

Resolving Multitask Competition for Constrained Resources in Dispersed Computing: A Bilateral Matching Game

Hongjia Wu¹, Jiao Zhang¹, Zhiping Cai¹, *Member, IEEE*, Qiang Ni², *Senior Member, IEEE*, Tongqing Zhou¹, Jiaping Yu¹, Haiwen Chen¹, and Fang Liu¹, *Member, IEEE*

Abstract—With the explosive emergence of computation-intensive and latency-sensitive applications, data processing could be envisioned to perform closer to the data source. Similar to edge and fog computing, dispersed computing is considered as a complementary computing paradigm, which can excavate potential computation resources in the network to users, and serve as a supplement for sharing the computational burden when the edge is overloaded. In this article, we first make full use of idle and geographically dispersed computation resources via task offloading, contributing to conserve energy for mobile devices. Especially, a dispersed computing offloading framework concerning the interests of users and networked computation points is proposed. We further transform the initial problem into a multiobjective optimization problem subject to latency and resource constraints. To tackle such a complex problem, an energy-saving bilateral matching algorithm is designed to obtain the optimal task offloading strategy. The simulation results demonstrate that our proposed algorithm can outperform the benchmark schemes in terms of user fairness and can achieve a relatively balanced energy cost ratio. Furthermore, comparative experiments with edge computing are implemented in Amber Response and Disaster Relief scenarios, respectively, to reveal the advantages of the proposed framework.

Index Terms—Bilateral matching, dispersed computing, energy saving, idle computation resources, multiobjective, offloading.

Manuscript received January 27, 2021; revised April 12, 2021; accepted April 17, 2021. Date of publication April 26, 2021; date of current version November 19, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62072465, Grant 62001483, and Grant U19B2024; in part by the National Key Research and Development Program of China under Grant 2018YFB0204301; in part by the Science and Technology Innovation Program of Hunan Province under Grant 2020RC2045; and in part by the National University of Defense Technology (NUDT) Research Grants under Grant ZK19-38. (*Corresponding authors: Zhiping Cai; Fang Liu.*)

Hongjia Wu is with the School of Design, Hunan University, Changsha 410082, China, and also with the College of Computer, National University of Defense Technology, Changsha 410082, China (e-mail: wuhongjia19@nudt.edu.cn).

Jiao Zhang is with the College of Electronic Science and Technology, National University of Defense Technology, Changsha 410082, China (e-mail: zhangjiao16@nudt.edu.cn).

Zhiping Cai, Tongqing Zhou, Jiaping Yu, and Haiwen Chen are with the College of Computer, National University of Defense Technology, Changsha 410082, China (e-mail: zpcai@nudt.edu.cn; zhoutongqing@nudt.edu.cn; yujiaping19@nudt.edu.cn; chenhaiwen13@nudt.edu.cn).

Qiang Ni is with the School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, U.K. (e-mail: q.ni@lancaster.ac.uk).

Fang Liu is with the School of Design, Hunan University, Changsha 410082, China (e-mail: fangli@hnu.edu.cn).

Digital Object Identifier 10.1109/IIOT.2021.3075673

I. INTRODUCTION

WITH the increasing evolution of embedding processing into “smart” network nodes [1] (such as smartphones, watches, automobiles, etc.), the in-network computing paradigms, such as edge computing [2] and fog computing, have drawn unprecedented attention. Through offloading the computation-intensive tasks to the edge servers of networks [3]–[5], the computation experience and the battery lifetime of smart devices can be greatly improved. However, due to the constrained computation capacity, economic cost, and network scalability, edge servers are generally distributed deployment. The conflict between computation tasks and limited resources is inevitable, when massive mobile devices send requests to the edge servers.

To deal with this conflict, some methods [6]–[9] propose to leverage the collaboration among edge servers to achieve load balancing, thereby reducing the computational burden on the high-load edge servers, whereas such edge collaboration-based methods are usually designed for scenarios with sufficient network state information, and are not suitable for dynamically changing network environments. To better cope with the network dynamics, a series of learning-based methods are proposed [10]–[12], wherein the optimal offloading strategy can be learned without knowing the prior knowledge of network dynamics. However, these proposals rely on the wide deployment of computation infrastructure (e.g., edge servers), which cannot scale to the sudden surge of traffic or network service damages due to disasters. Introducing mobile devices (e.g., mobile phones and drones) [13]–[16] to overcome the computation resources limitations in these situations provides a promising alternative, especially considering the occasionally vacant computation capacity onboard. Yet, it is nontrivial to accommodate the distributed devices for dynamic computation tasks as the involvement will incur additional costs.

Driven by the popularity of Internet of Things (IoT) devices, dispersed computing [17], [18] as a new computing paradigm is eye-catching. Based on the mutual aid idea of “one for all and all for one,” dispersed computing can adequately leverage the idle computation resources of surrounding available devices. Fig. 1 shows a dispersed computing offloading framework composed of a group of networked computation points (NCPs) [19], which form a collaborative organic network to provide users with scattered, closer, and diverse offloading

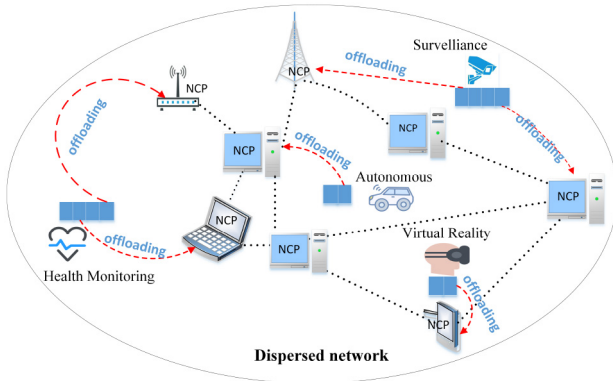


Fig. 1. Dispersed computing offloading framework.

services. At the same time, some offloading techniques for dispersed computing [19]–[23] have attracted attention, involving graph task allocation and throughput optimization. However, the offloading problem of multiuser multitask occupying multi-NCP resources has not been taken into account, especially the energy consumption concerns for both types of entities during the offloading. Note that it is critical to attain energy efficiency when the computation is conducted by resource-limited end devices. Therefore, the goal of this work is to provide a complementary dispersed computing framework that can adapt to dynamic tasks request with energy awareness for both the user devices and the NCPs.

Nevertheless, due to the collaboration of computation resources and the sharing of iterative information, the management of dispersed computing is inherently difficult for several reasons. First, the computation resources and lifetime of NCPs are limited. NCPs should have the right to select the offloading requests to maximize their interests. Second, there is competition among users for the limited resources of NCPs. Moreover, the tasks generated by different users might differ in computational complexity and data volumes. The allocation of heterogeneous resources should be compatible with the users' interests. Third, most devices are private, so the unnecessary information interaction between devices should be avoided. Therefore, the design of dispersed computing offloading algorithms that balance the interests of users and NCPs to efficiently utilize dispersed and heterogeneous idle resources is a critical problem. Solving such a problem yields three open technical challenges.

- 1) How to make full use of idle and geographically dispersed NCPs?
- 2) How to handle user competition for limited resources?
- 3) How to reduce the information interaction among devices and achieve dispersed computing?

To address these challenges, users and NCPs are encouraged to make independent decisions to maximize their interests. We further formulate the task offloading process as a distributed multiobjective minimization problem. Then, we propose an energy-saving dispersed matching algorithm for balancing the interests of multiple objectives. In detail, the main contributions of this article can be summarized as follows.

- 1) We first present a dispersed computing offloading framework involving multiuser, multitask, and multi-NCP,

which takes into account the interests of users and NCPs, as well as the allocation of computation and communication resources.

- 2) An energy-saving bilateral matching-based dispersed offloading strategy is proposed, which can effectively realize the dispersed task offloading and reduce the interaction between users and NCPs. On this basis, the stability, local optimality, and Price of Anarchy (PoA) of the proposed algorithms are analyzed and proved.
- 3) Extensive simulations and scene-based experiments are developed to evaluate the performance of our algorithm and highlight the advantages of dispersed computing compared with edge computing. The results show that the user fairness of the proposed algorithm is improved by 14.2% and 20.1% on average compared to the full search (FS) and random (RAN) algorithms. Moreover, our proposed framework outperforms the edge computing scheme on average by 54.9% for user energy cost (UEC), especially in a harsh network environment.

The remainder of this article is organized as follows. In Section II, we briefly introduce related works. Then, we present our system model and problem formulation in Section III. We propose an energy-saving bilateral matching-based dispersed offloading strategy in Section IV and theoretical analysis in Section V. Extensive simulations and scene comparison experiments are conducted in Sections VI and VII, respectively. Finally, we conclude this article in Section VIII.

II. RELATED WORKS

A. Tasks Offloading for Edge Computing

There are many offloading methods for edge computing scenarios. Among them, more efforts have been dedicated to utilizing edge collaboration [6], [7], artificial intelligence [10]–[12], and additional assistance (phones, drones, etc.) [13]–[15]. For example, a collaborative task offloading mechanism (CTOM) for mobile cloudlet networks was designed in [6] to achieve more efficient and low-cost load balancing. Anajemba *et al.* [7] introduced a distributed multiaccess MEC collaborative offloading technology based on the suboptimal Lagrangian method, whereas these edge collaborations consider situations with sufficient network status information and cannot handle the changing network environment well. In order to cope with the dynamic and real-time changes of the network environment, Chen *et al.* [10] proposed an offloading algorithm based on the double-deep Q network (DQN). The optimal strategy can be learned without knowing the prior knowledge of network dynamics. In [11], a deep reinforcement learning (DRL) algorithm was proposed to learn the optimal computing offloading and packet scheduling strategy. Feng *et al.* [12] developed a collaborative computing offload and resource allocation framework for blockchain-supported MEC systems. Moreover, an asynchronous advantage actor-critic (A3C) reinforcement learning algorithm is developed for the dynamic characteristics. However, these works are based on fixed infrastructures and lack scalability. As a result, it is unable to respond flexibly to the sudden surge of traffic or damage due to disasters and other reasons.

TABLE I
SYSTEM MODEL PARAMETER DEFINITIONS AND NOTATIONS

Notation	Definition
\mathcal{M}	Number of users
\mathcal{K}	Number of tasks per user
\mathcal{N}	Number of NCPs
X_{ikj}	Indicates if user i 's k -th task is offloaded to NCP j
τ_{ik}	The k -th task of user i
D_{ik}	Amount of data for user i 's k -th task
C_{ik}	Required CPU cycles for user i 's k -th task
T_{ik}^{max}	Maximum latency for user i 's k -th task
r_{ikj}	Transmission rate of user i offloading k -th task to NCP j
g_{ikj}	Channel gain of user i offloading k -th task to NCP j
s_{ikj}	Computation resources allocated by NCP j to i 's k -th task
s_j	Processing speed of NCP j
B_j	Total bandwidth of NCP j
q_j	The number of tasks that NCP j has accepted
q_j^{max}	Maximum number of tasks that NCP j can accept
p_{ikj}	Transmission power of user i offloading k -th task to NCP j

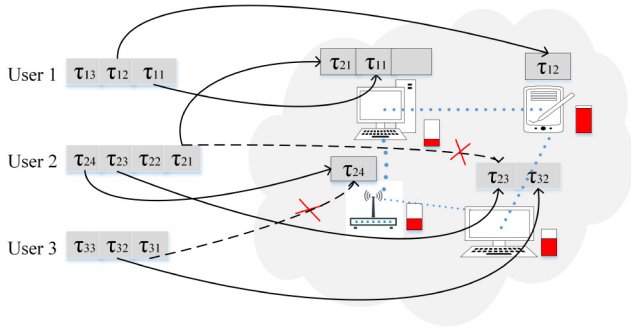


Fig. 2. Example of a dispersed network that consists of $M = 3$ users, $K = 3$ or 4 tasks, and $N = 4$ NCPs.

To increase the flexibility of the offloading framework, the work assisted by mobile devices has been focused on. Feng *et al.* [13] designed a Device-to-Device (D2D) communication-assisted traffic offloading scheme, which exploits D2D communications to assist traffic offloading from cellular to WiFi in integrated cellular-WiFi networks. Shang *et al.* [14] increased their overall profits by encouraging users to act as D2D transmitters, broadcasting their popular content to nearby users. In [15], the drone plays the role of a cloud in the sky, assisting the MEC to collect and process the computation tasks offloaded by the ground users. Nevertheless, these introduced auxiliary devices require additional costs. Furthermore, the above studies ignore the potential idle resources in the network and the utilization value of nearby computation points.

B. Tasks Offloading for Dispersed Computing

Different from the edge-oriented researches, some people began to pay attention to task offloading in dispersed computing [19]–[22], considering the ubiquitous idle devices with limited resources in the network. For example, Knezevic *et al.* [20] developed a runtime scheduling software tool for dispersed computing. It can deploy pipeline computing on multiple geographically dispersed computation points in the form of a directed acyclic graph. Ghosh *et al.* [21] designed a container orchestration architecture for dispersed computing. The system automatically and efficiently distributes tasks

among a group of networked computation nodes. Hu and Krishnamachari [22] proposed a throughput-optimized task scheduler for computer vision and video processing applications. To capture the heterogeneity of computation and communication in the network, Yang *et al.* [19] investigated a max-weight type scheduling policy, which is throughput-optimal for the proposed virtual queuing network. However, energy cost is not well studied in dispersed computing. Note that the energy cost is a critical metric for mobile devices due to the limited energy of devices and NCPs, particularly in dispersed computing scenarios that motivate our work in this article.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. Network Model

In this article, we consider a multiuser, multitask, and multi-NCP dispersed network shown in Fig. 2, which consists of M users and N NCPs. The sets of users and NCPs are denoted by $\mathcal{M} = \{1, 2, \dots, M\}$ and $\mathcal{N} = \{1, 2, \dots, N\}$, respectively. Without loss of generality, we assume that each user has K independent computation-intensive tasks denoted by $\mathcal{K} = \{1, 2, \dots, K\}$. In the dispersed network, each task can be processed locally or offloaded to an appropriate NCP with idle resources. The location of each node is assumed to be known [24] (e.g., using GPS or some positioning methods). The system model parameters and descriptions are provided in Table I.

Let $\tau_{ik} = \{D_{ik}, C_{ik}, T_{ik}^{max}\}$ be the k th computation task of user i , then it can be described as follows.

- 1) The input D_{ik} is defined as the amount of data for user i 's k th task, including system settings and input parameters.
- 2) C_{ik} is the number of CPU cycles required to complete the task.
- 3) T_{ik}^{max} represents the maximum acceptable latency for the user i 's k th task.

We assume that the computation task is atomic and cannot be divided into subtasks. To facilitate the analysis, we make the common assumption that the set of devices changes slowly [25].

B. Energy Cost Model

In what follows, we introduce the energy cost model under the constraints of computation and communication resources in the dispersed computing system.

1) *Computation Energy Cost:* We assume that the CPU processor of each device has speed scaling capability, characterized by a speed–power curve, which is defined as a function of its processing speed s_j . Here, we consider different devices have distinct processing speeds. The power consumption of a processor increases in proportion to the speed. Furthermore, the speed–power curve is [26] usually modeled by the polynomial function $P(s_j) = \mu_j s_j^\alpha$, where μ_j and α are the device-related parameter.

Given $P(s_j)$, the computation energy cost required for NCP j to perform user i 's k th task is expressed as

$$e_{ikj}^{com} = P(s_{ikj}) \frac{C_{ik}}{s_{ikj}} \quad (1)$$

where s_{ikj} is the computation resources allocated by NCP j to the user i 's k th task, defined as $s_{ikj} = s_j/q_j$. $q_j = \sum_{i=1}^M \sum_{k=1}^K X_{ikj}$ is the number of offloading tasks to NCP j . It should be noted that e_{ikj}^{com} is not only determined by C_{ik} , s_{ikj} , and s_j but also affected by the maximum number of offloading tasks q_j^{max} that the NCP j can accept, that is, $q_j \leq q_j^{\text{max}}$.

2) *Communication Energy Cost*: We adopt the orthogonal frequency-division multiple access (OFDMA) technology as the communication mode for data transmission to avoid mutual interference between communication links [27]. The achievable uplink transmission rate between user i and NCP j can be defined as

$$r_{ikj}(X_{ikj}, X_{-ikj}) = \frac{B_j}{q_j} \cdot \log_2 \left(1 + \frac{p_{ikj} g_{ikj}}{\sigma^2} \right) \quad (2)$$

where B_j is the total bandwidth assigned by NCP j . p_{ikj} is the transmission power from user i offloading task k to NCP j , and σ^2 is the noise power. g_{ikj} is the channel gain between user i and NCP j , which is related to the distance between the devices.

So the transmission energy cost of user i offloading task k to NCP j can be denoted as

$$e'_{ikj} = p_{ikj} \frac{D_{ik}}{r_{ikj}(X_{ikj}, X_{-ikj})}. \quad (3)$$

Meanwhile, the energy cost required for NCP j to receive the user i 's k th task can be obtained as

$$e^r_{ikj} = p_{ikj} g_{ikj} \frac{D_{ik}}{r_{ikj}(X_{ikj}, X_{-ikj})}. \quad (4)$$

3) *Total Energy Cost*: We define the total energy cost from the perspective of users and NCPs, respectively. Each user needs to decide whether to handle tasks locally or offload tasks to the idle NCPs for execution. A binary variable $X_{ikj} \in \{0, 1\}$ is denoted as the offloading decision, where $i \in M$, $k \in K$, and $j \in N$. Specifically, $X_{ikj} = 1$ means that user i decides to offload its k th task to NCP j . Overall, each task can only be handled by one NCP or local device ($j = 0$ means local computing).

a) *Energy cost of users*: The energy cost of user i consists of the computation energy for task execution, or the communication energy required to transmit the offloading tasks. Thus, the energy cost e_{ikj}^{task} of task k can be expressed as

$$e_{ikj}^{\text{task}}(X_{ikj}, X_{-ikj}) = \begin{cases} X_{ik0} e_{ik0}^{\text{com}}, & j = 0 \\ X_{ikj} e'_{ikj}, & j \in N. \end{cases} \quad (5)$$

Using the above notation, the energy cost of user E_i^{user} can be calculated by

$$\begin{aligned} E_i^{\text{user}} &= \sum_{k=1}^K \sum_{j=0}^N e_{ikj}^{\text{task}}(X_{ikj}, X_{-ikj}) \\ &= \sum_{k=1}^K X_{ik0} e_{ik0}^{\text{com}} + \sum_{k=1}^K \sum_{j=1}^N X_{ikj} e'_{ikj}. \end{aligned} \quad (6)$$

We ignore the energy consumption of other hardware components (e.g., RAM), which are generally much less significant [28].

b) *Energy cost of NCPs*: The energy cost of NCP j includes the energy needed to perform offloaded tasks and the energy needed to receive the tasks. Thus, the energy cost e_{ikj}^{nep} of task k can be expressed as

$$e_{ikj}^{\text{nep}}(X_{ikj}, X_{-ikj}) = \begin{cases} 0, & j = 0 \\ X_{ikj} (e_{ikj}^{\text{com}} + e^r_{ikj}), & j \in N. \end{cases} \quad (7)$$

Therefore, the energy cost E_j^{nep} of NCP j can be calculated by

$$\begin{aligned} E_j^{\text{nep}} &= \sum_{i=1}^M \sum_{k=1}^K e_{ikj}^{\text{nep}}(X_{ikj}, X_{-ikj}) \\ &= \sum_{i=1}^M \sum_{k=1}^K X_{ikj} (e_{ikj}^{\text{com}} + e^r_{ikj}). \end{aligned} \quad (8)$$

Assume that the energy and time used to transmit the communication results from the NCP j to the user i can be neglected [29]. Finally, we define the total energy cost of users and NCPs as E^{user} and E^{nep} , respectively.

C. Problem Definition

Based on the above formulation, we then formally define the offloading problem between multiuser and multi-NCP as the optimization of both users' and NCPs' energy costs during the involvement. On the one hand, the users, which have tasks to execute and offload, will compete with each other for obtaining NCPs' computation resources. During the competition, each of them makes the decision of whether to offload the local tasks so that all the tasks can be performed under the completion time constraints. Hence, the optimization problem for the user side is formulated as

$$\begin{aligned} \text{User : } \min_{X_{ikj} \in \{0,1\}} & E_i^{\text{user}} \\ \text{s.t. } & C1 : X_{ik0} \frac{C_{ik}}{s_0} + (1 - X_{ik0}) X_{ikj} \left(\frac{D_{ik}}{r_{ikj}} + \frac{C_{ik}}{s_{ikj}} \right) \\ & \leq T_{ik}^{\text{max}} \\ & C2 : X_{ikj} \in \{0, 1\} \quad \forall j \in N \cup \{0\} \\ & C3 : \sum_{j=0}^N X_{ikj} = 1 \\ & C4 : i \in \mathcal{M}, k \in \mathcal{K}, j \in \mathcal{N} \end{aligned} \quad (9)$$

where C1 ensures that task τ_{ik} is completed within the latency constraint. Constraint C2 indicates whether the user i 's k th task is offloaded to the j th NCP, which is also affected by the following optimization problem (10). Constraint C3 ensures that each task can either be processed by one NCP or finally conducted by the user device locally, indicating that the task is atomic and cannot be divided.

On the other hand, from the perspective of NCPs, energy-saving tasks within the constraints of their computation and communication resources are favored. To this end, each NCP attempts to find the optimal acceptable offloading tasks as

$$\begin{aligned} \text{NCP : } \min_{X_{ikj} \in \{0,1\}} & E_j^{\text{nep}} \\ \text{s.t. } & C1 : E_j^{\text{total}} - E_j^{\text{nep}} \geq \delta_j \end{aligned}$$

$$\begin{aligned}
C2 : & \sum_{i=1}^M \sum_{k=1}^K X_{ikj} s_{ikj} \leq s_j \\
C3 : & \sum_{i=1}^M \sum_{k=1}^K X_{ikj} B_{ikj} \leq B_j \\
C4 : & \sum_{i=1}^M \sum_{k=1}^K X_{ikj} = \min \left[\sum_{i=1}^M \sum_{k=1}^K X_{ikj}, q_j^{\max} \right] \\
C5 : & \sum_{i=1}^M \sum_{k=1}^K X_{ikj} \leq q_j^{\max} \\
C6 : & i \in \mathcal{M}, k \in \mathcal{K}, j \in \mathcal{N}
\end{aligned} \tag{10}$$

where C1 means the minimum energy limit for each NCP, which is individualized; C2 and C3 represent the computation and communication resource constraints of each NCP; and C4 and C5 indicate the lower and upper limits on the number of tasks that the j th NCP needs to receive, which are related to the offloading decision given by the users. One might argue why the NCPs are willing to contribute their idle resources. The reason behind the design is twofold.

- 1) It is the underlying idea of dispersed computing that one with idle computation resources shall share the available capacity with the peer devices. As a reward, an involved party can get help from the community to release the local computation burden.
- 2) The computation capacity s_j for the j th NCP is in fact its available resources out of its whole resource pool, thus can be shared without impacting the normal execution of its own tasks.

Considering the optimization process on both sides, we state that the offloading problem in fact is transformed into a distributed multiobjective optimization problem. Besides, since the two optimization problems have a common solution, it is also a two-way selection problem. We hope that with this solution, each user will be matched to an NCP, while each NCP can get the appropriate offloading tasks.

IV. ENERGY-SAVING BILATERAL MATCHING-BASED DISPERSED OFFLOADING STRATEGY

The multiobjective optimization problem is proved to be NP-hard [30]. Its solution can be found by searching for all possible offloading decisions. However, the computational complexity of the FS is exponential. To efficiently solve the problem with lower complexity, we propose a matching game-based task offloading strategy due to the two disjoint sets of users and NCPs.

A. Matching Concepts

To model the distributed multiobjective optimization problem as a one-to-many matching game with resource and latency constraints, we consider the task set $\Gamma = \{1, 2, \dots, M\} \times \{1, 2, \dots, K\}$ and NCP set $\mathcal{N} = \{1, 2, \dots, N\}$ as two teams of players. Specifically, in the matching game, each NCP plays the vendor role and the tasks act as the buyers. It is notable that matching is bilateral, in the sense that a

Algorithm 1: Preference List Generation for Each Task With Respect to $\{T_\tau^{\max}\}_{\tau \in \Gamma}$

Input: Task set $\Gamma = \{\tau_1, \tau_2, \dots, \tau_\Gamma\}$, NCP set $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$, the knowledge of NCPs: $d_{\tau n}, s_n, q_n$.

Output: The set of task decisions and preference list of tasks: $X_\tau, L_\tau, \forall \tau \in \Gamma$.

- 1 **Initialization:** $L_\tau = 0, X = 0$;
- 2 **for** $\tau = 1$ **to** $|\Gamma|$ **do**
- 3 **for** $j = 1$ **to** $|\mathcal{N}|$ **do**
- 4 **if** $\frac{D_\tau}{r_{\tau j}} + \frac{C_\tau}{s_{\tau j}} < T_\tau^{\max}$ **then**
- 5 Generate L_τ using (11), and sort in descending order;
- 6 Find the top ranked j' in L_τ and setting $X_{\tau j'} = 1$;
- 7 **return** L_τ, X_τ

task is admitted at a given NCP only if the NCP admits that task. Formally, we define the matching game as follows.

Definition 1: Given two disjoint finite sets of players, $\Gamma = \{\tau_i\}_{i=1}^{|\Gamma|}$ and $\mathcal{N} = \{n_j\}_{j=1}^{|\mathcal{N}|}$, let two disjoint finite sets of $S^\Gamma = \{1, \dots, |\Gamma|\}$ and $S^\mathcal{N} = \{1, \dots, |\mathcal{N}|\}$, then a one-to-many matching function Ψ [31] : $\{\Gamma\} \cup \{\mathcal{N}\} \rightarrow \{\Gamma\} \cup \{\mathcal{N}\}$ is defined such that for all $i \in M \times K$ and $j \in \mathcal{N}$.

- 1) $\Psi(\tau_i) \in \{n_{j'} \in S^\mathcal{N} \mid \Psi(\tau_i) \in \{0, 1\}\}$.
- 2) $\Psi(n_j) \subseteq \{\tau_{i'} \in S^\Gamma \mid \Psi(n_j) \leq q_n^{\max}\}$.
- 3) $\Psi(\tau_i) = n_j \Leftrightarrow \Psi(n_j) = \tau_i$.

Thus, the matching game Ψ can be defined by a tuple $(\Gamma, \mathcal{N}, q_n^{\max}, \succ_\tau, \succ_n)$, where q_n^{\max} is the NCP n 's quota vector. Here, \succ_τ and \succ_n represent the set of the preference relations of tasks and NCPs, respectively. Condition 1) implies that each task can only be offloaded to one NCP at most. Condition 2) represents the maximum number of offloading tasks that each NCP can accept, corresponding to C5. Condition 3) indicates that if τ_i matches n_j , then n_j is also matched to task τ_i . The output of this game is a set of matching pairs $\langle \tau_i, n_j \rangle_{i \in \Gamma, j \in \mathcal{N}}$ between tasks and NCPs.

B. Preference Profiles of Players

For each player, the preference profile is used to rank the other players. In the proposed game Ψ , tasks and NCPs could use available information to build their preference profiles, respectively. For simplicity, we denote τ as any task in the task set Γ .

Definition 2: The preference $P_\tau(n)$ of task τ for different NCPs can be defined as

$$P_\tau(n) = \frac{B_n}{q_n} \log_2 \left(1 + \frac{P_{\tau n} g_{\tau n}}{\sigma^2} \right). \tag{11}$$

This preference function is established to achieve the energy saving of users, where each task prefers to associate with NCP at the maximum transmission rate. According to the preference function, the task prefers NCPs with larger bandwidth, more quota vectors, and shorter distance.

Algorithm 2: Matching Evaluation for Each NCP With Respect to $\{\delta_j\}_{j \in \mathcal{N}}$

Input: $L_{\text{unmatched}}$, L_τ and X_τ , $\forall \tau \in \Gamma$.
Output: $L_{\text{unmatched}}$, L_{matched} and L_τ .

```

1 for  $j = 1$  to  $|\mathcal{N}|$  do
2   for  $\tau = 1$  to  $|L_{\text{unmatched}}|$  do
3     if  $X_{\tau j} = 1$  and  $E_j^{\text{total}} - E_j^{\text{nep}} \geq \delta_j$  then
4       Generate  $L_j$  using (12), and sort in
         descending order;
5     if  $q_j < q_j^{\text{max}}$  then
6       Accept the top ranked  $\tau'$  in  $L_j$ ;
7        $L_j^{\text{accept}} = L_j^{\text{accept}} \cup \tau', j >$ ;
8        $q_j = q_j + 1$ ;
9        $L_{\text{unmatched}} = \{L_{\text{unmatched}} \setminus \tau'\}$ ;
10    else
11       $L_j = L_j \cup L_j^{\text{accept}}$ ;
12      Select the top  $q_j^{\text{max}}$  tasks:  $L_j^{\text{accept}} = L_j[:q_j^{\text{max}}]$ ;
13       $L_{\text{unmatched}} = \{L_{\text{unmatched}} \setminus L_j[:q_j^{\text{max}}]\}$ ;
14       $L_{\text{unmatched}} = L_{\text{unmatched}} \cup \{L_j \setminus L_j[:q_j^{\text{max}}]\}$ ;
15    for  $\tau$  to  $|L_{\text{unmatched}}|$  do
16       $L_\tau = \{L_\tau \setminus j\}$ 
17       $L_{\text{matched}} \leftarrow L_j^{\text{accept}}$ 
18 return  $L_{\text{unmatched}}$ ,  $L_{\text{matched}}$  and  $L_\tau$ 

```

Definition 3: The preference $P_n(\tau)$ of NCP n for different tasks can be defined as

$$P_n(\tau) = 1/C_\tau. \quad (12)$$

For the NCP, a small amount of computation can save its computation overhead and energy consumption, so that it has the opportunity to provide computing services to more users. Thus, NCPs prefer tasks that require fewer CPU cycles.

C. Task-NCP Bilateral Matching Algorithm

The Task-NCP bilateral matching algorithm (TN-matching) consists of two major subalgorithms, as shown in Algorithms 1 and 2, respectively. In our design, Algorithm 1 solves the task assignment problem of users. A user finds the best offloading decision to minimize energy consumption based on the task preference value. Algorithm 2 solves the problem of NCPs' offloading task selection. Based on the offloading requests given by the users, the NCP chooses to accept the offloading tasks which make it energy saving. Algorithm 3 combines Algorithms 1 and 2 to solve the matching problem between tasks and NCPs and achieves the final stable matching between them iteratively.

For Algorithm 1, it has two phases, namely, the network information phase and the task preference list generation phase.

1) *Network Information:* Each NCP periodically broadcasts its idle resource information; Each user discovers idle resources around and collects required network parameters.

Algorithm 3: Task-NCP Bilateral Matching Algorithm

Input: The prior knowledge: Γ , \mathcal{N} , $d_{\tau j}$, s_j .
Output: The set of matching results for stable matching Ψ^* : L_{matched} .

```

1 Initialization:  $L_{\text{matched}} = 0$ ,  $L_{\text{unmatched}} = \Gamma$ ;
2 repeat
3   Run Algorithm 1;
4   if  $L_{\text{unmatched}} \neq \text{null}$  then
5     if  $L_\tau \neq \text{null}$  then
6       Run Algorithm 2;
7       Update  $L_\tau$  according to the latest information
         from NCPs;
8     else
9       Choose local processing:  $j = 0$ ;
10    Update  $L_{\text{matched}}$ ;
11  else
12    The algorithm is terminated;
13 until convergence to a stable matching;
14 return  $L_{\text{matched}} : < \tau_i, n_j >_{i \in \Gamma, j \in \mathcal{N}}$ 

```

The network information includes the available NCPs' computation capacities, distances, and the number of tasks that can be received at the moment. This step provides important information for subsequent evaluation.

2) *Task Preference List Generation:* First, each task selects the NCPs within the delay constraint based on the parameters obtained in phase 1. Then, based on the task preference function of (11), calculate the preference values of the selected NCPs and sort them in descending order (lines 2–6). It is worth noting that the preference function here is the key to Algorithm 1, which directly affects the final task assignment. Finally, we select the top-ranked NCP in the task preference value list L_τ as the current decision. Meanwhile, make a record locally and send it to the corresponding NCP for feedback. Based on the above content, the users complete the best task assignment decision for the purpose of minimizing energy consumption.

For Algorithm 2, it has two phases, namely, the NCP preference list generation phase and the matching evaluation phase.

3) *NCP Preference List Generation:* After Algorithm 1 is executed, NCPs will receive offloading requests from the users. The NCP receiving the requests first eliminates tasks that will consume more than its energy constraints (line 3). Then, each NCP calculates the preference values for the remaining tasks according to (12) and ranks them in descending order (line 4). Note that the preference function calculation (line 4) here is an important step of Algorithm 2, which aims to help NCPs prioritize tasks with low computation and energy saving.

4) *Matching Evaluation:* Each NCP accepts the top q_j^{max} tasks in preference list L_j that meet the energy constraints, and the others are rejected. If quota q_j is not full, the NCP selects the offloading task in the ranking order. Meanwhile, the matching relationship is established, and the accepted tasks are removed from $L_{\text{unmatched}}$ (lines 5–9); otherwise, the

new task request is added to the preference list L_j , and L_j is reordered (lines 10–14). Then, update the preference list L_τ of rejected tasks by deleting the current top-ranked NCP j (lines 15 and 16). The above process is the key to NCP's independent evaluation and decision making.

Finally, the TN-matching process is concluded in Algorithm 3, where the matching result is iteratively updated until it converges to a stable solution. Rematch the tasks in the unmatched set $L_{\text{unmatched}}$ and find all lower ranked j' than j until j can be admitted. If there is no match with any j until the preference list of task τ is null, the task selects local processing, that is, $j = 0$. We should know that each user and NCP run Algorithms 1 and 2, respectively, and certain communication and interaction are required between them.

V. THEORETICAL ANALYSIS

In this section, we give a series of theoretical analyses on the stability, local optimality, PoA, and complexity of the Task-NCP bilateral matching algorithm, which are defined to evaluate the principle and performance of the algorithm.

A. Stability Analysis

The goal of the algorithm is to find a stable offloading decision, where stability is the key concept in matching theory [32], which is defined as follows.

Definition 4 (Blocking Pair): The pair (τ', n') is a blocking pair [33] for the matching Ψ , only if $\tau' \succ_{n'} \tau$, $\tau \in \Psi(n')$ and $n' \succ_{\tau'} n$, $n \in \Psi(\tau')$, for $\tau' \notin \Psi(n')$ and $n' \notin \Psi(\tau')$. Particularly, the definition of the blocking pair can be presented mathematically as

$$(\forall \tau_i \in \Gamma, n_j \in N)((\tau_i, \Psi(\tau_i)), (n_j, \Psi(n_j))) \Rightarrow (\exists \tau' \in \Gamma, n' \in N)((\tau', \Psi(\tau_i)), (n', \Psi(n_j))). \quad (13)$$

In other words, there exists a partnership (τ', n') such that τ' and n' are not matched with each other under the current matching Ψ but prefer to be matched with each other.

Definition 5 (Stable Matching): A matching Ψ is said to be stable if it admits no blocking pair [34].

Theorem 1: The matching Ψ^* produced by TN-matching is stable.

Proof: Assume that tasks τ_1 and τ_2 match n_1 and n_2 , respectively. But $n_2 \succ_{\tau_1} n_1$, so τ_1 must have sent a request to n_2 at a certain stage. Unfortunately, n_2 's own preference list rejected τ_1 . So, we can clearly see that $\tau_2 \succ_{n_2} \tau_1$. There is no possibility that τ_1 prefers n_2 , and n_2 also prefers τ_1 . According to Definition 4, there are no blocking pairs. Furthermore, by Definition 5, the matching Ψ^* produced by TN-matching is proved to be stable. ■

The stability analysis of the algorithms in this article is on the theoretical level, but it will be affected by some factors in the real environment. There are two main influencing factors, one is the network communication status, and the other is the total number of tasks that all NCPs can receive. In terms of network status, real-time communication of basic information should be guaranteed. When the network status is poor, the information transmission speed is slow, but fortunately, the

amount of information required for our proposed scheme is small. Moreover, the distance between the devices is short compared to the cloud or edge server, thus the impact is not significant but there will be a slight delay in decision making, whereas if all devices are broken or the network is completely disconnected, the stability cannot be guaranteed. There are two situations in terms of the total number of tasks: 1) when the number of tasks NCP can receive is greater than or equal to the total number of tasks, all tasks can find the appropriate NCP, and the matching is stable and 2) when the number of tasks that the NCP can receive is less than the total number of tasks, there are tasks that cannot match the NCP. However, it is assumed in our algorithm that all tasks do not match any NCP, and the local computing is allowed as well, which ensures the final stable matching and convergence. In summary, we find that the proposed algorithms still have good stability in reality.

B. Local Optimality Analysis

Theorem 2: The outcome of TN-matching is a local optimal solution to the multiobjective optimization problem.

Proof: Suppose that n_1 selects q tasks $\tau_1, \tau_2, \dots, \tau_q$ from all the requested tasks, and rejects other tasks including task τ' . It can be seen that tasks $\tau_1, \tau_2, \dots, \tau_q$ (including tasks that have been rejected by other NCPs) prefer n_1 to other NCP n . To make a hypothetical arrangement here, we match task τ' to n_1 , then at least one τ_i has to choose an n' with a preference less than n_1 . However, if you follow this arrangement, the utility of both players will be reduced, represented by the following mathematical form:

$$P_{\tau_i}(n') < P_{\tau_i}(n_1) \text{ and } P_{n_1}(\tau') < P_{n_1}(\tau_i) \quad (14)$$

which will cause the algorithm to be unstable.

According to Definitions 4 and 5 and Theorem 1, there are no blocking pairs in the final matching, and the algorithm is stable. In addition, only the NCP side performs “reject” during the iterative process of TN-matching. As a result, the TN-matching converges to a local optimal solution to the problem. ■

C. Price of Anarchy

We have shown that the Ψ^* is a stable matching and now address the important question how far the total UEC would be from the optimal in a stable matching. To quantify the difference from the optimal performance, we use the PoA, defined as the ratio of the worst case UEC to the minimal UEC

$$\text{PoA} = \frac{\max \sum_{i \in M} E_i^{\text{user}}(\Psi^*)}{\min \sum_{i \in M} E_i^{\text{user}}(\Psi)} \quad (15)$$

where Ψ^* means a stable matching. In what follows, we give an upper bound to the PoA. Note that here we only consider the PoA for the one-sided (user) market.

Theorem 3: The PoA for the matching game Ψ^* is upper bounded by

$$\frac{\sum_{\tau \in M \times K} e_{\tau 0}^{\text{com}}}{\sum_{\tau \in M \times K} \min\{e_{\tau 0}^{\text{com}}, e_{\tau 1}, e_{\tau 2}, \dots, e_{\tau N}\}}. \quad (16)$$

Proof: First, we show that if there is a stability in which all tasks are executed locally then it is the worst case for stable matching. Let Ψ^* be an arbitrary stable matching, then X^* is the offloading decision of Ψ^* . Observe that $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) \leq e_{\tau 0}^{\text{com}}$ holds for every task $\tau \in M \times K$. Otherwise, if $\exists \tau \in M \times K$ enables $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) > e_{\tau 0}^{\text{com}}$, task τ would have an incentive to deviate from decision $X_{\tau j}^*$, which contradicts our initial assumption that $X_{\tau j}^*$ is the offloading decision of a stable matching Ψ^* . Thus, for any stable matching Ψ^* , $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) \leq e_{\tau 0}^{\text{com}}$ holds.

Next, let us consider an arbitrary offloading decision $(X_{\tau j}, X_{-\tau j}) \in X$. If $X_{\tau} = 0$ ($X_{\tau} = \sum_{j=1}^N X_{\tau j}$), then $e_{\tau j}^{\text{task}}(X_{\tau j}, X_{-\tau j}) = e_{\tau 0}^{\text{com}}$. Otherwise, if $X_{\tau j} = 1$ for a $j \in N$, we have the best case $X_{\tau'} = 0$ for every $\tau' \in \{M \times K \setminus \tau\}$, and thus $\sum_{j=1}^N q_j = 1$.

Therefore, $r_{\tau j}(X_{\tau j}, X_{-\tau j}) \leq B_j \log_2(1 + \frac{p_{\tau j} g_{\tau j}}{\sigma^2})$, which implies

$$p_{\tau j} \frac{D_{\tau}}{r_{\tau j}(X_{\tau j}, X_{-\tau j})} \geq p_{\tau j} \frac{D_{\tau}}{B_j \log_2(1 + \frac{p_{\tau j} g_{\tau j}}{\sigma^2})} = \bar{e}_{\tau j}. \quad (17)$$

Hence, $e_{\tau j}^t(X_{\tau j}, X_{-\tau j}) \geq \min\{e_{\tau 0}^{\text{com}}, \bar{e}_{\tau 1}, \bar{e}_{\tau 2}, \dots, \bar{e}_{\tau N}\}$ and

$$\sum_{\tau \in M \times K} e_{\tau j}^t(X_{\tau j}, X_{-\tau j}) \geq \sum_{\tau \in M \times K} \min\{e_{\tau 0}^{\text{com}}, \bar{e}_{\tau 1}, \bar{e}_{\tau 2}, \dots, \bar{e}_{\tau N}\}.$$

Using these expressions, we can establish the following bound as:

$$\begin{aligned} \text{PoA} &= \frac{\max \sum_{i \in M} E_i^{\text{user}}(\Psi^*)}{\min \sum_{i \in M} E_i^{\text{user}}(\Psi)} \\ &\leq \frac{\sum_{\tau \in M \times K} e_{\tau 0}^{\text{com}}}{\sum_{\tau \in M \times K} \min\{e_{\tau 0}^{\text{com}}, \bar{e}_{\tau 1}, \bar{e}_{\tau 2}, \dots, \bar{e}_{\tau N}\}} \end{aligned} \quad (18)$$

which proves the theorem. \blacksquare

We provide an upper bound for the PoA in case of the users, that is, when all users have the same parameters.

D. Complexity Analysis

Theorem 4: The TN-matching algorithm converges within a limited number of iterations, and the computational complexity is $O(\max(|L_{\tau}|_{\tau \in \Gamma}) \cdot \max(|L_j|_{j \in N})) + O(N \cdot \max(|L_{\tau}|_{\tau \in \Gamma}))$.

Proof: The complexity required to generate preference lists for users and NCPs is $O(\max(|L_{\tau}|_{\tau \in \Gamma}) \cdot \max(|L_j|_{j \in N}))$. And the running time to sort the preference list of users and NCP using the standard sorting algorithm is $O(\max(|L_{\tau}|_{\tau \in \Gamma}) \cdot \log^{\max(|L_{\tau}|_{\tau \in \Gamma})})$ and $O(\max(|L_j|_{j \in N}) \cdot \log^{\max(|L_j|_{j \in N})})$, respectively. Note that in a matching game $\forall \tau \in \Gamma, |L_{\tau}| \leq |N|$ and $\forall j \in N, |L_j| \leq |\Gamma|$.

To analyze the complexity, we first consider the best case, where each task is accepted by its preferred NCP, and the complexity is $O(1)$. In the worst case, each task is rejected N times, which means that $N + 1$ rounds of matching are needed to get the final result (not received by NCP, processed locally). In each round, once an NCP has to decide whether to replace the existing task, they need to calculate the satisfaction and the complexity of this process is $O(\max(|L_j| + 1)_{j \in N})$. At the end of each round, the preference list of each task needs to be updated with a complexity of $O(\max(|L_{\tau}| - 1)_{\tau \in \Gamma})$.

Therefore, the time bottleneck of the algorithm in the best case is the generation of the preference list, and the computational complexity is $O(\max(|L_{\tau}|_{\tau \in \Gamma}) \cdot \max(|L_j|_{j \in N}))$. In the worst case, the computational complexity is $O(\max(|L_{\tau}|_{\tau \in \Gamma}) \cdot \max(|L_j|_{j \in N})) + O(N \cdot \max(|L_{\tau}|_{\tau \in \Gamma}))$.

Based on the above analysis, we find that the complexity of the algorithm is polynomial, where $|\Gamma| = |M \times K|$ is the number of tasks and $|N|$ is the number of NCPs. Thus, the algorithm is able to achieve the converged state within several iterations since the sizes of two preference sets Γ and N are finite. \blacksquare

VI. NUMERICAL EVALUATIONS

In this section, we show simulation results under different network settings to evaluate the performance and computational efficiency of our proposed algorithm and reveal insights about it.

A. Evaluation Setup

We consider that users and NCPs are randomly distributed in a 1 km \times 1 km 2-D plane. We assume that the channel gain g_{ikj} is proportional to $d_{ikj}^{-\gamma}$, where d_{ikj} is the distance from user i to NCP j , and $\gamma = 4$ [35], [36] is the path-loss index. The channel bandwidth B_j of the j th NCP is uniformly distributed within [3, 10] MHz, while the transmission power p_{ikj} is set to 0.5 W. The computation capacity s_0 of users is uniformly set to 0.5 GHz [37], while the NCPs are drawn from a continuous normal distribution on [0.4, 2] GHz. The input data size D_{ik} and the required CPU cycles C_{ik} are uniformly distributed across [3, 10] Mb and [0.1, 1] Gcycles, respectively. The residual energy threshold δ_j of the j th NCP is randomly distributed in [5, 10] J. The device-related parameter α has been usually assumed to be around 3 [26]. The default number of NCPs is 4 with a quota of [3, 1, 2, 4], and the number of tasks per user is 2.

B. Baseline Setup

For effective and fair comparisons, we introduce the FS algorithm and random algorithm (RAN) for comparison. We devise FS using exhaustive traversal, where each task is matched with each NCP once to find the offloading decision with the optimal total energy cost. For RAN, the user selects NCP in a random manner and the NCP randomly accepts offloaded tasks. Based on the series of settings above, we show a comprehensive comparison and analysis for the algorithms in terms of total energy cost, UEC ratio, NCP energy cost (NEC) ratio, Jain's fairness index of users [38], and iteration times.

C. Evaluation Results and Analysis

In general, the proposed TN-matching outperforms RAN substantially and is close to the FS in terms of total energy cost, which validates the theoretical results. For fairness of users, the TN-matching improves on average 14.2% and 20.1% compared to the FS and the RAN. Moreover, which effectively promotes the balance of energy cost between the users and the NCPs, rather than just focusing on the total energy cost.

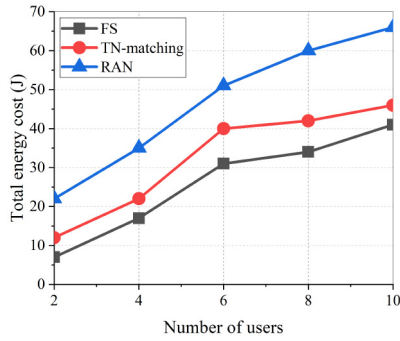


Fig. 3. Total energy cost.

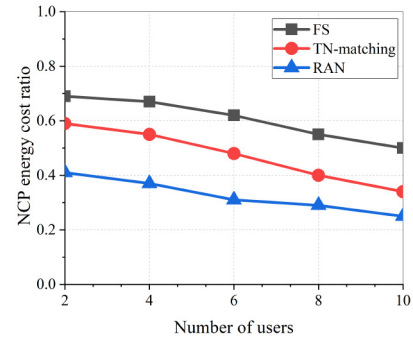


Fig. 5. NEC ratio.

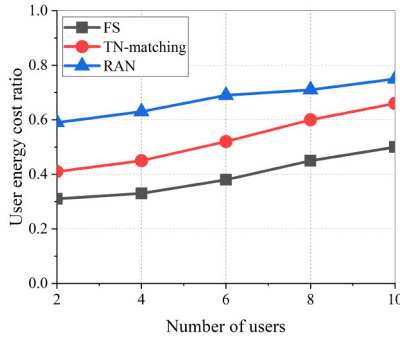


Fig. 4. UEC ratio.

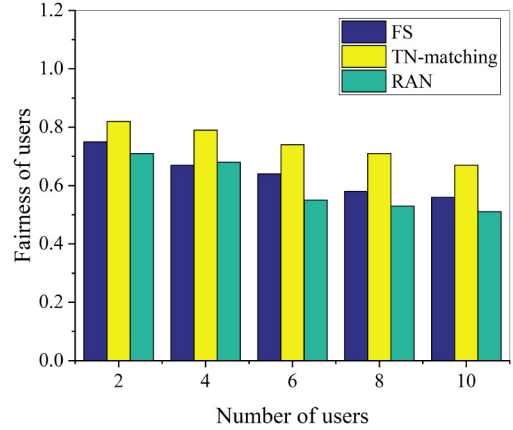


Fig. 6. Fairness index of users.

1) *Total Energy Cost*: As shown in Fig. 3, we can see that the TN-matching saves 30.8% more energy on average than RAN in total energy cost and is close to that of FS optimal solution as the number of users increases from 2 to 10. This is because the FS algorithm is optimized for the total energy cost. However, TN-matching takes the energy cost of each user and NCP as the optimization objectives, which is a multiobjective optimization. To balance the interests among the objectives, resulting in a loss of total energy cost. For RAN, the user's offloading decision is randomly selected, and the NCP also accepts the requests randomly without considering its energy cost, leading to poor performance. We can also find that the total energy cost of the three algorithms is on the rise as the number of users increases. This is because more users mean more data to be processed, which leads to an increase in transmission cost or computation cost, thus increasing total energy consumption.

2) *Energy Cost Ratio of Users and NCPs*: From Figs. 4 and 5, we can observe the gap between the UEC ratio and NEC ratio for FS and RAN is average 21.2% and 34.8%, whereas TN-matching is only 5.6%. The reason for these results is that the UEC ratio and NEC ratio are two game parameters, and TN-matching can maintain the balance of energy cost of both sides. Furthermore, TN matching makes the offloading decision from the perspectives of users and NCPs, respectively. It optimizes the total energy cost while promoting the balance of bilateral energy consumption and avoids excessive unilateral cost. However, FS only ensures that total energy costs are minimized and does not pay attention to the energy costs of NCPs and users.

3) *Fairness of Users*: Our simulation results show that the TN-matching outperforms RAN and FS in terms of user fairness as the number of users increases from 2 to 10. We can observe from Fig. 6 that the fairness of users obtained by the TN-matching is 14.2% and 20.1% better than FS and RAN, respectively, thus reflecting better user fairness. The reason accounts for these results is that TN-matching is a dispersed algorithm, each user makes the offloading decision for itself. The advantage is that users can maximize their respective interests, thereby narrowing the gap between them. However, FS focuses on minimizing total energy cost, ignoring individual interests. This will result in a low total cost, but the energy cost of a certain node is extremely high, which is unfair. For RAN, the random selection of task offloading is the essence of its fluctuating fairness. We can also see that the user fairness shows a downward trend with the user number. The reason is that NCP will be fully utilized as the task demand increases, but the fairness of users will be reduced due to the different quotas of NCPs.

4) *Number of Iterations*: To evaluate the computational complexity of the TN-matching algorithm, we investigate the number of iterations versus the number of users. As shown in Fig. 7, TN-matching is 86% iterations less than FS on average, which grows slowly as the user increases from 2 to 10. This is consistent with the complexity analysis in Section V-D. As we know, FS is an exhaustive global search that needs to traverse each NCP, so the number of iterations increases monotonically with the number of users. However, TN-matching takes each

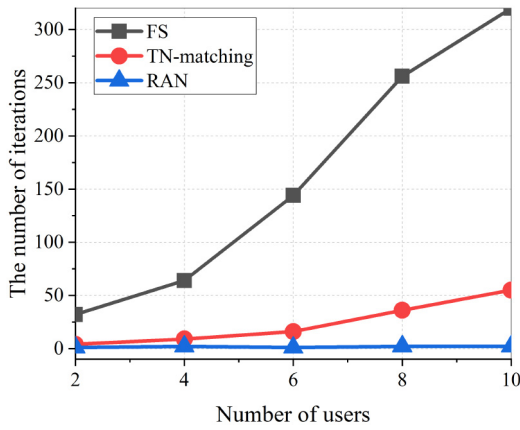


Fig. 7. Number of iterations.

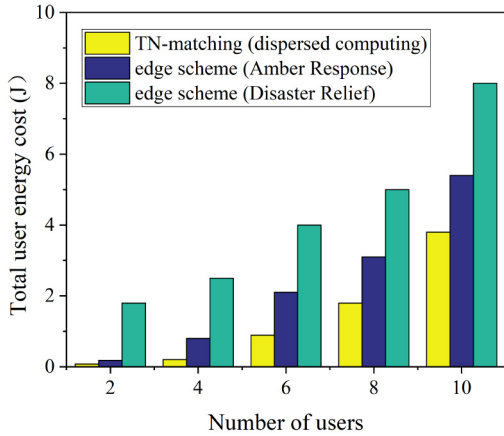


Fig. 8. Dispersed computing versus edge computing.

user's preference sequence as a reference and chooses NCP with preference, which greatly reduces the complexity of the algorithm and can achieve fast convergence. For RAN, the task randomly selects one NCP each time. Once rejected, it will be processed locally. Therefore, the number of iterations is not affected by the number of users and the maximum is 2.

VII. DISPERSED COMPUTING VERSUS EDGE COMPUTING

To further evaluate our proposed algorithm, we conduct two comparison experiments with the edge computing scheme in terms of total UEC, as shown in Fig. 8.

A. Evaluation Setup

We assume that an edge server is randomly distributed on a 10 km \times 10 km 2-D plane to provide services for users within its communication range. Meanwhile, $N = 4$ idle computation devices (mobile phones, computers, etc.) are randomly distributed around the users, and the distances are within the connected range [0.01, 1] km. The powerful computation capacity of edge server is set to 10 GHz, while the NCPs are normally distributed on [0.4, 2] GHz. According to the different network environments, it can be divided into two scenarios: 1) Amber Response (good network environment): $B = 10$ MHz and 2) Disaster Relief (poor network environment): $B = 1$ MHz.

B. Experimental Results and Analysis

For the Amber Response scenario, we can observe from Fig. 8 that the total UEC of TN-matching outperforms the edge scheme by 41.5% on average as the number of users increases from 2 to 10. The reason is that when a large number of devices in the coverage area make offloading requests for the edge cloud at the same time, it will inevitably cause problems such as excessive load pressure and low transmission efficiency. However, our proposed TN-matching for dispersed computing makes full use of idle computation resources around users, which not only shortens the transmission distance during offloading but also alleviates the congestion problem. To a certain extent, the transmission energy consumption of users is greatly reduced. We can also see that the energy cost of both TN-matching and edge scheme increases monotonically with the user number. For TN-matching, the idle computation resources are limited in dispersed computing. When the number of tasks far exceeds the available resources, some tasks cannot be offloaded, resulting in a greater increase in UEC. Faced with the same situation, the edge solution consumes more energy during offloading due to its relatively long communication distance and congestion.

The obtained result from the Disaster Relief experiment highlights the TN-matching more than that from the Amber Response experiment. As shown in Fig. 8, the total UEC of TN-matching lower than that of the edge scheme by 68.2% on average with the user number. We find that our proposed TN-matching is more suitable for scenarios with harsh network environments. We can explain that when the network condition is bad, the network speed is very slow or even unable to connect. For the edge computing scheme, the transmission cost of offloading will increase dramatically. However, due to the proximity of the devices, the dispersed computing scheme can self-organize into an organic network, so that the communication is not affected by the adverse environment, and users' energy can be greatly conserved by offloading.

Through the comparison and analysis of the two scenarios, we reveal insights that TN-matching has advantages over the edge scheme, especially in poor network environments.

VIII. CONCLUSION

In this article, we propose a dispersed computing offloading framework involving users and NCPs. To realize the framework, the task offloading process is modeled as a distributed multiobjective minimization problem. Then, we propose an energy-saving dispersed offloading algorithm named "Task-NCP bilateral matching," and present the specific theoretical analysis. The simulation results show that in terms of user fairness, the proposed algorithms improve on average by 14.2% and 20.1% compared with the baselines. Moreover, scene comparison experiments demonstrate that the proposed framework provides a feasible energy-saving scheme for the poor network environment.

Some works have focused on resource allocation and task offloading in dispersed computing, involving static and

dynamic algorithms. Whereas there is a fact that cannot be ignored, that is, most of the NCPs in the dispersed network are personal private devices, and it is difficult for them to participate in computing services for free in reality. Therefore, an incentive-driven offloading mechanism for dispersed computing is needed in future work. However, designing such a dispersed incentive mechanism will face some new challenges. For example, how to encourage idle devices to actively participate in the offloading mechanism to help reduce the computational load on the network? How to efficiently manage various idle resources? Our next work will focus on these new challenges and study how to design a situation-aware dispersed computing framework based on adaptive incentives.

REFERENCES

- [1] J. Yu, H. Chen, K. Wu, T. Zhou, Z. Cai, and F. Liu, "Centipede: Leveraging the distributed camera crowd for cooperative video data storage," *IEEE Internet Things J.*, early access, Apr. 22, 2021, doi: [10.1109/JIOT.2021.3074823](https://doi.org/10.1109/JIOT.2021.3074823).
- [2] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge computing enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, Oct. 2020.
- [3] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.
- [4] H. Wu, J. Zhang, Z. Cai, F. Liu, Y. Li, and A. Liu, "Toward energy-aware caching for intelligent connected vehicles," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8157–8166, Sep. 2020.
- [5] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou, and Q. Ni, "An offloading method using decentralized P2P-enabled mobile edge servers in edge computing," *J. Syst. Archit.*, vol. 94, pp. 1–13, Mar. 2019.
- [6] X. Fan *et al.*, "CTOM: Collaborative task offloading mechanism for mobile cloudlet networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6.
- [7] J. H. Anajemba, T. Yue, C. Iwendu, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020.
- [8] E. Baccour, A. Erbad, A. Mohamed, and M. Guizani, "CE-D2D: Dual framework chunks caching and offloading in collaborative edge networks with D2D communication," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Tangier, Morocco, 2019, pp. 1550–1556.
- [9] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.
- [10] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
- [11] X. Chen *et al.*, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Oct. 2019.
- [12] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.
- [13] B. Feng, C. Zhang, J. Liu, and Y. Fang, "D2D communications-assisted traffic offloading in integrated cellular-WiFi networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8670–8680, Oct. 2019.
- [14] B. Shang, L. Zhao, K.-C. Chen, and X. Chu, "An economic aspect of device-to-device assisted offloading in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2289–2304, Apr. 2018.
- [15] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
- [16] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020.
- [17] M. R. Schurgot, M. Wang, A. E. Conway, L. G. Greenwald, and P. D. Lebling, "A dispersed computing architecture for resource-centric computation and communication," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 13–19, Jul. 2019.
- [18] D. Siemon, F. Becker, L. Eckardt, and S. Robra-Bissantz, "One for all and all for one—Towards a framework for collaboration support systems," *Educ. Inf. Technol.*, vol. 24, no. 2, pp. 1837–1861, 2019.
- [19] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Communication-aware scheduling of serial tasks for dispersed computing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1330–1343, Aug. 2019.
- [20] A. Knezevic *et al.*, "CIRCE—A runtime scheduler for DAG-based dispersed computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, pp. 1–2.
- [21] P. Ghosh, Q. Nguyen, and B. Krishnamachari, "Container orchestration for dispersed computing," in *Proc. 5th Int. Workshop Container Technol. Container Clouds*, 2019, pp. 19–24.
- [22] D. Hu and B. Krishnamachari, "Throughput optimized scheduler for dispersed computing systems," in *Proc. 7th IEEE Int. Conf. Mobile Cloud Comput. Serv. Eng. (MobileCloud)*, Newark, CA, USA, 2019, pp. 76–84.
- [23] Q. Nguyen, P. Ghosh, and B. Krishnamachari, "End-to-end network performance monitoring for dispersed computing," in *Proc. IEEE Int. Conf. Comput. Netw. Commun. (ICNC)*, Maui, HI, USA, 2018, pp. 707–711.
- [24] H. Alamleh and A. A. S. AlQahtani, "A cheat-proof system to validate GPS location data," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, Chicago, IL, USA, 2020, pp. 190–193.
- [25] S. Jošilo and G. Dan, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Trans. Cloud Comput.*, early access, Jun. 20, 2019, doi: [10.1109/TCC.2019.2923768](https://doi.org/10.1109/TCC.2019.2923768).
- [26] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, San Diego, CA, USA, 2010, pp. 1–9.
- [27] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 48–54, Aug. 2018.
- [28] J. Liu, P. Li, J. Liu, and J. Lai, "Joint offloading and transmission power control for mobile edge computing," *IEEE Access*, vol. 7, pp. 81640–81651, 2019.
- [29] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
- [30] M. Caramia and P. Dell'Olmo, "Multi-objective optimization," in *Multi-Objective Management in Freight Logistics*. Cham, Switzerland: Springer, 2020, pp. 21–51.
- [31] C. Swain *et al.*, "METO: Matching theory based efficient task offloading in IoT-fog interconnection networks," *IEEE Internet Things J.*, early access, Sep. 21, 2020, doi: [10.1109/JIOT.2020.3025631](https://doi.org/10.1109/JIOT.2020.3025631).
- [32] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 103–122, Nov. 2016.
- [33] B. Wang, Y. Sun, H. M. Nguyen, and T. Q. Duong, "A novel socially stable matching model for secure relay selection in D2D communications," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 162–165, Feb. 2020.
- [34] F. Cooper, "Fair and large stable matchings in the stable marriage and student-project allocation problems," Ph.D. dissertation, Dept. School Comput. Sci., Univ. Glasgow, Glasgow, Scotland, 2020.
- [35] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, 2017, pp. 1–9.
- [36] C. Huang, G. Huang, W. Liu, R. Wang, and M. Xie, "A parallel joint optimized relay selection protocol for wake-up radio enabled WSNs," *Phys. Commun.*, vol. 47, Aug. 2021, Art. no. 101320.
- [37] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 207–220, Jan. 2019.
- [38] W. Wang, X. Wu, L. Xie, and S. Lu, "Femto-matching: Efficient traffic offloading in heterogeneous cellular networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, 2015, pp. 325–333.



Hongjia Wu received the B.S. degree from Liaoning Technical University, Fuxin, China, in 2016, and the M.S. degrees from Guangxi University, Nanning, China, in 2018. She is currently pursuing the Ph.D. degree with the College of Computer, National University of Defense Technology, Changsha, China.

She is a Research Assistant with Hunan University, Changsha. Her research interests include scheduling and resource allocation in dispersed computing.



Tongqing Zhou received the bachelor's, master's, and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 2012, 2014, and 2018, respectively.

He is currently a Postdoctoral Fellow with the College of Computer, NUDT. His main research interests include ubiquitous computing, mobile sensing, and data privacy.



Jiao Zhang received the Ph.D. degree from the National University of Defense Technology (NUDT), Changsha, China, in 2019.

She is currently a Postdoctoral Researcher with the College of Electronic Science and Technology, NUDT. Her research interests include mobile-edge computing and resource allocation in heterogeneous networks.



Jiaping Yu received the M.S. degree in information science and technology from Temple University, Philadelphia, PA, USA, in 2017. He is currently pursuing the Ph.D. degree with the College of Computer, National University of Defense Technology, Changsha, China.

His current research interests include distributed systems, blockchain, and edge computing.



Zhiping Cai (Member, IEEE) received the B.Eng., M.A.Sc., and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1996, 2002, and 2005, respectively.

He is a Full Professor with the College of Computer, NUDT. His current research interests include artificial intelligence, network security, and big data.

Dr. Cai is a Senior Member of CCF.



Haiwen Chen received the M.S. degree in computer science and technology from the National University of Defense Technology, Changsha, China, in 2017, where he is currently pursuing the Ph.D. degree with the College of Computer.

His current research interests include distributed systems, edge computing, and machine learning.



Qiang Ni (Senior Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees from the Huazhong University of Science & Technology, Wuhan, China, in 1993, 1996, and 1999, respectively.

He is a Professor with the School of Computing and Communications, Lancaster University, Lancaster, U.K. His research interests include the area of future generation communications and networking, including green communications and networking, millimeter-wave wireless communications, cognitive radio network systems, nonorthogonal multiple access, heterogeneous networks, 5G and 6G, SDN, cloud networks, edge computing, dispersed computing, energy harvesting, wireless information and power transfer, IoT, cyber-physical systems, AI and machine learning, big data analytics, and vehicular networks. He has authored or coauthored more than 300 papers in these areas.

Dr. Ni was an IEEE 802.11 Wireless Standard Working Group Voting Member and a contributor to various IEEE wireless standards.



Fang Liu (Member, IEEE) received the Ph.D. degree in computer science and technology from the National University of Defense Technology, Changsha, China, in 2005.

She is a Professor and a Ph.D. Supervisor with Hunan University, Changsha. Her main research interests include edge computing and storage system.