Matching-Theory-Based Multi-User Cooperative Computing Framework
Ya Zhou, Guopeng Zhang, Kezhi Wang and Kun Yang

Abstract—In this paper, we propose a matching theory based multi-user cooperative computing (MUCC) scheme to minimize the overall energy consumption of a group of user equipments (UEs), where the UEs can be classified into the following roles: resource demander (RD), resource provider (RP), and standalone UE (SU). We first determine the role of each UE by leveraging the roommate matching method. Then, we propose the college admission based algorithm to divide the UEs into multiple cooperation groups, each consisting of one RP and multiple RDs. Next, we propose the rotation swap operation to further improve the performance without deteriorating the system stability. Finally, we present an effective task offloading algorithm to minimize the energy consumption of all the cooperation groups. The simulation results verify the effectiveness of the proposed scheme.

Index Terms—multi-user cooperative computing, matching theory, computing task offloading.

I. INTRODUCTION

With the rapid deployment of the computationally-intensive tasks, e.g., virtual reality, the requirement for UE in terms of battery life and computing resource are also increasing. Although traditional fixed infrastructure-based mobile edge computing (MEC) may help to provide computing resources to UEs, they may be inaccessible in some situations, like disasters or emergency cases where infrastructures are unavailable. MUCC [1] has recently been proposed to allow a UE to utilize available computing resources from neighboring UEs. In the framework of MUCC, the UEs can be classified as one of the following roles: 1) RD, which has a computationally-intensive task to be processed; 2) RP, which has available computing resource to provide for other RDs; and 3) SU, which may do the task itself.

A one-to-one MUCC scheme was proposed in [2] to minimize the long-term energy consumption of two UEs. In [3], a RD has the option to offload the task to a MEC server or a RP. In [4], a one-to-multiple MUCC scheme was proposed to allow an RD to partition its task into multiple parts and offload them to multiple RPs for parallel execution. In [5], an RD is allowed to select one RP from a group of potential RPs. The purpose of the above works is mainly to reduce the energy consumption or execution delay of individual users. However, they do not address the following issues: (1) how to determine the role of a UE, i.e., as an RD, an RP, or a SU; (2) how to determine the association between the RDs and the RPs; and (3) how each RP allocates its computing and communication resources to the served RDs.

Against the above background, the main contribution of this paper is as follows: (1) The aim is to minimize the overall energy consumption of all UEs in a MUCC system through optimizing the user association and resource allocation. This problem is formulated as a mix integer nonlinear programming (MINLP) problem; (2) Due the high computational complexity of MINLP, we employ matching theory to design a low-complexity algorithm for finding the suboptimal solution of the problem. Specially, the energy consumption-related user preference functions are defined to solve the user role assignment and user association problems under the matching theory framework. Furthermore, a successive convex approximation (SCA) based task offloading algorithm is proposed to minimize the energy consumption of the whole system.

II. SYSTEM MODEL

As shown in Fig. 1, we consider a MUCC system consisting of a set \( \mathcal{N} \) \( \triangleq \{1, 2, \ldots, N\} \) of \( N \) UEs which can establish connections with each other via direct communications. Let \( \mathcal{T} \) \( \triangleq \{t = 1, 2, \ldots\} \) denote the sequence of the time slots. The length of slot \( t (\forall t \in \mathcal{T}) \) is \( \tau \). In slot \( t \), UE \( m (\forall m \in \mathcal{N}) \) has a computing task to execute, which is represented by \( \phi_m = (L_m, C_m) \), where \( L_m \) (in bits) denotes the amount of task data to be processed, whereas \( C_m \) (in CPU cycles/bit) denotes the number of CPU cycles required to be executed for each data bit. To complete task \( \phi_m \) in a slot, the CPU frequency of UE \( m \) is adjusted to \( f_m = C_m L_m / \tau \), by using the dynamic voltage and frequency scaling (DVFS) technique [2]. Let \( \gamma_m \) denote the effective capacitance coefficient of the CPU of UE \( m \). Then, the energy spent by UE \( m \) is as

\[
P_m^S = \gamma_m f_m^2 C_m L_m = \gamma_m C_m^3 L_m / \tau^2, \quad \forall m \in \mathcal{N}. \tag{1}
\]
In any slot $t$, the UEs in set $\mathcal{N}$ can be divided into the following three subsets, i.e., the RD set $\mathcal{N}^D$, the RP set $\mathcal{N}^P$, and the SU set $\mathcal{N}^S$. Then, we have
\[
\mathcal{N}^x \cap \mathcal{N}^y = \emptyset, \quad \forall x, y \in \{\mathcal{D}, \mathcal{P}, \mathcal{S}\} \text{ and } x \neq y,
\]
(2)

Let $i$ ($\forall i \in \mathcal{N}^D$) denote the index of an RD, and $j$ ($\forall j \in \mathcal{N}^P$) the index of an RP. Any RD $i$ can partition its task $\phi_i$ into two parts. One part is processed locally, while the other part, with data size $l_{i,j}$ is offloaded to RP $j$ for cooperative computing. Then, we have
\[
0 \leq l_{i,j} \leq L_i, \quad \forall i \in \mathcal{N}^D, \quad \forall j \in \mathcal{N}^P.
\]
(4)

We define the indicator function $\mathbb{1}_{0 \leq l_{i,j} \leq L_i} \in \{0, 1\}$ to represent the association between the RDs and the RPs. If RD $i$ offloads part of its task $\phi_i$ to RP $j$, $\mathbb{1}_{0 \leq l_{i,j} \leq L_i} = 1$, otherwise, $\mathbb{1}_{l_{i,j}=0} = 0$. In addition, RD $i$ can choose at most one RP $j$ to offload task $\phi_i$, that is
\[
\sum_{j \in \mathcal{N}^P} \mathbb{1}_{l_{i,j}} \leq 1, \quad \forall i \in \mathcal{N}^D,
\]
(5)

In order to avoid allocating too many RDs to RP $j$ in a time slot, we preset a parameter $a_j$ ($a_j \geq 1$) representing the maximum number of RDs that can be served by RP $j$ in a slot, that is
\[
\sum_{i \in \mathcal{N}^D} \mathbb{1}_{l_{i,j}} \leq a_j, \quad \forall j \in \mathcal{N}^P.
\]
(6)

The set of RDs that are associated with RP $j$ can thus be denoted by $\mathcal{M}_j \triangleq \{i \in \mathcal{N}^D \mid \mathbb{1}_{l_{i,j}} = 1\}$.

Since the UEs have limited battery energy, any UE $j$ can be used as a RP only when the available energy $E^A_j$ is greater than the threshold $E^\text{min}_j$. So we get the following constraint
\[
E^A_j \geq E^\text{min}_j, \quad \forall j \in \mathcal{N}^P.
\]
(7)

Next, we introduce the task offloading process in a cooperation group composed of RP $j$ and the associated RDs in set $\mathcal{M}_j$. As illustrated in Fig. 2, the task offloading will span the following two consecutive slots.

**Slot $t-1$ (Task data offloading):** In slot $t-1$, all the RDs in set $\mathcal{M}_j$ simultaneously transfer their task data to RP $j$ by using non-orthogonal multiple access (NOMA) [3], while the system allocates orthogonal channels (of the same size of $u$ MHz) for different RPs to receive the task data. Let $p_{i,j}$ denote the transmit power of RD $i$ associated with RP $j$, and $g_{i,j}$ denote the channel gain between RD $i$ and RP $j$. When successive interference cancellation (SIC) [3] is used, the achievable data rate of RD $i$ is given by
\[
r_{i,j} = w \log_2 \left(1 + \frac{p_{i,j}g_{i,j}}{\sum_{n \in \mathcal{M}_j \setminus \{i\}} p_{n,j}g_{n,j} + \sigma^2}\right),
\]
(8)

where $\sigma^2$ is the noise power at the receiver of RP $j$. For correct receiving the task data of RD $i$ of $l_{i,j}$ bits, the lower bound of the transmit power of RD $i$ is obtained as [3]
\[
p_{i,j} = \frac{\sigma^2}{g_{i,j}} \left(2^{\frac{l_{i,j}}{uT_s}} - 1\right) 2^\frac{1}{uT_s} \sum_{n \in \mathcal{M}_j \setminus \{i\}} l_{n,j} \leq P^\text{max}_i,
\]
(9)

where $P^\text{max}_i$ is the maximum transmit power of RD $i$.

By using eq. (9), the energy consumption of RD $i$ in slot $t-1$ for task data offloading is obtained as
\[
E^\text{TX}_{i,j} = p_{i,j} \tau, \quad \forall i \in \mathcal{M}_j, \quad \forall j \in \mathcal{N}^P.
\]
(10)

**Slot $t$ (Cooperative computing):** After collecting the task data of all the RPs in $\mathcal{M}_j$, RP $j$ can increase the CPU frequency as $f_j = \left(L_j + \sum_{i \in \mathcal{M}_j} l_{i,j} \right) / \tau$ for completing the accepted tasks as well as its own task in slot $t$. The energy spent by RP $j$ for processing the tasks is given by
\[
E^\text{P}_j = \gamma_j c^3_j \left(L_j + \sum_{i \in \mathcal{M}_j} l_{i,j} \right) / \tau^2, \quad \forall j \in \mathcal{N}^P.
\]
(11)

Since the size of the result is small, the energy spent by RP $j$ for feeding back the result is ignored.

In the meanwhile, RD $i$ ($\forall i \in \mathcal{M}_j$) can save its energy by decreasing the CPU frequency to $f_i = \left(L_i - \sum_{j \in \mathcal{N}^P} l_{i,j} \right) / \tau$ for processing the remaining part of task $\phi_i$. Therefore, the energy spent by RD $i$ for local execution in slot $t$ is given by
\[
E^\text{CX}_i = \gamma_i c^3_i \left(L_i - \sum_{j \in \mathcal{N}^P} l_{i,j} \right) / \tau^2, \quad \forall i \in \mathcal{N}^D.
\]
(12)

Then, the total energy spent by RD $i$ to complete task $\phi_i$ is given by
\[
E^\text{T}_i = E^\text{CX}_i + \sum_{j \in \mathcal{N}^P} E^\text{TX}_{i,j}, \quad \forall i \in \mathcal{N}^D.
\]
(13)

### III. Problem Presentation and Solution

Our objective is to minimize the overall energy consumption of all the UEs in the system, which can be given by
\[
\min_{l_{i,j}, 1_{i,j}} \sum_{i \in \mathcal{N}^D} E^\text{D}_i + \sum_{j \in \mathcal{N}^P} E^\text{P}_j + \sum_{n \in \mathcal{N}^S} E^\text{n}_n \quad \text{s.t.} \quad (2), (3), (4), (5), (6), (7), (9).
\]
(14)

Problem (14) is a MINLP, which is very difficult to solve in general. Next, we present an effective algorithm to find the sub-optimal solution of this problem based on matching theory. Note that we first do not consider constraint (7), thus allowing any UE to be used as a RP no matter how much energy it remains. This constraint will be dealt with in Sec. III-D by proposing a user role control operation. Furthermore, it should be noted that the cooperative UEs in slot $(t-1)$ can perform the computing tasks accepted in slot $(t-2)$, and the communication link in slot $t$ can be used to transfer the task data required in slot $(t+1)$. Hence, problem (14) minimizes the energy consumption of the cooperative UEs in each slot.
A. User role assignment

From the view of one-to-one roommate matching [6], an RP and the associated RD can be seen as a pair of roommate, and the UEs who do not get matched as the SUs. The user benefit is defined as the measure for two agents if they can be matched. Assume UE m \((\forall m \in N)\) is matched to UE k \((\forall k \in N' \text{ and } m \neq k)\), where UE m and UE k respectively act as the RD and the RP in the cooperation, then, they will get the same benefit as

\[
U_{m,k} = U_{k,m} = \max_{l_{i,j}} \left( (E^S_m + E^S_k) - (E^D_m + E^D_k) \right), \tag{15}
\]

where \((E^S_m + E^S_k)\) denotes the total energy consumption of UEs m and k when they work independently, whereas \((E^D_m + E^D_k)\) denotes the total energy consumption when they cooperate. Hence, the proposed benefit function (15) gives the maximum energy that UEs m and k can save through cooperation. Function (15) is a special case of problem (18) which can be solved by the proposed SCA algorithm.

By solving function (15), any UE m can rank the other UEs in set \(N\) according to the benefits that they can get from matching with other UEs. The preference list for UE m to choose roommates is set to \(\mathcal{PL}_m = \{U_{m,k}^{(1)}, \cdots, U_{m,n}^{(N-1)}\}\), where \(U_{m,k}^{(1)}\) and \(U_{m,n}^{(N-1)}\) denote the most preferred and the least preferred UEs for UE m, respectively. Then, the role of each UE can be determined by using the Irving algorithm [6], which is presented in Appendix A of the supplementary material1. The RP and RD in each successful matching pair are put into sets \(\mathcal{NP}\) and \(\mathcal{ND}\), respectively, but the UEs which can not successfully match a roommate are put into the SU set \(\mathcal{NS}\), rather than simply rejected as in the classical Irving algorithm. As a result, the UE set \(N\) is partitioned into three subsets \(\mathcal{ND}\), \(\mathcal{NP}\), and \(\mathcal{NS}\).

If two UEs i and j \((i \neq j)\) both have the incentive to leave their current partners and form a new pair with each other, these two UEs form a blocking pair \((i,j)\). One can see from Appendix B of the supplementary material1 that the matching result of the proposed user role assignment algorithm is stable as there exists no blocking pair in the matching result.

B. User association

After obtaining the RD and RP sets, we consider that one RP can serve multiple RDs in a slot. This user association problem can be seen as a multi-to-one two-side matching, which can be transformed into the college admission problem (CAP) [7] with the aim to achieve a stable multi-to-one matching. In the user association problem, the two disjoint UE sets, i.e., the RP set \(\mathcal{NP}\) and the RD set \(\mathcal{ND}\) correspond to the college set and the student set, respectively. The benefit that RD i can get after associating with RP j is shown as the energy saving for RD i with the help of RP j, then, one has

\[
U_{i,j}^T = \max_{l_{i,j}} \left( (E^S_i - E^D_i) \right). \tag{16}
\]

The benefit or the energy saving which RP j can get is

\[
U_{j,i}^T = \max_{l_{i,j}} \left( (E^S_j + E^D_j) - (E^D_i + E^D_j) \right). \tag{17}
\]

Following (16), the preference list of RD i is defined as \(\mathcal{PL}_{i}^T\), which is a ranking of RD i over all the RPs in set \(\mathcal{NP}\) in descending order according to \(U_{i,j}^T\) \((\forall j \in \mathcal{NP})\). Similarly, following (17), the preference list of RP j is defined as \(\mathcal{PL}_{j}^T\), which is a ranking of RP j over all the RDs in set \(\mathcal{ND}\) in descending order according to \(U_{j,i}^T\) \((\forall i \in \mathcal{ND})\).  

Up to now, we have mapped the user association problem into the CAP. It can then be solved by using the Gale-Shapley (GS) algorithm, which is presented in Appendix C of the supplementary material1. Although the obtained matching result is stable, some RDs may not match their preferred RPs [7], [8]. Hence we propose the rotation swap operation (RSO) which allows RDs to associate with other RPs with better performance through exchanging their currently matched RPs without losing the stability. Before presenting the RSO, some definitions and notations are given below.

Definition 1. Cabal: a cabal \(K = \{k_1, \ldots, k_x, \ldots, k_K\}\) is a subset of \(\mathcal{ND}\), such that \(\Omega(k_x-1) \succ k_x \Omega(k_x)\) according to \(\mathcal{PL}_{k_x}^T\), \(\forall k_x \in K\) \((x = 1 \Leftrightarrow K\) when \(x = 1\).

Definition 2. Accomplice: the accomplice set \(\mathcal{H}(K)\) of cabal \(K\) is a subset of \(\mathcal{ND}\), such that \(h \in \mathcal{H}(K)\) if \(1 \Leftrightarrow x \neq k_x\), for any \(k_x \in K\) if \(\Omega(k_x) \succ h\ \Omega(h)\) and \(h \succ x\ k_x\), or \(2\) \(h \in K\) and \(h = k_x\ \forall k_x \in K\), for any \(k_x \in K\) and \(x \neq 1\) if \(\Omega(k_x) \succ x\ k_x\ \Omega(k_x)\) and \(k_x \succ x\ \Omega(k_x)\).

Definitions 1 and 2 show that the RDs in set \(\mathcal{H}(K)\) may have prevented the RDs in set \(K\) from matching their more preferred RPs. Next, we propose the falsify operation, which enables the RDs in set \(\mathcal{H}(K)\) to help the RDs in set \(K\) match more preferred RPs. The detail is given below.

Let \(s \succ r\ n\) denote that RD i prefers RP s over RP n in \(\mathcal{PL}_{i}^T\). Let \(\Omega(i)\) denote the partner of RD i obtained by using the GS algorithm. Assume \(\Omega(i)\) denotes the partner of RD i obtained by further performing the RSO. If \(\Omega(i) \succ i\),\(\Omega(i)\), we say that matching \(\Omega(i)\) is “at least as good as” matching \(\Omega\), which is denoted by \(\Omega_S \succ \Omega\). We rewrite the preference list of RD i as \(\mathcal{PL}_{i}^T = (\mathcal{PL}_{i}^T(i), \Omega(i), \mathcal{PL}_{i}^D(i))\), where \(\mathcal{PL}_{i}^T(i)\) and \(\mathcal{PL}_{i}^D(i)\) denote the RPs that are ranked higher and lower than \(\Omega(i)\), respectively. Then, the falsify operation is to move RP \(\theta\) \((\theta \in \mathcal{PL}_{i}^D(i))\) from \(\mathcal{PL}_{i}^D(i)\) to \(\mathcal{PL}_{i}^T(i)\). Let \(\pi_r(\mathcal{PL}_{i}^T)\) denote the random permutation of \(\mathcal{PL}_{i}^T\). Then, we can have the following Lemma.

Lemma 1. Let \(J \subseteq \mathcal{ND}\). If all the RDs in set \(J\) \((\forall i \in J)\) submit their falsified lists in the form \((\pi_r(\mathcal{PL}_{i}^D(i) - \theta), \Omega(i), \pi_r(\mathcal{PL}_{i}^T(i) + \theta))\), then \(\Omega_S \succ \Omega\).  

Proof. See Appendix D of the supplementary material1.
the existing algorithms, e.g. [7], [8], allow only two users to swap resources with each other, while Algorithm 1 allows more than two users to swap resources at the same time, thus greatly improving the execution efficiency of the algorithm.

Algorithm 1 Rotation Swap Operation (RSO)
1. Let $\Omega = \Omega_0$, where $\Omega_0$ is the matching result obtained by performing the Irving algorithm and the GS algorithm sequentially.
2. Find the largest cabal $C$ from $\Omega$ by using Definition 1.
3. Find the accomplice $H(\Omega)$ of cabal $C$ by using Definition 2.
4. for all RD $i \in C$ do
5. if $i \in H(\Omega) - C$ then
6. RD $i$ falsifies its preference list $P_{\Omega_i}$ as $\pi_r(P_{\Omega_i}(i) - \theta)$, $\Omega(i)$, $\pi_r(P_{\Omega_i}(i) + \theta)$, where $\theta = \{c \mid \Omega(m) \in \Omega(\Omega_i), m \succeq m + 1\}$.
7. else
8. RD $i$ (i.e., $i \in C$) falsifies its preference list $P_{\Omega_i}$ as $\pi_r(P_{\Omega_i}(i) - \theta)$, $\Omega(i)$, $\pi_r(P_{\Omega_i}(i) + \theta)$, where $\theta = \{c \mid \Omega(m) \in \Omega(\Omega_i), m \succeq m + 1\}.
9. end if
10: end for
11: return The modified preference lists.

C. Resource allocation

Now, the UEs in the system have been divided into multiple cooperation groups, each consisting of one RP and multiple RDs. Therefore, problem (14) can be rewritten as

$$\min_{l_{ij}, p_{ij}} E_j^p + \sum_{i \in M_j} E_i^d$$

s.t. (4), (9).

We note that problem (18) is non-convex due to the non-convexity of constraint (9). To address this issue, we define $N_0 = \sigma^2 / g_{i,j}$ and $\alpha = 2^2 / \tau w$, and convert constraint (9) into the following form

$$p_{ij} = N_0 \left( \alpha \sum_{i \in M_j} l_{ij} - \alpha \sum_{n \in M_j \setminus \{i\}} l_{nj} \right) \leq p_{ij}^{\text{Max}}. \quad (19)$$

We obtain the partial derivatives of the $p_{ij}$ with respect to $l_{ij}$ and $l_{nj}$ as

$$\nabla_{l_{ij}} p_{ij} = N_0 \ln \alpha \left( \alpha \sum_{i \in M_j} l_{ij} \right). \quad (20)$$

and

$$\nabla_{l_{nj}} p_{ij} = N_0 \ln \alpha \left( \alpha \sum_{i \in M_j \setminus \{i\}} l_{nj} \right). \quad (21)$$

respectively. Then, we get the following Lemma.

Lemma 2. $\nabla_{l_{ij}} p_{ij}$ and $\nabla_{l_{nj}} p_{ij}$ are Lipschitz continuous on $l_{ij}$ and $l_{nj}$ with the constants $L_{C}$ and $L_{F}$, respectively, where $L_{C}$ and $L_{F}$ are the Lipschitz constants.

Proof. See Appendix E of the supplementary material. \Box

If the first derivative of a non-convex function is Lipschitz continuous on the variables with the corresponding Lipschitz constants and each variable is nonempty, closed, and convex, then, the non-convex function can be converted to a convex one by SCA [3]. Lemma 2 indicates that $p_{ij}$ meets this rule and thus can be converted to a convex one by using SCA.

Let $D_j \triangleq (l_{1,j}, ..., l_{M_j,j})^T$ denote the offloading strategy profile of the RDs in set $M_j$. Let $\delta = 1, 2, ...$ denote the iterative numbers. Then, $l_{ij}[\delta]$ represents the amount of task data offloaded from RD $i$ to RP $j$ in the $\delta$th iteration. By using eqs. (20) and (21), we derive the gradient of $p_{ij}$ on $D_j[\delta]$ as

$$\nabla_{D_j[\delta]} p_{ij} = \left( \frac{\partial p_{ij}}{\partial l_{1,j}[\delta]}, \frac{\partial p_{ij}}{\partial l_{2,j}[\delta]}, ..., \frac{\partial p_{ij}}{\partial l_{M_j,j}[\delta]} \right)^T. \quad (22)$$

According to [9], a strongly convex function to approximate $p_{ij}$ can then be constructed as

$$\bar{p}_{ij} \triangleq p_{ij}[\delta] + (\bar{D}_j - D_j[\delta])^T \nabla_{D_j[\delta]} p_{ij} + \frac{\lambda}{2} ||D_j - D_j[\delta]||_2^2 + \frac{1}{2} \lambda ||D_j - D_j[\delta]||^2 \nabla_{D_j[\delta]} p_{ij}. \quad (23)$$

where $\lambda > 0$. Now, we can approximate constraint (9) as

$$\bar{p}_{ij} \leq \bar{p}_{ij}^{\text{Max}}, \forall i \in M_j. \quad (24)$$

By replacing constraint (9) with constraint (24), problem (18) is converted to a convex one and can be solved by using convex optimization tools, e.g., CVX [3]. The detail of the SCA algorithm is presented in Appendix F of the supplementary material.

D. Overall algorithm

Algorithm 2 Overall algorithm to solve problem (14)
1: Perform the user role control to limit the UEs with available energy less than the threshold, i.e., $E_{\text{max}} \leq E_{\text{min}}$ to become RPs.
2: Perform the Irving algorithm to partition the UE set $N$ into three subsets $N^p$, $N^c$, $N^f$.
3: Perform the GS algorithm to establish the association between the UEs in set $N^p$ and the UEs in set $N^c$.
4: repeat
5: Perform Algorithm 1, i.e., the RSO to find cabal and falsify the preference lists of the UEs in the cabal.
6: Perform the GS algorithm by using the modified preference lists to update the UE association, that is
7: if there exist no $j \in N^c$ such that $\Omega(j) \succ j \Omega_i(j)$ then
8: update the UE association.
9: end if
10: until the matching result has no cabal.
11: Perform the SCA algorithm for each cooperation group to obtain the resource allocation.

We present the overall algorithm for implementing the MUCC in Algorithm 2. The algorithm first performs the user role control operation (line 1), which limits the UEs with the remaining energy less than the threshold to become RPs, thus satisfying constraint (7). The complexity of the SCA algorithm (line 11) is related to the number of iterations $\delta$ as well as the complexity of the solver for the convex problems [3]. Since the adopted CVX toolbox is based on the standard interior point method, whose complexity is $O(N^3)$ [10], the complexity of the SCA algorithm is thus $O(\delta N^3)$. As analyzed in Appendix G of the supplementary material, the computational complexity of Algorithm 2 is $O(\delta N^3)$.

The proposed MUCC scheme is enabled by D2D communications. The effective range of the D2D communications is limited to one-hop, and a cellular base station (BS) is required to establish and manage the D2D links. Therefore, Algorithm 2 can be performed at a BS in a centralized manner, as the BS can obtain the channel state information (CSI), and the
IV. SIMULATION RESULTS

We consider a 100 $\times$ 100 m$^2$ rectangular area, where multiple UEs are randomly distributed. This is a general WLAN case that enables distributed UEs to perform D2D communications and further perform cooperative computing. Other parameters used in the simulation are given below. The number of CPU cycles required to execute one task bit is $C_m = 500$, and the effective capacitance coefficient is $\gamma_m = 10^{-28}$ [2]. The noise power is $\sigma^2 = 10^{-9}$. The maximum transmit power of the UE is set to 0.1 W. The duration of a time slot is $\tau = 0.2$ s. The bandwidth is $w = 1$ MHz. The channel power gain is set to $g_{i,j} = \zeta_i/j/d^\beta$ [7], where $d$ is the distance between two UEs (in meters), $\beta = 3$ is the power-scaling path-loss factor, and $\zeta_i/j$ follows an exponential distribution with a unit mean, which captures the fading and shadowing effects. The maximum connection constraint of the UE is $a_m = 2$.

To verify the effectiveness of the proposed MUCC scheme, we compare our proposed algorithm with the exhaustive search (ES) algorithm. Although the ES algorithm can find the optimal solution of problem (14), the computational complexity is too high to be used in practice. The number of UEs in the system is set to less than 12. The simulation results in shown in Fig. 3. One can see from Fig. 3 that the performance loss of our proposed algorithm is small compared to the optimal solution obtained by using the ES algorithm, but our algorithm far outperforms the local computing method. It indicates that the proposed algorithm achieves a better compromise between computational complexity and optimal performance.

Next, we compare our proposed algorithm with the one-to-one random-matching algorithm, the pure Irving algorithm, and the Irving plus GS algorithm in a network with a larger number of UEs. The amount of task data for each UE is randomly distributed in $[0, 1]$ Mbits. Fig. 4 shows the overall energy versus the number of the UEs. One can see that our proposed algorithm achieves better energy efficiency over the other algorithms in all cases. The reason behind is that the proposed algorithm realizes the global user matching and resource allocation according to the available computing resources to the RPs, the required computing resources for the RDs, and the channel conditions between them, thus greatly reducing the overall energy consumption of the system.

V. CONCLUSION AND FUTURE WORKS

This paper presents a MUCC scheme based on matching theory that answers the questions about who and how to cooperate with. In the future work, we will extend the single-hop MUCC scheme to the multi-hop MUCC scheme, where the task data of a RD could be forwarded by multiple relay UEs to reach the RP. In such a case, the energy consumption of all the relay UEs along a route should be considered when balancing the energy consumption for task execution and that for data transmission. In addition, meeting the delay requirements for user computing tasks is another huge challenge.

REFERENCES

[1] X. Chen and J. Zhang, “When d2d meets cloud: Hybrid mobile task offloadings in fog computing,” in 2017 ICC, May 2017, pp. 1–6.
[2] Q. Lin, F. Wang, and J. Xu, “Optimal task offloading scheduling for energy efficient d2d cooperative computing,” IEEE Commun. Lett., vol. 23, no. 10, pp. 1816–1820, Oct 2019.
[3] D. Wu, F. Wang, X. Cao, and J. Xu, “Wireless powered user cooperative computation in mobile edge computing systems,” in 2018 GC Wkshps, Dec 2018, pp. 1–7.
[4] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, “Joint computation and communication cooperation for energy-efficient mobile edge computing,” IEEE Internet Things J., vol. 6, no. 3, pp. 4188–4200, Jun. 2019.
[5] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, “Energy efficient cooperative computing in mobile wireless sensor networks,” IEEE Trans. on Cloud Comput., vol. 6, no. 1, pp. 114–126, Jan 2018.
[6] R. W. Irving, “An efficient algorithm for the “stable roommates” problem,” J Algorithms, vol. 6, no. 4, pp. 577–595, 1985.
[7] Y. Gu, Y. Zhang, M. Pan, and Z. Han, “Matching and cheating in device to device communications underlying cellular networks,” IEEE J Sel. Areas Commun., vol. 33, no. 10, pp. 2156–2166, Oct 2015.
[8] B. Wang, Y. Sun, S. Li, and Q. Cao, “Hierarchical matching with peer effect for low-latency and high-reliable caching in social IoT,” IEEE Internet Things J., vol. 6, no. 1, pp. 1193–1209, Feb 2019.
[9] F. Facchinei, S. Sagratella, and G. Scutari, “Flexible parallel algorithms for big data optimization,” in 2014 ICASSP, May 2014, pp. 7208–7212.
[10] J. Xu and L. Qiu, “Energy efficiency optimization for mimo broadcast channels,” IEEE Trans. Wireless Commun., vol. 12, no. 2, pp. 690–701, February 2013.