A Bilevel Optimization Approach for Joint Offloading Decision and Resource Allocation in Cooperative Mobile Edge Computing

Pei-Qiu Huang, Yong Wang, *Senior Member, IEEE*, Kezhi Wang, *Member, IEEE*, and Zhi-Zhong Liu

***Abstract*—This paper studies a multi-user cooperative mobile edge computing offloading (CoMECO) system in a multi-user interference environment, in which delay-sensitive tasks may be executed on local devices, cooperative devices, or the primary MEC server. In this system, we jointly optimize the offloading decision and computation resource allocation for minimizing the total energy consumption of all mobile users under the delay constraint. If this problem is solved directly, the offloading decision and computation resource allocation are generally generated separately at the same time. Note, however, that they are closely coupled. Therefore, under this condition, their dependency is not well considered, thus leading to poor performance. We transform this problem into a bilevel optimization problem, in which the offloading decision is generated in the upper level, and then the optimal allocation of computation resources is obtained in the lower level based on the given offloading decision. In this way, the dependency between the offloading decision and computation resource allocation can be fully taken into account. Subsequently, a bilevel optimization approach, called BiJOR, is proposed. In BiJOR, candidate modes are first pruned to reduce the number of infeasible offloading decisions. Afterward, the upper level optimization problem is solved by ant colony system (ACS). Furthermore, a sorting strategy is incorporated into ACS to construct feasible offloading decisions with a higher probability and a local search operator is designed in ACS to accelerate the convergence. For the lower level optimization problem, it is solved by the monotonic optimization method. In addition, BiJOR is extended to deal with a complex scenario with the channel selection. Extensive experiments are carried out to investigate the performance of BiJOR on two sets of instances with up to 400 mobile users. The experimental results demonstrate the effectiveness of BiJOR and the superiority of the CoMECO system.**

***Index Terms*—Mobile edge computing, offloading decision, computation resource allocation, bilevel optimization, ant colony system.**

I. Introduction

With the popularity of mobile devices, such as smart phones, tablets, and wearable devices, a mass of novel mobile applications are emerging. Among them, an increasing number of applications are resource-intensive and delay-sensitive, for instance, face recognition [1] and augmented reality applications [2]. However, the growing capacity of mobile devices is still lagging behind the needs of people due to physical limitations such as CPU, battery life, and so on [3]. Therefore, it is a very challenging task to execute resource-intensive and delay-sensitive tasks on mobile devices.

Such a challenge motivates the development of mobile cloud computing (MCC) [4] in recent decades, which can provide rich computation resources for mobile users by offloading tasks from their own devices to the cloud server. Whereas, as MCC is a centralized computing paradigm, mobile users should exchange data with the cloud server in the remote data center. To be specific, users must first upload data to the remote cloud server through a wide area network, and after completing computation, results will be transmitted back to users via downlinks. Obviously, the remote transmission causes high costs in terms of time and energy, which degrades quality of service (QoS) [5].

To alleviate the drawback of remote transmission, a novel paradigm of mobile computing has been proposed, known as mobile edge computing (MEC)[1] [7], which encourages a shift of deploying servers from the core network to network edge such as base stations. Compared with MCC, MEC consumes a shorter transmission time and lower energy since data is only uploaded to the server close to mobile users. Therefore, it has more potential to speed up the process of task execution and/or to save the energy of mobile devices [8]. Although MEC can avoid remote task transmission, it still has some disadvantages. For instance, if too many tasks are offloaded to the MEC server simultaneously, mobile devices may generate severe interference with each other, which has side effects on task execution in two aspects [9]. Firstly, with the increasing number of tasks offloaded to the MEC server, the data transmission rate will decrease drastically. Consequently, the transmission time and energy consumption will be increased. Secondly, due to the delay constraint, the computation time will decrease with the growth of the transmission time. As a result, each task should be allocated more computation resources, which may cause higher overloads of the MEC server. To summarize, there is a performance bottleneck in the large-scale offloading system.

It is worth noting that an increasing number of mobile

This work was supported in part by the Innovation-Driven Plan in Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grant 61673397, in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (Grant No. 2016JJ1018), and in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06. (*Corresponding author: Yong Wang and Kezhi Wang*).

P.-Q. Huang, Y. Wang, and Z.-Z. Liu are with the School of Automation, Central South University, Changsha 410083, China. (Email: pqhuang@csu.edu.cn; ywang@csu.edu.cn; zhizhongliu@csu.edu.cn)

K. Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK. (Email: kezhi.wang@northumbria.ac.uk)

[1]Since September 2016, ETSI has renamed mobile edge computing to multi-access edge computing to broaden its applicability [6].

devices are distributed at the network edge and are often idle. These idle devices can potentially share their unused computation resources with nearby devices to help them execute tasks. This technology is called cooperative computing (CC) [10], and the idle devices are called the cooperative devices of nearby devices. Recently, some attempts [11]–[14] have been made to incorporate CC into MEC to enhance the capabilities of MEC, in which mobile devices are not only network terminals but also "virtual" MEC servers. In this way, it can not only save the computation resources of the real MEC server, but also effectively alleviate the network congestion for MEC. In this paper, a multi-user cooperative MEC offloading (CoMECO) system is studied, in which tasks may be executed on local devices, cooperative devices, or the primary MEC server (denoted as P-MEC server) in the base station. Different from existing multi-user cooperative MEC offloading systems [13], [14], we take the delay constraint and multi-user interference into account. Under this condition, the offloading system becomes more complicated.

In offloading systems, offloading decision and resource allocation are two key issues which have a direct impact on QoS in terms of energy consumption and delay [15]. The former is to decide whether to offload or not. In some cases, a further question is where to offload. While the latter is to decide how many resources are allocated. In this paper, first of all, a joint offloading decision and computation resource allocation problem in the CoMECO system is formulated, with the aim of minimizing the total energy consumption of all mobile users under the delay constraint. When solving this problem directly, in general the offloading decision and computation resource allocation are generated separately at the same time, thus ignoring their dependency unreasonably. To this end, we transform this problem into a bilevel optimization problem and prove that the original problem and the bilevel optimization problem have the same optimal solution. In the bilevel optimization problem, the upper level aims to optimize the offloading decision and the lower level is to find the optimal allocation of computation resources under the given offloading decision. In this way, the dependency between the offloading decision and computation resource allocation can be fully considered.

The main contributions of this work are summarized as follows:

- A multi-user MEC offloading system, i.e., the CoMECO system, is investigated in a multi-user interference environment, in which delay-sensitive tasks may be executed on local devices, cooperative devices, or the P-MEC server.
- In order to effectively tackle the bilevel optimization problem, a bilevel optimization approach, called BiJOR, is proposed, which employs ant colony system (ACS) and the monotonic optimization method to solve the offloading decision and computation resource allocation, respectively. ACS has the ability to handle combinatorial optimization problems with categorical variables. To construct feasible offloading decisions efficiently, the candidate mode pruning is implemented in the initialization phase and a sorting strategy is incorporated into ACS. Also, a local search operator is devised in ACS, with the aim of accelerating the convergence. In addition, without adding any additional components, BiJOR can be extended to deal with the channel selection to further allocate communication resources.
- Extensive experiments have been carried out on two instance suites with up to 400 mobile users. The experimental results have demonstrated the effectiveness of BiJOR. In addition, the superiority of the CoMECO system has been verified by comparing BiJOR with four other algorithms.

The rest of this paper is organized as follows. In Section II, the related work is summarized. Section III introduces the system model and presents the joint offloading decision and computation resource allocation problem. Section IV describes the details of our proposed approach, followed by the extension in Section V. The experimental studies are shown in Section VI. Finally, Section VII concludes this paper.

II. Related Work

A. Offloading Decision and Resource Allocation in Offloading Systems

Many methods have been presented for offloading decision. Chen [5] proposed a game theoretic approach for multi-user offloading decision making and obtained a Nash equilibrium solution. This approach is then used to optimize multi-user offloading decision in a multi-channel environment [9]. Plachy *et al.* [16] developed a novel path selection algorithm for offloading tasks between mobile users and cloud-enhanced small cells in a dynamic scenario. Considering the NP-hardness of offloading decision, Deng *et al.* [17] exploited a genetic algorithm to produce the offloading decision for service workflow. Other heuristic methods have also been successfully applied to optimize offloading decision, such as greedy method [18] and particle swarm optimization (PSO) [19]. On the other hand, resource allocation is optimized in a single-user MEC offloading system [20], and multi-user MEC offloading systems [21], [22].

However, the above-mentioned studies deal with offloading decision and resource allocation separately. Due to the fact that both of them are related to QoS, many methods have been proposed to deal with the joint optimization of offloading decision and resource allocation. For instance, in [23] and [24], the joint offloading decision and resource allocation problems in the binary MEC offloading system are optimized. Tran *et al.* [25] designed the joint optimization of offloading decision and resource allocation for a multi-cell, multi-server offloading system. Wang *et al.* [26] developed a decentralization algorithm for jointly optimizing offloading decision, resource allocation, and Internet content caching in heterogeneous wireless cellular networks with MEC. Nevertheless, these methods are designed for delay-tolerant tasks. Very recently, Lyu *et al.* [27] investigated a joint offloading decision and resource allocation problem of delay-sensitive tasks. Note that, no interference among mobile users is considered in [27].

TABLE I
COMPARISON BETWEEN THE COMECO SYSTEM (OUR WORK) AND THE CURRENT WORK.

Ref.	Offloading decision	Resource allocation	Cooperative computing	Multiple users	Multi-user interference	Delay-sensitive task
[5]	√			√	√	
[9]	√			√	√	
[16]	√			√		
[17]	√			√		
[18]	√			√		
[19]	√					√
[20]		√				
[21]		√		√		√
[22]		√		√	√	√
[23]	√	√		√		
[24]	√	√		√		
[25]	√	√		√	√	
[26]	√	√		√	√	
[27]	√	√		√		√
[11]	√		√			√
[12]	√	√	√			√
[13]	√		√	√		
[14]	√	√	√	√		
Our work	√	√	√	√	√	√

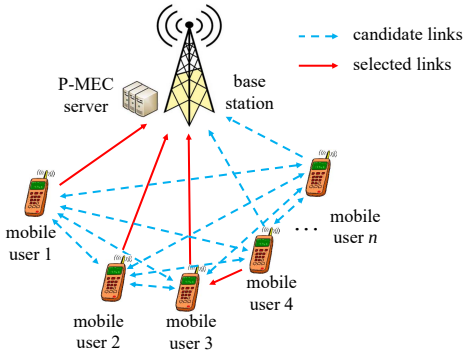


Fig. 1. The CoMECO system involving a base station with a P-MEC server and a set of n mobile users. In this scenario, U_1, U_2, and U_3 are executed on the P-MEC server; U_4 is executed on the mobile device of mobile user 3; and U_n is executed locally.

B. Combination of CC and MEC

As a means to enhance MEC's capabilities, the combination of CC and MEC has recently attracted significant attention. Tao *et al.* [11] considered a cooperative MEC offloading system with two mobile users. In this system, one mobile user can share the computation resources with the other. Cao *et al.* [12] studied a three-node MEC offloading system that consists of a user node, a helper node, and an access point node attached with the MEC server. In contrast to the two-user cooperative MEC offloading systems in [11] and [12], Pu *et al.* [13] researched a multi-user MEC offloading system based on network-assisted D2D collaboration. In this system, each task may be executed locally or offloaded to one of nearby devices. Cui *et al.* [14] developed a novel multi-user MEC offloading system, which integrates the hybrid computation resources of the MEC servers and Internet of Things devices. As a result, the tasks may be executed on local devices, cooperative devices, or the MEC servers. Subsequently, a fully distributed online learning approach is proposed to asymptotically minimize the time-average cost of the system. Despite in both the CoMECO system and [14], tasks may be executed on local devices, cooperative devices, or MEC servers, the CoMECO system is different from [14] by taking delay-sensitive tasks and multi-user interference into account. Under this condition, the CoMECO system is more complicated. In addition, we need to consider whether the tasks can be completed under the delay constraint. A comparison between the CoMECO system and the current work is summarized in Table I.

III. SYSTEM MODEL AND PROBLEM FORMULATION

As shown in Fig. 1, we consider the CoMECO system involving a base station with a P-MEC server and a set of n mobile users[2] denoted as $\mathcal{N} = \{1, 2, \ldots, n\}$. Note that each mobile user has a task to be executed. For the sake of simplicity, we denote the task of mobile user i as U_i, which can be defined by a 4-tuple: $U_i = (C_i, D_i, B_i, T_i^{max})$, $i \in \mathcal{N}$. Herein, C_i describes the total computation resource (i.e., the number of the CPU cycles) to complete task U_i, D_i and B_i denote the size of input data and results, respectively, and T_i^{max} specifies the delay constraint of U_i. In other words, U_i should be completed within time $[0, T_i^{max}]$.

In the CoMECO system, we assume that all mobile devices can be considered as MEC servers and are willing to share computation resources with each other. As a result, tasks may be executed on local devices, cooperative devices, or the P-MEC server in the base station. Therefore, there are $(n+1)$ candidate modes for U_i, denoted as $\mathcal{M} = \{0, 1, \ldots, n\}$. Specifically, for $j \in \mathcal{M}$

- $j = 0$ indicates the P-MEC computing mode;
- $j = i$ indicates the local computing mode;
- Otherwise, j indicates the cooperative computing mode, in which the mobile device of mobile user j is the cooperative device.

In addition, we assume that each mobile device can only accept at most one task but the P-MEC server can accept multiple tasks. As a result, multiple tasks may be executed in the P-MEC computing mode simultaneously. We also assume that there are $(n+1)$ wireless channels. The tasks executed in the same mode are transmitted through the same channel, and the tasks executed in different modes are transmitted through different channels. Furthermore, we define matrix $\mathbf{o}$ as the offloading decision, where $o_{ij} = 1$ ($i \in \mathcal{N}$ and $j \in \mathcal{M}$) if U_i is executed in mode j; otherwise, $o_{ij} = 0$. In Fig. 1, U_1, U_2, and U_3 are executed on the P-MEC server; U_4 is executed on the mobile device of mobile user 3; and U_n is executed locally. Therefore, o_{10}, o_{20}, o_{30}, o_{43}, and $o_{nn} = 1$. Additionally, we define another matrix $\mathbf{r}$ as the computation resource allocation, where r_{ij} ($i \in \mathcal{N}$ and $j \in \mathcal{M}$) represents the computation resources allocated to U_i in mode j.

We next introduce the computation model, transmission model, and delay model, respectively.

A. Computation Model

Given computation resource r_{ij}, the computation time of U_i in mode j is [28]

$$T_{ij}^{c}(r_{ij}) = \frac{C_i}{r_{ij}}, \ \forall i \in \mathcal{N}, j \in \mathcal{M} \tag{1}$$

[2]In this paper, we assume that mobile users are static unless otherwise stated.

Due to the fact that the computation of tasks on the P-MEC server does not consume any energy of mobile devices, similar to [29], we ignore the computation energy consumption of the P-MEC server. If U_i is executed in mode j, the computation energy consumption is given as [28]

$$E_{ij}^{c}(r_{ij}) = k(r_{ij})^2 C_i,\ \forall i \in \mathcal{N}, j \in \mathcal{M}\backslash\{0\} \tag{2}$$

where $k > 0$ is the effective capacitance coefficient.

B. Transmission Model

For the P-MEC computing mode and cooperative computing mode, mobile users must first upload data to the P-MEC server or cooperative devices. After the computation is completed, results are returned to mobile users via downlinks. The uplink data rate of U_i in mode j is given as [5]

$$R_{ij}(\mathbf{o}) = W\log_2\left(1 + \frac{p_i^t H_{ij}}{w + \sum\limits_{h=1,h\neq i}^{n} o_{hj} p_h^t H_{hj}}\right), \quad \forall i \in \mathcal{N}, j \in \mathcal{M}\backslash\{i\} \tag{3}$$

where W is the channel bandwidth, p_i^t is the transmission power of mobile device i and can be obtained by the power control algorithm [30], w denotes the background noise power, H_{ij} denotes the channel gain of mobile device i in mode j, and $\sum\limits_{h=1,h\neq i}^{n} o_{hj} p_h^t H_{hj}$ is the interference among mobile uers in the same channel.

Remark 1. In a static scenario, the uplink data rate R_{ij} and downlink data rate R_{ji} are symmetric. In a dynamic scenario, for example, in the case that mobile users are moving, the conditions in uplink and downlink are usually different [16]. As a result, R_{ij} and R_{ji} are asymmetric, the effect of which is investigated in the experimental studies.

Afterward, the transmission time of U_i in mode j is computed as [13]

$$T_{ij}^{t}(\mathbf{o}) = \frac{D_i}{R_{ij}(\mathbf{o})} + \frac{B_i}{R_{ji}(\mathbf{o})},\ \forall i \in \mathcal{N}, j \in \mathcal{M}\backslash\{i\} \tag{4}$$

For the P-MEC computing mode, the transmission energy consumption of U_i is the energy consumption of mobile device i for transmitting input data and receiving results, which is given as [13]

$$E_{ij}^{t}(\mathbf{o}) = \frac{p_i^t D_i}{R_{ij}(\mathbf{o})} + \frac{p_i^r B_i}{R_{ji}(\mathbf{o})},\ \forall i \in \mathcal{N}, j = 0 \tag{5}$$

where p_i^r represents the power of mobile user i for receiving results.

For the cooperative computing mode, the transmission energy consumption of U_i consists of the energy consumption of mobile device i and cooperative mobile device j for transmitting input data and receiving results. In addition, the time of cooperative mobile device j for receiving results is the same as that of mobile device i for transmitting input data and vice versa [31]. Therefore, the transmission energy consumption of U_i is computed as [13]

$$E_{ij}^{t}(\mathbf{o}) = \frac{p_i^t D_i}{R_{ij}(\mathbf{o})} + \frac{p_j^r D_i}{R_{ij}(\mathbf{o})} + \frac{p_j^t B_i}{R_{ji}(\mathbf{o})} + \frac{p_i^r B_i}{R_{ji}(\mathbf{o})}, \quad \forall i \in \mathcal{N}, j \in \mathcal{M}\backslash\{0, i\}. \tag{6}$$

Note that in the transmission model, we ignore the circuit power consumed by mobile devices. This assumption is used widely in the literature [5], [13], [28].

C. Delay Model

The total time spent in executing U_i in mode j is given as

$$T_{ij}(\mathbf{o}, r_{ij}) = \begin{cases} T_{ij}^{c}(r_{ij}), & \text{if } j = i \\ T_{ij}^{t}(\mathbf{o}) + T_{ij}^{c}(r_{ij}), & \text{otherwise} \end{cases},\ \forall i \in \mathcal{N} \tag{7}$$

D. Problem Formulation

Considering that both the offloading decision and computation resource allocation are related to QoS, we jointly optimize them to minimize the total energy consumption of all mobile users, which consists of the total computation and transmission energy consumption. The joint offloading decision and computation resource allocation problem is expressed as follows:

$$\begin{aligned} \mathcal{P}: \min_{\mathbf{o},\mathbf{r}} \quad & \sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij} E_{ij}^{c}(r_{ij}) + \sum_{j=0,j\neq i}^{n} o_{ij} E_{ij}^{t}(\mathbf{o})\right) \\ \text{s.t.} \quad & C1: \sum_{j=0}^{n} o_{ij} = 1, \forall i \in \mathcal{N}, \\ & C2: \sum_{i=1}^{n} o_{ij} \leq 1, \forall j \in \mathcal{M}\backslash\{0\}, \\ & C3: \sum_{i=1}^{n} o_{ij} r_{ij} \leq r_j^{max}, \forall j \in \mathcal{M}, \\ & C4: r_{ij} > 0, \forall o_{ij} = 1\ (i \in \mathcal{N} \text{ and } j \in \mathcal{M}), \\ & C5: r_{ij} = 0, \forall o_{ij} = 0\ (i \in \mathcal{N} \text{ and } j \in \mathcal{M}), \\ & C6: 0 \leq \sum_{j=0}^{n} o_{ij} T_{ij}(\mathbf{o}, r_{ij}) \leq T_i^{max}, \forall i \in \mathcal{N}. \end{aligned} \tag{8}$$

where $C1$ guarantees that each task is executed; $C2$ states that each mobile device can only accept at most one task; $C3$ assumes that at most computation resource r_j^{max} is provided in mode j; $C4$ and $C5$ represent that if U_i is executed in mode j, r_{ij} should be greater than 0; otherwise, r_{ij} should be equal to 0; and $C6$ ensures that each task has to be completed under the delay constraint.

We can observe that $\mathcal{P}$ is a mixed-variable nonlinear optimization problem, since $\mathbf{o}$ is an integer matrix and $\mathbf{r}$ is a continuous matrix. It is difficult to solve $\mathcal{P}$ by using traditional optimization methods. By further analyzing $\mathcal{P}$, we can find the following two characteristics. Firstly, the amount of available computation resources may vary with different modes depending on the result of offloading decision. Secondly, the performance of offloading decision cannot be assessed until the computation resource allocation has been generated; hence, it is affected by the result of computation resource allocation.

Furthermore, only if the computation resource allocation is optimal, the quality of offloading decision can be assessed accurately. In brief, the offloading decision and computation resource allocation are closely coupled. If $\mathcal{P}$ is solved directly, the offloading decision and computation resource allocation are generated separately; thus, their dependence is ignored unreasonably. Therefore, in order to solve $\mathcal{P}$ effectively, there are two challenges:

- Can this mixed-variable nonlinear optimization problem be transformed into another optimization problem which is easy to be solved?
- How can we take the dependency between the offloading decision and computation resource allocation into account?

IV. Proposed Approach

A. Problem Transformation

Considering the above two challenges, we first transform $\mathcal{P}$ into a bilevel optimization problem. Bilevel optimization involves addressing the upper level optimization problem under the premise of ensuring the optimality of the lower level optimization problem [32], [33]. In this paper, the offloading decision is regarded as the upper level optimization problem with the aim of minimizing the total energy consumption of all mobile users, and the computation resource allocation is considered as the lower level optimization problem with the purpose of minimizing the total computation energy consumption of all mobile users. This bilevel optimization problem is formulated as

$$\begin{aligned}\mathcal{P}1: \min_{\mathbf{o},\mathbf{r}} \ & \sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij}E_{ij}^{c}(r_{ij}) + \sum_{j=0,j\neq i}^{n} o_{ij}E_{ij}^{t}(\mathbf{o})\right) \quad (9)\\ \text{s.t. } & \mathbf{r} \in \arg\min_{\mathbf{r}} \left\{\sum_{i=1}^{n}\sum_{j=1}^{n} o_{ij}E_{ij}^{c}(r_{ij}) : C3 - C6\right\}\\ & C1 \text{ and } C2.\end{aligned}$$

Before presenting the relationship between $\mathcal{P}$ and $\mathcal{P}1$, lemma 1 is given below.

Lemma 1. *The optimal solution of $\mathcal{P}$ is a feasible solution of $\mathcal{P}1$.*

Proof: A detailed proof is given in Appendix A of the supplementary file.

Theorem 1. *$\mathcal{P}$ and $\mathcal{P}1$ have the same optimal solution.*

Proof: A detailed proof is given in Appendix B of the supplementary file.

Apart from keeping the optimal solution of $\mathcal{P}$, the above transformation has the following additional advantages:

- The original mixed-variable nonlinear optimization problem is transformed into a combinatorial optimization problem in the upper level and a continuous optimization problem in the lower level, which are more tractable than the original one.
- In $\mathcal{P}1$, the dependency between the offloading decision and computation resource allocation can be fully considered. To be specific, since the computation resource allocation is generated based on the offloading decision, the limitation of available computation resources enforced by the offloading decision is taken into account. In addition, since the optimal allocation of computation resources corresponding to each offloading decision is obtained in the lower level, the quality of each offloading decision can be assessed accurately.

Algorithm 1 General Framework of BiJOR

1: $gen = 0$;
2: Generate the feasible candidate mode sets $\mathcal{M}_1, \ldots, \mathcal{M}_n$ based on **Algorithm 2**;
3: **while** $gen < gen_{max}$ **do**
4: Construct NP offloading decisions via ACS based on **Algorithm 3**: $\mathcal{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_{NP}\}$;
5: Calculate the optimal allocations of computation resources under the given $\mathcal{O}$: $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_{NP}\}$;
6: Evaluate the total energy consumption of each offloading decision with the corresponding optimal allocation of computation resources;
7: Perform the local search operator on the iteration-best solution $\{\mathbf{o}^b, \mathbf{r}^b\}$;
8: Update the global pheromone in ACS based on (28);
9: $gen = gen + 1$;
10: **end while**
11: **Output**: the best offloading decision and the corresponding optimal allocation of computation resources

The offloading decision is a typical combinatorial optimization problem. Due to the NP-hard characteristic [9], seeking the optimal offloading decision is generally deemed to be a challenging task. Although deterministic algorithms have the capability to find the optimal offloading decision, they suffer from high computational burden in large-scale problems, and the optimal solution cannot be provided within a reasonable time [34]. Further, since there is no logical ordering relationship among candidate modes, the offloading decision is also considered as a combinatorial optimization problem with categorical variables; thus, a categorical optimization approach is necessary [35].

Due to the resource limitation, it is likely that some tasks cannot be completed in most candidate modes when considering the delay constraint. As a result, most of offloading decisions are infeasible. It is because an offloading decision is infeasible as long as any task is not completed under the delay constraint. Therefore, it poses a grand challenge to generate feasible offloading decisions. Additionally, in order to obtain the optimal offloading decision as soon as possible, a strategy that can speed up the optimization is required.

Therefore, the following three challenges should be considered:

- How to deal with the combinatorial optimization problem with categorical variables in the upper level?
- How can we construct feasible offloading decisions efficiently?
- How to design a strategy to speed up the optimization?

B. BiJOR

To address the above three challenges, a bilevel optimization approach named BiJOR is proposed to solve $\mathcal{P}1$. The general framework of BiJOR is presented in **Algorithm 1**. In the initialization phase, the feasible candidate mode set of each task is generated by the candidate mode pruning (line 2). In the main loop, both the upper level optimization and lower level optimization are implemented at each generation, where the latter is nested within the former. To be specific, ACS with the sorting strategy is used to construct NP offloading decisions (line 4). In the lower level optimization, the monotonic optimization method is adopted to obtain the corresponding optimal allocations of computation resources based on the given offloading decisions (line 5). Then, the performance of the offloading decisions and their corresponding optimal allocations of computation resources is evaluated (line 6). Subsequently, the local search operator is performed on the iteration-best solution to accelerate the convergence (line 7). Finally, the global pheromone in ACS is updated (line 8). The above procedure will continue until the stopping criterion is met. In the following, we will introduce BiJOR in detail.

C. Candidate Mode Pruning

The size of the solution space for the upper level optimization problem, i.e., the number of offloading decisions, depends on the number of mobile users in the CoMECO system. For instance, as shown in Fig. 1, since each task has $(n+1)$ candidate modes, the number of offloading decisions is $(n+1)^n$. Obviously, it is extremely large. Indeed, as stated in Section IV-A, most of them are infeasible. In order to reduce the number of infeasible offload decisions, we prune infeasible candidate modes of each task in the initialization phase. Next, the conditions to judge the feasibility of a candidate mode are given below.

Because the computation resources available to mobile users are limited and may vary with different modes, if the minimum demand of computation resources of a task in a mode is not satisfied, this mode is an infeasible candidate mode for this task. Specifically, based on $C6$, the minimum demand of computation resources of U_i in mode j can be given as

$$r_{ij}^{min} = \begin{cases} \dfrac{C_i}{T_i^{max}}, & \text{if } j = i \\ \dfrac{C_i}{T_i^{max} - \frac{D_i}{R_{ij}} - \frac{B_i}{R_{ji}}}, & \text{otherwise} \end{cases}, \ \forall i \in \mathcal{N} \quad (10)$$

Then, based on $C3 - C5$, a feasible candidate mode should satisfy the following conditions:

$$\sum_{i=1}^{n} o_{ij} r_{ij}^{min} \leq r_j^{max} \text{ and } r_{ij}^{min} \geq 0, \ \forall i \in \mathcal{N}, j \in \mathcal{M} \quad (11)$$

For a mode that can only be assigned to one task, such as the local computing mode or cooperative computing mode, the minimum demand of computation resources is readily available in (10). Then, by substituting it to (11), the feasible candidate modes in the local computing mode and cooperative computing mode are subject to (12) and (13), respectively,

$$0 \leq \frac{C_i}{T_i^{max}} \leq r_j^{max}, \ \forall i \in \mathcal{N}, j = i \quad (12)$$

and

$$0 \leq \frac{C_i}{T_i^{max} - \frac{D_i}{R_{ij}} - \frac{B_i}{R_{ji}}} \leq r_j^{max}, \ \forall i \in \mathcal{N}, j \neq i, \text{ and } j \neq 0 \quad (13)$$

However, for a mode that can be assigned to multiple tasks, such as the P-MEC computing mode, since it is impossible to predict which tasks are executed in this mode in advance during the initialization phase, the interference in the same channel cannot be obtained. As a result, the data rate is also not obtained, which results in the minimum demand of computation resources in (10) being unavailable. In this situation, (11) cannot be used directly for judging the feasibility of a candidate mode. Although the P-MEC server can accept multiple tasks simultaneously, if we assume that only one task is executed in the P-MEC server, the upper bound of the uplink data rate $\overline{R_{ij}}$ can be obtained based on (3)

$$\overline{R_{ij}} = W log_2(1 + \frac{p_i H_{ij}}{w}), \ \forall i \in \mathcal{N}, j = 0 \quad (14)$$

The upper bound of the downlink data rate $\overline{R_{ji}}$ can be obtained in a similar way.

Then, the lower bound of the minimum demand of computation resources is given as

$$\underline{r_{ij}^{min}} = \frac{C_i}{T_i^{max} - \frac{D_i}{\overline{R_{ij}}} - \frac{B_i}{\overline{R_{ji}}}}, \ \forall i \in \mathcal{N}, j = 0 \quad (15)$$

From (11) and (15), if the P-MEC computing mode is feasible, it at least satisfies

$$0 \leq \frac{C_i}{T_i^{max} - \frac{D_i}{\overline{R_{ij}}} - \frac{D_i}{\overline{R_{ji}}}} \leq r_j^{max}, \ \forall i \in \mathcal{N}, j = 0 \quad (16)$$

Note that, (16) is a necessary condition that the P-MEC computing mode is feasible. That is, even if (16) is satisfied, there is no guarantee that a task can be executed on the P-MEC server. Therefore, in the process of offloading decision construction, it is necessary to check whether (11) is satisfied to ensure that the P-MEC computing mode is feasible.

Since R_{ij} and R_{ji} are equal to $\overline{R_{ij}}$ and $\overline{R_{ji}}$ in the cooperative computing mode, respectively, (13) is equivalent to (16).

Algorithm 2 Candidate Mode Pruning

1: **for** $i = 1$ to n **do**
2: $\quad \mathcal{M}_i = \emptyset$;
3: $\quad$ **for** $j = 0$ to n **do**
4: $\quad\quad$ **if** $j = i$ and $0 \leq \frac{C_i}{T_i^{max}} \leq r_j^{max}$ **then**
5: $\quad\quad\quad \mathcal{M}_i \leftarrow \mathcal{M}_i \bigcup \{i\}$;
6: $\quad\quad$ **else if** $j \neq i$ and $0 \leq \frac{C_i}{T_i^{max} - \frac{D_i}{R_{ij}} - \frac{B_i}{R_{ji}}} \leq r_j^{max}$ **then**
7: $\quad\quad\quad \mathcal{M}_i \leftarrow \mathcal{M}_i \bigcup \{j\}$;
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**
11: **return** $\mathcal{M}_1, \ldots, \mathcal{M}_n$

In conclusion, a feasible candidate mode of U_i is subject to

$$\begin{cases} 0 \leq \dfrac{C_i}{T_i^{max}} \leq r_j^{max}, & \text{if } j = i \\ 0 \leq \dfrac{C_i}{T_i^{max} - \frac{D_i}{R_{ij}} - \frac{B_i}{R_{ji}}} \leq r_j^{max}, & \text{if } j \neq i \end{cases}, \ \forall i \in \mathcal{N} \tag{17}$$

The details of the candidate mode pruning are shown in **Algorithm 2**. Based on (17), the feasibility of each candidate mode for each task is checked in turn. For a candidate mode, if (17) is satisfied, it is considered feasible and added to the feasible candidate mode set; otherwise, it is pruned. Finally, the feasible candidate mode set of each task is generated. After pruning, the size of the solution space is $\prod_{i=1}^{n} |\mathcal{M}_i|$, where $\mathcal{M}_i$ denotes the feasible candidate mode set of U_i and $|\mathcal{M}_i|$ denotes its size. Due to $|\mathcal{M}_i| \leq n+1$, $\prod_{i=1}^{n} |\mathcal{M}_i| \leq (n+1)^n$. As a result, the size of the solution space may be reduced significantly. More importantly, since infeasible modes have been pruned, it is beneficial to rapidly construct feasible offloading decisions.

D. Lower Level Optimization

The objective of the lower level optimization is to minimize the total computation energy consumption of all mobile users through optimizing the computation resource allocation $\mathbf{r}$ under given offloading decision $\mathbf{o}$, which can be formulated as

$$\begin{aligned} \min_{\mathbf{r}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} o_{ij} E_{ij}^{c}(r_{ij}) \\ \text{s.t.} \quad & C3 - C6. \end{aligned} \tag{18}$$

Substituting (2) to (18), the lower level optimization problem can be rewritten as

$$\begin{aligned} \min_{\mathbf{r}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} o_{ij} k (r_{ij})^2 C_i \\ \text{s.t.} \quad & C3 - C6. \end{aligned} \tag{19}$$

It can be seen from (19) that there is a strictly monotonic increasing relationship between computation resources and the total computation energy consumption. In order to minimize the total computation energy consumption of all mobile users, the computation resources allocated to each task should be as few as possible. However, when U_i is executed in mode j (i.e., $o_{ij} = 1$), to ensure that $C6$ is satisfied, the amount of the computation resources allocated to U_i must at least satisfy the minimum demand r_{ij}^{min} in (10). In addition, if $r_{ij} = r_{ij}^{min}$, $C3 - C5$ are certainly satisfied. The reason is that if they are not satisfied, mode j cannot be assigned to U_i when constructing the offloading decision. Therefore, the optimal resource allocation can be given as

$$r_{ij}^{*} = \begin{cases} r_{ij}^{min}, & \text{if } o_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{20}$$

Algorithm 3 Offloading Decision Construction

```
Generate task priorities based on the sorting strategy: L = {l_1, ..., l_n};
// l_k (k ∈ {1, ..., n}) denotes the index of the task with priority level k
o = 0; // Initialize the offloading decision
for k = 1 to n do
    i ← l_k;
    if {0} ⊂ M_i then
        if (11) is not satisfied then
            M_i ← M_i \ {0};
        end if
    end if
    if M_i ≠ ∅ then
        Calculate the probability that each feasible candidate mode in M_i
        is assigned to U_i based on (21);
        Select mode j from M_i based on (24) and set o_ij = 1;
        if j > 0 then
            Update the feasible candidate mode sets of the tasks that have
            not yet been assigned any modes;
        end if
        Update the local pheromone in ACS based on (27);
    end if
end for
return o
```

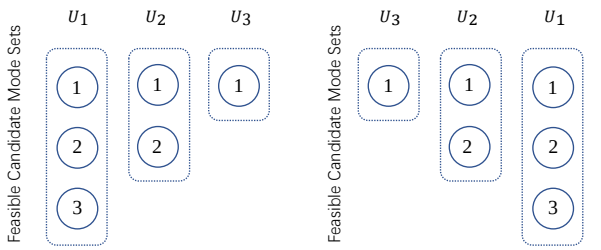
E. Upper Level Optimization

The goal of the upper level optimization is to optimize the offloading decision for minimizing the total energy consumption of all mobile users including the total transmission and computation energy consumption. As introduced in Section IV-A, making use of deterministic algorithms such as branching is not a good choice to seek the optimal solution of this problem. As population-based stochastic algorithms, evolutionary algorithms (EAs), such as PSO, differential evolution (DE), and ACS, are widely applied to address NP-hard problems [36]–[38]. In this paper, ACS is applied to optimize the offloading decision. The reasons are listed as follows:

- The offloading decision in this paper is a combinatorial optimization problem with categorical variables, and ACS has been shown to be particularly successful in solving such kind of problems [35].
- Because all variables in the solution obtained by other EAs, such as PSO and DE, are generated simultaneously, all tasks are assigned modes simultaneously when using other EAs to optimize the offload decision. As a result, multiple tasks may be executed on the same mobile device. However, each mobile device can only accept at most one task in this paper (i.e., $C2$). In contrast, since each variable in the solution obtained by ACS is generated one by one, each task is assigned a mode one by one when using ACS to optimize the offload decision. Once a task is assigned to a mobile device, this mobile device will not accept other tasks, which ensures that each mobile device can only accept at most one task.

In this paper, ACS consists of four components: offloading decision construction, fitness evaluation, local search, and pheromone management.

1) Offloading Decision Construction: For each task, an ant selects a mode from its feasible candidate mode set. By doing this, an offloading decision is obtained. Note that if the feasible candidate mode set is empty, the corresponding task is abandoned. **Algorithm 3** presents the construction of an



(a) Random priorities (b) Task priorities after applying the sorting strategy

Fig. 2. Illustration of the working principle of the proposed sorting strategy. There are three tasks, i.e., U_1, U_2, and U_3, and three feasible candidate mode sets. Each task is assigned one of the feasible candidate modes in turn.

offloading decision.

Before assigning the mode, task priorities should be generated, which are random in general [36], [39]. However, in this way, it is not conducive to constructing feasible offloading decisions. Fig. 2(a) shows an example with random priorities, where U_1, U_2, and U_3 are assigned modes in turn. In this case, if U_1 or U_2 is executed in mode 1, U_3 cannot be executed since it can only be executed in mode 1. To construct feasible offloading decisions efficiently, in this paper, a sorting strategy is proposed to generate task priorities $L = \{l_1, \ldots, l_n\}$, where l_k $(k \in \{1, \ldots, n\})$ denotes the index of the task with priority level k (line 1). This sorting strategy sorts all tasks in ascending order of the number of feasible candidate modes, which means that the tasks with fewer feasible candidate modes have higher priorities. Fig. 2(b) shows the task priorities after applying the sorting strategy, where $L = \{3, 2, 1\}$, i.e., U_3 is the first, U_2 is the second, and U_1 is the last. It is easy to find that all tasks can be executed (i.e., U_1, U_2, and U_3 are executed in modes 3, 2, and 1, respectively).

After task priorities are generated, the offloading decision is initialized (i.e., $\mathbf{o} = 0$). According to the priority of a task, a mode selected from the feasible candidate mode set is assigned to this task. For U_i, if mode 0 (i.e., the P-MEC computing mode) is one of its feasible candidate modes, as mentioned earlier, it is necessary to check whether (11) is satisfied when U_i is executed on the P-MEC server. If not, this mode should be removed from its feasible candidate mode set $\mathcal{M}_i$ (lines $5-9$).

Next, if $\mathcal{M}_i$ is not empty, the probability that each feasible candidate mode is assigned to U_i is given as [40]

$$pr_{ij} = \frac{\tau_{ij}\eta_{ij}^{\beta}}{\sum\limits_{k \in \mathcal{M}_i} \tau_{ik}\eta_{ik}^{\beta}}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{21}$$

where τ is the pheromone, η is the heuristic information, and β is a parameter to determine the relative importance between the pheromone and heuristic information. In this paper, the heuristic information is defined as

$$\eta_{ij} = \frac{1}{E_{ij}^I}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{22}$$

where E_{ij}^I is the increment of the total energy consumption when mode j is assigned to U_i. Thus, the smaller the increment of the total energy consumption, the higher the probability that mode j is assigned to U_i in (21). E_{ij}^I is calculated as

$$E_{ij}^I = \begin{cases} \sum\limits_{k \in \mathcal{A} \bigcup \{i\}} E_{kj}^t(\mathbf{o}) - \sum\limits_{k \in \mathcal{A}} E_{kj}^t(\mathbf{o}), & \text{if } j = 0 \\ E_{ij}^c(r_{ij}^{min}), & \text{if } j = i \\ E_{ij}^t(\mathbf{o}) + E_{ij}^c(r_{ij}^{min}), & \text{otherwise} \end{cases}, \ \forall i \in \mathcal{N} \tag{23}$$

where $\mathcal{A}$ denotes the set of all tasks that have been implemented in the P-MEC computing mode, and r_{ij}^{min} is the minimum demand of computation resources given in (10).

Then, mode j is selected and assigned to U_i (i.e., $o_{ij} = 1$), according to the following rule [40]

$$j = \begin{cases} \arg\max\limits_{k \in \mathcal{M}_i} \tau_{ik}\eta_{ik}^{\beta}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{24}$$

where q is a random number uniformly distributed over [0, 1], q_0 is a parameter, and J is a mode selected from $\mathcal{M}_i$ by using the roulette wheel selection based on (21).

Afterward, if $j > 0$, that is, U_i is executed in the local computing mode or cooperative computing mode, other tasks cannot be executed in mode j in order to ensure that $C2$ is satisfied. Therefore, the feasible candidate mode sets of the tasks that have not yet been assigned any modes are updated by removing mode j (lines $16-18$). Finally, the local pheromone is updated (line 19), which will be introduced in Section IV-E4. The above procedure is repeated until NP offloading decisions are constructed.

2) Fitness Evaluation: After offloading decisions have been constructed, the corresponding optimal allocation of computation resources of each offloading decision is obtained in the lower level optimization, which has been introduced in Section IV-D. Then, the performance of a solution consisting of an offloading decision and its optimal allocation of computation resources can be evaluated. In fact, it is difficult to guarantee that all tasks can be completed under a heavy load condition. It is because some tasks may have no feasible candidate mode under this condition. As a result, these tasks will be abandoned, which means that there may not exist any feasible solution for $\mathcal{P}$. To this end, we first maximize the number of tasks being completed, and then minimize the total energy consumption of these tasks. Thus, two fitness functions are proposed to evaluate the performance of solutions. The first fitness function calculates the number of tasks in a solution that can be completed, and the second fitness function calculates the total energy consumption to complete these tasks. These two fitness functions are given as

$$F_1(\mathbf{o}) = \sum_{i=1}^{n} \sum_{j=0}^{n} o_{ij}, \tag{25}$$

$$F_2(\mathbf{o}, \mathbf{r}^*) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} o_{ij} E_{ij}^c(r_{ij}^*) + \sum_{j=0, j \neq i}^{n} o_{ij} E_{ij}^t(\mathbf{o}) \right) \tag{26}$$

For two solutions, the solution with a large F_1 value is preferred. If they have the same F_1 value, the solution with a small F_2 value is preferred.

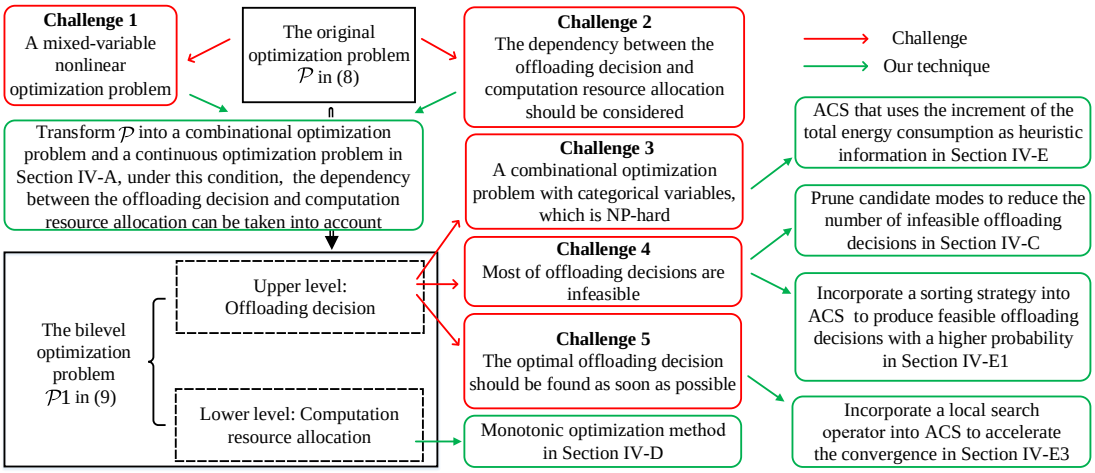


Fig. 3. Challenges and our techniques used in this paper.

3) Local Search: After the fitness evaluation, if the iteration-best solution $\{\mathbf{o}^b, \mathbf{r}^b\}$ is a feasible solution, a local search operator is performed on it to accelerate the convergence. Note that if the mode assigned to a task is switched to the P-MEC computing mode or from the P-MEC computing mode to another, the interference among mobile uers will be changed, resulting in a change in their uplink data rate. As a result, the optimal allocations of computation resources need to be recomputed. It is a complicated process. For simplicity, in this paper, we only consider that the mode of a task is switched from one to another, each of them is not the P-MEC computing mode. In addition, to ensure that each mobile device accepts at most one task, the switched mode should not be assigned to other tasks.

In the local search, firstly, task priorities are generated. Then, each task of $\{\mathbf{o}^b, \mathbf{r}^b\}$ is checked according to its priority. For each feasible candidate mode (denoted as j') of U_i, if the total energy consumption is reduced after the mode assigned to U_i is switched to j', then this adjustment is successful and acceptable. The above procedure is repeated until each feasible candidate mode of each task is checked.

4) Pheromone Management: After a task is assigned a mode, the local pheromone is updated to reduce the probability that the same task is assigned the same mode in different solutions. The local pheromone updating rule is given as

$$\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{27}$$

where φ is the pheromone decay coefficient and τ_0 is the initial value of the pheromone.

At the end of each generation, the global pheromone is updated based on $\{\mathbf{o}^b, \mathbf{r}^b\}$, with the aim of increasing the pheromone values associated with the promising solutions. The global pheromone updating rule is given as

$$\tau_{ij} = \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}, & \text{if } o^b_{ij} = 1 \\ \tau_{ij}, & \text{otherwise} \end{cases}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{28}$$

where ρ is the pheromone decay parameter, and

$$\Delta\tau_{ij} = \frac{1}{F_2(\mathbf{o}^b, \mathbf{r}^b)}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}_i \tag{29}$$

F. Discussion

1) Challenges and Our Techniques: Fig. 3 summarizes the challenges and our techniques used in this paper. To address challenges 1 and 2, $\mathcal{P}$ in (8) is first transformed into a bilevel optimization problem $\mathcal{P}1$ in (9). Then, an ACS with the sorting strategy and the local search is proposed for handing challenges 3, 4 and, 5 derived from the offloading decision in the upper level. In addition, we also perform the candidate mode pruning to alleviate challenge 4. Finally, the monotonic optimization method is employed to allocate computation resources in the lower level. By utilizing these techniques to tackle challenges 1-5, the optimal offloading decision and computation resource allocation can be obtained.

2) Computational Time Complexity of BiJOR: In the initialization phase, we check the feasibility of each mode for each task; thus, the computational time complexity of the candidate mode pruning is $O(n^2)$. In the offloading decision construction phase, each ant selects one mode from the feasible candidate mode set for each task. Therefore, each ant requires $n \cdot |\mathcal{M}_i|$ computations and NP ants require $NP \cdot n \cdot |\mathcal{M}_i| \leq NP \cdot n \cdot (n+1)$ computations due to $|\mathcal{M}_i| \leq n+1$. In the lower level optimization phase, for each task, it is necessary to calculate the optimal allocation of computation resources for the given offloading decision; thus, it requires $NP \cdot n$ computations. In addition, since the mode of each task is adjusted at most $(n-1)$ times in the local search, the computational time complexity of the local search is $O(n^2)$. Thus, the overall computational time complexity of BiJOR is $O(NP \cdot n^2)$.

3) Implementation of BiJOR: The implementation of BiJOR consists of the following three steps.

- Step 1: All mobile devices submit their own information, e.g., the channel states and the information of computation tasks (i.e., C_i, D_i, B_i, and T_i^{max}), to the base

station. Such information can be transmitted through the physical uplink control channel (PUCCH) in the current LTE network or the corresponding control channel in the future 5G network, as it only includes several bits and does not incur much cost.

- Step 2: The base station conducts BiJOR, which will decide the scheduling and the computation resource allocation for each mobile device. The base station then sends the above instructions to each mobile device. This information can be sent via the physical downlink control channel (PDCCH) in the current LTE or the future 5G network.
- Step 3: Each mobile device executes the offloading decision and applies the corresponding computation resource allocation based on the instructions received from the base station. The information transmission to the P-MEC can be done via the physical uplink shared channel (PUSCH) in the current LTE or the future 5G network. In addition, the information transmission to other mobile devices can be done via the Device to Device (D2D) mode, which has been standardized in LTE [41].

To sum up, our algorithm can be incorporated into existing LTE standard.

Remark 2. In real networks, poor network conditions may result in retransmissions, which can increase the transmission time of tasks. Therefore, tasks may not be completed under the preset delay constraint. A possible way to solve this problem is to tighten the delay constraint in the optimization process of BiJOR. This will be achieved in our future work by the following three steps: checking the network status, estimating the possible delay, and adjusting the delay constraint accordingly in our algorithm.

V. Extension

In this section, we extend BiJOR to deal with a complex scenario with the channel selection to further optimize communication resources. We assume that there are k wireless channels denoted as $\mathcal{K}=\{1,2,\ldots,k\}$, and matrix $\mathbf{c}$ indicates the channel selection, where $c_{il}=1$ ($i\in\mathcal{N}$ and $l\in\mathcal{K}$) if U_i is transmitted via channel l; otherwise, $c_{il}=0$. Different from Section III, we assume that the tasks executed in different modes can be transmitted through the same channel. In this scenario, the transmission model and delay model should be reformulated accordingly.

The uplink data rate of U_i in mode j via channel l can be formulated as

$$R^u_{ijl}(\mathbf{c})=Wlog_2\left(1+\frac{p^t_iH_{ij}}{w+\sum\limits_{h=1,h\neq i}^{n}c_{hl}p^t_hH_{hj}}\right),\quad \forall i\in\mathcal{N},j\in\mathcal{M}\backslash\{i\},k\in\mathcal{K} \tag{30}$$

Subsequently, the transmission time of U_i in mode j via channel l is given as

$$T^t_{ijl}(\mathbf{c})=\frac{D_i}{R^u_{ijl}(\mathbf{c})}+\frac{B_i}{R^d_{jil}(\mathbf{c})},\ \forall i\in\mathcal{N},j\in\mathcal{M}\backslash\{i\},l\in\mathcal{K} \tag{31}$$

We can obtain the transmission energy consumption of U_i in the P-MEC computing mode and cooperative computing mode via channel l. They are expressed as (32) and (33), respectively

$$E^t_{ijl}(\mathbf{c})=\frac{p^t_iD_i}{R_{ijl}(\mathbf{c})}+\frac{p^r_iB_i}{R_{jil}(\mathbf{c})},\ \forall i\in\mathcal{N},j=0,l\in\mathcal{K} \tag{32}$$

and

$$E^t_{ijl}(\mathbf{c})=\frac{p^t_iD_i}{R_{ijl}(\mathbf{c})}+\frac{p^r_jD_i}{R_{ijl}(\mathbf{c})}+\frac{p^t_jB_i}{R_{jil}(\mathbf{c})}+\frac{p^r_iB_i}{R_{jil}(\mathbf{c})},\quad \forall i\in\mathcal{N},j\in\mathcal{M}\backslash\{0,i\},l\in\mathcal{K} \tag{33}$$

Also, the total time to execute U_i in mode j is given as

$$T_{ij}(\mathbf{c},r_{ij})=\begin{cases}T^c_{ij}(r_{ij}), & \text{if } j=i\\ \sum\limits_{l=1}^{k}c_{il}T^t_{ijl}(\mathbf{c})+T^c_{ij}(r_{ij}), & \text{otherwise}\end{cases},\ \forall i\in\mathcal{N} \tag{34}$$

For minimizing the total energy consumption of all mobile users, we jointly optimize the offloading decision, channel selection, and computation resource allocation, which is formulated as

$$\mathcal{P}2:\min_{\mathbf{o},\mathbf{c},\mathbf{r}}\quad \sum_{i=1}^{n}\left(\sum_{j=1}^{n}o_{ij}E^c_{ij}(r_{ij})+\sum_{j=0,j\neq i}^{n}\sum_{l=1}^{k}o_{ij}c_{il}E^t_{ijl}(\mathbf{c})\right) \tag{35}$$

$$\begin{aligned}\text{s.t.}\quad & C1-C5,\\ & C6:0\leq\sum_{j=0}^{n}o_{ij}T_{ij}(\mathbf{c},r_{ij})\leq T^{max}_i,\forall i\in\mathcal{N},\\ & C7:\sum_{l=1}^{k}c_{il}\leq 1,\forall i\in\mathcal{N}\end{aligned}$$

where $C6$ ensures that each task is completed under the delay constraint, and $C7$ states that data can only be uploaded through at most one channel.

$\mathcal{P}2$ can also be transformed into a bilevel optimization problem. The offloading decision and channel selection are optimized in the upper level to minimize the total energy consumption of all mobile users, and the computation resource allocation is obtained in the lower level, with the purpose of minimizing the total computation energy consumption of all mobile users. This bilevel optimization problem is formulated as

$$\mathcal{P}3:\min_{\mathbf{o},\mathbf{c},\mathbf{r}}\quad \sum_{i=1}^{n}\left(\sum_{j=1}^{n}o_{ij}E^c_{ij}(r_{ij})+\sum_{j=0,j\neq i}^{n}\sum_{l=1}^{k}o_{ij}c_{il}E^t_{ijl}(\mathbf{c})\right) \tag{36}$$

$$\begin{aligned}\text{s.t. }\mathbf{r}\in\arg\min_{\mathbf{r}}&\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}o_{ij}E^c_{ij}(r_{ij}):C3-C6\right\}\\ & C1,C2,\text{and }C7\end{aligned}$$

We can see that $\mathcal{P}2$ and $\mathcal{P}3$ have the same optimal solution.

Proof: The proof is similar to Theorem 1 in Appendix of the supplementary file and it is omitted.

In fact, the change from $\mathcal{P}1$ to $\mathcal{P}3$ does not affect the

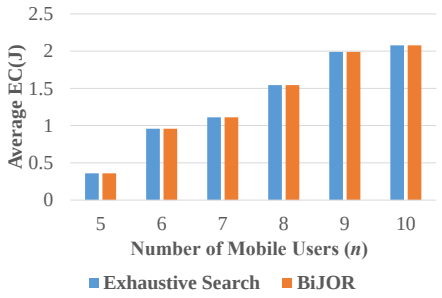


Fig. 4. Performance comparison of exhaustive search and BiJOR in terms of the average EC(J) value on instance suite I.

candidate mode pruning and the lower level optimization in BiJOR. However, compared with $\mathcal{P}1$, we need to not only optimize the offloading decision, but also select the channel in the upper level optimization of $\mathcal{P}3$; therefore, the upper level optimization of BiJOR should be updated.

In the upper level optimization, for each task, an ant selects a combination of a mode and a channel. The probability that each combination of mode j and channel l is assigned to U_i is calculated as

$$pr_{ijl} = \frac{\tau_{ijl}\eta_{ijl}^{\beta}}{\sum\limits_{h\in\mathcal{M}_i}\sum\limits_{v\in\mathcal{K}_i}\tau_{ihv}\eta_{ihv}^{\beta}},\ \forall i\in\mathcal{N}, j\in\mathcal{M}_i, l\in\mathcal{K}_i \quad (37)$$

where $\mathcal{K}_i$ denotes the candidate channel set of U_i.

The heuristic information is calculated as

$$\eta_{ijl} = \frac{1}{E_{ijl}^{I}},\ \forall i\in\mathcal{N}, j\in\mathcal{M}_i, l\in\mathcal{K}_i \quad (38)$$

where E_{ijl}^{I} represents the increment of the total energy consumption when mode j and channel l are assigned to U_i.

Subsequently, the combination of mode j and channel l is obtained as

$$(j,l) = \begin{cases} \arg\max\limits_{h\in\mathcal{M}_i, v\in\mathcal{K}_i} \tau_{ihv}\eta_{ihv}^{\beta}, & \text{if } q\le q_0 \\ J', & \text{otherwise} \end{cases} \quad (39)$$

where J' is a combination of a mode and a channel selected by the roulette wheel selection based on (37).

The fitness evaluation and pheromone management are similar to those in Section IV-E. Since there exists interference among mobile users in the cooperative computing mode in this scenario, the implementation of the local search will be quite complicated; thus, it is no employed any more.

In addition, since each ant selects a combination of a mode and a channel for each task in the offloading decision construction phase, each ant requires $n\cdot|\mathcal{M}_i|\cdot k$ computations. Consequently, the overall computational time complexity of BiJOR increases from $O(NP\cdot n^2)$ to $O(NP\cdot n^2\cdot k)$. Note, however, that we do not need to add any extra components to BiJOR in this scenario.

VI. Experimental Studies

A. Experimental Settings

Two instance suites with different scales were used to verify the effectiveness of BiJOR, which are instance suite I with a small number of mobile users, i.e., $n=\{6,7,8,9,10\}$, and instance suite II with a large number of mobile users, i.e., $n=\{20,50,80,100,120,150,200,300,400\}$.

The population size and maximum generation number of BiJOR were set as $NP=50$ and $gen_{max}=300$, and other parameter settings of BiJOR were consistent with [40]: $\beta=2, q_0=0.9, \varphi=\rho=0.1$, and $\tau_0=(n\cdot EC)^{-1}$, where EC is the total energy consumption obtained by the greedy search with the sorting strategy in Section S-III of the supplementary file. Moreover, BiJOR was independently run 30 times on each instance.

We assumed that all mobile users were randomly distributed in a 2000m $*$ 2000m region and the base station was located in the center of the region. Besides, C_i $(i\in\mathcal{N})$ was randomly distributed within $[10,2500]$Mega Cycles (MCycles), D_i $(i\in\mathcal{N})$ was randomly distributed within $[1,600]$KB, B_i $(i\in\mathcal{N})$ was randomly distributed within $[0.1,100]$KB, and the computing capability of each mobile device was a random one of four specifications, which are 0.5GHz, 0.8GHz, 1.0GHz, and 1.5GHz, respectively. Similar to [5], the channel gain was given as $H=d^{-4}$, where d is the propagation distance. In addition, T_i^{max} $(i\in\mathcal{N})$ was set to 1.0s, W was 20MHz, p_i^t and p_i^r $(i\in\mathcal{N})$ were 1.3W and 0.8W, respectively, w was -100dBm, r_0^{max} was 200GHz, and k was 10^{-27}.

In this paper, three performance indicators were adopted, i.e., the acceptance number (AN) [42], success rate (SR) [43], and total energy consumption (EC) [28]. AN records the number of tasks that can be completed under the delay constraint. SR is the percentage of successful runs over 30 independent runs. In this paper, a run is regarded as a successful run if all the tasks can be completed. If an algorithm can achieve 100% SR, EC is calculated based on (26).

B. Comparison with Exhaustive Search

To verify the effectiveness of BiJOR on small-scale instances, it was compared with exhaustive search on instance suite I. Exhaustive search is the brute-force search method and is capable of finding the global optimal solution, but suffers from high computational burden. In this subsection, only the third performance indicator, i.e., EC, was employed for comparison, since all the tasks can be completed by these two algorithms on each instance.

Fig. 4 shows the average EC values provided by BiJOR over 30 independent runs and the EC values resulting from exhaustive search. From Fig. 4, we can observe that BiJOR obtains the same results with exhaustive search on each instance, which means that BiJOR is able to find all the global optimal solutions of instance suite I.

C. Comparison with ACS-CLPSO

Instance suite II was further utilized to investigate the effectiveness of BiJOR on large-scale instances. Due to the high computational burden of exhaustive search on large-scale instances, it was no longer a competitor. Instead, a

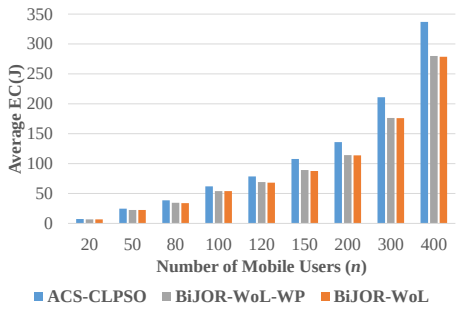
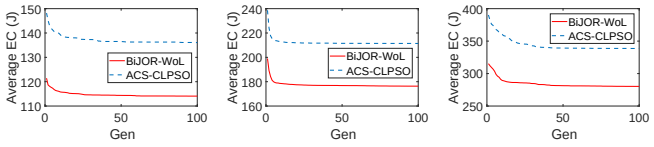


Fig. 5. Performance comparison of ACS-CLPSO, BiJOR-WoL-WP, and BiJOR-WoL in terms of the average EC(J) value on instance suite II.



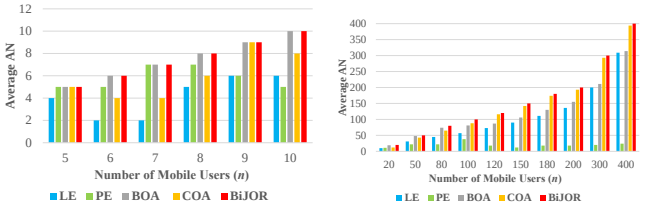
(a) $n = 200$ (b) $n = 300$ (c) $n = 400$

Fig. 6. Evolution of the average EC(J) values provided by ACS-CLPSO and BiJOR-WoL on three instances.

single-level optimization approach consisting of two state-of-the-art algorithms (i.e., ACS [40] and CLPSO [44]), called ACS-CLPSO, was under our consideration for comparison. CLPSO is a comprehensive learning particle swarm optimizer, in which a learning strategy is proposed to update a particle's velocity based on all other particles' historical best information. Different from the nested structure of BiJOR, the offloading decision and computation resource allocation of each solution in ACS-CLPSO were generated separately, so it was viewed as a single-level optimization algorithm. Both ACS-CLPSO and BiJOR employed ACS to generate the offloading decision, but the difference was that CLPSO, rather than the monotonic optimization method, was employed in ACS-CLPSO to optimize the computation resource allocation. The parameter settings of ACS in ACS-CLPSO were the same with those in BiJOR. In addition, the parameter settings of CLPSO were consistent with the original paper [44].

Since the local search of BiJOR is not suitable for ACS-CLPSO, a BiJOR variant without the local search, named as BiJOR-WoL, was compared with ACS-CLPSO for fairness. Both BiJOR-WoL and ACS-CLPSO were run 30 times independently.

Fig. 5 plots the average EC values of BiJOR-WoL and ACS-CLPSO over 30 independent runs. Moreover, the worst-case performance of BiJOR-WoL (called BiJOR-WoL-WP) over 30 independent runs was also presented. From Fig. 5, even BiJOR-WoL-WP shows better performance than ACS-CLPSO on all instances. In addition, BiJOR-WoL can provide an increasing advantage over ACS-CLPSO as the number of mobile users grows. Specifically, when $n = 20$, 50, 80, 100, 120, 150, 200, 300, and 400, the performance improvement rates of BiJOR-WoL against ACS-CLPSO are 6.45%, 8.88%, 12.34%, 12.82%, 13.30%, 15.00%, 16.29%, 16.58%, and



(a) Instance suite I (b) Instance suite II

Fig. 7. Performance comparison of BiJOR and four other algorithms in terms of the average AN value.

17.27%, respectively. In addition, Fig. 6 presents the evolution of the average EC values derived from BiJOR-WoL and ACS-CLPSO when $n = 200$, 300, and 400. From Fig. 6, it is clear that BiJOR-WoL can obtain high-quality solutions in the initial phase and then converges quickly. In contrast, ACS-CLPSO converges from poor initial solutions to local optimal solutions.

The above phenomenon can be explained as follows. In ACS-CLPSO, the offloading decision and computation resource allocation are generated independently, which may bring two drawbacks. On the one hand, the amount of the computation resources allocated may be greater than that of the computation resources available. It is because the available computation resources are determined by the offloading decision, but the allocated computation resources in ACS-CLPSO have no relationship with the offloading decision. On the other hand, it is very possible that ACS-CLPSO generates one solution consisting of a "good" offloading decision yet a "bad" computation resource allocation. Then, it is deemed unpromising as a whole based on (26) and will be discarded, which signifies that ACS-CLPSO cannot assess the quality of offloading decisions accurately. On the contrary, BiJOR-WoL is able to obtain the corresponding optimal allocation of computation resources and, as a result, the promising offloading decisions can be maintained.

D. Effectiveness of the CoMECO System

In this subsection, we are interested in studying the effectiveness of the CoMECO system. To this end, we compared BiJOR with the following four algorithms:

- Local execution (LE): All tasks are executed locally.
- P-MEC execution (PE): The P-MEC server accepts all the offloading requests.
- Binary MEC offloading algorithm (BOA): Tasks may be executed on local devices or the P-MEC server, as in [23] and [24].
- Cooperative MEC offloading algorithm (COA): Tasks may be executed on local devices or cooperative devices, as in [13].

Note that the resource allocation schemes of the above four algorithms are the same as that of BiJOR defined in Section IV-D.

The performance comparison between BiJOR and these four algorithms is presented in Fig. 7. Since LE, PE, BOA, and COA cannot guarantee 100% SR on all instances, the AN values were investigated. From Fig. 7, it can be observed that

BiJOR and BOA can obtain the best results on instance suite I. On instance suite II, BiJOR provides the best results, followed by COA. In addition, LE and PE show the worst performance on instance suite I and instance suite II, respectively. With respect to LE and PE, we would like to give the following comments:

- Owing to the physical limitations of mobile devices, it is difficult for them to complete tasks that require high computation resources. Therefore, LE is unable to ensure that all tasks can be completed even on small-scale instances. It is interesting to see that the rate between the mean AN value and the number of tasks in LE is similar on each case of instance suite II. It is because each task is executed locally and the interference among mobile users can be avoided. Thus, its performance is not affected significantly by the scale of the instances.
- The P-MEC server can provide rich computation resources for mobile users, which can help mobile users to execute resource-intensive tasks. Therefore, PE performs well on small-scale instances. However, due to the system bandwidth limitation, the P-MEC server can only accept a certain number of tasks. As a result, the performance of PE drastically degrades as the scale of the instances increases.

By combining the advantages of the P-MEC server and mobile devices, the CoMECO system can effectively alleviate the above problems and BiJOR achieves the best performance.

Remark 2. In the supplementary file, we investigated the feasibility of BiJOR in the dynamic scenario, effect of the upper level optimization method, and effectiveness of the sorting strategy and the local search. In addition, we also studied the effects of the number of mobile users and generations, channel number, and delay constraint.

VII. Conclusion

In this paper, a multi-user cooperative MEC offloading (CoMECO) system for delay-sensitive tasks in a multi-user interference environment was studied. A joint offloading decision and computation resource allocation problem was formulated to achieve the energy-efficient task execution in this system under the delay constraint. This problem was transformed into a bilevel optimization problem, in which the offloading decision is the upper level optimization problem and the computation resource allocation is the lower level optimization problem. Afterward, a bilevel optimization method (BiJOR) was proposed. In the upper level optimization of BiJOR, ant colony system (ACS) with a sorting strategy was adopted to seek the promising offloading decisions, and then the optimal resource allocation corresponding to each offloading decision was obtained by the monotonic optimization method in the lower level optimization. Moreover, if the iteration-best solution is a feasible solution, a local search operator was designed to accelerate the evolution. Additionally, BiJOR was extended to deal with a complex scenario with the channel selection.

BiJOR was applied to two instance suites with different scales and compared with exhaustive search and a single-level optimization method (ACS-CLPSO). The experimental results demonstrated the effectiveness of BiJOR. In addition, the superiority of the CoMECO system was tested by comparing BiJOR with four other algorithms. In the future, we will study the multi-objective optimization and take retransmissions into account in the CoMECO system.

References

[1] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *2012 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2012, pp. 59–66.

[2] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[3] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[5] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[6] I. Morris, "ETSI drops "mobile" from MEC," *Light Reading, New York, NY, USA, Sep*, 2016.

[7] 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.

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

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

[10] C. You and K. Huang, "Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4104–4117, 2018.

[11] Y. Tao, C. You, P. Zhang, and K. Huang, "Stochastic control of computation offloading to a dynamic helper," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[12] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet of Things Journal*, 2018, in press, DOI: 10.1109/JIOT.2018.2875246.

[13] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[14] C. Ren, X. Lyu, W. Ni, H. Tian, and R. P. Liu, "Distributed online learning of fog computing under nonuniform device cardinality," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1147–1159, 2019.

[15] K. Wang, K. Yang, H.-H. Chen, and L. Zhang, "Computation diversity in emerging networking paradigms," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 88–94, 2017.

[16] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Computer Networks*, vol. 108, pp. 357–370, 2016.

[17] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[18] S. Paris, F. Martignon, I. Filippini, and L. Chen, "An efficient auction-based mechanism for mobile data offloading," *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1573–1586, 2015.

[19] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *Communications Workshops (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 638–643.

[20] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[21] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–7.

[22] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[23] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[24] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*. IEEE, 2017, pp. 1–6.

[25] 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.

[26] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[27] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.

[28] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[29] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[30] M. Xiao, N. B. Shroff, and E. K. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 210–221, 2003.

[31] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy-aware offloading clustering approach (EAOCA) in fog computing," in *Wireless Communication Systems (ISWCS), 2017 International Symposium on*. IEEE, 2017, pp. 390–395.

[32] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: from classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2018.

[33] J. Xie, Y. Mei, A. T. Ernst, X. Li, and A. Song, "A bi-level optimization model for grouping constrained storage location assignment problems," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 385–398, 2018.

[34] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016.

[35] T. Liao, K. Socha, M. A. M. de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 503–518, 2014.

[36] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, 2017.

[37] Y. Wang, D. Yin, S. Yang, and G. Sun, "Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1642–1656, May 2019.

[38] Z. Liu, Y. Wang, S. Yang, and K. Tang, "An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1403–1416, April 2019.

[39] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113–128, 2018.

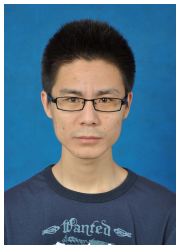
[40] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[41] J. Lee, Y. Kim, Y. Kwak, J. Zhang, A. Papasakellariou, T. Novlan, C. Sun, and Y. Li, "LTE-advanced in 3GPP Rel-13/14: an evolution toward 5G," *IEEE Communications Magazine*, vol. 54, no. 3, pp. 36–42, 2016.

[42] T. Li, C. S. Magurawalage, K. Wang, K. Xu, K. Yang, and H. Wang, "On efficient offloading control in cloud radio access network with mobile edge computing," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2258–2263.

[43] Y. Wang, B.-C. Wang, H.-X. Li, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2938–2952, 2016.

[44] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.



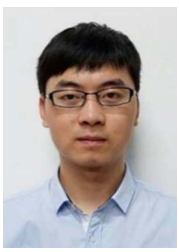
Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering both from the Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in control science and engineering, Central South University, Changsha, China. His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.

Yong Wang (M'08–SM'17) received the Ph.D. degree in control science and engineering from the Central South University, Changsha, China, in 2011.

He is a Professor with the School of Automation, Central South University, Changsha, China. His current research interests include the theory, algorithm design, and interdisciplinary applications of computational intelligence.

Dr. Wang is an Associate Editor for the *Swarm and Evolutionary Computation*. He was a Web of Science highly cited researcher in Computer Science in 2017 and 2018.

Kezhi Wang received the B.E. and M.E. degrees in School of Automation from Chongqing University, China, in 2008 and 2011, respectively. He received the Ph.D. degree in Engineering from the University of Warwick, U.K. in 2015. He was a senior research officer in University of Essex, U.K. Currently he is a Lecturer in Department of Computer and Information Sciences at Northumbria University, U.K. His research interests include wireless communication, mobile edge computing and artificial intelligence.



Zhi-Zhong Liu received the B.S. degree in automation from Central South University, Changsha, China, in 2013, where he is currently pursuing the Ph.D. degree in control science and engineering. His current research interests include evolutionary computation, bioinformatics, swarm intelligence, constrained optimization, and multimodal optimization.

Supplementary File for "A Bilevel Optimization Approach for Joint Offloading Decision and Resource Allocation in Cooperative Mobile Edge Computing"

S-I. Feasibility of BiJOR in the Dynamic Scenario

TABLE S-I
The Maximum Time to Complete All Tasks on Instance Suite II in the Dynamic Scenario.

n	20	50	80	100	120	150	200	300	400
T(s)	1.0011	1.0027	1.0017	1.0003	1.0017	1.002	1.0085	1.0017	1.0019

In previous experiments, all the scenarios are static. However, the real scenarios may be dynamic. To investigate the feasibility of BiJOR in the dynamic scenario, we considered a dynamic scenario where mobile users are moving and the maximum speed is 5m/s. Due to the fact that the delay constraint T_i^{max} was set to 1s in this paper, the distance between the positions where each mobile user uploads data and receives results is no more than 5m. Therefore, we first defined a circular area. The center of this circular area was the location where a mobile user uploads data and its radius was 5m. The location where the mobile user receives results was then randomly generated within this circular area. Subsequently, the offloading decision and computation resource allocation obtained by BiJOR in the static scenario were applied to this dynamic scenario and the maximum time to complete all tasks on instance suite II are recorded in Table S-I.

As shown in Table S-I, the maximum time to complete all tasks exceeds the delay constraint, which means that some tasks cannot be completed under the delay constraint due to the mobility of mobile users. However, the maximum time in each instance is less than 1.01s, i.e., the time exceeded is less than 0.01s. If this exceeded time is acceptable, it is clear that BiJOR is feasible. If not, we can tighten the delay constraint in the optimization process. Specifically, we set T_i^{max} to 0.99s. Through experiments, we find that in this way, all tasks are completed under the delay constraint. We can conclude that BiJOR in the dynamic scenario is still feasible.

S-II. Effect of the Upper Level Optimization Approach

TABLE S-II
PERFORMANCE COMPARISON OF BIJOR-GS, BIJOR-PSO, AND BIJOR IN TERMS OF THE AVERAGE AN VALUE, SR VALUE, AND AVERAGE EC(J) VALUE ON INSTANCE SUITE II.

n	BiJOR-GS			BiJOR-PSO			BiJOR		
	Average AN	SR	Average EC	Average AN	SR	Average EC	Average AN	SR	Average EC
20	20.0	100.0%	7.9624	14.2	0.0%	\	20.0	100.0%	6.8197
50	50.0	100.0%	30.6620	31.6	0.0%	\	50.0	100.0%	22.4660
80	79.8	90.0%	\	47.4	0.0%	\	80.0	100.0%	33.8958
100	98.2	6.7%	\	58.5	0.0%	\	100.0	100.0%	53.2827
120	117.7	0.0%	\	72.1	0.0%	\	120.0	100.0%	66.9925
150	141.1	0.0%	\	87.8	0.0%	\	150.0	100.0%	86.2902
200	198.2	0.0%	\	116.9	0.0%	\	200.0	100.0%	112.5012
300	282.3	0.0%	\	172.4	0.0%	\	300.0	100.0%	174.2470
400	395.0	0.0%	\	245.6	0.0%	\	400.0	100.0%	273.5903

The upper level optimization plays a vital role in BiJOR because BiJOR may be inefficient if the upper level optimization is unable to find the promising offloading decisions. To verify this, an additional experiment was conducted, in which greedy search and PSO were adopted to replace ACS as the upper level optimization approach of BiJOR, and the resultant algorithms were recorded as BiJOR-GS and BiJOR-PSO, respectively. BiJOR-GS firstly randomly generates task priorities, and then each task is assigned a mode in turn. Specifically, if a feasible candidate mode set of a task is not empty, the mode that generates the smallest increment of total energy consumption is assigned to this task; otherwise, this task will not be executed. In BiJOR-PSO, a discrete version PSO (BPSO) was adopted as the upper level optimization approach [1]. Since it cannot be directly used to optimize the offloading decision with multiple candidate modes, BPSO was slightly modified to assign the mode with the maximum speed to the task. Our experiment focused on instance suite II and the AN values, SR values, and average EC values of BiJOR-GS, BiJOR-PSO, and BiJOR over 30 independent runs are presented in Table S-II.

As shown in Table S-II, the average EC values derived from BiJOR-GS are greater than those derived from BiJOR on the instances with a small number of mobile users, e.g., $n = 20$ and 50. In addition, in the case of $n \geq 80$, BiJOR-GS cannot ensure that all tasks can be completed in each run. When $n \geq 120$, BiJOR-GS cannot provide any successful run. In addition, the average AN value derived from BiJOR-PSO is less than the number of mobile users on each instance, which means that BiJOR-PSO cannot guarantee that all tasks are completed under the delay constraint on each instance. Moreover, we can observe that the SR value provided by BiJOR-PSO is 0% on each instance, which further demonstrates that BiJOR-PSO cannot achieve any successful run. The above experimental results suggest that as the upper level optimization approach of BiJOR, ACS is significantly superior to greedy search and PSO, and that the upper level optimization method greatly affects the performance of BiJOR.

S-III. Effectiveness of the Sorting Strategy

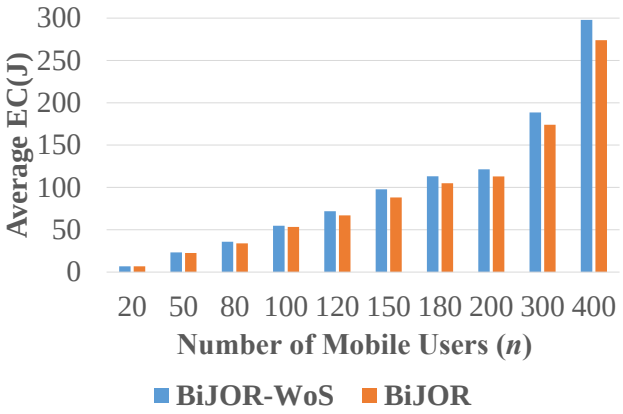


Fig. S-1. Performance comparison of BiJOR-WoS and BiJOR in terms of the average EC(J) value on instance suite II.

TABLE S-III
Performance comparison of BiJOR and BiJOR-GS-WS in terms of the SR value and average EC(J) value on instance suite II.

n	BiJOR-GS-WS		BiJOR	
	SR	Average EC	SR	Average EC
20	100.0%	7.4973	100.0%	6.8197
50	100.0%	22.4845	100.0%	22.4660
80	100.0%	49.1191	100.0%	33.8958
100	100.0%	62.9260	100.0%	53.2827
120	100.0%	80.3290	100.0%	66.9925
150	100.0%	113.0946	100.0%	86.2902
200	100.0%	127.2854	100.0%	112.5012
300	100.0%	202.3134	100.0%	174.2470
400	100.0%	321.1865	100.0%	273.5903

In this paper, a sorting strategy is implemented in the offloading decision construction of BiJOR. In order to study how the performance of BiJOR is influenced by this strategy, a variant of BiJOR, called BiJOR-WoS, was proposed, in which the sorting strategy was replaced with the random sorting. The experimental results of BiJOR-WoS and BiJOR are presented in Fig. S-1. As demonstrated in Fig. S-1, BiJOR-WoS and BiJOR almost achieve the same average EC values when $n = 20$, 50, and 80. However, the performance difference between BiJOR-WoS and BiJOR is clear in terms of a large number of mobile users. This is because the sorting strategy has the capability to produce feasible offloading decisions with a higher probability, which is beneficial to seek the optimal solution.

Actually, this sorting strategy can also be incorporated into greedy search since it needs to generate task priorities before assigning a mode to a task. A question which arises naturally is: how this sorting strategy affects the performance of greedy search. The performance of greedy search with this sorting strategy (denoted as BiJOR-GS-WS) was validated by comparing with BiJOR on instance suite II.

Table S-III presents the SR values and average EC values obtained by BiJOR-GS-WS and BiJOR over 30 independent runs. As shown in Table S-III, BiJOR-GS-WS can ensure that all tasks are completed under the delay constraint. Hence, this sorting strategy can significantly improve the performance of greedy search. But it is worth noting that the performance of BiJOR still has an edge over that of BiJOR-GS-WS in terms of the average EC value, which suggests again that ACS is a better choice for the upper level optimization approach of BiJOR.

S-IV. Effectiveness of the Local Search

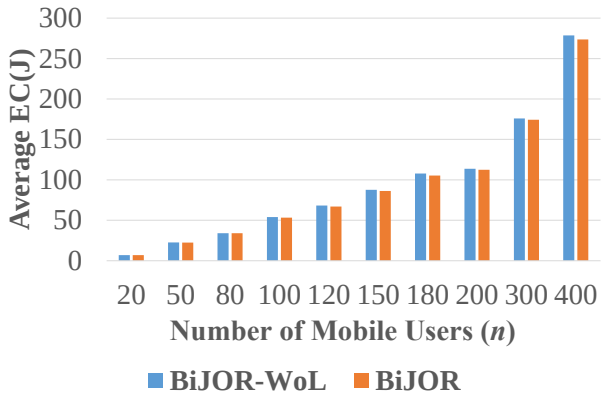
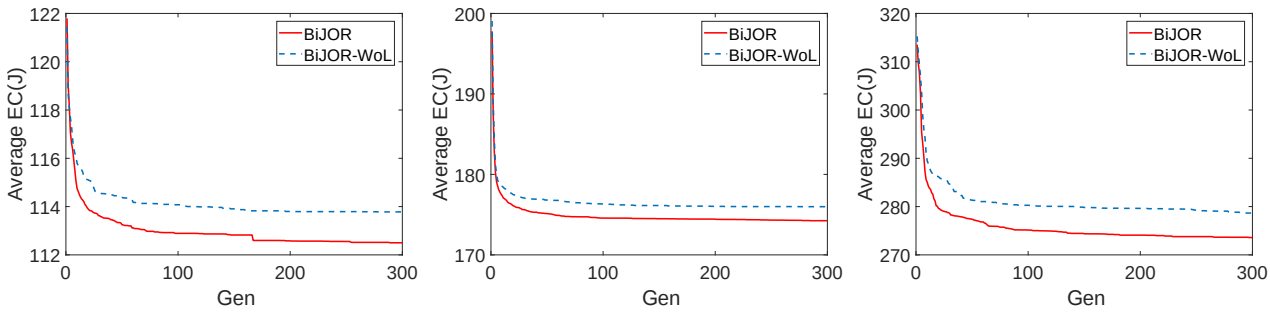


Fig. S-2. Performance comparison of BiJOR-WoL and BiJOR in terms of the average EC(J) value on instance suite II.



(a) $n = 200$ (b) $n = 300$ (c) $n = 400$

Fig. S-3. Evolution of the average EC(J) values provided by BiJOR-WoL and BiJOR on instance suite II.

In this subsection, we investigated the contribution of the proposed local search operator on the performance of BiJOR. To this end, we compared the performance of BiJOR with BiJOR-WoL on instance suite II. Fig. S-2 reports the comparison results of these two algorithms in terms of the average EC value.

As can be seen, on all instances except n=20, 50, 80, and 100, BiJOR performs better than BiJOR-WoL. To further validate the effectiveness of the local search operator, Fig. S-3 shows the evolution of the average EC values provided by BiJOR-WoL and BiJOR when n = 200, 300, and 400. From Fig. S-3, it is clear that the convergence of BiJOR is significantly faster than that of BiJOR-WoL. The above experimental results verify the effectiveness of the local search operator.

S-V. Effect of the Number of Mobile Users and Generations

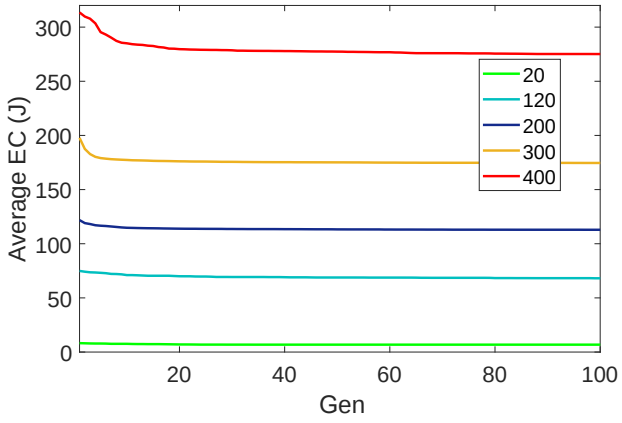


Fig. S-4. Evolution of the average EC(J) values provided by BiJOR for five different numbers of mobile users.

To study the effect of the number of mobile users and generations, we considered the number of mobile users with five different values: n = 20, 120, 200, 300, and 400. The evolution of the average EC values provided by BiJOR is shown in Fig. S-4. From Fig. S-4, with the increasing number of mobile users, BiJOR needs more generations to converge. However, BiJOR can converge in all cases before 100 generations.

S-VI. Effect of the Channel Number

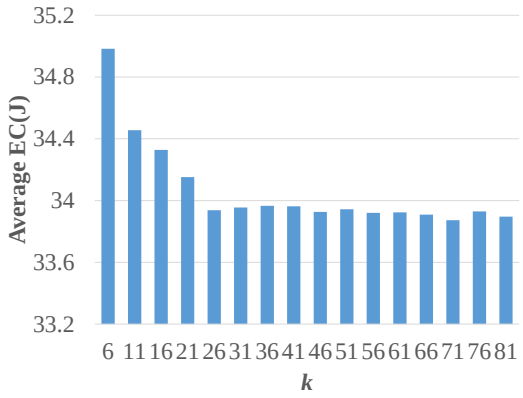


Fig. S-5. Average EC(J) values of the CoMECO system with 16 different k in the case of $n = 80$.

In this subsection, we studied the effect of the channel number on the performance of the CoMECO system in the complex scenario with the channel selection. We tested the CoMECO system with 16 different channel numbers in the case of $n = 80$: $k = 6, 11, \ldots, 81$. The experimental results are shown in Fig. S-5. From Fig. S-5, the average EC value decreases as k increases, and when $k \geq 26$, it remains nearly the same. The reason is explained in the following. When k is small, the interference among mobile devices is severe, resulting in a low transmission rate and high transmission energy consumption. Meanwhile, the demand of computation resources increases correspondingly, which leads to an increase in the computation energy consumption. As k increases, the interference among mobile devices becomes weak and the average EC value decreases. When k is large, some channels are redundant; thus, the average EC value remains nearly unchanged.

S-VII. Effect of the Delay Constraint

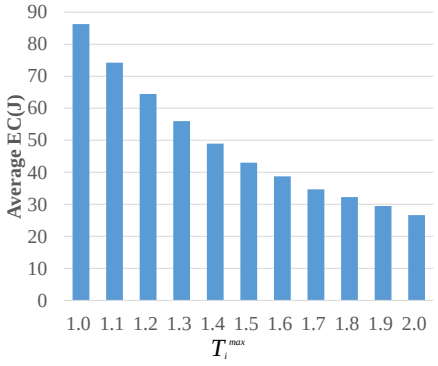
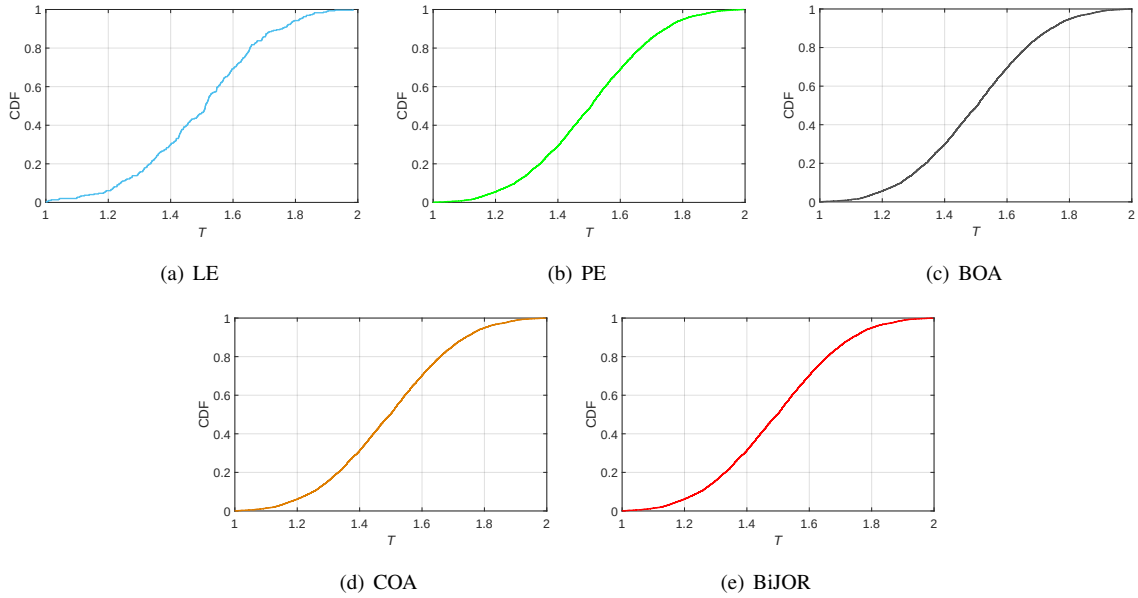


Fig. S-6. Average EC(J) values of the CoMECO system with 11 different T_i^{max} in the case of $n = 150$.



(a) LE (b) PE (c) BOA

(d) COA (e) BiJOR

Fig. S-7. CDFs of delays experienced by the mobile users for LE, PE, BOA, COA, and BiJOR.

To study the effect of the delay constraint T_i^{max}, we considered T_i^{max} with 11 different values in the case of $n = 150$: $T_i^{max} = 1.0, 1.1, \ldots, 2.0$. The average EC values over 30 independent runs are plotted in Fig. S-6. From Fig. S-6, it is clear that the average EC value continues to decrease as T_i^{max} grows. It can be attributed to the fact that the demand of computation resources decreases with the increase of T_i^{max}.

Furthermore, we studied the cumulative distribution functions (CDFs) of delays experienced by the mobile users for LE, PE, BOA, COA (see Section VII-D), and BiJOR. The delay constraint T_i^{max} follows a normal distribution over the interval [1, 2] with mean 1.5 and variance 0.2. Note that we only considered the mobile users whose tasks can be completed under the delay constraint. The experimental results are presented in Fig. S-7. From Fig. S-7, it is clear that the CDFs of delays experienced by the mobile users for LE, PE, BOA, COA, and BiJOR are almost the same. It can be attributed to the fact that the resource allocation schemes in these five algorithms are the same. Specifically, the optimal resource allocation is equal to the minimum demand of computation resources according to (20). Therefore, the delay experienced by each mobile user is T_i^{max}.

APPENDIX

A. Proof of Lemma 1

Proof: Let $\{\mathbf{o}^\star, \mathbf{r}^\star\}$ be the optimal solution of the original problem $\mathcal{P}$. Due to the fact that $\{\mathbf{o}^\star, \mathbf{r}^\star\}$ must satisfy constraints $C1-C6$, we only need to prove that $\mathbf{r}^\star$ is the optimal solution of the lower level optimization problem of the bilevel optimization problem $\mathcal{P}1$.

Assuming that $\mathbf{r}^\star$ is not the optimal solution of the lower level optimization problem. Then, there exits a feasible solution $\mathbf{r}'$ of the lower level optimization problem such that

$$\sum_{j=1}^{n} o_{ij}^\star E_{ij}^c(r_{ij}^{'}) < \sum_{j=1}^{n} o_{ij}^\star E_{ij}^c(r_{ij}^\star) \tag{S-1}$$

It is clear that $\{\mathbf{o}^\star, \mathbf{r}'\}$ is also a feasible solution of $\mathcal{P}$. Then, substituting (S-1) to the objective function of $\mathcal{P}$, one can obtain

$$\begin{aligned} &\sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij}^\star E_{ij}^c(r_{ij}^{'}) + \sum_{j=0,j\neq i}^{n} o_{ij}^\star E_{ij}^t(\mathbf{o}^\star)\right) \\ &< \sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij}^\star E_{ij}^c(r_{ij}^\star) + \sum_{j=0,j\neq i}^{n} o_{ij}^\star E_{ij}^t(\mathbf{o}^\star)\right) \end{aligned} \tag{S-2}$$

It is contradictory with the assumption that $\{\mathbf{o}^\star, \mathbf{r}^\star\}$ is the optimal solution of $\mathcal{P}$. Therefore, $\mathbf{r}^\star$ is the optimal solution of the lower level optimization problem, and further the optimal solution of $\mathcal{P}$ is a feasible solution of $\mathcal{P}1$. ■

B. Proof of Theorem 1

Proof: Let $\{\mathbf{o}^\star, \mathbf{r}^\star\}$ be the optimal solution of $\mathcal{P}$. By applying Lemma 1, it is a feasible solution of $\mathcal{P}1$. Suppose the optimal solution of $\mathcal{P}1$ is $\{\mathbf{o}', \mathbf{r}'\}$, rather than $\{\mathbf{o}^\star, \mathbf{r}^\star\}$.

Based on the upper level optimization objective of $\mathcal{P}1$, one can obtain

$$\begin{aligned} &\sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij}' E_{ij}^c(r_{ij}^{'}) + \sum_{j=0,j\neq i}^{n} o_{ij}^{'} E_{ij}^t(\mathbf{o}^{'})\right) \\ &< \sum_{i=1}^{n}\left(\sum_{j=1}^{n} o_{ij}^\star E_{ij}^c(r_{ij}^\star) + \sum_{j=0,j\neq i}^{n} o_{ij}^\star E_{ij}^t(\mathbf{o}^\star)\right) \end{aligned} \tag{S-3}$$

Obviously, $\{\mathbf{o}^\star, \mathbf{r}^\star\}$ is not the optimal solution of $\mathcal{P}$, so this is a contradiction. Therefore, the optimal solution of $\mathcal{P}$ is also the optimal solution of $\mathcal{P}1$. Similarly, it can also be proven that the optimal solution of $\mathcal{P}1$ is the optimal solution of $\mathcal{P}$. ■

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5. IEEE, 1997, pp. 4104–4108.