Cooperative Computing for Mobile Crowdsensing: Design and Optimization

Xin Xie, *Graduate Student Member, IEEE,* Tong Bai, *Member, IEEE,* Weiwei Guo, Zhipeng Wang, *Member, IEEE,* and Arumugam Nallanathan, *Fellow, IEEE*

Abstract—With the increasing number of mobile devices, mobile crowdsensing (MCS) has garnered significant attention in research. However, computing infrastructures such as edge/cloud nodes, which are necessary for processing sensor data, are not always readily available. To address this issue, we propose a cooperative computing framework that enables the offloading of sensor data to nearby mobile devices with unused computational resources (known as helpers) for processing. Our approach considers a scenario with multiple sources and multiple helpers, where computational tasks can be partially offloaded to several helpers. We jointly optimize task offloading strategy, communication resources, and computational resources to minimize the weighted sum energy consumption of mobile devices. We model the optimization problem as a mixed-integer nonlinear programming (MINLP), with the source-helper assignment solved using a distributed algorithm based on matching theory, and the joint task partition and resource allocation problem solved using an alternating optimization (AO) method. Simulation results demonstrate the efficacy of our cooperative computing framework and scheduling scheme, which offer significant advantages over local computing in terms of reducing the weighted sum energy consumption and improving the task completion ratio.

Index Terms—Mobile crowdsensing (MCS), cooperative computing

1 Introduction

Mobile crowdsensing (MCS) is emerging as a promising paradigm for a wide range of applications, thanks to the rapid development of the Internet-of-Things (IoT), embedded artificial intelligence, integrated sensing and communication (ISAC) technology, and the proliferation of mobile devices. This convergence of technologies has enabled large-scale and real-time data sensing [1], significantly improving the daily lives of citizens and providing fresh perspectives for urban societies [2]. In contrast to traditional sensor networks like IoT, which are the primary candidates for deploying sensing infrastructure for smart city applications, MCS stands out for three key characteristics: mobility, crowd participation, and diverse sensing tasks. The dynamic nature of mobile devices necessitates the development of strategies to adapt to their movements and changing environmental conditions. By leveraging the power of crowds, MCS utilizes the built-in sensors of mobile devices to collaboratively perform sensing tasks, eliminating the need for dedicated sensing devices. Furthermore, the diverse range of mobile devices with their multitude of sensors enables MCS to handle a wide variety of sensing tasks. As a result, MCS offers several advantages, including low network costs, flexible deployment, and easy maintenance. Its applications span various domains, such as traffic management [3], environment monitoring [4], and collision warning [5].

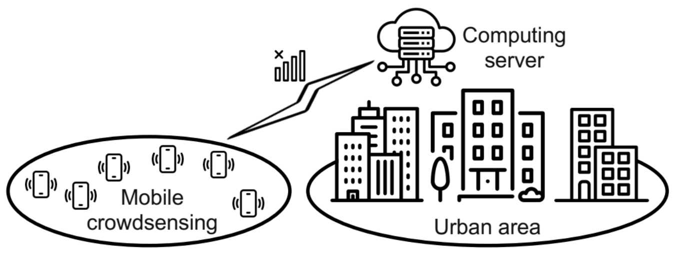

Fig. 1. Illustration of MCS in the absence of power computing infrastructure.

The state-of-the-art techniques in MCS have primarily focused on various aspects. Due to differences in location and data accuracy of mobile devices, some researchers have studied the assignment problem of sensing tasks [6], [7]. The authors in [8], [9], [10] have investigated the issues of user privacy and data aggregation. Additionally, data analysis plays a crucial role in extracting useful information from various types of user sensor data [11], [12]. However, it is worth noting that most of the current research predominantly focuses on the sensor data aspect, with little consideration given to the computation of data itself.

As the volume of sensor data grows, mobile devices are typically incapable of processing it by themselves [13], [14], [15]. To tackle this challenge, conventional solutions like mobile edge computing (MEC) [16], [17], fog computing [18], and cloud computing [19] have been proposed. Nevertheless, accessing a powerful computing infrastructure is not always convenient or readily available for all scenarios [20], [21], [22], [23]. Therefore, the network needs to be

- *X. Xie and Z. Wang are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China. E-mail: {xiexincqbz, wangzhipeng}@buaa.edu.cn.*
- *T. Bai is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China. E-mail: tongbai@buaa.edu.cn.*
- *W. Guo is with the Shanxi Supercomputing Center, Shanxi 033000, China. E-mail: 13700500052@163.com.*
- *A. Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. E-mail: a.nallanathan@qmul.ac.uk.*

capable of processing sensor data using its own computational resources. Bearing in mind that due to the mobility and random distribution of devices in MCS systems, not all devices are suitable for participating in sensing tasks at a given time. This means that sensing tasks are assigned to specific mobile devices selectively [6], [7] and the computational tasks typically occur sporadically or impulsively [24], it implies that there might be idle devices within the network that can provide additional computational resources. This specific advantages has inspired us to propose a mobile cooperative computing framework that can adapt to situations where access to powerful computing nodes is absent, as depicted in Figure 1. Specifically, we group the mobile sensing devices in MCS into source devices (SDs) and helper devices (HDs) during each time slot. A portion or even all of the computational tasks at the SDs can be offloaded to nearby helpers. By doing so, we can leverage the computational resources available on idle devices to execute computational tasks beyond the capabilities of a single device. Consequently, this approach eliminates the dependency on edge/cloud computing servers.

To fully exploit the benefits of our center-less cooperative computing framework, it is crucial to carefully design both the task offloading strategy decision and the allocation of communication and computational resources. Numerous researchers have investigated resource scheduling schemes in similar frameworks. For instance, authors in [25] proposed an efficient low-complexity algorithm to solve the resource allocation problem in ISAC-aided wireless ad hoc networks, aiming to achieve lower transmission delay. In [26], the control of power allocation, sensing, compression, and transmission was jointly investigated within a wirelessly powered crowdsensing framework. However, jointly optimizing coupled variables in the proposed distributed system is still a challenging task due to several factors. Firstly, the formulation of the proposed framework takes into account all design requirements and constraints pertaining to both communications and computations. The optimization of coupled variables gives rise to a highly complex mixed-integer nonlinear programming (MINLP) problem, which is notoriously difficult to solve. Secondly, the absence of centralized scheduling in the distributed system, coupled with the need to schedule multiple resources, can lead to substantial overhead during information interaction for scheduling purposes. If the scheduling schemes employed are not rational, there is a risk of the sensing tasks not being completed on time.

In order to facilitate the on-time computational task execution for the system in the absence of powerful computing infrastructure, we conceive a novel cooperative computing framework, which enables the source-helper assignment and task partition as well as communication and computational resource allocation in a distributed manner. The main contributions of this paper are highlighted as follows.

- *Cooperative computing framework design:* We propose a cooperative computing framework for MCS that leverages orthogonal frequency division multiple access (OFDMA)-based device-to-device (D2D) links for task offloading. This is the first treatise jointly considering offloading strategy decision and resource allocation problem for distributed MCS systems. The weighted sum energy consumption of all sensing and computing devices is minimized, by jointly optimizing the offloading strategy, including source-helper assignment and task partition, the communication resource allocation, including the sub-carriers and transmission power, and the computational resource allocation, subject to the application delay constraint. The optimization problem is formulated as a MINLP problem, which is solved by a sophisticated solution that uses a matching theory-based distributed algorithm to partition the mobile devices into several clusters, while optimizing the remaining variables using an alternating optimization (AO) method.
- *Cooperative computing cluster formulation algorithm:* To reduce the overhead of information interaction, we design a dispersive resource efficiency based source-helper assignment (DRESHA) algorithm for solving the source-helper assignment problem. Specifically, helpers are assigned to SDs to form multiple non-overlapped cooperative computing clusters, and then some helpers may be reassigned within clusters based on the task partition result. The cooperative computing clusters are obtained after a number of iterations, until the convergence is attained.
- *Task partition and resource allocation algorithm:* To solve the weighted sum energy consumption minimization problem in each cooperative cluster, we propose a joint task partition and resource allocation (JTPRA) algorithm, which decomposes the original problem into two subproblems, namely, task partition problem and resource allocation problem. A low-complexity solution was proposed relying on the AO method.
- *Numerical validations and evaluations:* Extensive numerical results are presented to demonstrate the effectiveness of the proposed framework and scheduling scheme proposed, showing significant reductions in energy consumption while ensuring task completion ratios under given delay constraints.

The remainder of the paper is structured as follows: Section 2 presents the related works of cooperative computing, Section 3 introduces the system model and problem formulation, Section 4 explains the cooperative computing cluster formulation algorithm, Section 5 proposes a solution for the weighted sum energy consumption minimization problem in each cluster, Section 6 presents numerical results obtained from simulations, and finally, Section 7 summarizes our findings and draws conclusions.

2 Related Works

In this section, the related works of cooperative computing are reviewed by classifying them into MCS, D2D-enabled MEC, and fog computing scenarios, as follows.

2.1 Cooperative Computing in MCS

As sensor data becomes increasingly complex and fine-grained, computing offloading techniques are introduced to enhance MCS, reducing time delays and high bandwidth costs. In recent years, most work has been done under the framework of centralized servers [27], [28], [29], such as MEC, fog computing, and cloud computing. For example, Zhou *et al.* [30] combined deep learning and edge computing techniques with MCS to provide robust data validation and local data processing. To preserve privacy and deal with malicious participants, Ma *et al.* [31] proposed two privacy-preserving reputation management schemes for edge computing enhanced MCS. In [32], the authors proposed a novel framework integrating mobile sensing and crowd computing, utilizing the crowd to solve problems without involving cloud servers in the backend. While the authors verified the feasibility of the framework, they did not address the resource scheduling issues inherent in the framework. This consolidate our motivation to study the problem of offloading strategy decision and resource allocation in generic scenarios.

2.2 Cooperative Computing in D2D-enabled MEC

Similar to our mobile cooperative computing framework for MCS, there has been some researches on D2D-enabled MEC, which allows mobile devices to offload tasks to neighboring devices via D2D communication links. Specifically, Chen *et al.* [33] proposed a novel D2D crowd framework with a bipartite-matching-based binary task assignment policy, but without considering communication and computational resource allocation. Furthermore, He *et al.* [34] conceived a resource allocation scheme for an OFDMA-based D2D-enabled MEC scenario, where the number of devices supported by the cellular networks is maximized by jointly optimized task splitting, transmission power, and computational resource, but they limited the number of available D2D links per task device. For eliminating this shortage, a D2D-enabled single-user and multi-helper MEC system was proposed in [35], the computing delay is minimized by optimizing users task assignment jointly with the time and rate for task offloading, as well as the computing frequency. Authors in [36], [37] studied a multi-user cooperative mobile edge computing offloading system, in order to minimize the total energy consumption of all mobile users under the delay constraint, a two-level alternation method framework was proposed to solve the challenging optimization problem. However, these aforementioned studies deal with offloading strategy and resource allocation in a centralized manner, which is not feasible for the center-less architecture of our proposed framework. Therefore, it is necessary to design a distributed scheduling scheme for the mobile cooperative computing framework in MCS.

2.3 Cooperative Computing in Fog Computing

The task offloading strategy has been extensively researched in the field of fog computing due to the high demand for computing offloading from mobile devices, the limited number of tasks that a single fog node can support requires the collaboration of multiple fog nodes through cooperative computing. To optimize the use of fog computing, various offloading strategies have been studied in the past few years. Most existing scheduling schemes utilize self-organizing distributed methods such as game theory [38], [39], [40] and matching theory [41], [42], [43]. Unfortunately, the majority of these studies focus on the offloading strategy component and do not consider resource allocation. For instance, Chen [38] proposed a game theory-based approach to the distributed computing offloading decision-making problem in a multi-channel wireless interference environment. In [41], a matching-theory based distributed algorithm was introduced to solve the joint source-helper assignment and task partition problem in a heterogeneous fog network. While there are some differences between the these studies and the problem at hand, their distributed algorithm design ideas are valuable references for this paper.

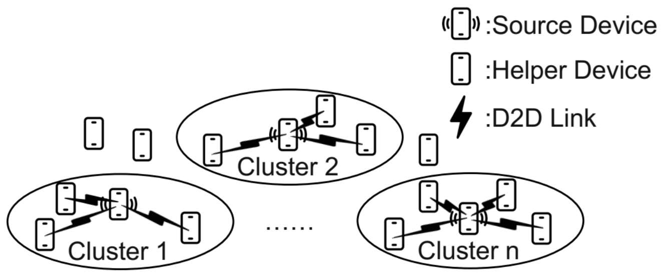

Fig. 2. Illustration of cooperative computing framework

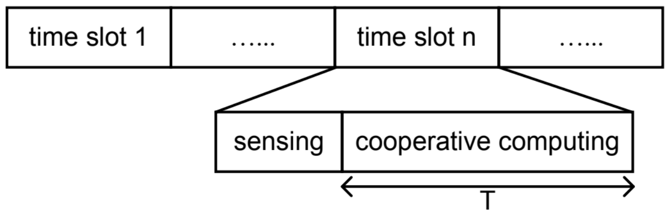

Fig. 3. Illustration of cooperative computing workflow

In summary, previous researches have focused on centralized server frameworks, but we propose a distributed scheduling scheme for cooperative computing in MCS, considering both offloading strategy decision and resource allocation. We draw inspiration from game theory-based approaches and matching-theory based algorithms in fog computing. Our contribution fills the gap in existing research by addressing resource scheduling issues and proposing a center-less architecture for cooperative computing.

3 System Model and Problem Formulation

As shown in Figure 2, we consider an MCS scenario, comprising $|\mathcal{U}|$ single-antenna mobile sensing devices. A subset of these devices is responsible for executing sensing tasks within a specified delay constraint. The offloading strategy decision and resource allocation are based on time slots as the fundamental unit [44], [45], [46]. As depicted in Figure 3, each time slot consists of two phases, namely, sensing and computing phases. In the sensing phase, mobile devices gather sensor data, which is subsequently processed in the computing phase. It is assumed that the data processing must be completed within a duration of T. During the computing phase, mobile devices are categorized into two types: SDs and HDs. To elaborate, SDs refer to the devices engaged in sensing tasks during the sensing phase, denoted by set $\mathcal{N}$. The remaining idle devices are referred to as HDs and are represented by set $\mathcal{K}$ (i.e., $\mathcal{U} = \mathcal{K} \cup \mathcal{N}$). Each HD can assist at most one SD in any given time slot [1]. For ease of reference, we use $\mathcal{K}_n$ to denote the set of HDs assigned to SD n, forming a cooperative computing cluster as illustrated in Figure 2. The notation $\{0\} \cup \mathcal{K}_n$ represents all mobile devices in the n-th cluster, where the element 0 is the cluster head (i.e., SD n). In this notation, subscript n_k indicates mobile device k within cluster $\{0\} \cup \mathcal{K}_n$. Similar to most of the previous works [17], [48], [49], [50], [51], we assume partial offloading for each SD. To enable concurrent task offloading and effective utilization of channel resources, communication for task offloading is carried out using the OFDMA scheme, which comprises $|\mathcal{M}|$ orthogonal sub-carriers. Additionally, we assume that the sub-carriers cannot be reused in SD-HD links. The sub-carriers assigned to the n-th cluster are denoted by $\mathcal{M}_n$.

3.1 Channel Model

The channel gain $h_{a,b}^m$ between mobile device a and b over the sub-carrier m is modeled as the product of path loss $\Upsilon_{a,b}$ and small-scale fading coefficient $g_{a,b}^m$, i.e.,

$$h_{a,b}^m = \Upsilon_{a,b} \left| g_{a,b}^m \right|^2. \tag{1}$$

Here, the path loss is determined by the distance between the two mobile devices, represented by $d_{a,b}$, and is given by

$$\Upsilon_{a,b} = \hat{\Upsilon} d_{a,b}^{-\varepsilon}, \tag{2}$$

where $\hat{\Upsilon}$ denotes the average channel power gain at the reference distance of one meter, and ε is the path loss exponent. Small-scale fading is assumed to be independent and identically distributed (i.i.d.) and follows a complex Gaussian distribution having zero mean and unit variance. We assume a block fading channel model. In other words, the channel gain between any two mobile devices remains unchanged during each time slot.

3.2 Computing Model

The computational task of SD n can be represented as a two-tuple $\{V_n, C_n\}$, where V_n denotes the input data size of the task, and C_n denotes the number of CPU cycles required to compute one bit of the task. The deadline of all tasks is the duration of cooperative computing phase T. In this paper, we consider the partitionable tasks, where the input data can be arbitrarily partitioned for parallel processing. The computational task of SD n can be divided into multiple fractions for local computing and for cooperative computing, respectively. Denoting by V_{n_k} the data size of the task to be computed on mobile device n_k, we have

$$V_n = V_{n_0} + \sum_{k=1}^{|\mathcal{K}_n|} V_{n_k}. \tag{3}$$

1. In many-to-many task offloading framework (e.g. [47]), overlapped clusters require coordination in resource allocation during scheduling. This leads to information sharing between clusters. In a multi-carrier system, these interactions can be expensive, particularly given mobile devices' processing capabilities. Hence, as a practical compromise, we adopt this assumption.

In addition, denoting by f_{n_k} the computing capability of mobile device n_k in terms of CPU cycles per second, the delay for computing can be readily formulated as

$$t_{n_k}^{\text{exe}} \left(V_{n_k}, f_{n_k} \right) = \frac{V_{n_k} C_n}{f_{n_k}}. \tag{4}$$

Upon assuming the dynamic voltage and frequency scaling techniques used [52], the energy consumption of per CPU cycle becomes $\delta_{n_k} = \kappa f_{n_k}^2$, where κ is the energy coefficient that depends on the integrated chip structure. Therefore, the energy consumption for computing can be given by

$$e_{n_k}^{\text{exe}} \left(V_{n_k}, f_{n_k} \right) = V_{n_k} C_n \delta_{n_k}. \tag{5}$$

3.3 Data Transmission Model

Let $\alpha_{n_k}^m \in \{0, 1\}$ be a binary indicator of the sub-carrier allocation for the D2D link between SD n and HD n_k, where $\alpha_{n_k}^m = 1$ indicates that sub-carrier m is allocated to the link, while $\alpha_{n_k}^m = 0$ otherwise. We have $\mathcal{M}_n = \left\{ m | a_{n_k}^m = 1, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}_n \right\}$. Since the rapid development of source coding [53] and data compression [26] technologies, and the data size of computing outcomes is much smaller than that of the input data, its transmission time can be either neglected or regarded as a small constant that does not affect our analyses [54].

As the sub-carriers are not reused within D2D links, the receive signal noise ratio (SNR) of HD n_k from SD n on sub-carrier m becomes

$$\text{SNR}_{n_k}^m \left(p_{n_k}^m \right) = \frac{p_{n_k}^m h_{n_0, n_k}^m}{\sigma^2}, \tag{6}$$

where $p_{n_k}^m$ is the transmission power of SD n allocated to HD n_k on sub-carrier m, and σ^2 is the power of the additive white gaussian noise (AWGN). The maximum achievable transmission data rate is given by

$$r_{n_k} \left(\alpha_{n_k}^m, p_{n_k}^m \right) = \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m B \log_2 \left(1 + \text{SNR}_{n_k}^m \right), \tag{7}$$

where B is the bandwidth of each sub-carrier.

The total cooperative computing delay consists of the transmission delay and the computing delay. The transmission delay can be expressed as

$$t_{n_k}^{\text{off}} \left(V_{n_k}, \alpha_{n_k}^m, p_{n_k}^m \right) = \frac{V_{n_k}}{r_{n_k}}. \tag{8}$$

Similarly, the total energy consumption is composed of transmission energy consumption and computing energy consumption. The transmission energy consumption is given by

$$e_{n_k}^{\text{off}} \left(V_{n_k}, \alpha_{n_k}^m, p_{n_k}^m \right) = t_{n_k}^{\text{off}} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m. \tag{9}$$

3.4 Weighted Sum Energy Consumption Minimization Problem Formulation

The network lifetime is a non-negligible performance metric for wireless sensor network, due to the limited energy capacity of devices [55], hence our goal is to minimize the weighted sum energy consumption, by performing jointly optimizing source-helper assignment, task partition, sub-carrier allocation, transmission power allocation, and computing frequency allocation. The weighted sum energy consumption in the n-th cooperative computing cluster is given by

$$\Phi_n\left(\mathcal{K}_n, V_{n_k}, f_{n_k}, \alpha_{n_k}^m, p_{n_k}^m\right) = \sum_{k=0}^{|\mathcal{K}_n|} \lambda_{n_k} e_{n_k}^{\text{exe}} + \sum_{k=1}^{|\mathcal{K}_n|} \lambda_{n_0} e_{n_k}^{\text{off}}, \tag{10}$$

where the positive weighting factors λ_{n_k} account for fairness among devices based on their remaining battery life. Mathematically, the total weighted sum energy consumption minimization problem can be formulated as

$$\mathcal{P}0: \min_{\substack{\{\mathcal{K}_n\},\\ \{\boldsymbol{V}_{n_k}\},\{\boldsymbol{f}_{n_k}\},\\ \{\boldsymbol{\alpha}_{n_k}^m\},\{\boldsymbol{p}_{n_k}^m\}}} \Phi = \sum_{n=1}^{|\mathcal{N}|} \Phi_n, \tag{11a}$$

$$\text{s.t. } t_{n_0}^{\text{exe}} \leq T, \forall n \in \mathcal{N}, \tag{11b}$$

$$t_{n_k}^{\text{exe}} + t_{n_k}^{\text{off}} \leq T, \forall k \in \mathcal{K}_n, \forall n \in \mathcal{N}, \tag{11c}$$

$$\mathcal{K}_n \cap \mathcal{K}_{n'} = \emptyset, \forall n \neq n', \tag{11d}$$

$$\bigcup_{n=1}^{|\mathcal{N}|} \mathcal{K}_n \subseteq \mathcal{K}, \tag{11e}$$

$$V_n = V_{n_0} + \sum_{k=1}^{|\mathcal{K}_n|} V_{n_k}, \forall n \in \mathcal{N}, \tag{11f}$$

$$0 \leq V_{n_k} \leq V_n, \forall k \in \{0\} \cup \mathcal{K}_n, \forall n \in \mathcal{N}, \tag{11g}$$

$$0 \leq f_{n_k} \leq f_{\max}, \forall k \in \{0\} \cup \mathcal{K}_n, \forall n \in \mathcal{N}, \tag{11h}$$

$$\alpha_{n_k}^m \in \{0,1\}, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}_n, \forall n \in \mathcal{N}, \tag{11i}$$

$$\sum_{n=1}^{|\mathcal{N}|} \sum_{k=1}^{|\mathcal{K}_n|} \alpha_{n_k}^m \leq 1, \forall m \in \mathcal{M}, \tag{11j}$$

$$\sum_{k=1}^{|\mathcal{K}_n|} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m \leq p_{\max}, \forall n \in \mathcal{N}, \tag{11k}$$

$$0 \leq p_{n_k}^m \leq p_{\max}, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}_n, \forall n \in \mathcal{N}. \tag{11l}$$

Here, constraints (11b) and (11c) ensure that the task execution delay of SD does not exceed the maximum tolerable delay. Constraint (11d) guarantees each HD can be assigned to at most one SD, and constraint (11e) indicates all assigned HDs must come from the original HD set. Constraint (11f) guarantees that the task partition sizes add up to the original task size. Constraints (11g), (11h), and (11l) give the bounds on task dividing, computing capability, and transmission power, respectively. Constraint (11j) enforces that each sub-carrier can only be allocated to at most one D2D link. Finally, constraint (11k) implies that the sum transmission power does not exceed the total power constraint for each SD. Table 1 presents a summary of the key notations used in this paper.

TABLE 1
List of key notations

Symbol	Definition
T	Delay limit of computational tasks
$\mathcal{U}$	Set of mobile sensing devices
$\mathcal{N}$	Set of SDs
$\mathcal{K}$	Set of HDs
$\mathcal{K}_n$	Set of HDs in n-th cluster
$\mathcal{M}$	Set of orthogonal sub-carriers
$\mathcal{M}_n$	Set of sub-carriers assigned to n-th cluster
$h_{a,b}^m$	Channel gain between device a and b on sub-carrier m
$\hat{h}_{a,b}^m$	Location-based channel gain between device a and b
$\Upsilon_{a,b}$	Path loss between device a and b
$\hat{\Upsilon}$	Average channel power gain at the reference distance of 1 m
$d_{a,b}$	3D distance between device a and b
ε	Path loss exponent
$g_{a,b}^m$	Small-scale fading between device a and b on sub-carrier m
V_n	Input data size of task n in bit
C_n	Computing complexity of task n in CPU cycle
V_{n_k}	Offloading partition size of task n to device n_k
f_{n_k}	Computing capability of device n_k
$f_{\max}$	Maximum computing frequency of each device
$t_{n_k}^{\text{exe}}$	Delay of device n_k for computing
δ_{n_k}	Energy consumption by each CPU cycle of device n_k
κ	Energy coefficient of computing
$e_{n_k}^{\text{exe}}$	Energy consumption of device n_k for computing
$\alpha_{n_k}^m$	Indicator if sub-carrier m allocated to SD n and HD n_k
$\text{SNR}_{n_k}^m$	SNR between SD n and HD n_k on sub-carrier m
$p_{n_k}^m$	Transmission power of SD n to HD n_k on sub-carrier m
$p_{\max}$	Maximum transmission power of each device
σ^2	Additive white Gaussian noise
r_{n_k}	Transmission data rate from SD n to HD n_k
B	Bandwidth of sub-carrier
$t_{n_k}^{\text{off}}$	Delay of transmission task to HD n_k
$e_{n_k}^{\text{off}}$	Energy consumption of transmission task to HD n_k
Φ_n	Weighted sum energy consumption in n-th cluster
θ	Resource efficiency
λ_{n_k}	Weight factor of mobile device n_k
γ_k	Lagrange multiplier of constraint (25c) in $\mathcal{P}1\text{-}2C''$
ω_k	Lagrange multiplier of constraint (12f) in $\mathcal{P}1\text{-}2C''$
η	Lagrange multiplier of constraint (12i) in $\mathcal{P}1\text{-}2C''$
u_k	Auxiliary variable for $\mathcal{P}1\text{-}2C''$
β_k	Auxiliary variable for $\mathcal{P}1\text{-}2C''$
ϵ	Convergence criterion

Remark 1. *Problem $\mathcal{P}0$ is a MINLP problem and does not have an exact solution in polynomial time. Based on further analysis of $\mathcal{P}0$, the following features can be found. First, the task partition and resource allocation results can vary depending on the source-helper assignment decisions. Second, the performance of source-helper assignment cannot be evaluated until the task partition and resource allocation results are generated. Finally, the number of assigned HDs in each cluster does not follow the principle of "the more, the better" since the task is offloaded to HDs with the lowest weighted energy consumption, and assigning too many inefficient HDs does not aid in reducing the weighted sum energy consumption.*

A distributed scheduling scheme is proposed to address Problem $\mathcal{P}0$. The scheme consists of two algorithms, namely, a matching theory-based distributed algorithm called DRESHA to solve the cooperative computing cluster formulation (source-helper assignment) problem in the outer loop, and a JTPRA algorithm, aiming to solve the joint task partition and resource allocation problem in each cluster, in the inner loop. The scheme comprises the following three phases.

(1) Cooperative Computing Cluster Formulation: A distributed source-helper assignment scheme is utilized to determine the HD set $\mathcal{K}_n$ for any SD, forming cooperative computing clusters based on the resource efficiency;

(2) Joint Task Partition and Resource Allocation: After determining source-helper assignment, the task partition and resource allocation problem $\mathcal{P}1$ is formulated in each cooperative computing cluster, and its sub-optimal solution can be obtained by solving the problem below

$$\mathcal{P}1: \min_{\substack{\boldsymbol{V}_{n_k}, \boldsymbol{f}_{n_k}, \\ \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m}} \sum_{k=0}^{|\mathcal{K}_n|} \lambda_{n_k} e_{n_k}^{\text{exe}} + \sum_{k=1}^{|\mathcal{K}_n|} \lambda_{n_0} e_{n_k}^{\text{off}}, \tag{12a}$$

$$\text{s.t. } t_{n_0}^{\text{exe}} \leq T, \tag{12b}$$

$$t_{n_k}^{\text{exe}} + t_{n_k}^{\text{off}} \leq T, \forall k \in \mathcal{K}_n, \tag{12c}$$

$$V_n = V_{n_0} + \sum_{k=1}^{|\mathcal{K}_n|} V_{n_k}, \tag{12d}$$

$$0 \leq V_{n_k} \leq V_n, \forall k \in \{0\} \cup \mathcal{K}_n, \tag{12e}$$

$$0 \leq f_{n_k} \leq f_{\max}, \forall k \in \{0\} \cup \mathcal{K}_n, \tag{12f}$$

$$\alpha_{n_k}^m \in \{0,1\}, \forall m \in \mathcal{M}_n, \forall k \in \mathcal{K}_n, \tag{12g}$$

$$\sum_{k=1}^{|\mathcal{K}_n|} \alpha_{n_k}^m \leq 1, \forall m \in \mathcal{M}_n, \tag{12h}$$

$$\sum_{k=1}^{|\mathcal{K}_n|} \sum_{m=1}^{|\mathcal{K}_n|} \alpha_{n_k}^m p_{n_k}^m \leq p_{\max}, \tag{12i}$$

$$0 \leq p_{n_k}^m \leq p_{\max}, \forall m \in \mathcal{M}_n, \forall k \in \mathcal{K}_n; \tag{12j}$$

(3) Redundant HDs Rejection: To remove redundant HDs in each cooperative computing cluster, only HDs that have been assigned with a task are accepted by the SD, HDs without assigned tasks are rejected and may subsequently be assigned to other clusters in the next iteration.

The details of phases (1) and (3) are provided in Section 4, while the detail of phase (2) is shown in Section 5.

4 Cooperative Computing Cluster Formulation

The challenges associated with the formulation of cooperative computing clusters are as follows: i) the weighted sum energy consumption cannot be calculated until the source-helper assignment is complete, owing to the strong coupling of the optimization variables; ii) source-helper assignment is a combinatorial optimization problem, which is NP-hard; and iii) information interaction is a significant burden in the distributed systems if centralized methods are used. Therefore, it is intractable to obtain the globally optimal solution. To overcome these difficulties, we propose the DRESHA algorithm, which is based on the matching theory. We introduce the resource efficiency to describe the preference of HDs to SDs, which is calculated using easy-to-access information such as device location, task complexity, and remaining battery life. The source-helper assignment problem is formulated as a many-to-one matching problem, and the DRESHA algorithm is used to obtain a helper-optimal [56] solution for source-helper assignment in a distributed manner.

4.1 Preliminary of Matching

Bipartite matching problems with two-sided preferences is to map agents of one set to agents of the another disjoint set. The source-helper assignment problem considered can be readily formulated as a many-to-one bipartite matching between the set of SDs $\mathcal{N}$ and the set of HDs $\mathcal{K}$, where each HD k is allowed to choose almost one SD from set $\mathcal{N}$. Formally, the source-helper assignment matching problem can be defined as follows.

Definition 1. *The source-helper assignment matching is defined as a many-to-one mapping function μ from set $\mathcal{N} \cup \mathcal{K}$ into the set of all subsets of $\mathcal{N} \cup \mathcal{K}$ such that:*

- *For every agent $n \in \mathcal{N}$, $|\mu(n)| \leq |\mathcal{K}|$ and $\mu(n) \in 2^k$;*
- *For every agent $k \in \mathcal{K}$, $|\mu(k)| \leq 1$ and $\mu(k) \in \mathcal{N} \cup \emptyset$;*
- *$n = \mu(k)$ if and only if $k \in \mu(n)$.*

The stable matching theory is based on preference relation, which is a strict, transitive, and complete relation between agents from two disjoint sets. We use $k \succ_n k'$ to denote the preference ordering if agent n prefers agent k to k'; Given a set of agents $H \subseteq \mathcal{K}$, let $\mathrm{Ch}_n(H)$ denote n's most preferred subset of H according to its preference relation. Stability is considered a fundamental requirement in any sensible matching, which means no agent has incentives to deviate from the matching result. The specific definitions for stability are as follows.

Definition 2. *A matching function μ is stable, if there exists no blocking pair (n,k): if $k \notin \mu(n), n \notin \mu(k)$, such that $k \in \mathrm{Ch}_n(\mu(n) \cup k)$ and $n \succ_k \mu(k)$.*

4.1.1 Preference of HD to SD

Furthermore, acquiring and exchanging channel state information of all D2D links is a crucial task that incurs a complexity overhead of $O(|\mathcal{N}||\mathcal{K}||\mathcal{M}|)$. To reduce the cost of information interaction and complexity of the algorithm, we use location-based channel gain instead of actual channel gain in the source-helper assignment scheme, and it is pressed as

$$\hat{h}_{a,b}^m \approx \Upsilon_{a,b} = \hat{\Upsilon} d_{a,b}^{-\varepsilon}, \tag{13}$$

basing on the location-based channel gain we can define resource efficiency as following.

The primary objective of source-helper assignment is to reduce the weighted sum energy consumption. HDs prefer to provide task offloading services to SDs with lower energy consumption. However, achieving weighted energy efficiency is not possible until resource allocation is complete. Thus we use resource efficiency to indicates the affection of HD on SD, and it is defined as

$$\theta(n,k) = \frac{1}{\frac{\lambda_n}{B\log_2\left(1+\frac{\hat{\Upsilon} d_{n,k}^{-\varepsilon}}{\sigma^2}\right)} + \lambda_k \kappa C_n}. \tag{14}$$

Therefore, the preference of HD to SD can be constructed as

$$n \succ_k n', \theta(n,k) > \theta(n',k), \forall k \in \mathcal{K}. \tag{15}$$

4.1.2 *Preference of SD to HD*

Then, the preference of SD to HD depends on the task partition result given by the JTPRA algorithm, which is constructed as

$$k \succ_n k', V_{n_k} > 0 \text{ and } V_{n'_k} = 0, \forall n \in \mathcal{N}. \tag{16}$$

4.2 Dispersive Resource Efficiency based Source-Helper Assignment Algorithm

Based on the classical Gale-Shapley algorithm, the DRESHA algorithm is proposed to solve the source-helper assignment problem, and the task partition and resource allocation results are obtained by JTPRA within each iteration. The overall working flow is depicted in Algorithm 1. The outputs are the results of source-helper assignment $\{\mathcal{K}_n\}$, task partition $\{\boldsymbol{V}_{n_k}\}$, sub-carrier allocation $\{\boldsymbol{\alpha}^m_{n_k}\}$, transmission power allocation $\{\boldsymbol{p}^m_{n_k}\}$, and computing frequency allocation $\{\boldsymbol{f}_{n_k}\}$. Initially, all mobile devices exchange the necessary information so that HDs can set up preference lists (steps 3-5). Then, each unassigned HD sends a proposal to the most preferred SD that has not proposed yet (steps 8 and 9). After all unassigned HDs send proposals, the cooperative computing clusters are formulated, and the joint task partition and resource allocation problem is solved by the JTPRA algorithm, and SD rejects HDs that are not assigned with any tasks (steps 13-17). After rejecting all redundant HDs in each cluster, new cooperative computing clusters are formulated. The algorithm repeats the above steps until all cooperative computing clusters converge.

Algorithm 1 DRESHA Algorithm

```
Ensure: $\{\mathcal{K}_n\}$, $\{\boldsymbol{V}_{n_k}\}$, $\{\boldsymbol{\alpha}^m_{n_k}\}$, $\{\boldsymbol{p}^m_{n_k}\}$, and $\{\boldsymbol{f}_{n_k}\}$.
 1: Set iteration index $l$ = 0.
 2: Randomly and equally allocate sub-carriers to each cluster [2] to ensure $\{\mathcal{M}_n\}$.
 3: Calculate $\boldsymbol{\lambda}_{n_k}$ according to the remaining battery life of each mobile device.
 4: Every SD $n$ broadcasts the device location and task complexity message to its achievable HDs in parallel, and sets $\mathcal{K}_n^{(l)} = \emptyset$.
 5: Every HD $k$ sets up preference list $L_k$ according to equation (14), and sets its status as unassigned.
 6: repeat
 7:   for $\exists$ unassigned HD $k$ with non-empty preference list $L_k$ do
 8:     Find the most preferred SD $n$ from $L_k$.
 9:     Send proposes message to SD $n$, remove $n$ from $L_k$, and set status as assigned.
10:     Add HD $k$ to the temporary assignment set $\mathcal{K}_n^{(l)}$ of SD $n$.
11:   end for
12:   for each cluster head SD $n \in \mathcal{N}$, that $\mathcal{K}_n^{(l-1)} \neq \mathcal{K}_n^{(l)}$ do
13:     Given $\mathcal{K}_n^{(l)}$ and $\mathcal{M}_n$, obtain $\boldsymbol{V}_{n_k}$, $\boldsymbol{f}_{n_k}$, $\boldsymbol{\alpha}^m_{n_k}$, and $\boldsymbol{p}^m_{n_k}$ by solving $\mathcal{P}1$ using JTPRA (Algorithm 6).
14:     $\mathcal{K}_n^{(l+1)} \leftarrow \mathcal{K}_n^{(l)}$.
15:     for each HD $n_k \in \mathcal{K}_n^{(l+1)}$ do
16:       if $V_{n_k}$ is 0 then
17:         SD $n$ sends reject message to HD $n_k$, and remove $n$ from $\mathcal{K}_n^{(l+1)}$.
18:         Set HD $k$ as unassigned.
19:       end if
20:     end for
21:   end for
22:   $l \leftarrow l + 1$.
23: until $\mathcal{K}_n^{(l-1)} == \mathcal{K}_n^{(l)}, \forall n \in \mathcal{K}_n$.
24: return $\{\mathcal{K}_n\}$, $\{\boldsymbol{V}_{n_k}\}$, $\{\boldsymbol{\alpha}^m_{n_k}\}$, $\{\boldsymbol{p}^m_{n_k}\}$, and $\{\boldsymbol{f}_{n_k}\}$.
```

4.3 Stability and Complexity Analysis

For stability and complexity, there are two key results as follows.

Proposition 1. *The DRESHA algorithm is stable.*

Proof. The proposition can be proved by contradiction. Suppose there is a blocking pair (n, k). Since $n \succ_k \mu(k)$, HD n must have proposed to SD n (steps 8 and 9), if SD n accept HD k in the end, $k \notin \mu(n)$ will not hold. On the other hand, SD n reject HD k in the end, $k \in \mathrm{CH_n}(\mu(\mathrm{n}) \cup \mathrm{k})$ will not hold, because SD must not reject preferred agent (step 17). In the conclusion, DRESHA algorithm is stable. □

Proposition 2. *The overall time complexity of DRESHA algorithm is $O\left(|\mathcal{N}|^2|\mathcal{K}|^5 \log_2 f_{\max} + |\mathcal{N}|^2|\mathcal{K}|^5|\mathcal{M}|\right)$, which is polynomial.*

Proof. Considering the worst case, for $|\mathcal{K}|$ SDs, the complexity of setting up preference list is $O(|\mathcal{K}|\,|\mathcal{N}|)$; the maximum iteration loop is $|\mathcal{K}|\,|\mathcal{N}|$, in each iteration: the complexity of find the most preferred SD is $O(|\mathcal{N}|)$, the complexity of JTPRA algorithm is $O\left(|\mathcal{N}|\,|\mathcal{K}|^4 \log_2 f_{\max} + |\mathcal{N}|\,|\mathcal{K}|^4|\mathcal{M}|\right)$ (proven in Subsection 5.4), and the complexity of rejecting redundant HDs is $O(|\mathcal{K}|)$. Therefore the overall time complexity is $O(|\mathcal{K}|\,|\mathcal{N}|) + O\left(|\mathcal{K}|\,|\mathcal{N}|^2\right) + O\left(|\mathcal{N}|^2|\mathcal{K}|^5 \log_2 f_{\max} + |\mathcal{N}|^2|\mathcal{K}|^5|\mathcal{M}|\right) + O\left(|\mathcal{K}|^2\,|\mathcal{N}|\right)$, which an be simplified as $O\left(|\mathcal{N}|^2|\mathcal{K}|^5 \log_2 f_{\max} + |\mathcal{N}|^2|\mathcal{K}|^5|\mathcal{M}|\right)$. □

5 Joint Task Partition and Resource Allocation Design

The JTPRA algorithm is proposed to optimize the design of task partition and resource allocation within each cooperative computing cluster. The original problem, $\mathcal{P}1$, is intractable due to the coupling of the optimization variables. The JTPRA algorithm decomposes $\mathcal{P}1$ into two subproblems, namely, the task partition problem $\mathcal{P}1$-1, and the communication and computational resource allocation problem $\mathcal{P}1$-2. The AO method is applied to optimize the design of the computational task partition ($\boldsymbol{V}_{n_k}$) while fixing the communication and computational resource allocation ($\boldsymbol{\alpha}^m_{n_k}$, $\boldsymbol{p}^m_{n_k}$, and $\boldsymbol{f}_{n_k}$), followed by optimizing the resource allocation while fixing the task assignment. These steps are

2. Since precise sub-carrier allocation between clusters requires extensive channel state information exchange, and there is no clear mathematical relationship between the state of mobile devices and the number of sub-carriers required by each cluster, we choose this allocation method between clusters as a compromise.

Algorithm 2 Task Partition Algorithm

```
Require: $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
Ensure: $\boldsymbol{V}_{n_k}$.
1: Set unassigned task bits $V_n^{\mathrm{R}} = V_n$ and device index $i = 0$.
2: Calculate the efficiency $E_n$ of each device $k \in \{0\} \cup \mathcal{K}_n$ using following equations
   $$E_n = \begin{cases} \lambda_{n_0} \kappa C_n f_{n_0}^2, & n = 0 \\ \frac{\lambda_{n_0} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m}{r_{n_k}} + \lambda_{n_k} \kappa C_n f_{n_k}^2, & \text{otherwise.} \end{cases} \quad (20)$$
3: Sort devices in ascending order of $E_n$, and the sequence is expressed as: $e_0, e_1, ..., e_n$.
4: while $V_n^{\mathrm{R}} > 0$ do
5:     $n \leftarrow e_i$.
6:     Assign task according following equations
       $$V_{n_k} = \begin{cases} \frac{T f_{n_0}}{C_n}, & n = 0 \\ \frac{T f_{n_k} r_{n_k}}{C_n r_{n_k} + f_{n_k}}, & \text{otherwise.} \end{cases} \quad (21)$$
7:     $i \leftarrow i + 1$.
8: end while
9: return $\boldsymbol{V}_{n_k}$.
```

iteratively updated in the outer loop until convergence is achieved. The details of the task partition and resource allocation are presented in the following sections.

5.1 Task Partition Optimization

Given a fixed resource allocation results $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$, the task assignment problem can be given by

$$\mathcal{P}1\text{-}1: \min_{\boldsymbol{V}_{n_k}} \lambda_{n_0} \kappa C_n f_{n_0}^2 V_{n_0} + \sum_{k=1}^{|\mathcal{K}_n|} \left(\frac{\lambda_{n_0} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m}{r_{n_k}} + \lambda_{n_k} \kappa C_n f_{n_k}^2 \right) V_{n_k}, \tag{17a}$$

$$\text{s.t.} \quad (12b), (12c), (12d), (12e).$$

where constraints (12b), (12c), (12d) can be recast as

$$0 \le V_{n_0} \le \frac{T f_{n_0}}{C_n}, \tag{18}$$

and

$$0 \le V_{n_k} \le \frac{T f_{n_k} r_{n_k}}{C_n r_{n_k} + f_{n_k}}, \forall k \in \mathcal{K}_n. \tag{19}$$

It is observed that $\mathcal{P}1$-1 is a linear programming problem, and the optimal partition strategy is assigning the task to the most efficient device until V_{n_k} (or V_{n_0}) reaches its upper bound. This problem is readily to be solved by a method similar to the water filling algorithm, and the solution is summarized in Algorithm 2.

5.2 Communication and Computational Resource Allocation

Given a fixed computational task partition $\boldsymbol{V}_{n_k}$, the communication and computational resource allocation problem can be formulated as

$$\mathcal{P}1\text{-}2: \min_{\substack{\boldsymbol{f}_{n_k}, \\ \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m}} \sum_{k=0}^{|\mathcal{K}_n|} \lambda_{n_k} \kappa C_n V_{n_k} f_{n_k}^2 + \sum_{k=1}^{|\mathcal{K}_n|} \frac{\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m}{r_{n_k}}, \tag{22a}$$

$$\text{s.t.} \quad (12b), (12c), (12f), (12g), (12h), (12i), (12j).$$

It is readily seen that constraint (12b) is tight when the optimal computing frequency allocation of local computing is obtained, which can be expressed as

$$f_{n_0} = \frac{V_{n_0} C_n}{T}, \tag{23}$$

then the resource allocation solution of cooperative computing can be obtained by solving the following problem:

$$\mathcal{P}1\text{-}2\mathrm{C}: \min_{\substack{\boldsymbol{f}_{n_k}, \\ \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m}} \sum_{k=1}^{|\mathcal{K}_n|} \lambda_{n_k} \kappa C_n V_{n_k} f_{n_k}^2 + \sum_{k=1}^{|\mathcal{K}_n|} \frac{\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m}{r_{n_k}}, \tag{24a}$$

$$\text{s.t.} \quad (12c), (12f), (12g), (12h), (12i), (12j),$$

and $\mathcal{P}1$-2C is a non-convex MINLP problem since the binary optimization variable $\boldsymbol{\alpha}_{n_k}^m$ and the sum-of-ratio minimization in the objective function (OF). In order to tackle these issues, firstly, it is rewritten as the following equivalent form:

$$\mathcal{P}1\text{-}2\mathrm{C}': \min_{\substack{\boldsymbol{\beta}_n, \boldsymbol{f}_{n_k}, \\ \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m}} \sum_{k=1}^{|\mathcal{K}_n|} \left(\lambda_{n_k} \kappa C_n V_{n_k} f_{n_k}^2 + \beta_k \right), \tag{25a}$$

$$\text{s.t.} \quad \frac{\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m}{r_{n_k}} \le \beta_k, \forall k \in \mathcal{K}_n, \tag{25b}$$

$$r_{n_k} \ge \frac{V_{n_k} f_{n_k}}{T f_{n_k} - C_n V_{n_k}}, \forall k \in \mathcal{K}_n, \tag{25c}$$

$$(12f), (12g), (12h), (12i), (12j).$$

The following proposition lends us convenient to solving $\mathcal{P}1$-2C$'$.

Proposition 3. *If $\left(\hat{\boldsymbol{\beta}}_k, \hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ is the optimal solution of $\mathcal{P}1$-2C$'$, then there exist $\hat{\boldsymbol{u}}_k = \{u_1, u_2, ..., u_k\}$ such that*

$\left(\hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ is the optimal solution of the following problem for $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ and $\boldsymbol{\beta}_k = \hat{\boldsymbol{\beta}}_k$.

$$\mathcal{P}1\text{-}2\mathrm{C}'' : \min_{\substack{\boldsymbol{f}_{n_k}, \\ \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m}} \sum_{k=1}^{|\mathcal{K}_n|} \Bigg[\lambda_{n_k}\kappa C_n V_{n_k} f_{n_k}^2 \tag{26a}$$

$$+u_k \left(\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m - \beta_k r_{n_k}\right)\Bigg],$$

$$\text{s.t.} \quad (25c), (12f), (12g), (12h), (12i), (12j),$$

and $\left(\hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ also satisfies the following equations for $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ and $\boldsymbol{\beta}_k = \hat{\boldsymbol{\beta}}_k$:

$$u_k = \frac{1}{r_{n_k}}, \forall k \in \mathcal{K}_n; \tag{27}$$

$$\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \alpha_{n_k}^m p_{n_k}^m - \beta_k r_{n_k} = 0, \forall k \in \mathcal{K}_n. \tag{28}$$

Correspondingly, if $\left(\hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ is the optimal solution of $\mathcal{P}1$-2C$''$ and satisfies (27), (28) when set $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ and $\boldsymbol{\beta}_k = \hat{\boldsymbol{\beta}}_k$, $\left(\hat{\boldsymbol{\beta}}_k, \hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ is the optimal solution of $\mathcal{P}1$-2C$'$.

Proof. The Lagrangian duality gap of the optimization problem is zero in multi-carrier systems when the number of sub-carriers goes to infinity according to the time-sharing condition [57], [58]. It is verified that the duality gap vanishes, even in systems with a practical number of sub-carriers [59]. So the duality gap of $\mathcal{P}1$-2C$'$ and $\mathcal{P}1$-2C$''$ vanishes. if $\left(\hat{\boldsymbol{\beta}}_k, \hat{\boldsymbol{f}}_{n_k}, \hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m\right)$ is the solution of $\mathcal{P}1$-2C$'$, there exists non-negative Lagrange multiplier $\hat{\boldsymbol{u}}_k$ satisfying the following KKT conditions

$$\frac{\partial \mathcal{L}}{\partial \beta_k} = 1 - \hat{u}_k \hat{r}_{n_k} = 0, \forall k \in \mathcal{K}_n, \tag{29}$$

$$\hat{u}_k \left(\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \hat{\alpha}_{n_k}^m \hat{p}_{n_k}^m - \beta_k \hat{r}_{n_k}\right) = 0, \forall k \in \mathcal{K}_n, \tag{30}$$

$$\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \hat{\alpha}_{n_k}^m \hat{p}_{n_k}^m - \beta_k \hat{r}_{n_k} \leq 0, \forall k \in \mathcal{K}_n, \tag{31}$$

$$\hat{u}_k \geq 0, \forall k \in \mathcal{K}_n, \tag{32}$$

where $\mathcal{L}$ is the Lagrange function of $\mathcal{P}1$-2C$'$. Since $r_{n_k} > 0$, (29) is equivalently to

$$\hat{u}_k = \frac{1}{r_{n_k}}, \forall k \in \mathcal{K}_n. \tag{33}$$

and since $\hat{u}_k > 0$, (30) is equivalently to

$$\lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \hat{\alpha}_{n_k}^m \hat{p}_{n_k}^m - \beta_k \hat{r}_{n_k} = 0, \forall k \in \mathcal{K}_n. \tag{34}$$

Furthermore it is readily to prove the rest KKT conditions of $\mathcal{P}1$-2C$'$ are exactly the KKT conditions of $\mathcal{P}1$-2C$''$, when $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ and $\boldsymbol{\beta}_k = \hat{\boldsymbol{\beta}}_k$. So the first conclusion is proved. Following the same procedure the second conclusion can be proved. □

The optimization problem $\mathcal{P}1$-2C$''$ remains a non-convex MINLP problem that cannot be solved in polynomial time due to its high computational complexity. Fortunately, the duality gap of the optimization problem vanishes in multi-carrier systems. Thus, the Lagrangian duality method [60] is invoked for solving $\mathcal{P}1$-2C$''$.

Given $\boldsymbol{u}_k$ and $\boldsymbol{\beta}_k$, the Lagrangian function of $\mathcal{P}1$-2C$''$ is formulated as (35), shown at the top of next page, where $\boldsymbol{\gamma}_k = [\gamma_1, \gamma_2, \ldots, \gamma_k]^T$, $\boldsymbol{\omega}_k = [\omega_1, \omega_2, \ldots, \omega_k]^T$ and η are the Lagrange multiplier vectors. The dual function is obtained as

$$\mathcal{G}(\gamma_k, \omega_k, \eta) = \begin{cases} \min\limits_{\boldsymbol{f}_{n_k}, \boldsymbol{\alpha}_{n_k}^m, \boldsymbol{p}_{n_k}^m} \mathcal{L}\left(f_{n_k}, \alpha_{n_k}^m, p_{n_k}^m, \gamma_k, \omega_k, \eta\right) \\ \text{s.t.} \quad (12\mathrm{f}), (12\mathrm{g}), (12\mathrm{h}), (12\mathrm{j}), \end{cases} \tag{36}$$

and the Lagrange dual problem is formulated as

$$\mathcal{P}1\text{-}2\mathrm{C}''\text{-dual} : \max_{\boldsymbol{\gamma}_k, \boldsymbol{\omega}_k,, \eta} \mathcal{G}(\gamma_k, \omega_k, \eta), \tag{37a}$$

$$\text{s.t.} \quad \gamma_k \geq 0, \forall k \in \mathcal{K}_n, \tag{37b}$$

$$\omega_k \geq 0, \forall k \in \mathcal{K}_n, \tag{37c}$$

$$\eta \geq 0. \tag{37d}$$

Since the duality gap vanishes, we can obtain a near-optimal solution of $\mathcal{P}1$-2C$''$ by solving $\mathcal{P}1$-2C$''$-dual, the dual function (36) can be reformulated as

$$\begin{aligned} \mathcal{G}(\gamma_k, \omega_k, \eta) = &\sum_{m=1}^{|\mathcal{K}_n|} \hat{F}_m(\gamma_k, \omega_k) + \sum_{m=1}^{|\mathcal{M}_n|} \hat{P}_m(\gamma_k, \eta) \\ &- \sum_{k=1}^{|\mathcal{K}_n|} \omega_k f_{\max} - \eta p_{\max}, \end{aligned} \tag{38}$$

where

$$\begin{aligned} &\hat{F}_m(\gamma_k, \omega_k) \\ &= \min_{f_{n_k} \in \mathcal{D}_f} \left(\lambda_{n_k}\kappa C_n V_{n_k} f_{n_k}^2 + \frac{\gamma_k V_{n_k} f_{n_k}}{T f_{n_k} - C_n V_{n_k}} + \omega_k f_{n_k}\right), \end{aligned} \tag{39}$$

and

$$\begin{aligned} &\hat{P}_m(\gamma_k, \eta) \\ &= \min_{\substack{\alpha_{n_k}^m \in \mathcal{D}_\alpha, \\ p_{n_k}^m \in \mathcal{D}_p}} \sum_{k=1}^{|\mathcal{K}_n|} \alpha_{n_k}^m \Big[(u_k \lambda_{n_0} V_{n_k} + \eta) p_{n_k}^m \\ &\qquad - (u_k \beta_k + \gamma_k) B \log_2\left(1 + \frac{p_{n_k}^m h_{n_0, n_k}^m}{\sigma^2}\right)\Big], \end{aligned} \tag{40}$$

domain $\mathcal{D}_f$ is defined by constraint (12f), domain $\mathcal{D}_\alpha$ is defined by constraints (12g) and (12h), domain $\mathcal{D}_p$ is defined by constraint in (12j). The solution of communication and computational resource allocation can be obtained by the primal-dual method as follows.

5.2.1 Computational Resource Allocation

Since the strong duality holds for the primal and dual problem, the optimal computing frequency allocation vec-

$$\begin{aligned}&\mathcal{L}\left(f_{n_k},\alpha_{n_k}^m,p_{n_k}^m,\gamma_k,\omega_k,\eta\right)\\&=\sum_{k=1}^{|\mathcal{K}_n|}\left\{\lambda_{n_k}\kappa C_n V_{n_k} f_{n_k}^2+u_k\left[\lambda_{n_0}V_{n_k}\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m p_{n_k}^m-\beta_k\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m B\log_2\left(1+\frac{p_{n_k}^m h_{n_0,n_k}^m}{\sigma^2}\right)\right]\right\}\\&+\sum_{k=1}^{|\mathcal{K}_n|}\gamma_k\left[\frac{V_{n_k}f_{n_k}}{Tf_{n_k}-C_nV_{n_k}}-\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m B\log_2\left(1+\frac{p_{n_k^m}h_{n_0,n_k}^m}{\sigma^2}\right)\right]+\sum_{k=1}^{|\mathcal{K}_n|}\omega_k\left(f_{n_k}-f_{\max}\right)+\eta\left(\sum_{k=1}^{|\mathcal{K}_n|}\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m p_{n_k}^m-p_{\max}\right).\end{aligned}\tag{35}$$

Algorithm 3 Computational Resource Allocation

```
Require: $\boldsymbol{V}_{n_k}$, $\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$, and $\epsilon$.
Ensure: $\boldsymbol{f}_{n_k}$.
 1: for $k \in \mathcal{K}_n$ do
 2:   Set $f_{n_k}^{\mathrm{LB}} = C_n V_{n_k}/T$ and $f_{n_k}^{\mathrm{UB}} = f_{\max}$.
 3:   repeat
 4:     Set $f_{n_k} = \left(f_{n_k}^{\mathrm{LB}} + f_{n_k}^{\mathrm{UB}}\right)/2$.
 5:     Calculate $\frac{\partial \mathcal{L}\left(f_{n_k},\alpha_{n_k}^m,p_{n_k}^m,\gamma_k,\omega_k,\eta\right)}{\partial f_{n_k}}$ according to equation (41).
 6:     if $\frac{\partial \mathcal{L}\left(f_{n_k},\alpha_{n_k}^m,p_{n_k}^m,\gamma_k,\omega_k,\eta\right)}{\partial f_{n_k}} > 0$ then
 7:       Set $f_{n_k}^{\mathrm{UB}} = f_{n_k}$.
 8:     else
 9:       Set $f_{n_k}^{\mathrm{LB}} = f_{n_k}$.
10:     end if
11:   until $\left|\frac{\partial \mathcal{L}\left(f_{n_k},\alpha_{n_k}^m,p_{n_k}^m,\gamma_k,\omega_k,\eta\right)}{\partial f_{n_k}}\right| \le \epsilon$.
12: end for
13: return $\boldsymbol{f}_{n_k}$.
```

tor $\hat{\boldsymbol{f}}_{n_k}$ should satisfy the following Karush-Kuhn-Tucker (KKT) condition:

$$\begin{aligned}&\frac{\partial\mathcal{L}\left(f_{n_k},\alpha_{n_k}^m,p_{n_k}^m,\gamma_k,\omega_k,\eta\right)}{\partial f_{n_k}}\\&=\frac{\partial F\left(f_{n_k},\gamma_k,\omega_k\right)}{\partial f_{n_k}}=2\lambda_{n_k}\kappa C_n V_{n_k} f_{n_k}\\&\quad-\frac{\gamma_k C_n V_{n_k}^2}{\left(Tf_{n_k}-C_nV_{n_k}\right)^2}+\omega_k=0,\forall k\in\mathcal{K}_n,\end{aligned}\tag{41}$$

where $F\left(f_{n_k},\gamma_k,\omega_k\right) = \lambda_{n_k}\kappa C_n V_{n_k} f_{n_k}^2 + \frac{\gamma_k V_{n_k} f_{n_k}}{Tf_{n_k}-C_nV_{n_k}} + \omega_k f_{n_k}$. However, it is difficult to write the closed-form expression of the optimal solution. Fortunately, $\mathcal{L}$ is convex regarding f_{n_k} and $\frac{\partial\mathcal{L}}{\partial f_{n_k}}$ increases monotonically among f_{n_k} since $Tf_{n_k} > C_nV_{n_k}$. Thus, the bisection method can be adopted to obtain the optimal $\hat{f}_{n_k}$ within $C_nV_{n_k}/T < f_{n_k} \le f_{\max}$. The procedure is detailed in Algorithm 3.

5.2.2 Communication Resource Allocation

According to (38), after determining $\hat{\boldsymbol{f}}_{n_k}$, $\mathcal{G}\left(\gamma_k,\omega_k,\eta\right)$ is decomposed into $|\mathcal{M}_n|$ sub-problems which can be independently solved at each sub-carrier. It is readily seen that (40) is convex regarding $p_{n_k}^m$, we get the optimal power allocation to D2D link k on sub-carrier m, expressed as

$$\hat{p}_{n_k}^m=\left[\frac{\left(u_k\beta_k+\gamma_k\right)B}{\ln 2\left(\lambda_{n_0}u_kV_{n_k}+\eta\right)}-\frac{\sigma^2}{h_{n_0,n_k}^m}\right]^+.\tag{42}$$

Since each sub-carrier can be allocated to almost one D2D link, $\hat{P}_m\left(\gamma_k,\eta\right)$ can be obtained, by searching over all $|\mathcal{K}_n|$ possible D2D links, as follows:

$$\begin{aligned}\hat{P}_m\left(\gamma_k,\eta\right)=\min_{k\in\mathcal{K}_n}&\left\{\left(u_k\lambda_{n_0}V_{n_k}+\eta\right)\hat{p}_{n_k}^m\right.\\&\left.-\left(u_k\beta_k+\gamma_k\right)B\log_2\left(1+\frac{\hat{p}_{n_k}^m h_{n_0,n_k}^m}{\sigma^2}\right)\right\}.\end{aligned}\tag{43}$$

Hence, the sub-carrier allocation result is achieved as

$$\alpha_{n_k}^m=\begin{cases}1, & \text{if } k=k^*=\arg\hat{P}_m\left(\gamma_k,\eta\right)\\0, & \text{otherwise,}\end{cases}\tag{44}$$

and the optimal power allocation is

$$p_{n_k}^m=\begin{cases}\hat{p}_{n_k}^m, & \text{if } \alpha_{n_k}^m=1\\0, & \text{otherwise.}\end{cases}\tag{45}$$

5.2.3 Lagrange Multipliers Update

After solving sub-problems in (39) and (40), $\mathcal{G}\left(\gamma_k,\omega_k,\eta\right)$ can be calculated by equation (38). Then we have to find a suitable set of $\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$, and η to maximize $\mathcal{G}\left(\gamma_k,\omega_k,\eta\right)$, which can be realized by the sub-gradient method [57]. Specifically, the Lagrange multipliers are updated by a step size sequence δ in the sub-gradient direction s, the update of $\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$, and η can be performed as follows:

$$\gamma_k\left(l+1\right)=\left[\gamma_k\left(l\right)+\delta_{\gamma_k}\left(l\right)s_{\gamma_k}\right]^+,\tag{46}$$

$$\omega_k\left(l+1\right)=\left[\omega_k\left(l\right)+\delta_{\omega_k}\left(l\right)s_{\omega_k}\right]^+,\tag{47}$$

$$\eta\left(l+1\right)=\left[\eta\left(l\right)+\delta_{\eta}\left(l\right)s_{\eta}\right]^+,\tag{48}$$

where l is the iteration index, $\delta_{\gamma_k}\left(l\right)=\delta_{\gamma_k}\left(1\right)/l$, $\delta_{\omega_k}\left(l\right)=\delta_{\omega_k}\left(1\right)/l$, $\delta_{\eta}\left(l\right)=\delta_{\eta}\left(1\right)/l$, and the corresponding sub-gradients are given as

$$s_{\gamma_k}=\frac{V_{n_k}f_{n_k}}{Tf_{n_k}-C_nV_{n_k}}-\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m B\log_2\left(1+\frac{p_{n_k}^m h_{n_0,n_k}^m}{\sigma^2}\right),\tag{49}$$

$$s_{\omega_k}=f_{n_k}-f_{\max},\tag{50}$$

$$s_{\eta}=\sum_{k=1}^{|\mathcal{K}_n|}\sum_{m=1}^{|\mathcal{M}_n|}\alpha_{n_k}^m p_{n_k}^m-p_{\max}.\tag{51}$$

With given $\boldsymbol{u}_k$ and $\boldsymbol{\beta}_k$ the dual-based communication and computational resource allocation algorithm can be summarized in Algorithm 4.

Algorithm 4 Dual-based Resource Allocation

```
Require: $\boldsymbol{u}_k$, $\boldsymbol{\beta}_k$, $l_{\max}$, and $\epsilon$.
Ensure: $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
1: Set initial Lagrange multipliers $\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$ and $\eta$.
2: Set iteration index $l = 0$, $\mathcal{L}^{(0)} = 0$.
3: repeat
4:   Obtain $\boldsymbol{f}_{n_k}$ via Algorithm 3.
5:   for each sub-carrier $m \in \mathcal{M}_n$ do
6:     for $k \in \mathcal{K}_n$ do
7:       Calculate $\hat{p}_{n_k}^m$ according to equation (42).
8:     end for
9:     Calculate $\hat{P}_m(\boldsymbol{\gamma}_k, \eta)$ according to equation (43).
10:    Obtain sub-carrier allocation result $\alpha_{n_k}^m$ and optimal
       power allocation $p_{n_k}^m$ using equation (44) and (45).
11:  end for
12:  Calculate $\mathcal{L}^{(l+1)}$ according to equation (35).
13:  Update $\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$ and $\eta$ according to equation (46), (47)
     and (48), respectively.
14:  $l \leftarrow l + 1$.
15: until $\frac{\left|\mathcal{L}^{(l)} - \mathcal{L}^{(l-1)}\right|}{\left|\mathcal{L}^{(l)}\right|} \le \epsilon$ or $l \ge l_{\max}$.
16: return $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
```

5.2.4 *Auxiliary Variables Update*

Then, we can update parameter $\boldsymbol{u}_k$ and $\boldsymbol{\beta}_k$ to obtain the global near-optimal solution $\left(\hat{\boldsymbol{\alpha}}_{n_k}^m, \hat{\boldsymbol{p}}_{n_k}^m, \hat{\boldsymbol{f}}_{n_k}\right)$, which satisfies the following conditions:

$$u_k \hat{r}_{n_k} - 1 = 0, \forall k \in \mathcal{K}_n, \tag{52}$$

and

$$\beta_k \hat{r}_{n_k} - \lambda_{n_0} V_{n_k} \sum_{m=1}^{|\mathcal{M}_n|} \hat{\alpha}_{n_k}^m \hat{p}_{n_k}^m = 0, \forall k \in \mathcal{K}_n. \tag{53}$$

Referring to [61], we define $\chi(u_k) = u_k \hat{r}_{n_k} - 1$ and $\psi(\beta_k) = \beta_k \hat{r}_{n_k} - \lambda_{n_0} V_{n_k} \sum\limits_{m=1}^{|\mathcal{M}_n|} \hat{\alpha}_{n_k}^m \hat{p}_{n_k}^m$, and $\boldsymbol{\beta}_k$ and $\boldsymbol{u}_k$ is updated by a modified Newton's method as follows:

$$u_k^{(l+1)} = u_k^{(l)} - \frac{\zeta^{i^{(l)}} \chi\left(u_k^{(l)}\right)}{r_{n_k}^{(l)}}, \tag{54}$$

and

$$\beta_k^{(l+1)} = \beta_k^{(l)} - \frac{\zeta^{i^{(l)}} \psi\left(\beta_k^{(l)}\right)}{r_{n_k}^{(l)}}, \tag{55}$$

where $i^{(l)}$ is the smallest integer among $i \in \{0, 1, 2, ...\}$ satisfying

$$\begin{aligned}\sum_{k=1}^{\mathcal{K}_n} &\left\{ \chi\left(u_k^{(l)} - \frac{\zeta^i \chi\left(u_k^{(l)}\right)}{r_{n_k}^{(l+1)}}\right)^2 + \psi\left(\beta_k^{(l)} - \frac{\zeta^i \psi\left(\beta_k^{(l)}\right)}{r_{n_k}^{(l+1)}}\right)^2 \right\} \\ &\le \left(1 - \sigma\zeta^i\right)^2 \sum_{k=1}^{|\mathcal{K}_n|} \left[\chi\left(u_k^{(l)}\right)^2 + \psi\left(\beta_k^{(l)}\right)^2\right],\end{aligned} \tag{56}$$

where $\zeta \in (0, 1)$ and $\sigma \in (0, 1)$.

The procedure of solving Problem $\mathcal{P}1$-2 is summarized in Algorithm 5.

Algorithm 5 Communication and Computational Resource Allocation

```
Require: $\boldsymbol{V}_{n_k}$ and $\epsilon$.
Ensure: $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
1: Obtain optimal $f_{n_0}$ according to equation (23).
2: Set iteration index $l = 0$, $\zeta \in (0, 1)$, $\sigma \in (0, 1)$, and
   randomly generate feasible solution $\boldsymbol{\alpha}_{n_k}^{m\ (0)}$, $\boldsymbol{p}_{n_k}^{m\ (0)}$.
3: Calculate $\boldsymbol{u}_k^{(0)}$ and $\boldsymbol{\beta}_k^{(0)}$ using (27) and (28), respectively.
4: repeat
5:   Given $\boldsymbol{u}_k^{(l)}$ and $\boldsymbol{\beta}_k^{(l)}$, obtain $\boldsymbol{\alpha}_{n_k}^{m\ (l+1)}$, $\boldsymbol{p}_{n_k}^{m\ (l+1)}$, and
     $\boldsymbol{f}_{n_k}^{(l+1)}$ by soling $\mathcal{P}1$-2C$''$ via Algorithm 4.
6:   Updata $\boldsymbol{u}_k^{(l+1)}$ and $\boldsymbol{\beta}_k^{(l+1)}$ according to equation (54)
     and (55).
7:   $l \leftarrow l + 1$.
8: until $\forall k \in \mathcal{K}_n, \left|\chi\left(u_k^{(l)}\right)\right| < \epsilon$ and $\left|\psi\left(\beta_k^{(l)}\right)\right| < \epsilon$.
9: return $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
```

Algorithm 6 JTPRA Algorithm

```
Require: $\{\mathcal{K}_n\}$, $\{\mathcal{M}_n\}$, $l_{\max}$, and $\epsilon$.
Ensure: $\boldsymbol{V}_{n_k}$, $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
1: Set iteration index $l = 0$, $\Phi_n^{(0)} = 0$.
2: Randomly generate feasible solution $\boldsymbol{\alpha}_{n_k}^{m\ (0)}$, $\boldsymbol{p}_{n_k}^{m\ (0)}$, $\boldsymbol{f}_{n_k}^{(0)}$.
3: repeat
4:   Given $\boldsymbol{\alpha}_{n_k}^{m\ (l)}$, $\boldsymbol{p}_{n_k}^{m\ (l)}$, and $\boldsymbol{f}_{n_k}^{(l)}$, obtain $\boldsymbol{V}_{n_k}^{(l+1)}$ by solving
     $\mathcal{P}1$-1 via Algorithm 2.
5:   Given $\boldsymbol{V}_{n_k}^{(l+1)}$, obtain $\boldsymbol{\alpha}_{n_k}^{m\ (l+1)}$, $\boldsymbol{p}_{n_k}^{m\ (l+1)}$, and $\boldsymbol{f}_{n_k}^{(l+1)}$ by
     solving $\mathcal{P}1$-2 via Algorithm 5.
6:   Calculate $\Phi_n^{(l+1)}$ according to equation (10).
7:   $l \leftarrow l + 1$.
8: until $\frac{\left|\Phi_n^{(l)} - \Phi_n^{(l-1)}\right|}{\left|\Phi_n^{(l)}\right|} \le \epsilon$ or $l \ge l_{\max}$.
9: return $\boldsymbol{V}_{n_k}$, $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$.
```

5.3 Overall Joint Task Partition and Resource Allocation Algorithm to Solve $\mathcal{P}1$

According to Subsections 5.1 and 5.2, the procedure of optimizing $\boldsymbol{V}_{n_k}$, $\boldsymbol{\alpha}_{n_k}^m$, $\boldsymbol{p}_{n_k}^m$, and $\boldsymbol{f}_{n_k}$ is shown in Algorithm 6. To this end, the task partition and communication and computational resource allocation problem $\mathcal{P}1$ is solved.

5.4 Computing Complexity Analysis

Proposition 4. *The overall complexity of joint task partition and resource allocation algorithm in all cooperative computing clusters is* $O\left(|\mathcal{N}|\,|\mathcal{K}|^4 \log_2 f_{\max} + |\mathcal{N}|\,|\mathcal{K}|^4 |\mathcal{M}|\right)$, *which is polynomial.*

Proof. In each cluster, the joint task partition and resource allocation problem $\mathcal{P}1$ is decomposed into two subproblems: $\mathcal{P}1$-1 and $\mathcal{P}1$-2, then they are solved iteratively with maximum number of iterations $l_{\max}$.

Firstly, for problem $\mathcal{P}1$-1, the complexity of sort operation is $O(|\mathcal{K}_n| \log_2 |\mathcal{K}_n|)$, and the complexity of task partition is $O(|\mathcal{K}_n|)$ in the worse case. So the the complexity of solving $\mathcal{P}1$-1 is $O(|\mathcal{K}_n| \log_2 |\mathcal{K}_n|)$.

Then, we have three layer iteration to solve problem $\mathcal{P}1$-2, the first layer (outermost) is updating auxiliary variables ($\boldsymbol{u}_k$, $\boldsymbol{\beta}_k$) with the complexity independent of $|\mathcal{K}_n|$,

TABLE 2
Default Parameters

Parameters	Values
T	100 ms
$\lvert\mathcal{N}\rvert$	4
$\lvert\mathcal{K}\rvert$	16
$\lvert\mathcal{M}\rvert$	64
$\hat{\Upsilon}$	1×10^{-3}
$d_{a,b}$	Uniform distributed in $[0, 1000]$ m
ε	2
V_n	Uniform distributed in $[100000, 140000]$ bits
C_n	Uniform distributed in $[900, 1000]$ cycle/bit
$f_{\max}$	400 MHz
κ	2×10^{-29}
$p_{\max}$	10 mW
σ^2	1×10^{-9} mW
B	12.5 KHz
ϵ	0.001

so the complexity is $O(|\mathcal{K}_n|)$; the second layer is updating Lagrange multipliers ($\boldsymbol{\gamma}_k$, $\boldsymbol{\omega}_k$, η) by sub-gradient method, with complexity converges in $O\left(|\mathcal{K}_n|^2\right)$ [57]; the third layer (innermost) is solving (39) and (40), the complexity of obtaining the optimal computing frequency by bisection method is $O(|\mathcal{K}_n| \log_2 f_{\max})$, and the complexity of obtaining sub-carrier and power allocation is $O(|\mathcal{M}_n| |\mathcal{K}_n|)$. Therefore, the complexity of solving $\mathcal{P}1$-2 is $O\left(|\mathcal{K}_n|^4 \log_2 f_{\max} + |\mathcal{K}_n|^4 |\mathcal{M}_n|\right)$.

Finally, the overall time complexity for solving $\mathcal{P}1$ can be abbreviated as $O\left(|\mathcal{K}_n|^4 \log_2 f_{\max} + |\mathcal{K}_n|^4 |\mathcal{M}_n|\right)$.

As such, the overall complexity of solving $\mathcal{P}1$ in all clusters is $O\left(|\mathcal{N}| |\mathcal{K}|^4 \log_2 f_{\max} + |\mathcal{N}| |\mathcal{K}|^4 |\mathcal{M}|\right)$. $\square$

6 Numerical Results

We present the numerical results that evaluate the properties and performance of proposed distributed scheduling scheme and the performance of cooperative computing framework. The maximum communication distance of each mobile device is set to 400 meters. The remaining battery life of each mobile device follows a uniform distribution between 80% and 100%, and λ_{n_k} is calculated by a percentage of the remaining battery life of device n_k to the total remaining battery life of all mobile devices[3]. The rest default parameters are shown in Table 2. To demonstrate the performance of our proposed DRESHA algorithm, we introduced a number of corresponding reference schemes:

- *DRESHA Without Reassignment (DRESHA-WR):* In each iteration of the DRESHA algorithm, SDs will not reject redundant HDs (steps 15-20 in Algorithm 1 will not be performed).
- *DRESHA Using Distance as Affection (DRESHA-Distance):* The preference of HD to SD is reconstructed as $n \succ_k n', d_{k,n} < d_{k,n'}, \forall k \in \mathcal{K}$. In other words, instead of resource efficiency, distance is used as the measure of the affection of HD on SD.
- *DRESHA Using Computing Complexity as Affection (DRESHA-Complexity):* The preference of HD to SD is reconstructed as $n \succ_k n', C_n < C_{n'}, \forall k \in \mathcal{K}$. In other words, instead of resource efficiency, computing complexity of task is used as the measure of the affection of HD on SD.
- *DRESHA Using Weight Factor as Affection (DRESHA-Factor):* The preference of HD to SD is reconstructed as $n \succ_k n', \lambda_{k,n} < \lambda_{k,n'}, \forall k \in \mathcal{K}$. In other words, instead of resource efficiency, weight factor of device is used as the measure of the affection of HD on SD.
- *Maximum Computing Frequency (MCF):* The computational resource is not optimized, all mobile devices use the maximum computing frequency ($f_{n_k} = f_{\max}$) to compute tasks.
- *Equal Power Allocation (EPA):* The transmission power is not optimized, SDs distribute transmission power equally among their allocated sub-carriers ($p_{n_k}^m = \frac{p_{\max}}{|\mathcal{M}_n|}$).

3. Although the calculation of λ_{n_k} has an impact on the maximization of lifetime of all mobile devices, this is not the key point of this paper, hence we choose this simple method for calculation.

6.1 Properties of Proposed Algorithm

In this subsection, we thoroughly examined two important properties of the proposed algorithm: convergence behavior and deviation from the globally optimal solution. The detailed results of our analysis are presented below.

6.1.1 Convergence Behavior

The proposed DRESHA algorithm consists of multiple layers of iterations. The convergence of the iterative algorithms involved has a significant impact on the execution time of the DRESHA algorithm. Therefore, we investigated the convergence of sum-of-ratios optimization algorithm (Algorithm 5), AO method (Algorithm 6), and DRESHA algorithm (Algorithm 1).

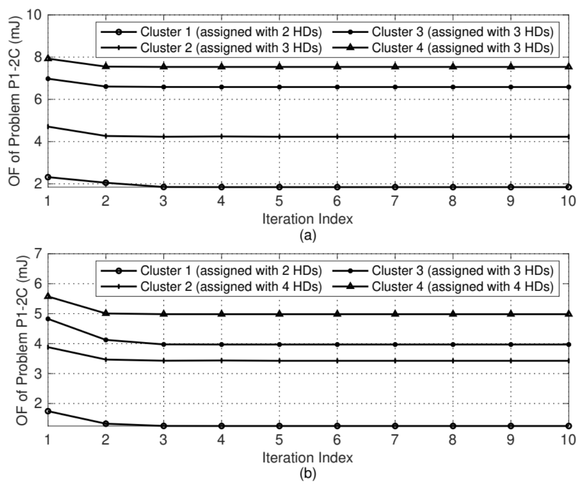

Fig. 4. Convergence of the Algorithm 5 in different cooperative computing clusters. The parameters are set as follows: (a) $|\mathcal{K}| = 16$, (b) $|\mathcal{K}| = 28$, and the rest are default.

Figure 4 illustrates the convergence behavior of Algorithm 5 with varying numbers of HDs. Across two diverse scenarios considered, the results indicate that the proposed sum-of-ratios optimization algorithm for $\mathcal{P}1$-$2C$ in each

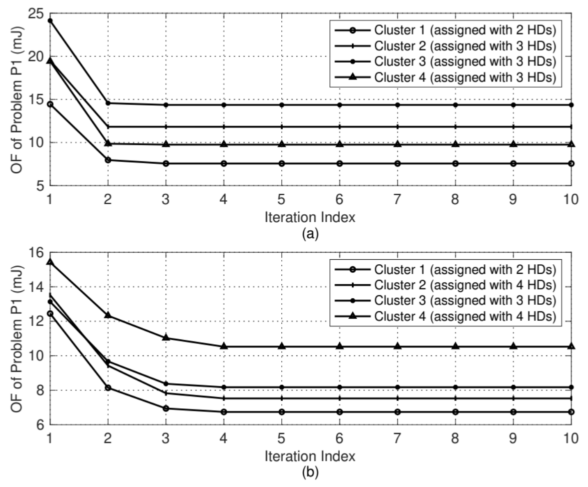

Fig. 5. Convergence of the Algorithm 6 in different cooperative computing clusters. The parameters are set as follows: (a) $|\mathcal{K}| = 16$, (b) $|\mathcal{K}| = 28$, and the rest are default.

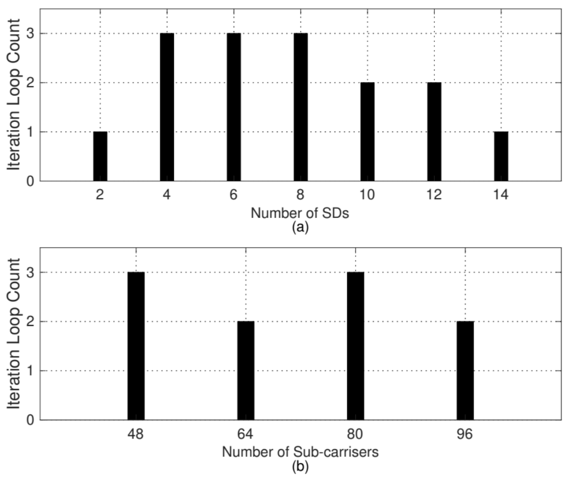

Fig. 6. Iteration loop count of the Algorithm 1 versus the number of source devices (SDs, which is equal to the number of clusters) (a) and the number of sub-carriers (b). The parameters are set as follows: (a) $|\mathcal{K}| = 24$, the rest are default. (b) all are default.

cooperative computing cluster achieves convergence within four iterations. Similarly, Figure 5 showcases the convergence behavior of Algorithm 6 for $\mathcal{P}$-1, where the weighted sum energy consumption in each cluster also converges within four iterations.

In Figure 6, we display the iteration count required for Algorithm 1 to achieve convergence (as specified in step 23 of Algorithm 1). Notably, the proposed DRESHA algorithm demonstrates rapid convergence in scenarios with varying numbers of SDs (clusters) and sub-carriers. This behavior is particularly intriguing. When the number of helpers is abundant or sparse, the algorithm converges swiftly. In rich helper environments, SDs encounter minimal competition for helpers from each other, leading to quick stabilization. Conversely, in scenarios with a scarcity of helper resources, SDs rarely reject matched helpers, resulting in similarly rapid stabilization. However, when the number of helpers

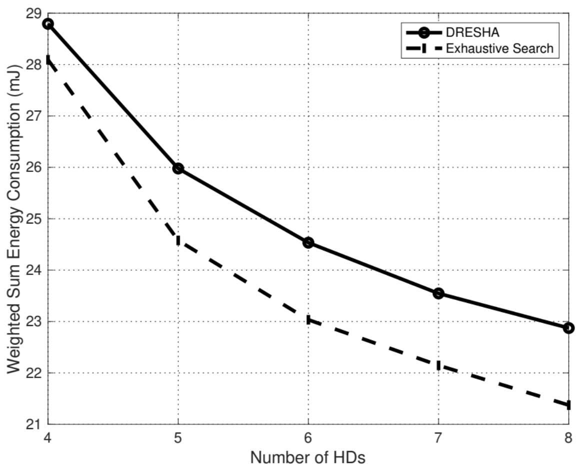

Fig. 7. Comparison of DRESHA and exhaustive search in terms of weighted sum energy consumption.

falls within an intermediate range, SDs engage in competitive interactions for helper, which extends the convergence process over several iterations.

Drawing from the analyses presented in Subsections 4.3 and 5.4, we can confidently conclude that the computing complexity of the proposed algorithms is polynomial, making them practically implementable.

6.1.2 Comparison With Exhaustive Search

In order to evaluate the gap between the outcomes achieved by the proposed DRESHA algorithm and the globally optimal solution, we performed a comprehensive comparative analysis utilizing an exhaustive search approach. The exhaustive search method, which employs a brute-force search strategy, is capable of attaining the globally optimal solution for optimization problems. However, it is important to note that the computational complexity of the exhaustive search method increases exponentially with the number of SDs, HDs, and sub-carriers involved. To mitigate this computational burden, we conducted the comparative analysis in a small-scale scenario, where the number of SDs was set to 2, the number of sub-carriers was set to 16, and all other parameters were maintained at their default values.

The obtained numerical results, depicted in Figure 7, reveal that the solutions provided by DRESHA are suboptimal. In comparison to the globally optimal solution, DRESHA leads to an increase in the weighted sum energy consumption ranging from 2.4% to 7.0%. Moreover, the deviation from the globally optimal solution escalates with a higher number of HDs. This can be attributed to the disparities between our proposed algorithm's source-helper assignment scheme and the globally optimal solution, which become more pronounced as the number of HDs increases. Nevertheless, it is crucial to acknowledge the intricate interplay between source-helper assignment and resource allocation, as well as the communication overhead inherent in distributed systems. Despite the observed suboptimality, our proposed algorithm remains practical and applicable, as it takes into account the complexities and

TABLE 3
The task completion ratio under local computing and cooperative computing framework, the results are obtained under 100 random scenario parameters, where $|\mathcal{N}| = 10$, $|\mathcal{K}| = 40$, $|\mathcal{M}| = 160$.

Framework (Algorithm)	Local Computing (None)	Cooperative Computing (DRESHA-WR)	Cooperative Computing (DRESHA)
$T = 50$ ms	0.000%	0.800%	1.000%
$T = 100$ ms	0.000%	58.800%	70.200%
$T = 150$ ms	0.000%	80.060%	98.890%
$T = 200$ ms	0.000%	98.350%	100.000%
$T = 250$ ms	12.500%	100.000%	100.000%
$T = 300$ ms	67.200%	100.000%	100.000%
$T = 350$ ms	100.000%	100.000%	100.000%

challenges inherent in achieving a globally optimal solution within real-world, resource-constrained environments.

6.2 Performance of Cooperative Computing

To evaluate the effectiveness of our proposed mobile cooperative computing framework, we compared the task completion ratio and the weighted sum energy consumption of our proposed framework with that of local computing. Our evaluation is presented below.

6.2.1 *Task Completion Ratio*

Table 3 shows the results obtained from local computing, cooperative computing using DRESHA-WR, and cooperative computing using DRESHA. The following observations were made: Firstly, in situations where tasks exhibit high delay sensitivity (computational demands exceeding the capacity of a single mobile device), cooperative computing can effectively increase the percentage of tasks completed on time compared to local computing. This underscores the effectiveness of our proposed cooperative computing framework in mitigating the insufficient computing capabilities of individual mobile devices. By harnessing the computational resources of the entire network, our framework addresses this challenge to a considerable extent. Secondly, under identical conditions, DRESHA demonstrates superior performance in terms of on-time task completion when contrasted with DRESHA-WR. This distinction arises from the iterative process employed by DRESHA, which intelligently reallocates redundant HDs from each cluster to an alternative resource pool. This iterative approach efficiently caters to the computational requirements of resource-constrained clusters, thereby ensuring a greater number of tasks are completed on time.

6.2.2 *Average Weighted Sum Energy Consumption*

Table 4 presents the average weighted sum energy consumption under different computing frameworks and algorithms, assuming all computational tasks are completed on time ($T = 350$ ms). Notably, our proposed cooperative computing framework consumes less weighted energy, and the result of DRESHA is better than that of DRESHA-WR. While wireless task offloading may introduce additional energy costs compared to local computing, cooperative computing remains advantageous in some heterogeneous networks (e.g. devices have different energy efficiency or energy weights) as SDs can offload tasks to more efficient

TABLE 4
Offloading Framework Comparison, when $T = 350ms$, $|\mathcal{N}| = 10$, $|\mathcal{K}| = 40$, $|\mathcal{M}| = 160$.

Framework (Algorithm)	Average Weighted Energy Consumption of Each Task
Local Computing (None)	18.2699 mW
Cooperative Computing (DRESHA-WR)	10.4637 mW
Cooperative Computing (DRESHA)	9.2262 mW

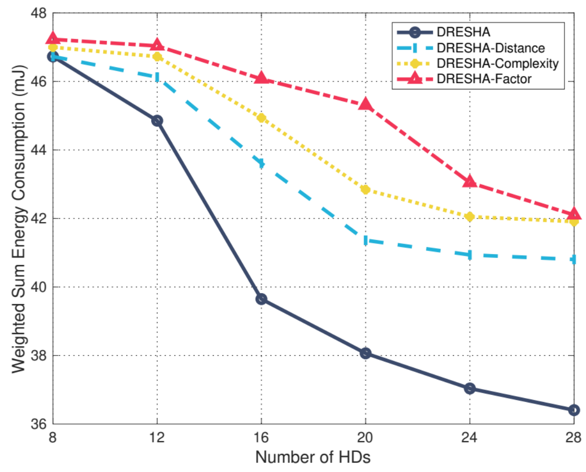

Fig. 8. The weighted sum energy consumption versus the number of helper devices (HDs) using different affections.

devices to reduce the overall network expense of computing.

6.3 Performance of Proposed Algorithm

In this subsection, we first evaluated the effect of using 'resource efficiency' as a matching preference. Then, the performance of the proposed algorithm was examined under different parameter settings.

6.3.1 *Performance of Proposed Affection*

Figure 8 depicts the numerical results of the weighted sum energy consumption as a function of the total number of HDs for four schemes employing different affection criteria. It is evident that the advantage of the resource efficiency based algorithm, DRESHA, over other algorithms (DRESHA-Distance, DRESHA-Complexity, and DRESHA-Factor) becomes increasingly pronounced with higher numbers of HDs. This phenomenon is attributed to the fact that in scenarios with a limited number of HDs, SDs have only a few choices for matching. Consequently, the distinctions between algorithms employing diverse affection criteria are marginal. However, as the number of HDs escalates, reliance on a singular criterion for preference can lead to inappropriate source-helper matches. This, in turn, inflates the overall system's weighted sum energy consumption. The resource efficiency based algorithm proposed in this paper accounts for factors including device distance, task complexity, and energy weight factor. This comprehensive

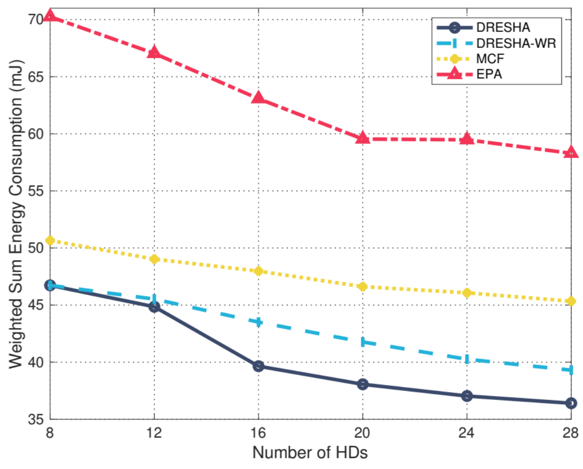

Fig. 9. The weighted sum energy consumption versus the number of helper devices (HDs).

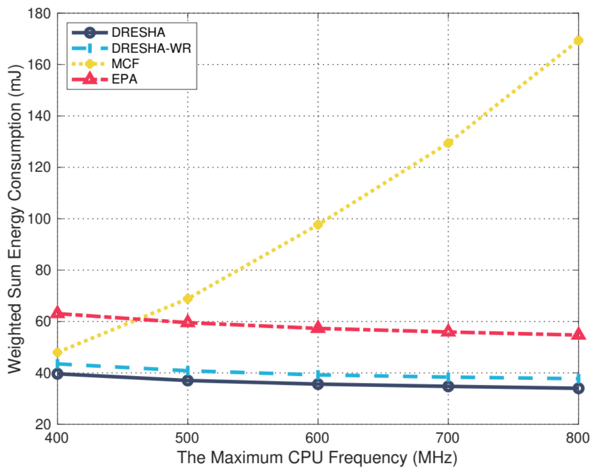

Fig. 11. The weighted sum energy consumption versus the maximum computing frequency of mobile devices.

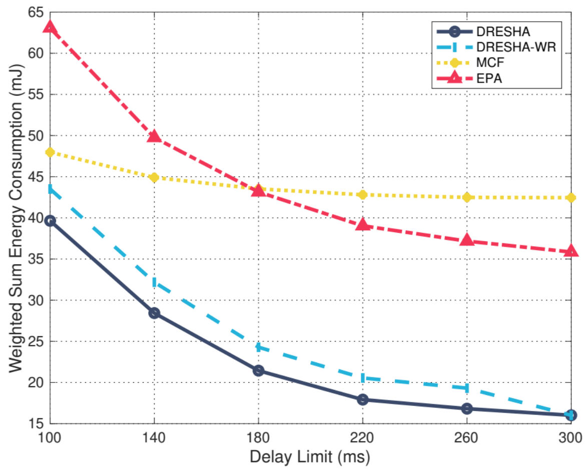

Fig. 10. The weighted sum energy consumption versus the delay limit.

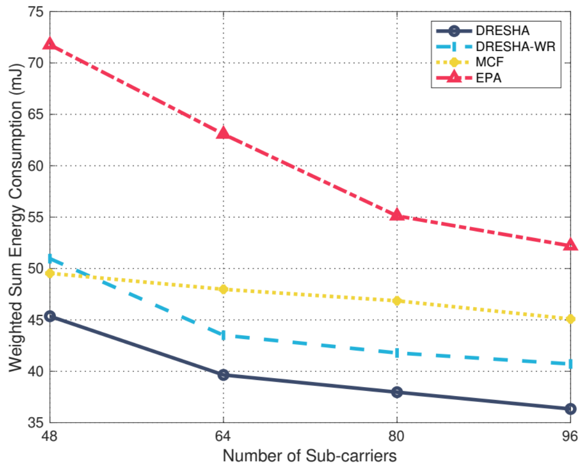

Fig. 12. The weighted sum energy consumption versus the number of sub-carriers.

consideration ensures the selection of matches that align with the optimization objective. Numerical outcomes validate that adopting resource efficiency as a preference metric results in a reduction of weighted energy consumption by 1% to 13.5%.

6.3.2 *Impact of Various Settings*

To further evaluate the performance of DRESHA, we compared it with reference schemes across various settings, encompassing variations in the number of HDs, delay limit, computational resource, and communication resource. The findings from these evaluations are detailed below.

Figure 9 displays the simulation results of the weighted sum energy consumption versus the total number of HDs for four schemes. Our observations are as follows. Firstly, an increase in the number of HDs corresponds to a decreasing trend in weighted sum energy consumption. The marginal benefit of introducing additional HDs stems from their capacity to efficiently share computing loads, thereby reducing the overall system's energy consumption. However, as the number of available helpers grows, the selection of efficient helpers becomes more critical, and an excess of helpers may not significantly impact energy consumption. It is this characteristic that inspired us to design DRESHA algorithm. Secondly, the disparity between DRESHA and DRESHA-WR results is minimal in scenarios with limited HDs. DRESHA's distinct advantage becomes apparent as the number of alternative HDs increases, resulting in a 1% to 11% reduction in weighted sum energy consumption. Finally, under different numbers of HDs, both DRESHA and DRESHA-WR outperform algorithms (MCF and EPA) that do not optimize computational or communication resources. Specifically, DRESHA reduces around 8% to 20% and 33% to 38% of weighted sum energy consumption compared to MCF and EPA, respectively.

Figure 10 displays the simulation results of the weighted sum energy consumption versus the delay limit of computational tasks. A comparative analysis of MCF, EPA, and DRESHA reveals intriguing insights. The weighted sum energy consumption using MCF decreases by 11%, while

EPA's decrease is more significant at 43%, both in response to increasing task delay tolerance. This trend underscores the greater impact of optimizing computational resources compared to communication resources. In contrast, the weighted sum energy consumption using DRESHA decreases by 59%.

Figure 11 presents the correlation between weighted sum energy consumption and the maximum computing frequency of mobile devices. Notably, an escalation in the maximum computing frequency leads to an 11% to 14% reduction in weighted sum energy consumption for computational resource optimization schemes (DRESHA, DRESHA-WR, and EPA). This is attributed to the improved likelihood of Algorithm 3 identifying optimal points with an extended search range. Conversely, MCF, which does not optimize computational resources, exhibits a gradual increase in energy consumption due to the imbalance between communication and computational resources. This is further highlighted by a 253% increase in MCF's weighted sum energy consumption when the maximum computing frequency is elevated from 400MHz to 800MHz.

Figure 12 represents the results of weighted sum energy consumption versus the number of sub-carriers, it can be seen that the energy consumption diminishes alongside an increasing number of sub-carriers. This is because the Shannon capacity formula is a concave function with respect to transmission power, and allocating power to a new subcarrier can yield greater gains than allocating the same power to an existing sub-carrier when the difference in gain between the sub-carriers is not significant. Furthermore, a larger number of sub-carriers amplifies the selection options for SDs, enabling the choice of sub-carriers with superior channel gains. These results reinforce DRESHA's superiority, evidenced by a 6% to 37% reduction in weighted sum energy consumption compared to other schemes.

7 Conclusion

The processing of sensor data on mobile devices with resource constraints and limited access to computing servers presents a challenge. To address this, we proposed a mobile cooperative computing framework for MCS that utilizes the idle resources of devices in the network to collaborate on tasks. Based on the model, a weighted sum energy consumption minimization problem was formulated, while considering task execution delay limits and practical constraints on communication and computing capabilities. We developed a distributed scheduling scheme to optimize offloading strategy, communication resources, and computational resources. A fast convergence of DRESHA algorithm was shown through numerical analysis, and we evaluated the performance of the cooperative computing framework under various simulation environments. Results demonstrated that cooperative computing can effectively improve the task completion ratio and reduce weighted sum energy consumption compared to local computing. Additionally, we evaluated the performance of DRESHA by comparing it to several schemes. In our future work, we intend to delve into the prediction of task allocation and sensor data volumes, thereby enhancing our ability to schedule resources across multiple time slots. Furthermore, we aim to address the complexity of our distributed algorithm for mitigating scheduling burdens on resource-limited mobile devices. Additionally, we aim to bridge the gap between the optimal solution and the results we obtained by exploring deep learning to solve the resource allocation problem. These forthcoming pursuits are poised to propel our research to new heights in order to address evolving demands and emerging challenges in the field.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] Z. Jiang, H. Zhu, B. Zhou, C. Lu, M. Sun, X. Ma, X. Fan, C. Wang, and L. Chen, "CrowdPatrol: A Mobile Crowdsensing Framework for Traffic Violation Hotspot Patrolling," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1401–1416, 2023.

[3] X. Wang, Z. Ning, X. Hu, E. C.-H. Ngai, L. Wang, B. Hu, and R. Y. K. Kwok, "A City-Wide Real-Time Traffic Management System: Enabling Crowdsensing in Social Internet of Vehicles," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 19–25, 2018.

[4] X. Kang, L. Liu, and H. Ma, "Data correlation based crowdsensing enhancement for environment monitoring," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[5] W. Yang, B. Wan, and X. Qu, "A Forward Collision Warning System Using Driving Intention Recognition of the Front Vehicle and V2V Communication," *IEEE Access*, vol. 8, pp. 11 268–11 278, 2020.

[6] L. Wang, Z. Yu, D. Zhang, B. Guo, and C. H. Liu, "Heterogeneous Multi-Task Assignment in Mobile Crowdsensing Using Spatiotemporal Correlation," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 84–97, 2019.

[7] X. Tao and W. Song, "Profit-Oriented Task Allocation for Mobile Crowdsensing With Worker Dynamics: Cooperative Offline Solution and Predictive Online Solution," *IEEE Transactions on Mobile Computing*, vol. 20, no. 8, pp. 2637–2653, 2021.

[8] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Incentive Mechanism for Privacy-Aware Data Aggregation in Mobile Crowd Sensing Systems," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2019–2032, 2018.

[9] H. Wu, L. Wang, K. Cheng, D. Yang, J. Tang, and G. Xue, "Privacy-Enhanced and Practical Truth Discovery in Two-Server Mobile Crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1740–1755, 2022.

[10] Y. Zhao, X. Gong, and X. Chen, "Privacy-Preserving Incentive Mechanisms for Truthful Data Quality in Data Crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2518–2532, 2022.

[11] S. Krishnaswamy, J. Gama, and M. M. Gaber, "Mobile Data Stream Mining: From Algorithms to Applications," in *2012 IEEE 13th International Conference on Mobile Data Management*, 2012, pp. 360–363.

[12] B. Wang, L. Kong, L. He, F. Wu, J. Yu, and G. Chen, "I(TS, CS): Detecting Faulty Location Data in Mobile Crowdsensing," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 808–817.

[13] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[14] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A Survey on Mobile Crowdsensing Systems: Challenges, Solutions, and Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[15] L. Ai, B. Tan, J. Zhang, R. Wang, and J. Wu, "Dynamic Offloading Strategy for Delay-Sensitive Task in Mobile-Edge Computing Networks," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 526–538, 2023.

[16] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami, "Human-Enabled Edge Computing: Exploiting the Crowd as a Dynamic Extension of Mobile Edge Computing," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 145–155, 2018.

[17] M. Marjanović, A. Antonić, and I. P. Žarko, "Edge Computing Architecture for Mobile Crowdsensing," *IEEE Access*, vol. 6, pp. 10 662–10 674, 2018.

[18] L. Lin, X. Lin, and X. Wang, "A Mobile Crowd Sensing Ecosystem based on Fog Computing Infrastructure," in *2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*, 2021, pp. 108–115.

[19] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich, and P. Bouvry, "Energy-Efficient Computation Offloading for Wearable Devices and Smartphones in Mobile Cloud Computing," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[20] R. Han, Y. Wen, L. Bai, J. Liu, and J. Choi, "Rate Splitting on Mobile Edge Computing for UAV-Aided IoT Systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1193–1203, 2020.

[21] Y. Yu, X. Bu, K. Yang, H. Yang, X. Gao, and Z. Han, "UAV-Aided Low Latency Multi-Access Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4955–4967, 2021.

[22] T. Bai, C. Pan, Y. Deng, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Latency Minimization for Intelligent Reflecting Surface Aided Mobile Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2666–2682, 2020.

[23] T. Bai, C. Pan, H. Ren, Y. Deng, M. Elkashlan, and A. Nallanathan, "Resource Allocation for Intelligent Reflecting Surface Aided Wireless Powered Mobile Edge Computing in OFDM Systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5389–5407, 2021.

[24] Y. Liu, L. Kong, and G. Chen, "Data-Oriented Mobile Crowdsensing: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2849–2885, 2019.

[25] J. Wang, L. Bai, J. Chen, and J. Wang, "Starling Flocks-Inspired Resource Allocation for ISAC-Aided Green Ad Hoc Networks," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 444–454, 2023.

[26] X. Li, C. You, S. Andreev, Y. Gong, and K. Huang, "Wirelessly Powered Crowd Sensing: Joint Power Transfer, Sensing, Compression, and Transmission," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 2, pp. 391–406, 2019.

[27] X. Xia, Y. Zhou, J. Li, and R. Yu, "Quality-Aware Sparse Data Collection in MEC-Enhanced Mobile Crowdsensing Systems," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1051–1062, 2019.

[28] D. Wu, J. Liu, and Z. Yang, "Bilateral Satisfaction Aware Participant Selection With MEC for Mobile Crowd Sensing," *IEEE Access*, vol. 8, pp. 48 110–48 122, 2020.

[29] L. Liu, L. Wang, Z. Lu, Y. Liu, W. Jing, and X. Wen, "Cost-and-Quality Aware Data Collection for Edge-Assisted Vehicular Crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5371–5386, 2022.

[30] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust Mobile Crowd Sensing: When Deep Learning Meets Edge Computing," *IEEE Network*, vol. 32, no. 4, pp. 54–60, 2018.

[31] L. Ma, X. Liu, Q. Pei, and Y. Xiang, "Privacy-Preserving Reputation Management for Edge Computing Enhanced Mobile Crowdsensing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 786–799, 2019.

[32] A. Ray, S. Mallick, S. Mondal, S. Paul, C. Chowdhury, and S. Roy, "A Framework for Mobile Crowd Sensing and Computing based Systems," in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2018, pp. 1–6.

[33] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting Massive D2D Collaboration for Energy-Efficient Mobile Edge Computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

[34] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D Communications Meet Mobile Edge Computing for Enhanced Computation Capacity in Cellular Networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, 2019.

[35] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint Task Assignment and Resource Allocation for D2D-Enabled Mobile-Edge Computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

[36] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A Bilevel Optimization Approach for Joint Offloading Decision and Resource Allocation in Cooperative Mobile Edge Computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4228–4241, 2020.

[37] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-Efficient Joint Task Offloading and Resource Allocation in OFDMA-Based Collaborative Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1960–1972, 2022.

[38] 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.

[39] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge, "POMT: Paired Offloading of Multiple Tasks in Heterogeneous Fog Networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8658–8669, 2019.

[40] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "POST: Parallel Offloading of Splittable Tasks in Heterogeneous Fog Networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3170–3183, 2020.

[41] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: Dispersive Stable Task Scheduling in Heterogeneous Fog Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3423–3436, 2019.

[42] Y. Zu, F. Shen, F. Yan, L. Shen, F. Qin, and R. Yang, "SMETO: Stable Matching for Energy-Minimized Task Offloading in Cloud-Fog Networks," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[43] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "METO: Matching-Theory-Based Efficient Task Offloading in IoT-Fog Interconnection Networks," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 705–12 715, 2021.

[44] S. Xu, X. Chen, X. Pi, C. Joe-Wong, P. Zhang, and H. Y. Noh, "iLOCuS: Incentivizing Vehicle Mobility to Optimize Sensing Distribution in Crowd Sensing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 8, pp. 1831–1847, 2020.

[45] H. Wu, J. Tao, and B. Xiao, "Towards a Stable and Truthful Incentive Mechanism for Task Delegation in Hierarchical Crowdsensing," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[46] T. Cai, Z. Yang, Y. Chen, W. Chen, Z. Zheng, Y. Yu, and H.-N. Dai, "Cooperative Data Sensing and Computation Offloading in UAV-Assisted Crowdsensing With Multi-Agent Deep Reinforcement Learning," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3197–3211, 2022.

[47] Z. Wei, B. Li, R. Zhang, X. Cheng, and L. Yang, "OCVC: An Overlapping-Enabled Cooperative Vehicular Fog Computing Protocol," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022.

[48] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "UAV-Enabled Mobile Edge Computing: Offloading Optimization and Trajectory Design," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[49] T. Cai, Z. Yang, Y. Chen, W. Chen, Z. Zheng, Y. Yu, and H.-N. Dai, "Cooperative Data Sensing and Computation Offloading in UAV-Assisted Crowdsensing With Multi-Agent Deep Reinforcement Learning," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3197–3211, 2022.

[50] X. Li, G. Feng, Y. Liu, L. Zhang, and S. Qin, "Cooperative Date Sensing, Communication and Computation in Resource Constrained Mobile Crowdsensing," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 4358–4363.

[51] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2897–2909, 2022.

[52] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[53] L. Bai, Z. Wu, and L. Zhou, "Achievable Refined Asymptotics for Successive Refinement Using Gaussian Codebooks," *IEEE Transactions on Information Theory*, pp. 1–1, 2023.

[54] M. Masoudi and C. Cavdar, "Device vs Edge Computing for Mobile Services: Delay-Aware Decision Making to Minimize Power Consumption," *IEEE Transactions on Mobile Computing*, vol. 20, no. 12, pp. 3324–3337, 2021.

[55] R. Han, J. Liu, L. Bai, and J. Liu, "Epidemic Theory-inspired Integrated Sensing and Communication Networks: Design and Analysis," *IEEE Communications Magazine*, pp. 1–7, 2023.

[56] R. Martínez, J. Massó, A. Neme, and J. Oviedo, "Single Agents and the Set of Many-to-One Stable Matchings," *J. Econ. Theory*, vol. 91, pp. 91–105, 2000.

[57] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1310–1322, 2006.

[58] K. Seong, M. Mohseni, and J. M. Cioffi, "Optimal Resource Allo-

cation for OFDMA Downlink Systems," in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1394–1398.

[59] T. Bai, C. Pan, H. Ren, Y. Deng, M. Elkashlan, and A. Nallanathan, "Resource Allocation for Intelligent Reflecting Surface Aided Wireless Powered Mobile Edge Computing in OFDM Systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5389–5407, 2021.

[60] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[61] Y. Jong, "An efficient global optimization algorithm for nonlinear sum-of-ratios problem," *Optimization Online*, pp. 1–21, 2012.