apr. 2020.

[26] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, 2016, pp. 1–6.
[27] L. Li, T. Q. S. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang, "An incentive-aware job offloading control framework for multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 63–75, Jan. 2021.
[28] H. Gedawy, K. Habak, K. A. Harras, and M. Hamdi, "RAMOS: A resource-aware multi-objective system for edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, pp. 2654–2670, Aug. 2021.
[29] M. T. Kabir and C. Masouros, "A scalable energy vs. latency trade-off in full-duplex mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5848–5861, Aug. 2019.
[30] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.
[31] F. Song, H. Xing, S. Luo, D. Zhan, P. Dai, and R. Qu, "A multiobjective computation offloading algorithm for mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8780–8799, Sep. 2020.
[32] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2021.
[33] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[34] B. Sklar, "Rayleigh fading channels in mobile digital communication systems .I. Characterization," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 90–100, Jul. 1997.
[35] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4800–4810, Jul. 2020.
[36] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
[37] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.
[38] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.
[39] X. Xu *et al.*, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.

Peng Wang received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2019.

He is currently a Postdoctoral Fellow with the College of Information Science and Engineering, Hunan University. He is also an Associate Professor with Hunan Institute of Technology, Hengyang, China. He has published many research articles in international conference and journals. His research interests include edge computing, cloud computing, computational intelligence, and artificial intelligence.

Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2003.

He was a Visiting Scholar with the University of Illinois at Urbana Champaign, Champaign, IL, USA, from 2004 to 2005. He is currently the Dean and a Full Professor of Computer Science and Technology with Hunan University, Changsha, China, and the Deputy Director of National Supercomputing Center in Changsha. He has published more than 150 research papers in international conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the *Journal of Parallel and Distributed Computing*, ICPP, and CCGrid. His major research areas include parallel computing, high-performance computing, grid, and cloud computing.

Prof. Li serves on the editorial board of the IEEE TRANSACTIONS ON COMPUTERS. He is an outstanding member of CCF.

Bin Xiao (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Fudan University, Shanghai, China, and the Ph.D. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2003.

After his Ph.D. graduation, he joined the Department of Computing of the Hong Kong Polytechnic University as an Assistant Professor. He is an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He published more than 180 technical papers in international top journals and conferences. His research interests include AI and network security, data privacy, and blockchain systems.

Dr. Xiao is currently the Associate Editor of IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and *Journal of Parallel and Distributed Computing* (Elsevier). He is the Vice Chair of IEEE ComSoc CISTC Committee. He has been the Symposium Co-Chair of IEEE ICC 2020, ICC 2018, and Globecom 2017, and the General Chair of IEEE SECON 2018. He is a member of ACM and CCF.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York, New Paltz, NY, USA. He is also a Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored more than 810 journal articles, book chapters, and refereed conference papers. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent, and soft computing.

Prof. Li has received several best paper awards. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.