Multi-Objective Optimized Cloudlet Deployment and Task Offloading for Mobile Edge Computing

Xiaojian Zhu and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Mobile edge computing provides an effective approach to reducing the workload of smart devices and the network delay induced by data transfer through deploying computational resources in the proximity of the devices. In a mobile edge computing system, it is of great importance to improve the quality of experience of users and reduce the deployment cost for service providers. This paper investigates a joint cloudlet deployment and task offloading problem with the objectives of minimizing energy consumption and task response delay of users and the number of deployed cloudlets. Since it is a multi-objective optimization problem, a set of tradeoff solutions ought to be found. After formulating this problem as a mixed integer nonlinear program and proving its NP-completeness, we propose a modified guided population archive whale optimization algorithm to solve it. The superiority of our devised algorithm over other methods is confirmed through extensive simulations.**

***Index Terms*—Mobile edge computing, offloading, multi-objective optimization, communication, Internet of Things.**

I. Introduction

WITH the development of the Internet of Things (IoT) technology, more and more smart devices are used, such as mobile phones, tablets, intelligent sensors, wearable devices, and smart cameras. More and more applications are run on them. If the applications are run on the remote central cloud, the task response delay of users tends to be large as a result of likely high network latency caused by, e.g., network congestion [1], [2]. To overcome this issue, mobile edge computing technology [1], [3], [4] emerges. Computational resources can be brought close to users via a mobile edge computing system (MECS) [1], [3], [5]. It consists of a number of cloudlets, and its deployment can decrease task response delay and network traffic. It is thus regarded as one of the key technologies to cope with the challenges posed by the proliferation of mobile devices and IoT applications [6]. In smart city applications, cloudlets can be co-located with some base stations to process huge amounts of data produced by IoT devices. In addition, cloudlets can be deployed at some important sites, such as railway stations, factories, shopping arcades, schools, and roadside units, to provide computing resources for intelligent terminal devices/equipment. In these applications, it is necessary to optimize energy consumption and task response delay of users and the number of deployed cloudlets. Some of mobile applications are computation-intensive, such as augmented reality, video processing, and natural language processing. Since they require more computational resources, their executions consume more energy. Because user devices are powered by batteries, an important consideration is how to save their energy [7], [8]. In MECS, user devices can reduce their energy consumption and extend their service life through offloading their non-urgent tasks to cloudlets.

Decreasing response latency of mobile applications improves user experience [9], [10]. Compared to local execution, the response delay of user tasks executed on cloudlets may be longer due to more waiting time including transmission, routing, and execution time. The large number of users sharing limited wireless bandwidth may cause long transmission time from them to their associated access point (AP) or base station (BS). The routing time from an AP to a cloudlet is long if their distance is large. When too many tasks are executed on cloudlets, execution time may become too long due to the cloudlets' limited computing capacities. Therefore, a good tradeoff between energy consumption and task response delay of users has to be made.

In addition to task offloading strategies, the optimal deployment of cloudlets according to user demand distribution is also essential to improve the quality of experience (QoE) of users. More cloudlets can accommodate more offloaded tasks of users, and thus more energy of users can be saved. The routing and running time of user tasks can be reduced by deploying more cloudlets at more appropriate sites. However, a larger number of deployed cloudlets leads to higher deployment cost. In order to reduce such deployment cost, the providers of edge computing services strive to deploy as few cloudlets as possible while guaranteeing user QoE. Accordingly, it is necessary to balance MECS deployment cost and user QoE.

Most existing work on cloudlet deployment such as [11]–[13] and [14] mainly focuses on the minimization of deployment cost or task response delay, but ignores the optimization of user energy consumption. Although there is some work studying task offloading strategies to minimize energy consumption and task response delay of users, e.g., [15]–[17], it does not take into account cloudlet deployment cost. On the other hand, most existing work on cloudlet deployment and task offloading such as [13], [15], [16], [18] and [19] employs the weighted sum method to deal with multi-objective optimization problems, while it is hard to select such weighting coefficients. Observing the limitations of the

This work was supported by the Natural Science Foundation of Jiangsu Province under Grant BK20160411.

X. Zhu is with the School of Computer Science and Engineering, Changshu Institute of Technology, Changshu 215500, China (e-mail: zhuxj5318@163.com).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

existing work, we are motivated to examine how to optimize energy consumption, task response delay of users and MECS deployment cost simultaneously, such that a comprehensive insight into MECS design can be offered to MECS service providers. In addition, we are motivated to develop an effective method for producing a set of tradeoff solutions, such that it is easy for them to make an informed decision.

In this work, we investigate a Multi-objective Optimization problem arising from Cloudlet-deployment and Task-offloading (MOCT) in MECS. In this problem, not only the minimization of energy consumption and task completion delay of users is required, but also the minimization of network deployment cost is demanded. The three objectives conflict with each other. Specifically, less energy consumption of users makes more tasks offloaded, and thus more cloudlets have to be deployed. On the other hand, smaller task completion delay of offloaded tasks requires less routing and running time, thus increasing the number of deployed cloudlets. Moreover, lower task response latency allows fewer tasks to be offloaded, yet increasing the energy consumption of user devices. As the three objectives are essentially very different, their weighted sum makes little sense to a decision-maker. Therefore, we do not use weighting coefficients to convert MOCT into a single-objective optimization problem. Instead, a set of Pareto optimal solutions have to be found. This work attempts to make the below contributions:

1) Formulating a novel problem of MOCT as a mixed integer nonlinear program (MINP) and proving it to be NP-complete; and
2) Proposing a modified guided population archive whale optimization algorithm for MOCT to produce a set of Pareto optimal solutions given an MINP. In addition, this work conducts extensive simulations to compare the performance of our proposed algorithm against those in the literature. The comparison results confirm the superiority of our proposed algorithm over its compared peers.

The organization of the rest of this paper is as follows. Section II reviews related work on mobile edge computing. Section III builds a mathematical model for MOCT and analyzes its complexity. Section IV gives a modified guided population archive whale optimizer-based cloudlet deployment and task offloading algorithm. Section V reports the simulation results. Section VI concludes the work.

II. RELATED WORK

An MECS can provide computational resources for users. On one hand, to reduce its energy consumption and task completion delay of users, the problems of task offloading and resource allocation should be effectively solved. On the other hand, to form it, the cloudlets should be properly deployed so as to enhance system performance while minimizing system cost.

During task offloading, communication and computation resources have to be optimally allocated subject to the resource and quality-of-service (QoS) limitations. The work in [15] focuses on how to optimize the computing frequencies and transmission powers of IoT mobile devices and make offloading decisions of their multiple types of tasks to minimize their task completion delay and energy consumption under the maximum completion latency constraint of each task. The work in [16] investigates how to optimally select a task offloading strategy, user transmission power, and allocation of computing frequencies of mobile edge computing servers to their associated users so as to maximize the system utility with respect to the task completion delay and energy consumption of users. The work in [18] stresses the minimization of the weighted sum of the energy consumed by users for offloading their tasks to cloudlets and the cost of the computational resources of cloudlets allocated to users by network operators while meeting users' maximum task completion delays. Three algorithms are presented to solve the problem. Liu *et al.* [20] investigate a task offloading problem where offloading probabilities and transmission powers of users are optimized to minimize their energy consumption, task completion latency, and the cost of the resources of the fog node and central cloud used by them. They propose an algorithm based on the interior point method to solve the problem. However, only one fog node is considered in the problem. Khalili *et al.* [17] consider the problem of minimizing the energy consumption of users by optimizing their offloading decisions and the assignment of subchannels and transmission powers used by them while guaranteeing their maximum service delay constraint. In [21], a modified NSGA-II algorithm is proposed to tackle the task offloading decision problem for the minimization of both energy consumption and task completion latency of smart mobile devices. Hu *et al.* [22] employ a UAV to provide computation resources for ground users. They investigate how to optimize a UAV trajectory and user task offloading to minimize task delay. This problem takes into account the user fairness and the energy consumption constraints of both users and UAV. The work in [23]–[25] investigates workload allocation and balancing to minimize response time of users.

Coupled with the schemes of task offloading and resource allocation, a cloudlet deployment strategy has a great influence on system performance and cost. The work in [26] examines the problem of deploying edge servers and assigning BSs to edge servers in order to minimize the maximum workload difference among edge servers and the maximum access delay between BSs and edge servers. To solve this problem, a mixed integer program is presented. Xu *et al.* [11] address how to place cloudlets and assign user requests to minimize the average access delay from users to cloudlets. In this problem, different cloudlets may have different computing capacities and different user requests may have different computational resource demands. They propose an integer linear program and a heuristic algorithm to solve the problem. In addition, two approximation algorithms are designed for a special case of the problem. The work in [19] reports the cloudlet deployment and task assignment problem with the minimum energy consumption and proves it to be NP-hard. In the problem, the total energy consumption consists of the energy consumed by the used links, placed cloudlets, and cloud servers, and the task completion delay requirement must be met. To formulate and solve the problem, a mixed integer linear program and an algorithm based on the Benders decomposition are presented, respectively. Jia *et al.* [12] investigate the cloudlet placement and user assignment problem with the minimum system response delay and devise two heuristic algorithms to solve it. In [13], a mixed integer program is proposed for the cloudlet deployment problem with the objective of minimizing the sum of the deployment cost and the delay from the regions of user requests to their assigned cloudlets. In [27], the cloudlets are deployed through a Lagrangian heuristic algorithm. The work in [28] presents a mixed integer linear program for the minimum cost deployment of the roadside units equipped with a server under the constraints of the minimum ratios of the network coverage and satisfied computational resource demand. The deployment cost is related to the number of the deployed roadside units, the distances between them and the cells they cover, and the transmission power levels assigned to them. Ren *et al.* [14] examine the problem of deploying the minimum number of cloudlets to satisfy the computation demand of each wireless access point subject to the maximum access delay constraint. They also present an integer linear program and a greedy algorithm to solve it.

Different from the previous work, we address the problem of the joint optimization of cloudlet deployment and task offloading to minimize the energy consumption and task completion delay of users and the number of deployed cloudlets simultaneously in this work. Because it is a triple-objective optimization problem, we need to devise an effective algorithm to obtain a set of Pareto optimal solutions.

III. PROBLEM MODEL

We consider a mobile edge computing network with M users and N APs. A number of cloudlets are deployed in the network to provide computing service for users. All APs are interconnected by wired links and each user can access the network via its associated AP. Let u_i $(1 \le i \le M)$ and v_j $(1 \le j \le N)$ represent the ith user and jth AP, respectively. Denote ϑ_i and λ_i as the index of the associated AP and average task arrival rate of u_i, respectively. Denote L as the maximum number of the cloudlets that can be deployed, which is the upper bound of the budget of a service provider. Let c_i stand for the ith $(1 \le i \le L)$ cloudlet. We assume that the potential sites where the cloudlets can be deployed are the positions of the APs, i.e., the deployed cloudlets are co-located with some of the APs. In addition, at most a cloudlet can be deployed at the location of an AP. Fig. 1 shows the MECS model we consider. A task of a user u_i can be characterized by its input data size and computational resource demand. Both of them are assumed to satisfy the exponential distribution, and their average values are represented by α_i (bits) and β_i (CPU cycles), respectively. Denote φ_i^u (CPU cycles/second) and ξ_i as the computing capacity and effective switched capacitance of the CPU of u_i, respectively.

The energy consumption model for processing a task in [15], [16], [29] is adopted, and thus the energy consumption for executing a task of a user u_i locally is given as:

$$e_i^c = \xi_i(\varphi_i^u)^2\beta_i. \tag{1}$$

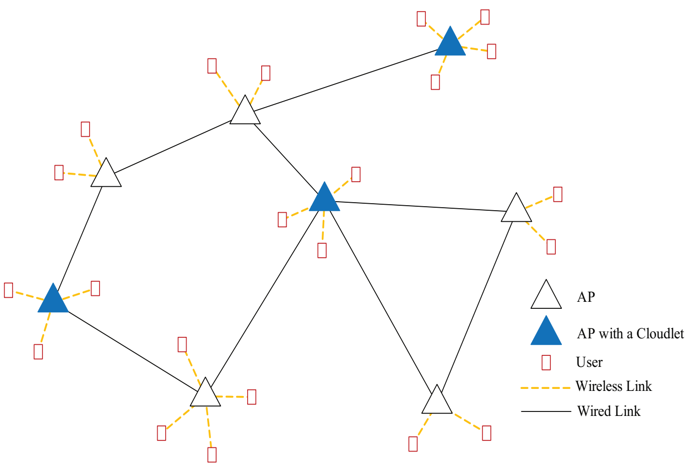


Fig. 1. Illustration of the MECS model.

As shown in Fig. 1, if a user wants to offload some of its tasks to some cloudlets, it has to first transmit them to its associated AP through the wireless link. Then, its associated AP sends them to the APs where the cloudlets are located through wired links. In the network, the orthogonal frequency division multiple access (OFDMA) is adopted. All users share the same system bandwidth B, and the users associated to the same AP employ orthogonal channels to transmit their offloaded tasks. Thus, there is no interference among users associated to the same AP, but users associated to different APs may interfere with each other. Let p_i and w_i stand for the transmission power and available channel bandwidth of user u_i, respectively. For the users associated to the same AP, the system bandwidth is evenly divided among them, and thus $w_i = B/\sum_{1 \le j \le M \wedge \vartheta_j = \vartheta_i} 1$. Denote $g_{i,j}$ and σ^2 as the channel gain from u_i to v_j and the background noise power, respectively. The transmission rate of u_i can be computed as [30]:

$$r_i^u = w_i log_2(1 + \frac{p_i g_{i,\vartheta_i}}{\sigma^2 + \sum_{1 \le j \le M \wedge \vartheta_j \ne \vartheta_i} p_j g_{j,\vartheta_i}}). \tag{2}$$

The energy consumption required for transmitting a task of a user u_i to its associated AP is

$$e_i^t = p_i \frac{\alpha_i}{r_i^u}. \tag{3}$$

Denote $z_{i,j}$ as the probability of offloading a task of user u_i to cloudlet c_j. It is assumed that the task arrival at a user is a Poisson process, and thus both the local execution and transmission of its tasks are modelled as an M/M/1 queue [31]. Accordingly, the average waiting time composed of the queue and execution time for executing a task of u_i locally is

$$\tau_i^l = \frac{\beta_i}{\varphi_i^u - \beta_i \lambda_i (1 - \sum_{1 \le j \le L} z_{i,j})}. \tag{4}$$

The average waiting time consisting of the queue and transmission time for transmitting an offloaded task of u_i to its associated AP is

$$\tau_i^t = \frac{\alpha_i}{r_i^u - \alpha_i \lambda_i \sum_{1 \le j \le L} z_{i,j}}. \tag{5}$$

Each cloudlet is equipped with only one server. Denote φ^c and μ as the computing capacity (CPU cycles/second) and maximum workload (CPU cycles/second) of each cloudlet, respectively. The maximum workload of a cloudlet is leveraged to avoid the extremely long waiting time of offloaded tasks. Since the amount of the resulting data of a task is very small, we ignore the energy consumption and delay caused by returning it from a cloudlet to a user. Since the task arrival at a cloudlet is a Poisson process, the task execution in it can be modelled as an M/M/1 queue [31]. Thus, the average waiting time comprising of the queue and execution time for executing a task in a cloudlet c_i is

$$\tau_i^c = \frac{\frac{\sum_{1\le j\le M} z_{j,i}\lambda_j\beta_j}{\sum_{1\le j\le M} z_{j,i}\lambda_j}}{\varphi^c - \sum_{1\le j\le M} z_{j,i}\lambda_j\beta_j}. \tag{6}$$

Denote $r_{i,j}^v$ as the transmission rate of a link $\langle v_i, v_j\rangle$. We model the connectivity of all APs as a graph G, whose vertex set and arc set are the set of all APs and the set of all links between APs, respectively. The weight of an arc $\langle v_i, v_j\rangle$ in G is the transmission delay of a bit data, i.e., $1/r_{i,j}^v$. Denote $\tau_{i,j}^p$ as the communication delay of a bit data from v_i to v_j, which is the length of the shortest path from v_i to v_j in G.

The total energy consumption of a user u_i is given as:

$$e_i = e_i^c\lambda_i(1 - \sum_{1\le j\le L} z_{i,j}) + e_i^t\lambda_i \sum_{1\le j\le L} z_{i,j}. \tag{7}$$

Denote χ_i as the index of an AP where a cloudlet c_i is deployed. The average response time of a task of user u_i is:

$$\tau_i = (1 - \sum_{1\le j\le L} z_{i,j})\tau_i^l + \sum_{1\le j\le L} z_{i,j}(\tau_i^t + \alpha_i\tau_{\vartheta_i,\chi_j}^p + \tau_j^c). \tag{8}$$

Before formulating MOCT, we explain some concepts related to multi-objective optimization problems [32]–[35]. A general form of multi-objective optimization problems can be described as:

$$\min \quad (f_1(Q), f_2(Q), \ldots, f_m(Q))^{\mathrm{T}} \tag{9}$$

subject to

$$Q \in \Omega \tag{10}$$

where $Q = (q_1, q_2, \ldots, q_n)^{\mathrm{T}}$ is the decision variable vector and Ω is the feasible solution space. A solution Q_1 dominates another solution Q_2 if $(f_1(Q_1), f_2(Q_1), \ldots, f_m(Q_1))^{\mathrm{T}} \neq (f_1(Q_2), f_2(Q_2), \ldots, f_m(Q_2))^{\mathrm{T}}$ and for each objective function f_i $(1 \le i \le m)$ $f_i(Q_1) \le f_i(Q_2)$ [32], [33]. A solution is Pareto optimal if it cannot be dominated by any other one. All Pareto optimal solutions and their objective vectors constitute the Pareto set and the Pareto front, repectively [32], [33].

The MOCT problem aims to select the optimal sites to deploy cloudlets and probabilities to offload tasks of users to cloudlets such that the average energy consumption, task response delay of users and the number of deployed cloudlets are minimized. Let $\chi_{i,j}'$ denote a binary variable that is 1 if c_i is deployed at v_j, and 0 otherwise. We formulate MOCT as the following MINP:

$$\min \quad \psi_1 = \frac{1}{M}\sum_{1\le i\le M} e_i \tag{11}$$

$$\min \quad \psi_2 = \frac{1}{M}\sum_{1\le i\le M} \tau_i \tag{12}$$

$$\min \quad \psi_3 = \sum_{1\le i\le L}\sum_{1\le j\le N} \chi_{i,j}' \tag{13}$$

subject to

$$\chi_{i,j}' \in \{0,1\}, \forall 1 \le i \le L, \forall 1 \le j \le N \tag{14}$$

$$0 \le z_{i,j} \le 1, \forall 1 \le i \le M, \forall 1 \le j \le L \tag{15}$$

$$\sum_{1\le j\le N} \chi_{i,j}' \le 1, \forall 1 \le i \le L \tag{16}$$

$$\sum_{1\le i\le L} \chi_{i,j}' \le 1, \forall 1 \le j \le N \tag{17}$$

$$\sum_{1\le j\le L} z_{i,j} \le 1, \forall 1 \le i \le M \tag{18}$$

$$\sum_{1\le j\le M} z_{j,i}\lambda_j\beta_j \le \mu, \forall 1 \le i \le L. \tag{19}$$

The objective functions (11) and (12) minimize the average energy consumption of all users and the average response delay of all tasks of all users, respectively, while the objective function (13) minimizes the number of deployed cloudlets. Since the energy and waiting time of users are very limited, it is very necessary to reduce their energy consumption and waiting time, which is embodied in (11) and (12). Constraints (14) and (15) define the domains of decision variables. Inequalities (16) and (17) guarantee that a cloudlet is deployed at no more than one site and that at most one cloudlet is deployed at each AP, respectively. Constraint (18) ensures that the total task offloading probability of each user is no more than 1. The maximum workload constraint of each cloudlet is expressed by inequality (19). The important notations used in the problem model are listed in Table I.

Theorem 1. *The MOCT problem is NP-complete.*

Proof. In order to prove the NP-completeness of MOCT, we first show that it belongs to the NP class and then reveal its NP-hardness.

Given a solution to MOCT, it takes $O(ML)$ time to verify both whether the sum of the task offloading probabilities of each user is no more than 1 and the maximum workload constraint of each cloudlet. In addition, it consumes $O(L)$ time to check whether the number of the cloudlets deployed at each site is no larger than one. Thus, to verify a solution to MOCT requires polynomial time. Therefore, MOCT belongs to the class of NP problems.

The NP-hardness of MOCT is proved by showing that the NP-hard p-median problem in [36] is a special case of MOCT. The p-median problem in [36] is to select p facilities from m $(m > p)$ potential ones into a subset such that the sum of the distances from users to their nearest facilities in the subset is minimized.

When $\forall 1 \le i \le M, \varphi_i^u = 0$, $\varphi^c = \infty$, and the price of each cloudlet is 0, the energy consumption of users is fixed, the network deployment cost can be ignored, and all L cloudlets

TABLE I
IMPORTANT NOTATIONS

Notation	Meaning
M	Number of users
N	Number of APs
u_i	User i
v_i	AP i
ϑ_i	Index of the associated AP of u_i
λ_i	Average task arrival rate of u_i
L	Maximum number of the cloudlets that can be deployed
c_i	Cloudlet i
α_i	Average input data size of a task of u_i (bits)
β_i	Average computational resource demand of a task of u_i (CPU cycles)
φ_i^u	Computing capacity of u_i (CPU cycles/second)
ξ_i	Effective switched capacitance of the CPU of u_i
e_i^c	Energy consumed by u_i for the local execution of its a task
p_i	Transmission power of u_i
B	System bandwidth
w_i	Available wireless channel bandwidth of u_i
$g_{i,j}$	Channel gain from u_i to v_j
σ^2	Background noise power
r_i^u	Transmission rate of u_i
e_i^t	Energy consumption required for transmitting a task of u_i to its associated AP
$z_{i,j}$	Ratio of the tasks of u_i offloaded to c_j
τ_i^l	Average waiting time for executing a task of u_i locally
τ_i^t	Average waiting time for transmitting a task of u_i to its associated AP
φ^c	Computing capacity of each cloudlet (CPU cycles/second)
μ	Maximum workload of each cloudlet (CPU cycles/second)
τ_i^c	Average waiting time for executing a task in c_i
$r_{i,j}^v$	Transmission rate of a link $\langle v_i, v_j \rangle$
$\tau_{i,j}^p$	Communication delay of a bit data from v_i to v_j
e_i	Total energy consumption of u_i
χ_i	Index of the AP where c_i is deployed
$\chi'_{i,j}$	A binary variable that is 1 if c_i is deployed at v_j, and 0 otherwise
τ_i	Average response time of a task of u_i

have to be deployed to minimize task response delay of users. Then, MOCT becomes the problem of selecting L APs from N ones to deploy L cloudlets and assigning each user to a cloudlet such that the sum of the communication delays from all users to the cloudlets to which they are assigned is minimized. To this end, each user has to be assigned to a cloudlet to which the communication delay from it is shortest. In this situation, MOCT becomes the p-median problem in [36]. Because the p-median problem is NP-hard [36], so is MOCT. Hence, MOCT is NP-complete. □

IV. HEURISTIC ALGORITHM

For an MECS service provider, it is hard to determine the constraints. If the constraints are inappropriate, no feasible solution can be found. If we address a constrained single-objective optimization problem rather than MOCT, only an objective is emphasized and only a solution can be generated, which may not make a satisfactory tradeoff among all the three objectives. By solving MOCT, a set of nondominated solutions can be produced. Therefore, a comprehensive insight into MECS design can be offered to MECS service providers, and thus it is highly useful for them to make an informed decision. Furthermore, no matter which objectives need to be stressed or what constraints are considered, the providers can select an appropriate solution from the set of nondominated solutions delivered by our proposed method.

Since MOCT is an NP-complete triple-objective optimization problem and also an MINP problem, it is infeasible for any exact solution methods to solve MOCT in a reasonable time by using scientific computing software, e.g., CPLEX. Therefore, an effective heuristic algorithm should be developed to solve MOCT. Fortunately, meta-heuristic algorithms are very powerful tools that can be adopted to solve MOCT. In this section, we propose a Modified Guided-population-archive Whale-optimizer-based cloudlet deployment and task offloading (called MGW for short) algorithm for MOCT.

A. Guided Population Archive Whale Optimization Algorithm

The whale optimization algorithm (WOA) [37] models the predatory behavior of a humpback whale swarm, and is a powerful tool for solving complex multidimensional single-objective optimization problems. In WOA [37], the positions of a number of whales are iteratively updated to search for a global optimal solution. There are three methods of updating a whale position, i.e., the prey encircling, spiral movement, and prey searching [37]. Denote H and K as the whale population size and the number of components of each whale position, respectively. Let $S_i^t = (s_{i,1}^t, s_{i,2}^t, \ldots, s_{i,K}^t)$ represent the position of the ith whale in the tth generation. Denote S^* as the best solution found so far. To update whale positions using the prey encircling and prey search, two factors $\delta = a(2\gamma - 1)$ and $\zeta = \varpi\gamma$ have to be first computed, where a is the distance coefficient, ϖ is the prey coefficient, and γ is a uniformly distributed random real number in the range of [0, 1]. If the prey encircling is used, the position of whale i is updated as [37]:

$$S_i^{t+1} = S^* - \delta|\zeta S^* - S_i^t|. \tag{20}$$

If the spiral movement is selected, the position of whale i is updated as [37]:

$$S_i^{t+1} = |S^* - S_i^t|e^{bl}cos(2\pi l) + S^*, \tag{21}$$

where b is the logarithmic spiral shape parameter and l is an evenly distributed random real number in the range of $[-1, 1]$. If the prey search is adopted, the position of whale i is updated as [37]:

$$S_i^{t+1} = S_{rand} - \delta|\zeta S_{rand} - S_i^t|, \tag{22}$$

where S_{rand} is the current position of a randomly selected whale.

To handle multi-objective optimization problems, a guided population archive whale optimization algorithm (GPAWOA) is presented in [32]. GPAWOA [32] extends WOA to deal with the problems with more than one optimization objective by making use of an external archive with a fixed capacity and the crowded-comparison operator in the nondominated sorting genetic algorithm II (NSGA-II) [38] to keep nondominated solutions. Specially, in each generation, after the positions of all whales are updated, the newly found nondominated solutions are used to update the archive. If the number of the nondominated solutions in the archive exceeds its capacity, those with the smallest crowding distances are removed. The method of computing the crowding distances is proposed in [38]. The crowding distance of a solution in a nondominated set is the sum of its elementary distances with respect to all objective functions. For each objective function, all solutions in the nondominated set have to be sorted by the value of this objective function first. For a solution, if the value of this objective function is maximum or minimum, its elementary distance regarding this objective function is set to the infinity; otherwise the absolute value of the difference between the normalized values of this objective function at its previous and next neighbors [38]. If the prey encircling or spiral movement is adopted to update a whale position, a leader S^* needs to be selected from the archive to guide the update.

B. Whale Position Representation

A position of whale i, i.e., S_i represents a solution for MOCT and is defined by a 3-tuple (X, Y, Z), where X is an L-vector of indexes of deployment sites of cloudlets, Y is an M-vector of workloads of users, and Z is an $M \times L$ matrix of task offloading probabilities of users. An element $S_i.X[j]$ of $S_i.X$ represents the index of the deployment site of a cloudlet c_j, and its lower bound is set to $2 - N$ so that c_j has the equal opportunities to be deployed and not deployed. If $S_i.X[j] \leq 0$, c_j is not deployed. An element $S_i.Y[j]$ of $S_i.Y$ denotes the workload of user u_j. An element $S_i.Z[j][k]$ of $S_i.Z$ stands for the probability of offloading a task of user u_j to cloudlet c_k.

During a search process, in a new position of whale i, $\lambda_j - S_i.Y[j]$ gives the upper bound of the total task offloading flow of user u_j. If only the task offloading probabilities of each user are updated, its total offloaded task flow is easy to be large, and thus its workload is easy to be small, which results in poor convergence and diversity of the obtained nondominated solution set. By means of updating both the workloads and the task offloading probabilities of users, the optimization ability of the proposed algorithm can be enhanced. Each component of $S_i.X$ is rounded to its nearest integer during the whale position updating. Under this representation, the length of a whale position is $K = L + M(L + 1)$.

C. Whale Position Initialization

In order to improve the convergence rate, each whale position is initialized with a feasible solution, which is produced by a random feasible solution construction algorithm (RFSCA). In RFSCA, after the number of the cloudlets to be deployed is randomly generated, the allocation of offloaded tasks of users is performed. For each user, RFSCA first randomly generates its total offloaded task flow, and then randomly allocates it to some cloudlets. During this allocation, it should be ensured that the workload of each cloudlet is not greater than its upper bound. At last, the cloudlets that have some workload are randomly deployed at some APs. During this deployment, it has to be guaranteed that at most one cloudlet is placed at an AP. In RFSCA, functions $rand()$ and $rint(i, j)$ generate an evenly distributed random real number in [0, 1] and a uniformly distributed random integer in $\{i, i+1, \ldots, j\}$, respectively.

RFSCA is illustrated in Algorithm 1. Herein, Lines 1-2 initialize the solution to be constructed. The number of the cloudlets to be placed is randomly produced at line 3. The while-loop at line 7 carries out the allocation of offloaded task flows to cloudlets for users until there is no usable cloudlet. The total offloaded task flow of a user is randomly generated at line 9. The while-loop at line 12 executes the offloaded task flow assignment for a single user, and the remainder is assigned by lines 24-34. A usable cloudlet is randomly chosen at line 13, and lines 15 and 16 randomly generate the offloaded task flow increment, whose computational resource demand cannot be more than the residual available computing capacity of the chosen cloudlet. Lines 17-19 update the task offloading probability matrix, residual offloaded task flow to be allocated, and workload of the chosen cloudlet, respectively. After the task offloading probabilities of all users are determined, their workloads are computed at line 36. The for-loop at line 39 deploys all the used cloudlets. Line 40 deploys an employed cloudlet at a randomly selected usable AP, and line 41 eliminates the AP where the cloudlet has been placed.

In RFSCA, the execution time of lines 1-6 is $O(ML)$. Since the while-loop at line 12 requires $O(L)$ time, it takes $O(ML)$ time to execute the while-loop at line 7. The for-loop at line 39 consumes $O(L)$ time. Therefore, the time complexity of RFSCA is $O(2ML + L) = O(ML)$.

D. Whale Position Repairing

In an invalid whale position, more than one cloudlet is deployed at an AP, the constraint of the maximum workload of each cloudlet is violated, or the sum of the offloaded task flows from a user to cloudlets exceeds the difference between the task arrival rate and workload of the user. During the search process of MGW, the invalid whale positions are repaired so as to speed up its convergence. During whale position repairing, the task offloading probabilities of users are first adjusted under the constraint of their total offloaded task flows determined by their workloads. Then, they are further modified subject to the maximum workload constraint of each cloudlet. After all task offloading probabilities of all users are determined, their workloads need to be adjusted such that the sum of the workload and total offloaded task flow of each of them is equal to its task arrival rate. Finally, cloudlet deployment is modified such that each cloudlet without any workload is not deployed and no two cloudlets are deployed at the same site.

A solution repairing (SR) algorithm is illustrated in Algorithm 2, where lines 2 and 3 restrict the cloudlet deployment and user workload components within their bounds. The values of the flows of the tasks offloaded to the cloudlets that are not deployed are cleared to zero at line 4. The for-loop at line 5 revises the task offloading probabilities of users based on their workloads. Line 6 computes the total task offloading probability for a user, and line 7 restricts its probabilities. The while-loop at line 11 finds the offloaded task flows from a given user that should be reduced, and lines 15 and 16 reduce

Algorithm 1 RFSCA

```
Input: $u_1, u_2, \ldots, u_M$; $v_1, v_2, \ldots, v_N$; $c_1, c_2, \ldots, c_L$; $\vartheta_1$, $\vartheta_2$, $\ldots$, $\vartheta_M$; $\lambda_1$, $\lambda_2$, $\ldots$, $\lambda_M$; $\alpha_1, \alpha_2, \ldots, \alpha_M$; $\beta_1, \beta_2, \ldots, \beta_M$; $\varphi_1^u, \varphi_2^u, \ldots, \varphi_M^u$; $B$; $\varphi^c$; $\mu$
Output: whale position $S$
1: $\forall 1 \le i \le L, S.X[i] \leftarrow 0$;
2: $\forall 1 \le i \le M, \forall 1 \le j \le L, S.Z[i][j] \leftarrow 0$;
3: $\eta \leftarrow rint(0, L)$;
4: $U \leftarrow \{i | 1 \le i \le M\}$;
5: $C \leftarrow \{i | 1 \le i \le \eta\}$;
6: $\forall 1 \le i \le \eta, W[i] \leftarrow 0$;
7: while $U \neq \emptyset \wedge C \neq \emptyset$ do
8:     $i \leftarrow U[rint(1, |U|)]$;
9:     $q \leftarrow rand()\lambda_i$;
10:    $f \leftarrow q$;
11:    $C' \leftarrow C$;
12:    while $f > 0 \wedge C' \neq \emptyset$ do
13:        $j \leftarrow C'[rint(1, |C'|)]$;
14:        $C' \leftarrow C' - \{j\}$;
15:        $f' \leftarrow \min\{f, rand()q\}$;
16:        $f' \leftarrow \min\{\mu - W[j], f'\beta_i\}/\beta_i$;
17:        $S.Z[i][j] \leftarrow S.Z[i][j] + f'/\lambda_i$;
18:        $f \leftarrow f - f'$;
19:        $W[j] \leftarrow W[j] + f'\beta_i$;
20:        if $W[j] \ge \mu$ then
21:            $C \leftarrow C - \{j\}$;
22:        end if
23:    end while
24:    if $C \neq \emptyset \wedge f > 0$ then
25:        $C' \leftarrow \{k | k \in C \wedge W[k] + f\beta_i \le \mu\}$;
26:        if $C' \neq \emptyset$ then
27:            $j \leftarrow C'[rint(1, |C'|)]$;
28:            $S.Z[i][j] \leftarrow S.Z[i][j] + f/\lambda_i$;
29:            $W[j] \leftarrow W[j] + f\beta_i$;
30:            if $W[j] \ge \mu$ then
31:                $C \leftarrow C - \{j\}$;
32:            end if
33:        end if
34:    end if
35: end while
36: $\forall 1 \le i \le M, S.Y[i] \leftarrow (1 - \sum_{1 \le j \le \eta} S.Z[i][j])\lambda_i$;
37: $A \leftarrow \{i | 1 \le i \le N\}$;
38: $C \leftarrow \{i | 1 \le i \le \eta \wedge W[i] > 0\}$;
39: for $i \in C$ do
40:     $S.X[i] \leftarrow A[rint(1, |A|)]$;
41:     $A \leftarrow A - \{S.X[i]\}$;
42: end for
43: return $S$;
```

Algorithm 2 SR

```
Input: whale position $S$
Output: whale position $S'$
1: $S' \leftarrow S$;
2: $\forall 1 \le i \le L, S'.X[i] \leftarrow \max\{\min\{S'.X[i], N\}, 2 - N\}$;
3: $\forall 1 \le i \le M, S'.Y[i] \leftarrow \max\{\min\{S'.Y[i], \lambda_i\}, 0\}$;
4: $\forall 1 \le i \le L \wedge S'.X[i] \le 0, \forall 1 \le j \le M, S'.Z[j][i] \leftarrow 0$;
5: for $i \leftarrow 1$ to $M$ do
6:     $\rho \leftarrow (\lambda_i - S'.Y[i])/\lambda_i$;
7:     $\forall 1 \le j \le L, S'.Z[i][j] \leftarrow \max\{\min\{S'.Z[i][j], \rho\}, 0\}$;
8:     if $\sum_{1 \le j \le L \wedge S'.X[j] > 0} S'.Z[i][j] > \rho$ then
9:         $C \leftarrow \{j | 1 \le j \le L \wedge S'.X[j] > 0 \wedge S'.Z[i][j] > 0\}$;
10:        $q \leftarrow 0$; $h \leftarrow 0$;
11:        while $q < \rho$ do
12:            $h \leftarrow C[rint(1, |C|)]$;
13:            $q \leftarrow q + S'.Z[i][h]$; $C \leftarrow C - \{h\}$;
14:        end while
15:        $S'.Z[i][h] \leftarrow S'.Z[i][h] - (q - \rho)$;
16:        $\forall j \in C, S'.Z[i][j] \leftarrow 0$;
17:    end if
18: end for
19: for $i \in \{i | 1 \le i \le L \wedge S'.X[i] > 0\}$ do
20:     $f \leftarrow \sum_{1 \le j \le M} S'.Z[j][i]\lambda_j\beta_j$;
21:     if $f > \mu$ then
22:         $U \leftarrow \{j | 1 \le j \le M \wedge S'.Z[j][i] > 0\}$;
23:         $q \leftarrow 0$; $h \leftarrow 0$;
24:         while $q < \mu$ do
25:             $h \leftarrow U[rint(1, |U|)]$;
26:             $q \leftarrow q + S'.Z[h][i]\lambda_h\beta_h$; $U \leftarrow U - \{h\}$;
27:         end while
28:         $S'.Z[h][i] \leftarrow S'.Z[h][i] - (q - \mu)/(\beta_h\lambda_h)$;
29:         $\forall j \in U, S'.Z[j][i] \leftarrow 0$;
30:     end if
31: end for
32: $\forall 1 \le i \le L, W[i] \leftarrow \sum_{1 \le j \le M} S'.Z[j][i]\lambda_j\beta_j$;
33: $\forall 1 \le i \le M, S'.Y[i] \leftarrow (1 - \sum_{1 \le j \le L} S'.Z[i][j])\lambda_i$;
34: $\forall 1 \le i \le L \wedge S'.X[i] > 0 \wedge W[i] \le 0, S'.X[i] \leftarrow 0$;
35: $A \leftarrow \emptyset$; $C \leftarrow \emptyset$; $C' \leftarrow \{i | 1 \le i \le L \wedge S'.X[i] > 0\}$;
36: while $C' \neq \emptyset$ do
37:     $i \leftarrow C'[rint(1, |C'|)]$; $C' \leftarrow C' - \{i\}$;
38:     if $S'.X[i] \notin A$ then
39:         $A \leftarrow A \cup \{S'.X[i]\}$; $C \leftarrow C \cup \{i\}$;
40:     end if
41: end while
42: $A' \leftarrow \{i | 1 \le i \le N \wedge i \notin A\}$;
43: $C'' \leftarrow \{i | 1 \le i \le L \wedge S'.X[i] > 0 \wedge i \notin C\}$;
44: while $C'' \neq \emptyset$ do
45:     $i \leftarrow C''[rint(1, |C''|)]$;
46:     $C'' \leftarrow C'' - \{i\}$;
47:     $S'.X[i] \leftarrow A'[rint(1, |A'|)]$;
48:     $A' \leftarrow A' - \{S'.X[i]\}$;
49: end while
50: return $S'$;
```

them. The violation of the maximum cloudlet workload is handled by the for-loop at line 19. The while-loop at line 24 finds the offloaded task flows to a given cloudlet that should be decreased, and lines 28 and 29 decrease them. The workloads of users are reset according to their final task offloading probabilities at line 33. Line 34 states that if no tasks are offloaded to a cloudlet, it does not need to be deployed. Lines 35-49 deal with the case that more than one cloudlet is placed at a site. Specially, the while-loop at line 36 determines the set of the cloudlets with valid deployment at random. The while-loop at line 44 places each cloudlet with invalid deployment at a randomly selected unused site.

In SR, it needs $O(ML)$ time to run lines 2-4. The while-loop at line 11 requires $O(L)$ time, and thus the for-loop at line 5 consumes $O(ML)$ time. The for-loop at line 19 consumes $O(ML)$ time, since the while-loop at line 24 takes $O(M)$ time. Lines 32-34 take $O(ML)$ time. It takes $O(L)$ time to execute the while-loop at line 36. The while-loop at line 44 consumes $O(L)$ time, because it has at most $O(L)$ iterations. Therefore, the time complexity of SR is $O(4ML + 2L)$=$O(ML)$.

E. Generalized and Quasi Opposition-based Learning

In order to avoid trapping into local optima and accelerate convergence, different from the application of the opposition-based learning (OBL) [39], [40] in [41], we make comprehensive use of the generalized opposition-based learning (GOBL) [42] and the quasi opposition-based learning (QOBL) [43], [44] to produce the opposites of whale positions, both of which are an extension of OBL [39], [40]. By introducing GOBL and QOBL, not only whale positions but also their opposites are used to update the archive. Therefore, the diversity of nondominated solutions stored in the archive can be improved, and the searching behavior of whales can be better guided.

In OBL [39], [40], GOBL [42], and QOBL [43], [44], the opposite numbers of a component q_i of a vector $Q = (q_1, q_2, \dots, q_n)^{\mathrm{T}}$ are respectively defined as follows:

$$\bar{q}_i^{o} = \check{q}_i + \hat{q}_i - q_i \tag{23}$$

$$\bar{q}_i^{go} = rand()(\check{q}_i + \hat{q}_i) - q_i \tag{24}$$

$$\bar{q}_i^{qo} = (\check{q}_i + \hat{q}_i)/2 + rand()(\bar{q}_i^{o} - (\check{q}_i + \hat{q}_i)/2) \tag{25}$$

where $\check{q}_i$ and $\hat{q}_i$ are the lower and upper bounds of q_i, respectively.

By combining QOBL and GOBL, we can obtain another opposite number of a component q_i, which is defined as

$$\bar{q}_i^{co} = (\check{q}_i + \hat{q}_i)/2 + rand()(\bar{q}_i^{go} - (\check{q}_i + \hat{q}_i)/2). \tag{26}$$

In the opposite $\bar{S}_i$ of a whale position S_i, the opposite number $\bar{S}_i.X[j]$ of $S_i.X[j]$ is calculated by GOBL, while the opposite numbers $\bar{S}_i.Y[j]$ and $\bar{S}_i.Z[j][k]$ of $S_i.Y[j]$ and $S_i.Z[j][k]$ are computed by the combination of QOBL and GOBL. Therefore, $\bar{S}_i.X[j] = 2rand() - S_i.X[j]$, $\bar{S}_i.Y[j] = \lambda_j/2 + rand()(rand()\lambda_j - S_i.Y[j] - \lambda_j/2)$, and $\bar{S}_i.Z[j][k] = 1/2 + rand()(rand() - S_i.Z[j][k] - 1/2)$.

F. MGW for MOCT

MGW utilizes a number of whales to search for the true Pareto front. After the whale positions are randomly initialized by RFSCA, their opposites are computed and repaired. Both the initial positions of whales and their opposites are employed to initialize the archive.

In order to improve the exploration and exploitation abilities of whales, the differential mutation and crossover in differential evolution (DE) [45]–[48] are employed to modify the whale foraging behavior. If the DE operations are selected to update a whale position, in order to balance exploration and exploitation, three mutually different solutions $\mathbb{S}$, $\mathbb{S}'$, and $\mathbb{S}''$ or two ones $\mathbb{S}$ and $\mathbb{S}'$ need to be randomly selected from the archive before the updating. The probabilities of selecting the three and two ones are equal. The crossover [45] is performed on the mutant result and $\mathbb{S}$. A random component of the whale position is updated using the differential mutation definitely. For each of the other components k of the whale position, it is updated using the differential mutation with the probability of CR, and equals $\mathbb{S}[k]$ with the probability of $1-CR$. Denote F as the scaling factor. When the differential mutation is chosen, if $\mathbb{S}$, $\mathbb{S}'$, and $\mathbb{S}''$ are selected, a whale position component $S_i[j]$ is updated as [45]:

$$S_i^{t+1}[j] = \mathbb{S}[j] + F(\mathbb{S}'[j] - \mathbb{S}''[j]); \tag{27}$$

otherwise, it is updated as:

$$S_i^{t+1}[j] = \mathbb{S}[j] + F(\mathbb{S}'[j] - \mathbb{S}[j]). \tag{28}$$

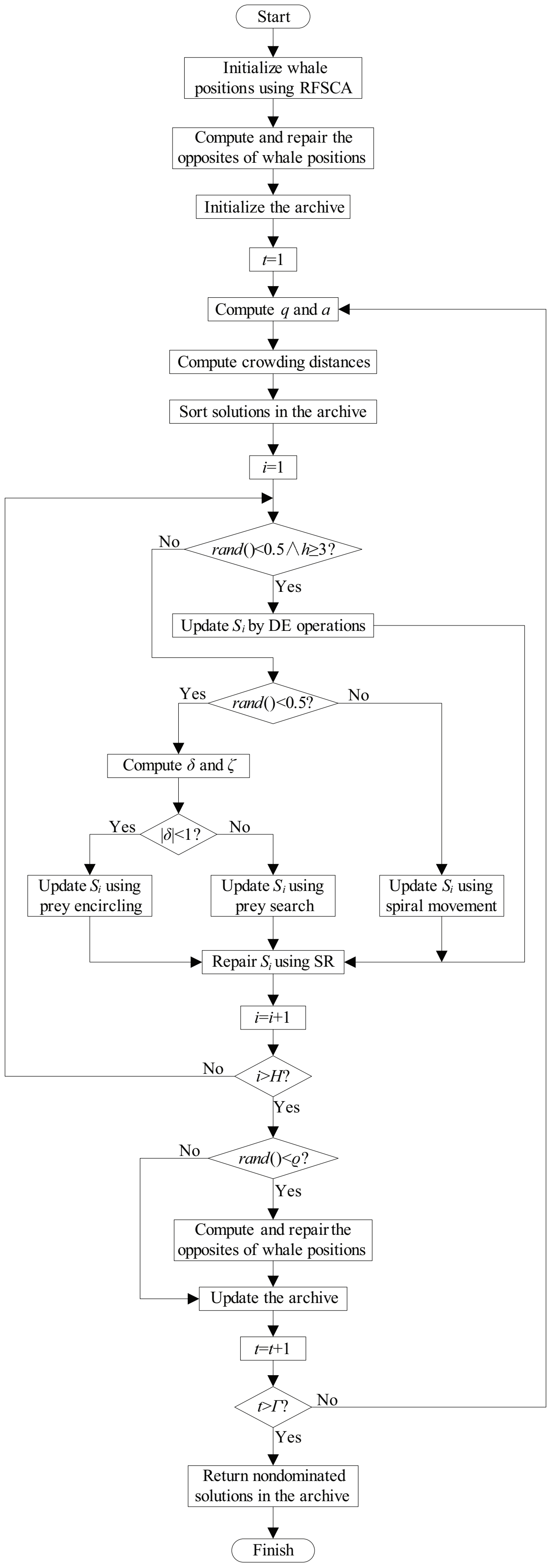

Start
Initialize whale positions using RFSCA
Compute and repair the opposites of whale positions
Initialize the archive
t=1
Compute q and a
Compute crowding distances
Sort solutions in the archive
i=1
No
rand()<0.5∧h≥3?
Yes
Update S_i by DE operations
Yes
rand()<0.5?
No
Compute δ and ζ
Yes
|δ|<1?
No
Update S_i using prey encircling
Update S_i using prey search
Update S_i using spiral movement
Repair S_i using SR
i=i+1
No
i>H?
Yes
No
rand()<ϱ?
Yes
Compute and repair the opposites of whale positions
Update the archive
t=t+1
t>T?
No
Yes
Return nondominated solutions in the archive
Finish

Fig. 2. Flowchart of MGW.

During the search process, each whale position is updated with the prey encircling, spiral movement, prey searching, or DE operations. If the first two methods are adopted, a leader S^* has to be randomly selected from a number of the solutions with the biggest crowding distances in the archive first, whose proportion is denoted by ς. A random whale ought to be first selected from the current population when the prey searching is used. If the DE operations are chosen, three or two mutually different solutions should be first selected from the archive at random. If a new position is invalid, it is repaired by SR. After the positions of all whales are updated, they are used to update the archive. In each generation, the opposites of the whale positions are computed with the probability of ϱ, which are also used to update the archive. The new archive contains the nondominated solutions identified from the whale positions, their opposites, and the solutions in the old archive. If the size of the new archive exceeds its capacity, some solutions with the smallest crowding distances are removed. Denote Υ and Γ as the archive capacity and the number of generations, respectively. In each generation t, $a = 2(1 - ((t-1)/\Gamma)^3)$. MGW is summarized in Algorithm 3, and its flowchart is shown in Fig. 2.

In MGW, line 1 consumes $O(MLH)$ time. Both for-loops at lines 2 and 11 take $O(MLH)$ time. Both lines 9 and 10 require $O(\Upsilon log_2\Upsilon)$ time. Because the crowding distance assignment requires $O((H+\Upsilon)log_2(H+\Upsilon))$ time [38] and it takes $O(H(H+\Upsilon))$ time to identify the nondominated solutions from the archive and the whale population, the archive updating at lines 42 and 44 consumes $O((H+\Upsilon)(H+log_2(H+\Upsilon)))$ time. As a result, it takes $O(\Gamma(\Upsilon log_2\Upsilon+MLH+(H+\Upsilon)(H+log_2(H+\Upsilon)))) = O(\Gamma(MLH+(H+\Upsilon)(H+log_2(H+\Upsilon))))$ time to run the for-loop at line 7. Therefore, the time complexity of MGW is $O(\Gamma(MLH+(H+\Upsilon)(H+log_2(H+\Upsilon))))$.

Algorithm 3 MGW

```
Input: $u_1, u_2, \ldots, u_M$; $v_1, v_2, \ldots, v_N$; $c_1, c_2, \ldots, c_L$; $\vartheta_1, \vartheta_2, \ldots, \vartheta_M$; $\lambda_1, \lambda_2, \ldots, \lambda_M$; $\alpha_1, \alpha_2, \ldots, \alpha_M$; $\beta_1, \beta_2, \ldots, \beta_M$; $\varphi_1^u, \varphi_2^u, \ldots, \varphi_M^u$; $B$; $\varphi^c$; $\mu$
Output: Set of nondominated solutions
1: $\forall 1 \le i \le H, S_i \leftarrow RFSCA()$;
2: for $i \leftarrow 1$ to $H$ do
3:   Compute the opposite $\bar{S}_i$ of $S_i$;
4:   $\bar{S}_i \leftarrow SR(\bar{S}_i)$;
5: end for
6: Initialize the archive using $\{S_i|1 \le i \le H\} \cup \{\bar{S}_i|1 \le i \le H\}$;
7: for $t \leftarrow 1$ to $\Gamma$ do
8:   $q \leftarrow \lceil \varsigma h \rceil$, where $h$ is the archive size; $a \leftarrow 2(1 - ((t-1)/\Gamma)^3)$;
9:   Compute the crowding distances of the solutions in the archive;
10:  Sort the solutions in the archive in descending order by their crowding distances;
11:  for $i \leftarrow 1$ to $H$ do
12:    if $rand() < 0.5$ and $h \ge 3$ then
13:      if $rand() < 0.5$ then
14:        Select three mutually different solutions $\mathbb{S}$, $\mathbb{S}'$, and $\mathbb{S}''$ from the archive randomly;
15:        Update $S_i$ using $\mathbb{S}$, $\mathbb{S}'$, $\mathbb{S}''$, (27), and crossover;
16:      else
17:        Select two different solutions $\mathbb{S}$ and $\mathbb{S}'$ from the archive randomly;
18:        Update $S_i$ using $\mathbb{S}$, $\mathbb{S}'$, (28), and crossover;
19:      end if
20:    else
21:      if $rand() < 0.5$ then
22:        Compute $\delta$ and $\zeta$;
23:        if $|\delta| < 1$ then
24:          Select a leader $S^*$ from the first $q$ solutions in the sorted archive at random;
25:          Update $S_i$ using $S^*$ and (20);
26:        else
27:          Choose a whale $j$ randomly;
28:          Update $S_i$ using $S_j$ and (22);
29:        end if
30:      else
31:        Select a leader $S^*$ from the $q$ solutions with the biggest crowding distances in the archive at random;
32:        Update $S_i$ using $S^*$ and (21);
33:      end if
34:    end if
35:    $S_i \leftarrow SR(S_i)$;
36:  end for
37:  if $rand() < \varrho$ then
38:    for $i \leftarrow 1$ to $H$ do
39:      Compute the opposite $\bar{S}_i$ of $S_i$;
40:      $\bar{S}_i \leftarrow SR(\bar{S}_i)$;
41:    end for
42:    Update the archive using $\{S_i|1 \le i \le H\} \cup \{\bar{S}_i|1 \le i \le H\}$;
43:  else
44:    Update the archive using $\{S_i|1 \le i \le H\}$;
45:  end if
46: end for
47: return nondominated solutions in the archive;
```

V. SIMULATION RESULTS

A. Comparison and Metrics

In this section, we compare our proposed MGW with NSGA-II [38], GPAWOA [32], multiobjective differential evolution (MODE) [49], multi-objective grey wolf optimizer (MOGWO) [50], and multiobjective particle swarm optimization (MOPSO) [33] adapted to handle MOCT. In them, the approaches of encoding, initializing, and repairing individuals are the same as those in MGW. NSGA-II [38], MOPSO [33], and MODE [49] are classic and commonly used methods, while GPAWOA [32] and MOGWO [50] are very recent ones. The comparison is in terms of the inverted generational distance (IGD) [51]–[53] and hypervolume (HV) [54], which are the two most effective performance indicators to evaluate the accuracy of a nondominated solution set. The IGD of a nondominated solution set S is defined as the average distance between each point in the true Pareto front and its nearest point in the Pareto front of S. The HV of S measures how much objective space S covers. Lower IGD or larger HV of S indicates that it has higher quality. Since the true Pareto front is unknown, the Pareto front extracted from the results of all the six algorithms is used as the substitute of the true Pareto front to compute IGD. Let $\check{\psi}_i$ and $\hat{\psi}_i$ $(1 \le i \le 2)$ denote the minimum and maximum values of the ith objective of the results of all the six algorithms, respectively. To compute IGD and HV, the objective values $\psi_1(S)$, $\psi_2(S)$, and $\psi_3(S)$ of each solution S are normalized as $\psi_1'(S) = (\hat{\psi}_1 - \psi_1(S))/(\hat{\psi}_1 - \check{\psi}_1)$, $\psi_2'(S) = (\hat{\psi}_2 - \psi_2(S))/(\hat{\psi}_2 - \check{\psi}_2)$, and $\psi_3'(S) = (L - \psi_3(S))/L$, respectively. For computing HV, the reference point is chosen to be $(-1/(\iota-1), -1/(\iota-1), -1/(\iota-1))$, where ι is the number of all different network deployment costs in the results of all the six algorithms, according to the recommendation in [55]. We use Java to develop the simulation program on MyEclipse 2014 and run it on a Lenovo ThinkPad T450s, which has 2.20 GHz CPU and 4 GB physical RAM.

B. Parameter Settings

All APs and users are located in a 10000 m $\times$ 10000 m area, which is divided into $N \times N$ grids. There is an AP in the center of each grid. Each user is randomly connected to an AP, and the connectivity graph of all APs is randomly generated. The position of each user is randomly generated, and the distance between each user and its associated AP is no more than $5000/N$ m. The average input data size and computational resource demand per bit data of a task of each user are two random values in the ranges of [200×1024, 500×1024] bytes and [50, 100] cycles/bit [30], respectively. The average task arrival rate of each user is randomly drawn from [0.1, 3] tasks/second. The effective switched capacitance of CPU, computing capacity, and transmission power of each user are set to 5×10^{-27} [16], 2 GHz, and 0.1 W, respectively. The system bandwidth is set to 40 MHz, and the channel gain from a user to its associated AP is set to d^{-4}, where d is the distance between them. The maximum workload μ of each cloudlet is set to $0.9\varphi^c$. The background noise power is set to -100 dBm, and the link transmission rate between two APs is randomly generated in [10^8, 2×10^8] bits/second.

In NSGA-II, the simulated binary crossover and polynomial mutation are used. The crossover and mutation probabilities and the distribution indexes for crossover and mutation are set to 0.9, $1/(L + M(L + 1))$, 20, and 20 following [38], respectively. In MOPSO, the mutation rate is set to 0.5 following [33]. In both MOPSO and MOGWO, the number of the subdivisions for each dimension in the grid mechanism is set to 30 [33]. The parameters' setting of GPAWOA is the same as that in [32]. In MGW, b, ς, ϖ, and ϱ are set to 3, $3/\Upsilon$, 3, and 0.15, respectively. In both MODE and MGW, the scaling factor and crossover probability are set to 0.5 and 0.9, respectively. In MOPSO, MOGWO, GPAWOA, and MGW, the archive capacity is set to 100. In all algorithms, the population size and the number of generations are set to 100 and 2000, respectively.

C. Results and Discussion

In the simulations, we consider different M, L, N, and φ^c. For each setting of the parameters, 30 instances are randomly generated. IGDs and HVs of the results of each algorithm for the 30 instances are averaged. Tables II - V display the simulation results. From them, we can see that the performance of MGW with respect to IGD and HV is better than that of other algorithms. This means that MGW can produce a better nondominated solution set than the others. In other words, the Pareto front obtained by MGW is closest to the true Pareto front. Meanwhile, the running time of MGW is only longer than that of GPAWOA and MOGWO. From Tables II and III, we can observe that the running time of the algorithms increases with the growth of the upper bound of the budget or the number of users. The reason is that larger upper bound of the budget or more users result in a bigger problem size. As shown in Table V, with the increase of the cloudlet computing capacity, IGD of the result of MGW decreases and its HV increases. This is due to the fact that larger cloudlet computing capacity permits shorter task response delay and less energy consumption of users and fewer deployed cloudlets, and thus makes the normalized objective values of solutions increase.

TABLE II
Performance comparison with M=240, N= 120, and φ^c=25 GHz under different upper bound of the budget

L	Method	IGD	HV	Time (s)
30	NSGA-II	0.2329	0.5762	235.65
	MOPSO	0.2098	0.6384	158.13
	MOGWO	0.2432	0.5565	64.9
	MODE	0.0454	0.9381	109.2
	GPAWOA	0.2281	0.5718	49.12
	MGW	0.0267	0.9423	98.55
35	NSGA-II	0.2393	0.5696	286.95
	MOPSO	0.223	0.6059	203.46
	MOGWO	0.2575	0.5366	88.27
	MODE	0.0446	0.9294	138.02
	GPAWOA	0.2335	0.5671	58.58
	MGW	0.0241	0.9405	122.25
40	NSGA-II	0.2321	0.5618	312.49
	MOPSO	0.2201	0.6023	220.6
	MOGWO	0.2483	0.5366	89.04
	MODE	0.04	0.9252	142.95
	GPAWOA	0.2243	0.5608	60.38
	MGW	0.0253	0.9401	129.26
45	NSGA-II	0.2475	0.5585	340.05
	MOPSO	0.2508	0.5731	242.51
	MOGWO	0.2628	0.5294	93.91
	MODE	0.048	0.9263	150.87
	GPAWOA	0.2314	0.5602	65.09
	MGW	0.029	0.9326	138.91
50	NSGA-II	0.2426	0.5581	377.98
	MOPSO	0.2504	0.5702	273.89
	MOGWO	0.2593	0.5244	103.72
	MODE	0.0487	0.9248	166.29
	GPAWOA	0.2379	0.548	73.25
	MGW	0.0311	0.9287	154.35

The Pareto fronts obtained by the algorithms for two random instances are shown in Figs. 3 and 4, respectively. For each instance, each algorithm is run 30 times independently, and the 100 best nondominated solutions are selected from the results of the 30 runs using the crowded-comparison operator in [38]. In Fig.3, the IGDs of the Pareto fronts delivered by MOPSO, MOGWO, NSGA-II, MODE, GPAWOA, and MGW are 0.1457, 0.2024, 0.1978, 0.0265, 0.1875, and 0.0211, respectively, and their HVs are 0.7216, 0.5712, 0.6033, 0.9336, 0.5842, and 0.9444, respectively. In Fig. 4, the IGDs of the Pareto fronts obtained by MOPSO, MOGWO, NSGA-II, MODE, GPAWOA, and MGW are 0.2444, 0.2356, 0.2242, 0.0292, 0.2216, and 0.0104, respectively, and their HVs are 0.562, 0.5428, 0.5689, 0.9359, 0.5561, and 0.9472, respectively. From the figures, we can observe that the Pareto front obtained by MGW has better accuracy and diversity than those obtained by other algorithms, because it is distributed over a larger area and covers more objective space. This is because the effective combination of GPAWOA, GOBL,

TABLE III
Performance comparison with $L = 40$, $N = 120$, and φ^c=25 GHz under different Number of users

M	Method	IGD	HV	Time (s)
200	NSGA-II	0.2637	0.569	250.9
	MOPSO	0.2443	0.6098	173.53
	MOGWO	0.282	0.538	66.5
	MODE	0.0435	0.9629	105.8
	GPAWOA	0.2586	0.558	41.61
	MGW	0.0211	0.9703	97.18
220	NSGA-II	0.252	0.5699	286.94
	MOPSO	0.2489	0.5858	200.39
	MOGWO	0.2729	0.5393	79.2
	MODE	0.0453	0.947	123.29
	GPAWOA	0.2454	0.5598	50.95
	MGW	0.0254	0.9519	112.67
240	NSGA-II	0.2295	0.5609	313.5
	MOPSO	0.2167	0.6014	218.94
	MOGWO	0.2493	0.5353	87.64
	MODE	0.0453	0.9152	137.64
	GPAWOA	0.2177	0.5602	54.94
	MGW	0.0257	0.9331	123.8
260	NSGA-II	0.2273	0.563	328.06
	MOPSO	0.2274	0.5899	225.77
	MOGWO	0.2456	0.5323	90.57
	MODE	0.0526	0.9073	140.81
	GPAWOA	0.2191	0.5579	58.11
	MGW	0.0317	0.922	129.77
280	NSGA-II	0.2181	0.5499	357.14
	MOPSO	0.2219	0.5615	248.3
	MOGWO	0.2295	0.5295	98.0
	MODE	0.0542	0.884	156.22
	GPAWOA	0.2078	0.5498	63.93
	MGW	0.0358	0.8905	144.3

TABLE IV
Performance comparison with M=240, L= 40, and φ^c=25 GHz under different number of APs

N	Method	IGD	HV	Time (s)
100	NSGA-II	0.2412	0.5601	286.13
	MOPSO	0.2495	0.5652	213.9
	MOGWO	0.2574	0.5363	80.86
	MODE	0.042	0.9253	131.5
	GPAWOA	0.2363	0.5577	52.29
	MGW	0.0289	0.9279	117.59
110	NSGA-II	0.2351	0.5619	287.29
	MOPSO	0.2338	0.585	210.61
	MOGWO	0.2512	0.5393	81.38
	MODE	0.0469	0.9212	132.92
	GPAWOA	0.229	0.5549	51.52
	MGW	0.0275	0.9349	118.1
120	NSGA-II	0.2427	0.562	286.53
	MOPSO	0.2366	0.6018	207.81
	MOGWO	0.26	0.535	79.81
	MODE	0.044	0.9318	131.04
	GPAWOA	0.2385	0.5566	50.17
	MGW	0.028	0.9387	115.69
130	NSGA-II	0.2401	0.5665	285.42
	MOPSO	0.2405	0.5886	210.01
	MOGWO	0.251	0.5412	80.27
	MODE	0.0452	0.9313	132.96
	GPAWOA	0.2319	0.559	51.51
	MGW	0.0265	0.9386	118.77
140	NSGA-II	0.2387	0.5634	291.43
	MOPSO	0.2335	0.5898	213.01
	MOGWO	0.2547	0.5335	83.34
	MODE	0.0426	0.9313	136.59
	GPAWOA	0.2297	0.5561	53.33
	MGW	0.0245	0.941	119.7

TABLE V
Performance comparison with M=240, L= 40, and N= 120 under different cloudlet computing capacity (GHz)

φ^c (GHz)	Method	IGD	HV	Time (s)
15	NSGA-II	0.1796	0.543	292.93
	MOPSO	0.2191	0.5047	216.99
	MOGWO	0.2045	0.5102	88.04
	MODE	0.0578	0.8192	143.87
	GPAWOA	0.1712	0.5426	56.05
	MGW	0.0466	0.8266	126.59
20	NSGA-II	0.2071	0.5558	292.94
	MOPSO	0.2184	0.5729	217.78
	MOGWO	0.2321	0.5276	85.19
	MODE	0.0539	0.8697	140.69
	GPAWOA	0.2038	0.5561	55.04
	MGW	0.0422	0.8736	124.55
25	NSGA-II	0.2376	0.5555	291.9
	MOPSO	0.2367	0.5871	214.85
	MOGWO	0.2606	0.5238	83.97
	MODE	0.0489	0.9187	135.18
	GPAWOA	0.2322	0.5535	53.79
	MGW	0.026	0.9328	118.35
30	NSGA-II	0.2579	0.568	303.19
	MOPSO	0.248	0.6002	227.54
	MOGWO	0.2827	0.5319	92.0
	MODE	0.0378	0.9643	146.02
	GPAWOA	0.2541	0.557	56.98
	MGW	0.0177	0.9759	125.62
35	NSGA-II	0.2521	0.5612	294.94
	MOPSO	0.2279	0.6148	216.43
	MOGWO	0.2762	0.5251	84.68
	MODE	0.0359	0.9838	139.46
	GPAWOA	0.2492	0.541	53.08
	MGW	0.0155	0.9965	120.11

QOBL, and DE enables MGW to have stronger exploration and exploitation abilities. From the figures, it can be observed that by and large the average task response delay of users and the network deployment cost grow with the reduction of the average energy consumption of users, and the average task response delay of users declines with the increase of the number of deployed cloudlets. This observation verifies that there exist conflicts among the objectives of MOCT. As shown in the figures, MGW can deliver solutions with lower energy consumption and task response delay of users and number of deployed cloudlets. This means that MGW can better optimize user QoE and network deployment cost, which can also be verified by the higher accuracy of MGW revealed by the tables.

VI. Conclusion

In this work, the problem of joint cloudlet deployment and task offloading is addressed. Its objectives are to minimize the energy consumption and task response delay of users and the number of deployed cloudlets. This optimization problem is intractable and formulated as a mixed integer nonlinear program. To deal with this NP-complete problem, a modified guided population archive whale optimization algorithm is presented, and its time complexity is analyzed as well. In this algorithm, the effective methods of encoding, initializing, and repairing a whale position are designed. Besides, the generalized and quasi opposition-based learning and the mutation and crossover in differential evolution are adopted to improve its optimization ability. Compared with other meta-heuristic al-

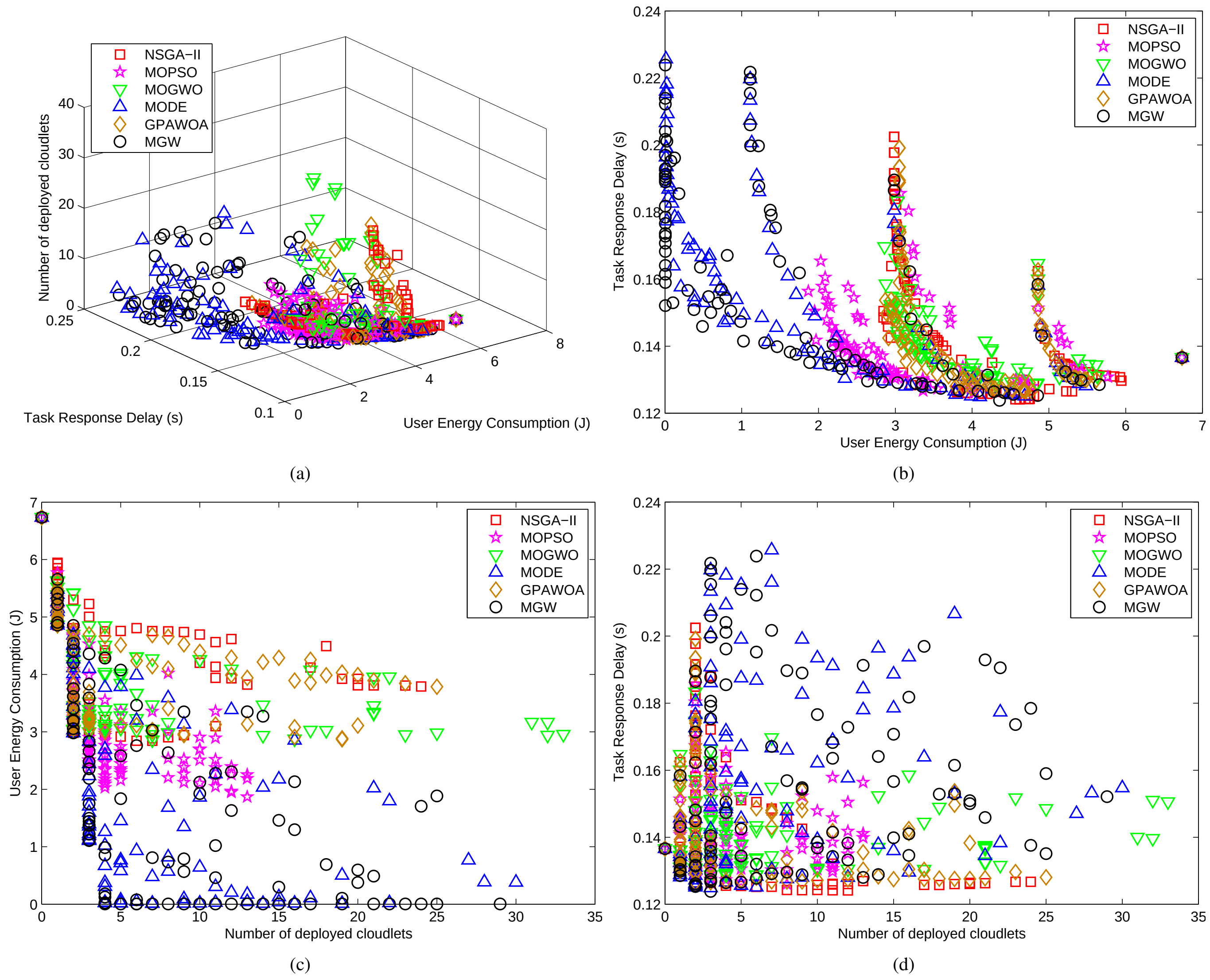


Fig. 3. Pareto fronts obtained by different algorithms for a random instance with M=240, N=120, L=40, and φ^c=25 GHz. (a) Visualization with three objectives. (b) Visualization with ψ_1 and ψ_2. (c) Visualization with ψ_1 and ψ_3. (d) Visualization with ψ_2 and ψ_3.

gorithms, the proposed algorithm can deliver a nondominated solution set with higher quality with competitive efficiency.

In this work, we investigate how to optimize the deployment sites of cloudlets. Other system parameters are known. Because the cloudlet deployment objectives depend on a task offloading scheme, the task offloading decisions have to be considered during cloudlet deployment optimization. According its actual demand, an MECS service provider can choose an appropriate cloudlet deployment scheme from the set of nondominated solutions delivered by our proposed method to make a satisfactory tradeoff among all the three objectives. Our future work includes designing a scheme to optimize channel bandwidth allocation to users.

References

[1] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, “Edge computing: A survey,” *Future Generation Computer Systems*, vol. 97, pp. 219 – 235, 2019.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[4] P. Zhang, M. Zhou, and G. Fortino, “Security and trust issues in fog computing: A survey,” *Future Generation Computer Systems*, vol. 88, pp. 16 – 27, 2018.

[5] Q. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, “Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach,” *IEEE Access*, vol. 6, pp. 75 868–75 885, 2018.

[6] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, “The role of edge computing in internet of things,” *IEEE Communications Magazine*, vol. 56, no. 11, pp. 110–115, 2018.

[7] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A survey of computation offloading for mobile systems,” *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[8] T. Mekonnen, M. Komu, R. Morabito, T. Kauppinen, E. Harjula, T. Koskela, and M. Ylianttila, “Energy consumption analysis of edge orchestrated virtualized wireless multimedia sensor networks,” *IEEE Access*, vol. 6, pp. 5090–5100, 2018.

[9] K. Intharawijitr, K. Iida, H. Koga, and K. Yamaoka, “Practical enhancement and evaluation of a low-latency network model using mobile edge computing,” in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2017, pp. 567–574.

[10] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, “Hybrid method

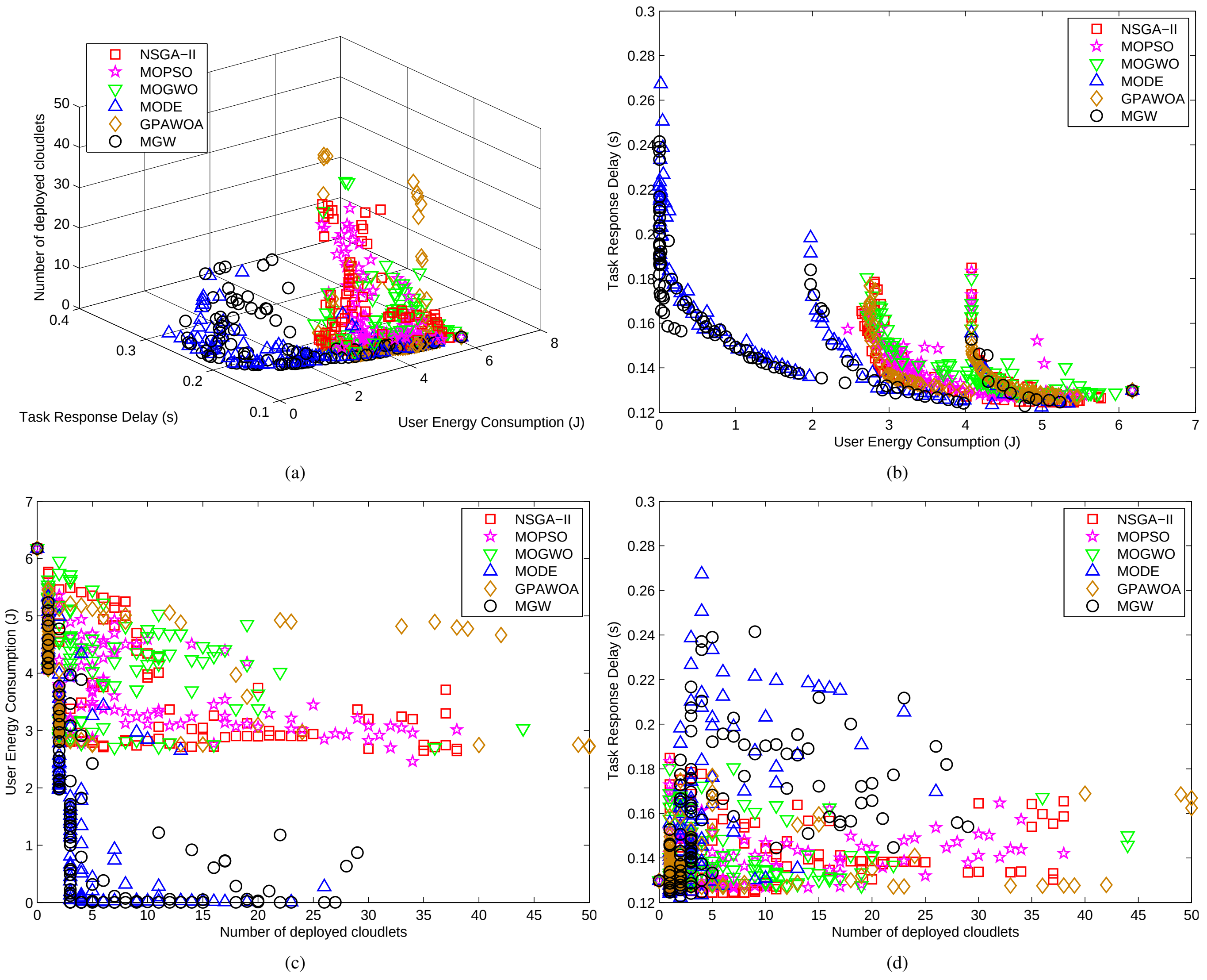


Fig. 4. Pareto fronts obtained by different algorithms for a random instance with M=300, N=150, L=50, and φ^c=35 GHz. (a) Visualization with three objectives. (b) Visualization with ψ_1 and ψ_2. (c) Visualization with ψ_1 and ψ_3. (d) Visualization with ψ_2 and ψ_3.

for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[11] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, 2016.

[12] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017.

[13] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[14] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 2018, pp. 1–6.

[15] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultradense IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4317–4329, 2019.

[16] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[17] A. Khalili, S. Zarandi, and M. Rasti, "Joint resource allocation and offloading decision in mobile edge computing," *IEEE Communications Letters*, vol. 23, no. 4, pp. 684–687, 2019.

[18] E. El Haber, T. M. Nguyen, and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3407–3421, 2019.

[19] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853–5863, 2019.

[20] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[21] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4791–4803, 2019.

[22] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2019.

[23] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Communications Letters*, vol. 22, no. 4, pp. 820–823, 2018.

[24] ——, "Towards workload balancing in fog computing empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, 2020.

[25] ——, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153,

2018.

[26] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160 – 168, 2019.

[27] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 926–937, 2019.

[28] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, "Efficient placement of edge computing devices for vehicular applications in smart cities," in *2018 IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, 2018, pp. 1–9.

[29] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2018.

[30] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[31] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 2013, pp. 494–502.

[32] A. Got, A. Moussaoui, and D. Zouache, "A guided population archive whale optimization algorithm for solving multiobjective optimization problems," *Expert Systems with Applications*, vol. 141, p. 112972, 2020.

[33] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[34] Y. Feng, M. Zhou, G. Tian, Z. Li, Z. Zhang, Q. Zhang, and J. Tan, "Target disassembly sequencing and scheme evaluation for CNC machine tools using improved multiobjective ant colony algorithm and fuzzy integral," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2438–2451, 2019.

[35] P. Wu, A. Che, F. Chu, and M. Zhou, "An improved exact ε-constraint and cut-and-solve combined method for biobjective robust lane reservation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1479–1492, 2015.

[36] Z. Ren, H. Jiang, J. Xuan, and Z. Luo, "An accelerated-limit-crossing-based multilevel algorithm for the p-median problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1187–1202, 2012.

[37] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51 – 67, 2016.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 1, 2005, pp. 695–701.

[40] Q. Kang, C. Xiong, M. Zhou, and L. Meng, "Opposition-based hybrid strategy for particle swarm optimization in noisy environments," *IEEE Access*, vol. 6, pp. 21 888–21 900, 2018.

[41] W. L. Wang, W. K. Li, Z. Wang, and L. Li, "Opposition-based multi-objective whale optimization algorithm with global grid ranking," *Neurocomputing*, vol. 341, pp. 41 – 59, 2019.

[42] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699 – 4714, 2011.

[43] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Quasi-oppositional differential evolution," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 2229–2236.

[44] K. H. Truong, P. Nallagownden, Z. Baharudin, and D. N. Vo, "A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems," *Applied Soft Computing*, vol. 77, pp. 567 – 583, 2019.

[45] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[46] P. Zhang, M. Zhou, and X. Wang, "An intelligent optimization method for optimal virtual machine allocation in cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1725–1735, 2020.

[47] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to be published, doi: 10.1109/TSMC.2019.2956121.

[48] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 233–246, 2020.

[49] U. K. Sikdar, A. Ekbal, and S. Saha, "MODE: multiobjective differential evolution for feature selection and classifier ensemble," *Soft Computing*, vol. 19, pp. 3529–3549, 2015.

[50] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. dos S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Systems with Applications*, vol. 47, pp. 106 – 119, 2016.

[51] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

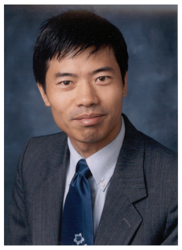
[52] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1944–1955, 2019.

[53] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, "Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, 2019.

[54] L. M. S. Russo and A. P. Francisco, "Quick hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 481–502, 2014.

[55] Y. Cao, B. J. Smucker, and T. J. Robinson, "On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design," *Journal of Statistical Planning and Inference*, vol. 160, pp. 60 – 74, 2015.

Xiaojian Zhu received the Ph.D. degree in Computer Science from Southeast University, Nanjing, China, in 2014. Currently, he is an Assistant Professor with the School of Computer Science and Engineering, Changshu Institute of Technology, Changshu, Jiangsu, China. His current research interests include mobile edge computing, wireless sensor networks, machine learning, green communications, and Internet of Things.



MengChu Zhou (Fellow, IEEE) received his B.S. degree in Control Engineering from Nanjing University of Science and Technology, Nanjing, China in 1983, M.S. degree in Automatic Control from Beijing Institute of Technology, Beijing, China in 1986, and Ph. D. degree in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY in 1990. He joined New Jersey Institute of Technology (NJIT), Newark, NJ in 1990, and is now Distinguished Professor in Electrical and Computer Engineering. His research interests are in Petri nets, intelligent automation, Internet of Things, big data, web services, and intelligent transportation. He has over 900 publications including 12 books, 600+ journal papers (500+ in IEEE transactions), 29 patents and 29 book-chapters. He is the founding Editor of IEEE Press Book Series on Systems Science and Engineering, Editor-in-Chief of IEEE/CAA Journal of Automatica Sinica, and Associate Editor of IEEE Internet of Things Journal, IEEE Transactions on Intelligent Transportation Systems, and IEEE Transactions on Systems, Man, and Cybernetics: Systems. He is a recipient of Humboldt Research Award for US Senior Scientists from Alexander von Humboldt Foundation, Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society, Excellence in Research Prize and Medal from NJIT, and Edison Patent Award from the Research & Development Council of New Jersey. He is a life member of Chinese Association for Science and Technology-USA and served as its President in 1999. He is a Fellow of International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation (CAA) and National Academy of Inventors (NAI).