Resolving Multi-task Competition for Constrained Resources in Dispersed Computing: A Bilateral Matching Game

Hongjia Wu, **Jiao Zhang**, **Zhiping Cai***, **Qiang Ni**, **Tongqing Zhou**, **Jiaping Yu**, **Haiwen Chen** and **Fang Liu**

***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 further bring the computation to users, and serve as a supplement for sharing computing pressure when the edge is overloaded. In this paper, we first make full use of idle and geographically dispersed computation resources via task offloading, contributing to conserve energy for mobile devices. Specially, a dispersed computing offloading framework concerning the interests of users and network computation points is proposed. We further transform the initial problem into a multi-objective 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*—Dispersed computing, idle computation resources, energy-saving, multi-objective, bilateral matching, offloading.**

I. Introduction

With the increasing evolution of embedding processing into "smart" network nodes (such as smartphones, watches, and automobiles, etc.), the in-network computing paradigms such as edge computing [1] and fog computing have drawn unprecedented attention. Through offloading the computation-intensive tasks to the edge servers of networks [2]–[4], the computation experience and the battery lifetime of smart devices can be greatly improved. However, due to the constrained computation capability, 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 propose [5]–[8] 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 [9]–[11], 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, drones) [12]–[15] to overcome the computation resources limitations in these situations provides a promising alternative, especially considering the occasionally vacant computation capacity on-board. Yet, it is non-trivial to accommodate the distributed devices for dynamic computation tasks as the involvement will incur additional costs.

Driven by the popularity of IoT devices, dispersed computing [16], [17] 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) [18], which form a collaborative organic network to provide users with scattered, closer, and diverse offloading services. At the same time, some offloading techniques for dispersed computing [18]–[22] have attracted attention, involving graph task allocation and throughput optimization. However, the offloading problem of multi-user multi-task 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

Hongjia Wu, Zhiping Cai (corresponding author), Tongqing Zhou, Jiaping Yu and Haiwen Chen are with the College of Computer, National University of Defense Technology, Changsha, Hunan, CN 410073 e-mails: wuhongjia19@nudt.edu.cn; zpcai@nudt.edu.cn; zhoutongqing@nudt.edu.cn; yujiaping19@nudt.edu.cn; chenhaiwen13@nudt.edu.cn.

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

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

Fang Liu is with the School of Design, Hunan University, Changsha, Hunan, CN 410073 e-mail: liufang06@gmail.com.

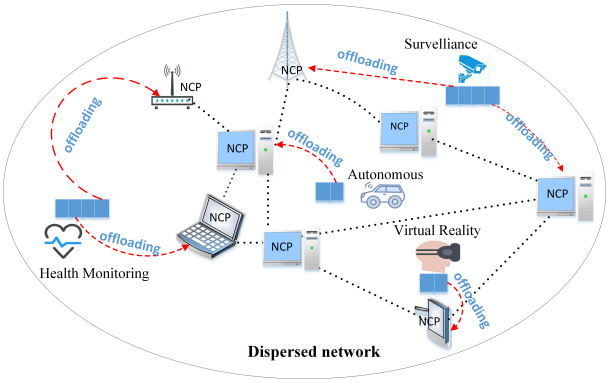

Fig. 1: Dispersed computing offloading framework

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 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:

- How to make full use of idle and geographically dispersed NCPs?
- How to handle user competition for limited resources?
- 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 multi-user multi-task occupying multi-NCP resources (MMOM) competition problem, and finally transform it into a multi-objective minimization problem. Then, we propose an energy-saving dispersed matching algorithm for balancing the interests of multiple objectives. In detail, the main contributions of the paper can be summarized as follows.

- We first present a dispersed computing offloading framework involving multi-user, multi-task, and multi-NCP, which takes into account the interests of users and NCPs, as well as the allocation of computation and communication resources.
- 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 of the proposed algorithm are analyzed and proved.
- 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, especially in a harsh network environment.

The rest of this paper 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 Section VI and VII, respectively. Finally, we conclude the paper 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 [5], [6], artificial intelligence [9]–[11] and additional assistance (phones, drones, etc.) [12]–[14]. For example, a collaborative task offloading mechanism (CTOM) for mobile cloudlet networks was designed in [5] to achieve more efficient and low-cost load balancing. Anajemba et al. [6] introduced a distributed multi-access MEC collaborative offloading technology based on the sub-optimal 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. [9] proposed a offloading algorithm based on double-deep Q network (DQN). The optimal strategy can be learned without knowing the prior knowledge of network dynamics. In [10], a deep reinforcement learning (DRL) algorithm was proposed to learn the optimal computing offloading and packet scheduling strategy. Feng et al. [11] 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 sudden surge of traffic or damage due to disasters and other reasons.

To increase the flexibility of the offloading framework, the work assisted by mobile devices has been focused on. Feng et al. [12] 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. [13] increased their overall profits by encouraging users to act as D2D transmitters, broadcasting their popular content to nearby users. In [14], 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 nodes.

B. Tasks offloading for dispersed computing

Different from the edge-oriented researches, some people began to pay attention to task offloading in dispersed computing [18]–[21], considering the ubiquitous idle devices with limited resources in the network. For example, Knezevic et al. [19] 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. [20] designed a container orchestration architecture for dispersed computing. The system automatically and efficiently distributes tasks among a group of networked computation nodes. Hu et al. [21] 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. [18] investigated a Max-Weight type scheduling policy, which is throughput-optimal for the proposed virtual queuing network. However, the offloading problem of multi-user multi-task occupying multi-NCP resources has not been taken into account, and the 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 which motivates our work in this paper.

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}	Data volume of 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

III. System Model and Problem Formulation

A. Network Model

In this paper, we consider a multi-user, multi-task, 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 $\mathcal{M}=\{1,2,\cdots,M\}$ and $\mathcal{N}=\{1,2,\cdots,n\}$, respectively. Without loss of generality, we assume that each user has K independent computation-intensive tasks denoted by $\mathcal{K}=\{1,2,\cdots,K\}$. 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

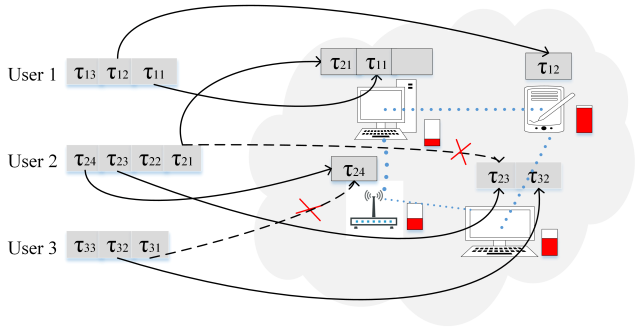

Fig. 2: An example of a dispersed network that consists of $M = 3$ users, $K = 3$ or 4 tasks and $N = 4$ NCPs

[23](e.g. using GPS or some positioning methods). The system model parameters and descriptions are provided in TABLE I.

Each user i has the computation-intensive tasks $\tau_{ik} = \{D_{ik}, C_{ik}, T_{ik}^{max}\}$, which can be described as:

- The input D_{ik} is defined as the amount of data for user i's k-th task, including system settings and input parameters.
- C_{ik} is the number of CPU cycles required to complete the task.
- 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 sub-tasks. To facilitate the analysis, we make the common assumption that the set of devices changes slowly [24].

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 how consumes power 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 [25] is 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

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

where, s_{ikj} is the computation resources allocated by NCP j to 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, $q_j \leq q_j^{max}$.

2) Communication energy cost: We adopt Orthogonal Frequency Division Multiple Access (OFDMA) technology as the communication mode for data transmission to avoid mutual interference between communication links [26]. The achievable uplink transmission rate between user i and NCP j can be defined as

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

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

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

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).

(1) Energy cost of the 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^{task}_{ikj} of task k can be expressed as

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

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

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

We ignore the energy consumption of other hardware components (e.g., RAM), which are generally much less significant [27].

(2) 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^{ncp}_{ikj} of task k can be expressed as

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

Therefore, the energy cost E^{ncp}_j of NCP j can be calculated by

$$\begin{aligned} E^{ncp}_j &= \sum_{i=1}^{M} \sum_{k=1}^{K} e^{ncp}_{ikj}(X_{ikj}, X_{-ikj}) \\ &= \sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj}(e^{com}_{ikj} + e^r_{ikj}). \end{aligned} \quad (8)$$

Assume that the energy and time used to transmit the communication results from the NCP j to the user i can be neglected [28]. Finally, we define the total energy cost of users and NCPs as E^{user} and E^{ncp}, respectively.

C. Problem Definition

Based on the above formulation, we then formally define the offloading problem between multi-user and multi-NCP as the optimization of both users' and NCPs' energy costs during the involvement. On 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

$$\begin{aligned} & User: \min_{X_{ikj} \in \{0,1\}} E^{user}_i \\ s.t. \; & C1: X_{ik0} \tfrac{C_{ik}}{s_0} + (1 - X_{ik0}) X_{ikj} (\tfrac{D_{ik}}{r_{ikj}} + \tfrac{C_{ik}}{s_{ikj}}) \le T^{max}_{ik} \\ & C2: X_{ikj} \in \{0, 1\}, \forall j \in N \cup \{0\} \\ & C3: \textstyle\sum_{j=0}^{N} X_{ikj} = 1 \\ & C4: i \in \mathcal{M}, k \in \mathcal{K}, j \in \mathcal{N} \end{aligned} \quad (9)$$

where, C1 ensures that task τ_{ik} is completed within the latency constraint. C2 indicates whether user i's k-th task is offloaded to the j-th NCP, which is the final offloading decision solution for the user. 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

$$\begin{aligned} & NCP: \min_{X_{ikj} \in \{0,1\}} E^{ncp}_j \\ s.t. \; & C1: E^{total}_j - E^{ncp}_j \ge \delta_j \\ & C2: \textstyle\sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj} s_{ikj} \le s_j \\ & C3: \textstyle\sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj} B_{ikj} \le B_j \\ & C4: \textstyle\sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj} = min[\sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj}, q^{max}_j] \\ & C5: \textstyle\sum_{i=1}^{M} \sum_{k=1}^{K} X_{ikj} \le q^{max}_j \\ & C6: i \in \mathcal{M}, k \in \mathcal{K}, j \in \mathcal{N} \end{aligned} \quad (10)$$

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; 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 rationale behind the design is: (1) the mutual aid idea of "one for all and all for one" in dispersed computing; (2) the available idle resources of NCPs are only a part of the resources they have and do not affect their normal work.

Considering the optimization process on both sides, we state that the offloading problem in fact transforms into a distributed multi-objective optimization problem. Besides, since the two optimization objectives have a common solution and their constraints are coupled, which is also a two-way selection. We hope that with this solution, each user will be matched to an NCP. For another, each NCP can get the appropriate offloading tasks.

IV. An Energy-Saving Bilateral Matching Based Dispersed Offloading Strategy

The multi-objective optimization problem is proved to be NP-hard [29]. Its solution can be found by searching for all possible offloading decisions. However, the computational complexity of the full search 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 multi-objective optimization problem as a one-to-many matching game with resource and latency constraints, we consider the task set $\Gamma = \{1, 2, \cdots, M\} \times \{1, 2, \cdots, K\}$ and NCP set $N = \{1, 2, \cdots, N\}$ 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 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 $N = \{n_j\}_{j=1}^{|N|}$, let two disjoint finite sets of $S^\Gamma = \{1, \cdots, |\Gamma|\}$ and $S^N = \{1, \cdots, |N|\}$, then a one to many matching function Ψ [30]: $\{\Gamma\} \cup \{N\} \to \{\Gamma\} \cup \{N\}$ is defined such that for all $i \in M \times K$ and $j \in N$

(i) $\Psi(\tau_i) \in \left\{ n_{j' \in S^\Gamma} \in N \right\} and |\Psi(\tau_i)| \in \{0, 1\}$

(ii) $\Psi(n_j) \subseteq \left\{ \tau_{i' \in S^N} \in \Gamma \right\} and |\Psi(n_j)| \leq q_n^{max}$

(iii) $\Psi(\tau_i) = n_j \Leftrightarrow \Psi(n_j) = \tau_i$.

Thus, the matching game Ψ can be defined by a tuple $\{\Gamma, N, q_n^{max}, \succ_\tau, \succ_n\}$, where q_n^{max} is the NCP n's quota vector. Here, $\succ_\tau$ and $\succ_n$ represent the set of the preference relations of tasks and NCPs, respectively. Condition (i) implies that each task can only be offloaded to one NCP at most. Condition (ii) represents the maximum number of offloaded tasks that each NCP can accept, corresponding to C5. The condition (iii) 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 $< \tau_i, n_j >_{i \in \Gamma, j \in N}$ between tasks and NCPs.

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

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

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

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

Algorithm 2: Matching evaluation for each NCP with respect to $\{\delta_j\}_{j \in N}$

Input: $L_{unmatched}$, L_τ and X.

Output: $L_{unmatched}$, $L_{matched}$ and L_τ.

for $j = 1$ *to* $|N|$ **do**
  for $\tau = 1$ *to* $|L_{unmatched}|$ **do**
    if $X_{\tau j} = 1$ *and* $E_j^{total} - E_j^{ncp} \geq \delta_j$ **then**
      Generate L_j using (12), and sort in descending order;
  if $q_j < q_j^{max}$ **then**
    Accept the top ranked τ' in L_j;
    $L_j^{accept} = L_j^{accept} \cup < \tau', j >$;
    $q_j = q_j + 1$;
    $L_{unmatched} = \{L_{unmatched} \setminus \tau'\}$;
  else
    $L_j = L_j \cup L_j^{accept}$;
    Select the top q_j^{max} tasks: $L_j^{accept} = L_j[: q_j^{max}]$;
    $L_{unmatched} = \left\{L_{unmatched} \setminus L_j[: q_j^{max}]\right\}$;
    $L_{unmatched} = L_{unmatched} \cup \left\{L_j \setminus L_j[: q_j^{max}]\right\}$;
  for τ *to* $|L_{unmatched}|$ **do**
    $L_\tau = \{L_\tau \setminus j\}$
  $L_{matched} \leftarrow L_j^{accept}$
return $L_{unmatched}$, $L_{matched}$ and L_τ

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

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

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.

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

$$P_n(\tau) = 1/C_\tau. \tag{12}$$

For NCP, a small amount of computation will greatly reduce the computation time and energy consumption of NCP. In this way, NCP can reduce its energy cost and provide computation services for more users.

Algorithm 3: Task-NCP Bilateral Matching Algorithm

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

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

C. Task-NCP Bilateral Matching Algorithm

The Task-NCP bilateral matching algorithm (TN-matching) consists of two major sub-algorithms, as shown in Algorithm 1 and Algorithm 2, respectively. Algorithm 1 solves the task assignment problem of users. The 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, NCP chooses to accept the offloading tasks which make it energy-saving. Algorithm 3 combines algorithm 1 and algorithm 2 to solve the matching problem between tasks and NCPs, and achieves the final stable matching between the two through iteration.

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. The network information includes the available NCPs' computation capabilities, 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 evaluates its task completion time according to the parameters obtained in phase 1, and filters out the NCPs within the delay constraint. Then, based on the task preference function of Eq. (11), calculate the preference values of the selected NCPs and sort them in descending order (line 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 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 (line3). Then, each NCP calculates the preference values for the remaining tasks according to Eq. (12) and ranks them in descending order (line4). Note that the preference function calculation (line4) here is an important step of algorithm 2, which aims to help NCP 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 set $L_{unmatched}$ (line 5-9); Otherwise, the new task request is added to the preference list L_j, and L_j is reordered (line10-14). Then, update the preference list L_τ of rejected tasks by deleting the current top-ranked NCP j (line 15-16). 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 $L_{unmatched}$ 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 Algorithm 1 and Algorithm 2 separately, and certain communication and interaction are required between them.

V. Theoretical Analysis

In this section, we give a series of theoretical analysis on the stability, local optimality, Price of Anarchy, 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 [31], which is defined as follows.

***Definition 4*: (Blocking Pair)** The pair (τ', n') is a blocking pair [32] 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

$$\begin{aligned}(\forall \tau_i \in \Gamma, n_j \in N)((\tau_i, \Psi(\tau_i)), (n_j, \Psi(n_j))) \Rightarrow \\ (\exists \tau' \in \Gamma, n' \in N)((\tau', \Psi(\tau_i)), (n', \Psi(n_j))).\end{aligned} \tag{13}$$

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 [33].

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 algorithm in this paper 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: 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; 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 algorithm still has good stability in reality.

B. Local Optimality Analysis

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

Proof. Suppose that n_1 selects q tasks τ_1, τ_2,... τ_q from all the requested tasks, and rejects other tasks including task τ'. It can be seen that tasks τ_1, τ_2,... τ_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 a 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

$$P_{\tau_i}(n') < P_{\tau_i}(n_1) \; and \; P_{n_1}(\tau') < P_{n_1}(\tau_i), \tag{14}$$

which will cause the algorithm to be unstable.

According to ***Definition 4***, ***Definition 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 MMOM 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 user energy cost would be from the optimal in a stable matching. To quantify the difference from the optimal performance, we use the Price of Anarchy (PoA), defined as the ratio of the worst case user energy cost to the minimal user energy cost

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

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 price of anarchy for the matching game Ψ^* is upper bounded by

$$\frac{\sum_{\tau \in M \times K} e_{\tau 0}^{com}}{\sum_{\tau \in M \times K} min\left\{e_{\tau 0}^{com}, e_{\tau 1}^{-}, e_{\tau 2}^{-}, \cdots, e_{\tau N}^{-}\right\}}. \tag{16}$$

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 $\boldsymbol{X}^*$ is the offloading decision of Ψ^*. Observe that $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) \leq e_{\tau 0}^{com}$ holds for every task $\tau \in M \times K$. Otherwise, if $\exists \tau \in M \times K$ enables $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) > e_{\tau 0}^{com}$, 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^*, $e_{\tau j}^t(X_{\tau j}^*, X_{-\tau j}^*) \leq e_{\tau 0}^{com}$ holds.

Next, let us consider an arbitrary offloading decision $(X_{\tau j}, X_{-\tau j}) \in \boldsymbol{X}$. If X_τ=0 ($X_\tau = \sum_{j=1}^N X_{\tau j}$), then $e_{\tau j}^{task}(X_{\tau j}, X_{-\tau j}) = e_{\tau 0}^{com}$. 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

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

Hence, $e^t_{\tau j}(X_{\tau j}, X_{-\tau j}) \geq min\left\{e^{com}_{\tau 0}, e^-_{\tau 1}, e^-_{\tau 2}, \cdots, e^-_{\tau N}\right\}$ and

$$\sum_{\tau \in M\times K} e^t_{\tau j}(X_{\tau j}, X_{-\tau j}) \geq \sum_{\tau \in M\times K} min\left\{e^{com}_{\tau 0}, e^-_{\tau 1}, e^-_{\tau 2}, \cdots, e^-_{\tau N}\right\}.$$

Using these expressions, we can establish the following bound as

$$\begin{aligned} PoA &= \frac{max\sum_{i\in M} E^{user}_i(\Psi^*)}{min\sum_{i\in M} E^{user}_i(\Psi)} \\ &\leq \frac{\sum\limits_{\tau\in M\times K} e^{com}_{\tau 0}}{\sum\limits_{\tau\in M\times K} min\left\{e^{com}_{\tau 0}, e^-_{\tau 1}, e^-_{\tau 2}, \cdots, e^-_{\tau N}\right\}}, \end{aligned} \quad (18)$$

which proves the theorem. □

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 a 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. □

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 $1km*1km$ 2D plane. We assume that the channel gain g_{ikj} is proportional to d^γ_{ikj}, where d_{ikj} is the distance from user i to NCP j, and $\gamma=4$ [34], [35] is the path loss index. The channel bandwidth B of each NCP is uniformly distributed within $[3,10]$MHz, while the transmission power p_{ikj} is set to 0.5W. The computation capability s_0 of users is set to 0.5GHz [36], 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 NCP j is random distributed in $[5,10]$J. The device-related parameter α has been usually assumed to be around 3 [25]. 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 full search algorithm (FS) 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, user energy cost ratio (UEC), NCP energy cost ratio (NEC), Jain's fairness index of users [37] 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.

1) Total energy cost: As shown in Fig. 3, we can see that 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 multi-objective optimization. In order 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

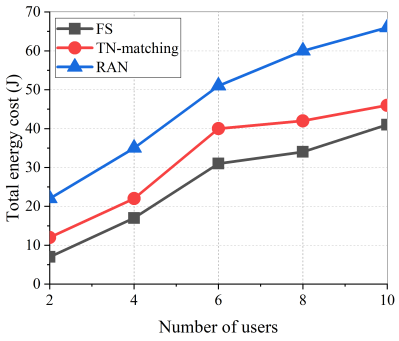

Fig. 3: Total Energy Cost

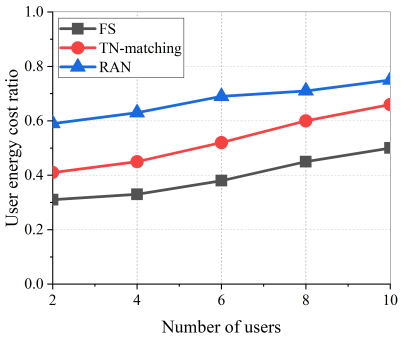

Fig. 4: User Energy Cost Ratio

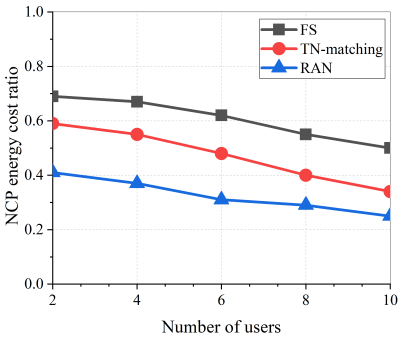

Fig. 5: NCP Energy Cost Ratio

that the total energy cost of the three algorithms is on the rise as the number of users increases. This is because more users means more data to be processed, which leads to an increase in transmission cost or computation cost, thus increasing total energy consumption.

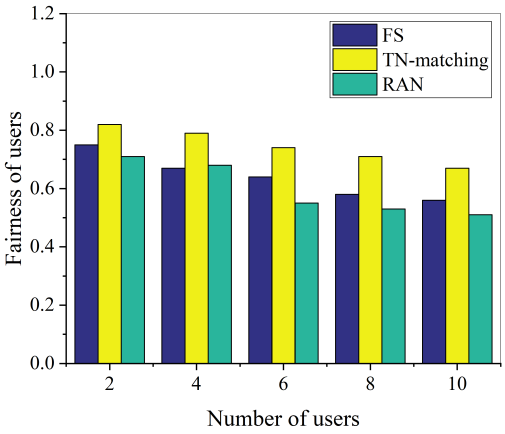

Fig. 6: Fairness Index of Users

2) Energy cost ratio of users and NCPs: From Fig. 4 and Fig. 5, we can observe the gap between 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 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 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.

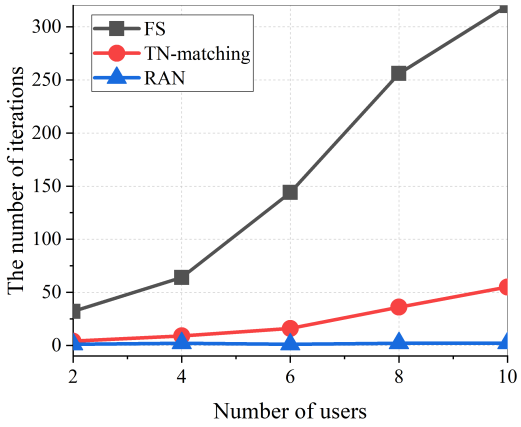

Fig. 7: The Number of Iterations

4) The number of iterations: To evaluate the computational complexity of TN-matching algorithm, we investigate the number of iterations versus the number of users. Our simulation results show that 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 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 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 Vs. Edge Computing

To further evaluate our proposed algorithm, we conduct a comparative experiment on the total energy cost of users with the edge computing scheme in two different scenarios, as shown in Fig. 8.

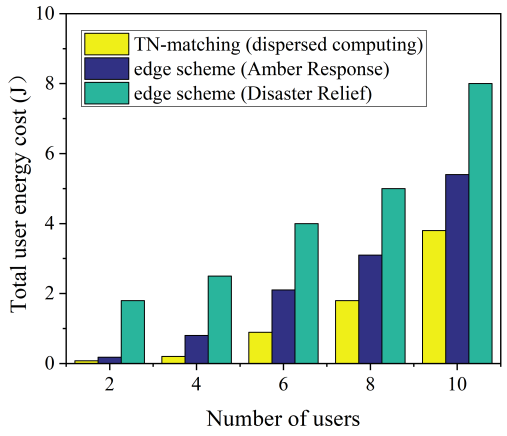

Fig. 8: Dispersed computing vs. Edge computing

A. Evaluation Setup

We assume that an edge server is randomly distributed on a $10km * 10km$ 2D 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 distance is within the connected range $[0.01, 1]$km. The powerful computation capability of edge sever is set to 10GHz, while the NCPs are uniformly distributed on $[0.4, 2]$GHz. According to the different network environment, it can be divided into two scenarios: 1) Amber Response (good network environment): B=10MHz; 2) Disaster Relief (poor network environment): B=1MHz.

B. Experimental Results and Analysis

For the Amber Response scenario, we can observe from Fig. 8 that the total user energy cost 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 user energy cost. 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 user energy cost of TN-matching lower than that of edge scheme by 68.2% on average with 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 paper, we propose a dispersed computing offloading framework involving users and NCPs. To realize the framework, the task offloading process is modeled as a multi-user, multi-task and multi-NCP competition problem, and further transformed into a multi-objective 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 algorithm improves 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 task allocation and offloading in dispersed computing, involving static and dynamic algorithms. Whereas, there is a fact that cannot be ignored, that is, most of the networked computation points in the dispersed network are personal private devices, and it is unrealistic for them to participate in computing services for free. 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.

IX. Acknowledgment

This work was supported in part by the National Key Research and Development Program of China (2018YFB1800202, 2016YFB1000302, SQ2019ZD090149, 2018YFB0204301), the National Natural Science Foundation of China (61832020, 61702569,62001483, U19B2024), Key-Area Research and Development Program of Guang Dong Province (2019B010107001) and the science and technology innovation Program of Hunan Province (2020RC2045).

References

[1] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge computing enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, 2020.

[2] 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 Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[3] H. Wu, J. Zhang, Z. Cai, F. Liu, Y. Li, and A. Liu, "Toward energy-aware caching for intelligent connected vehicles," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8157–8166, 2020.

[4] 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," *Journal of Systems Architecture*, vol. 94, pp. 1–13, 2019.

[5] X. Fan, X. He, D. Puthal, S. Chen, C. Xiang, P. Nanda, and X. Rao, "Ctom: Collaborative task offloading mechanism for mobile cloudlet networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[6] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020.

[7] 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 *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1550–1556, IEEE, 2019.

[8] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for d2d-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[9] 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 of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[10] X. Chen, Z. Zhao, C. Wu, M. Bennis, H. Liu, Y. Ji, and H. Zhang, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2377–2392, 2019.

[11] 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 of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2019.

[12] B. Feng, C. Zhang, J. Liu, and Y. Fang, "D2d communications-assisted traffic offloading in integrated cellular-wifi networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8670–8680, 2019.

[13] B. Shang, L. Zhao, K.-C. Chen, and X. Chu, "An economic aspect of device-to-device assisted offloading in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2289–2304, 2018.

[14] 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 Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, 2020.

[15] 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 of Things Journal*, vol. 7, no. 4, pp. 2777–2790, 2019.

[16] 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 Communications Magazine*, vol. 57, no. 7, pp. 13–19, 2019.

[17] D. Siemon, F. Becker, L. Eckardt, and S. Robra-Bissantz, "One for all and all for one-towards a framework for collaboration support systems," *Education and Information Technologies*, vol. 24, no. 2, pp. 1837–1861, 2019.

[18] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Communication-aware scheduling of serial tasks for dispersed computing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1330–1343, 2019.

[19] A. Knezevic, Q. Nguyen, J. A. Tran, P. Ghosh, P. Sakulkar, B. Krishnamachari, and M. Annavaram, "Circe-a runtime scheduler for dag-based dispersed computing," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–2, 2017.

[20] P. Ghosh, Q. Nguyen, and B. Krishnamachari, "Container orchestration for dispersed computing," in *Proceedings of the 5th International Workshop on Container Technologies and Container Clouds*, pp. 19–24, 2019.

[21] D. Hu and B. Krishnamachari, "Throughput optimized scheduler for dispersed computing systems," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 76–84, IEEE, 2019.

[22] Q. Nguyen, P. Ghosh, and B. Krishnamachari, "End-to-end network performance monitoring for dispersed computing," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 707–711, IEEE, 2018.

[23] H. Alamleh and A. A. S. AlQahtani, "A cheat-proof system to validate gps location data," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, pp. 190–193, IEEE, 2020.

[24] S. Jošilo and G. Dan, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Transactions on Cloud Computing*, 2019.

[25] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *2010 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–9, IEEE, 2010.

[26] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, 2018.

[27] 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.

[28] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[29] M. Caramia and P. Dell'Olmo, "Multi-objective optimization," in *Multi-objective management in freight logistics*, pp. 21–51, Springer, 2020.

[30] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. Rodrigues, and V. H. C. de Albuquerque, "Meto: Matching theory based efficient task offloading in iot-fog interconnection networks," *IEEE Internet of Things Journal*, 2020.

[31] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 103–122, 2016.

[32] 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 Communications Letters*, vol. 9, no. 2, pp. 162–165, 2019.

[33] F. Cooper, *Fair and large stable matchings in the stable marriage and student-project allocation problems*. PhD thesis, University of Glasgow, 2020.

[34] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *2017 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–9, IEEE, 2017.

[35] M. X. S. Z. C. Huang, R. Wang, "Artificial intelligence-aware contend node restricted joint consecutive packet transmissions receiver-initiated protocol for efficient wireless communications," vol. 7, 2021.

[36] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 207–220, 2018.

[37] W. Wang, X. Wu, L. Xie, and S. Lu, "Femto-matching: Efficient traffic offloading in heterogeneous cellular networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 325–333, IEEE, 2015.