User Matching on Blockchain for Computation Offloading in Ultra-Dense Wireless Networks

Shuming Seng, Changqing Luo, *Member, IEEE*, Xi Li, Heli Zhang, and Hong Ji, *Senior Member, IEEE*

***Abstract*—The popularity of mobile devices has driven the occurrence of ultra-dense wireless networks (UDNs) to accommodate high volumes of data therein generated by mobile users (MUs) running various mobile applications, such as virtual reality/augmented reality (VR/AR) and online gaming. Since many such mobile applications are computation-intensive, mobile edge computing (MEC), a promising computing paradigm that can provide on-demand computation resources (i.e., virtual machines (VMs)), is considered to adopt for allowing MUs to offload their computation tasks to EgSvrs in the proximity. Previous works have developed many schemes to schedule computation tasks to be offloaded to edge servers (EgSvrs). However, these developed schemes are centralized, which are vulnerable to congested coordination operations, and have no trustworthiness guarantee as well. In this paper, we propose to design a decentralized coordination scheme to orchestrate MUs and EgSvrs for scheduling computation tasks to their right VMs by taking advantage of blockchain technology. Specifically, we develop an efficient task-VM matching algorithm that jointly considers task execution time and energy consumption. Particularly, we prove the stability of the task-VM matching achieved by the developed matching algorithm. Besides, we further implement the developed task-VM matching algorithm on the blockchain by developing a smart matching contract to perform task-VM matching on the blockchain without trusted third parties. Extensive simulation results demonstrate that our decentralized coordination scheme can improve the performance significantly and converge to a stable state very quickly.**

***Index Terms*—Ultra-dense wireless networks (UDNs), mobile edge computing (MEC), blockchain, computation offloading, matching.**

I. Introduction

THE increasing popularity of mobile devices like smartphones and tablets has led to the ever-increasing data volumes, which hence have driven the evolution of wireless networks. In the past decades, we have witnessed the emergence of various wireless networks, such as the macrocell based, the microcell based, the picocell based, the femtocell based, etc., [1]–[4]. Their emergences are due to the demand for accommodating the ever-increasing data traffic therein. Specifically, they take advantage of the cell splitting technology [5] to continually divide the cells of wireless networks into smaller and smaller ones along with the time. As a result, cells in some areas (e.g., metro-cities) are very crowded, which naturally leads to an ultra-dense wireless network (UDN) with small-cell base stations (SBSs) [6], [7], for providing very high system capacity to accommodate high data volumes.

On the other hand, mobile devices' explosive growth has also driven the emergence of various mobile applications. More importantly, many of them are generally computation-intensive, such as virtual reality/augmented reality (VR/AR), object recognition, and online gaming [8], [9]. Running these mobile applications typically requires high computations from mobile devices. However, it is a well-known fact that current mobile devices usually have limited computation resources that cannot meet the demand sufficiently. Therefore, new computing frameworks are in dire need to enable mobile users (MUs) to play with various computation-intensive mobile applications smoothly.

For this purpose, academia and industry propose mobile edge computing (MEC) to empower MUs to run computation-intensive mobile applications. Specifically, the MEC paradigm enables MUs to offload computations to their nearby edge servers (EgSvrs), which provide on-demand computation resources, i.e., virtual machines (VMs) (e.g., Amazon EC2 M3.large instances [10]), to help conduct the required computation tasks [11], [12]. In particular, though offloading transmissions incur extra transmission delay, the time is acceptable because EgSvrs are very close to MUs (e.g., sitting at SBSs), which makes MEC very suitable for UDNs. In response to adopting MEC in UDNs, many researchers have been concerned about the most related research issues and developed quite a few computation offloading schemes. For example, Chen *et al.* [13] develop an online SBS peer offloading framework in MEC-enabled UDNs to maximize the long-term system performance without knowing future system dynamics information. Sun *et al.* [14] consider the energy consumption of the computation offloading in MEC-enabled UDNs.

Despite having been studied for years, many problems are still open, and extensive efforts are required to fulfill the

Manuscript received October 1, 2019; revised February 19, 2020 and May 10, 2020; accepted May 23, 2020. Date of publication June 9, 2020; date of current version July 7, 2021. This work was partially supported by the National Natural Science Foundation of China under Grants 61771070 and 61671088, Commonwealth Cyber Initiative (CCI) Smart Cities, and the Beijing University of Posts and Telecommunications (BUPT) Excellent Ph.D. Students Foundation under Grant CX2019219. Recommended for acceptance by Dr. Yulei Wu. *(Corresponding author: Xi Li.)*

Shuming Seng is with the Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China, and also with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: seng@bupt.edu.cn).

Changqing Luo is with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: cluo@vcu.edu).

Xi Li, Heli Zhang, and Hong Ji are with the Key Laboratory of Universal Wireless Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lixi@bupt.edu.cn; zhangheli@bupt.edu.cn; jihong@bupt.edu.cn).

Digital Object Identifier 10.1109/TNSE.2020.3001081

mission of adopting MEC in UDNs. First, previously developed schemes are centralized and vulnerable to a single point of failure and congested coordination operations among MUs and EgSvrs. Specifically, these schemes assume one centralized coordinator to orchestrate the scheduling of computation offloading among massive MUs and EgSvrs in a practical UDN. The fact is that most of the MUs may request computation offloading services at a moment. As a consequence, the concurrent service requests can cause considerable coordination operations, which hence challenges the single coordinator. In particular, UDNs in nature suffer higher mobility management demands, which can significantly increase the number of computation offloading service requests, further exacerbating the congested coordinations [14], [15]. Second, these previously developed schemes have not carefully considered the trustworthiness among MUs, EgSvrs, and the coordinator. In practice, there is no fully established trust among them, as all of them are individually rational and attempt to maximize their own profits [16]. As a result, the yielded coordination results are not trustworthy. Therefore, a decentralized computation offloading coordination scheme with trustworthiness is in desperate need.

The emergence of blockchain provides a promising alternative solution to the aforementioned issues. Blockchain is a decentralized public ledger shared and agreed among all participants in a peer-to-peer (P2P) network [17]–[19]. Each participant in the P2P network can view the contents in all blocks. All users can write their transactions into a block after creating this block, and all participants can verify the content in the block. Thus, contents stored on blockchain are unable to be manipulated, enhancing the trustworthiness of all users and participants. However, how to leverage blockchain technology to design decentralized computation offloading schemes is still a challenging problem. In the literature, there have been a few existing works on blockchain-based applications in wireless networks [20]–[28]. Nevertheless, most of them attempt to address research issues within wireless networks, such as data security, trustworthiness, and blockchain-based radio resource management. Exploiting blockchain technology in wireless networks is still in the infancy, particularly when applying it to perform the coordination among MUs and EgSvrs for computation offloading in MEC-enabled UDNs.

In this paper, we propose to take advantage of blockchain technology to design a decentralized coordination scheme for computation offloading in UDNs. Specifically, to schedule MUs' computations to the right EgSvrs, we base a deferred acceptance algorithm, a widely studied matching algorithm [29], to design a task-VM matching algorithm that jointly considers task execution time and energy consumption. In particular, we have proved that the designed matching algorithm can achieve a stable matching. Moreover, to perform the task-VM matching between MUs and EgSvrs without trusted third-parties, we leverage smart contract to develop a novel smart matching contract to implement the designed task-VM matching algorithm on the blockchain. Simulation results demonstrate the significant performance improvement and the fast convergence of the designed coordination scheme.

Our major contributions of this paper are summarized as follows.

1) We take advantage of blockchain technology to design a decentralized coordination scheme for computation offloading in UDNs.
2) We design an efficient task-VM matching algorithm to perform the matching between MUs and EgSvrs, which jointly takes into account task execution time and energy consumption.
3) We implement the designed task-VM matching algorithm on the blockchain by developing a new smart matching contract to perform matching on the blockchain without trusted third parties.
4) We have proved that the achieved task-VM matching is stable.
5) We conduct extensive and thorough simulations to evaluate the performance of the decentralized coordination scheme. The simulation results show the significant performance improvement and the fast convergence speed achieved by the developed scheme.

The rest of this paper is organized as follows. We first introduce the most related works in Section II. Then, we present the considered system model in Section III. Afterwards, we describe the developed decentralized coordination scheme for computation offloading in UDNs in Section IV. We subsequently present and discuss the performance results of the simulation thoroughly in Section V, and finally conclude this paper in Section VI.

II. Related Works

Due to the popularity of cryptocurrency systems, blockchain technology, one of the most fundamental techniques applied in cryptocurrency systems, has received extensive attention from academia and industry. Particularly, many researchers have explored blockchain technology in wireless networks and studied research issues therein, such as data security, trustworthiness, and blockchain-based radio resource management. We summarize them as follows.

A chunk of previous works has leveraged blockchain technology to protect data security and privacy in wireless networks. For example, Kotobi *et al.* [20] employed blockchain technology to design a verification scheme for secure spectrum sharing in cognitive radio networks. Raju *et al.* [21] took advantage of blockchain technology to develop a decentralized user identity management system for user privacy protection in cloud-centric cognitive cellular networks.

In addition to data security and privacy, some previous works are concerned about trustworthiness in wireless networks and have explored blockchain technology to address this issue. For instance, Liu *et al.* [22] designed a blockchain-based consensus algorithm for the trust establishment amongst electric vehicles in vehicle-to-vehicle (V2V) networks with the support of edge computing. Lin *et al.* [23] also developed a blockchain-based consensus scheme to verify the authenticity of channel state information for device-to-device (D2D) communications in wireless cellular networks. Yazdinejad *et al.* [24] designed a

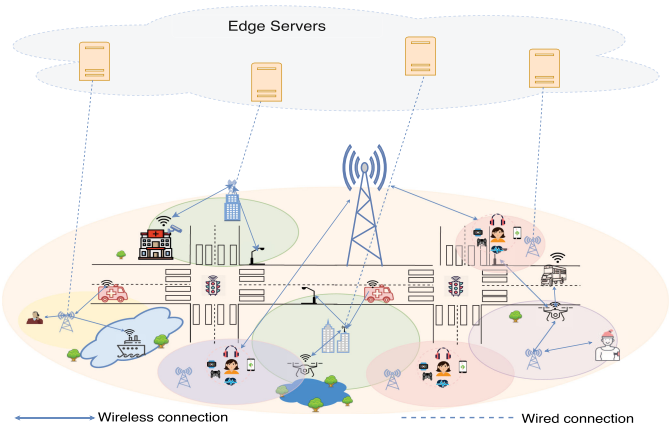

Fig. 1. An example of the considered UDN.

blockchain-enabled authentication approach for a re-authentication issue incurred by frequent handovers between two base stations. Guo *et al.* [25] designed a practical Byzantine fault tolerance consensus method to guarantee the trusted authentication of IoT (Internet of Things) terminals.

Some previous works have employed blockchain technology for managing radio resources in wireless networks. For example, Dai *et al.* [26] exploited blockchain and artificial intelligence techniques to construct a secure and intelligent network architecture to enable a secure and flexible resource sharing in wireless networks. Qiao *et al.* [27] applied blockchain technology to design a radio resource management framework for radio resource trading and task assignments in MEC-enabled wireless networks. Rawat *et al.* [28] also exploited one of the blockchain features, i.e., a public ledger, to construct a sublease system for radio resource allocation.

From the above analysis, we notice that applying blockchain technology in wireless networks has been widely studied. However, few of them take advantage of blockchain technology into computation offloading for UDNs.

III. System Model

In this section, we first present the mobile application model and then the computation offloading in the considered UDN.

A. The Mobile Application Model

We consider that M MUs in a UDN run computation-intensive mobile applications composed of many computation tasks[1]. Specifically, MU m runs a mobile application that consists of C_m tasks. Let $\mathcal{C}_m = \{1, 2, \ldots, C_m\}$ denote the set of tasks. Thus, we have $\mathbf{T} = \mathcal{C}_1 \cup \mathcal{C}_2 \cdots \cup \mathcal{C}_M$ to represent the tasks of the mobile applications run by the M MUs. Task c_m, $c_m \in \mathcal{C}_m$, is usually defined by a tuple $\langle l_{c_m}, d_{c_m}, \tau_{c_m} \rangle$, where l_{c_m} is the computation workload, indicating the number of CPU cycles, d_{c_m} is the data size, i.e., the number of bits, and τ_{c_m} is the task execution deadline.

[1] Note that we can obtain the information about a mobile application's computation tasks by parsing its profile, which is studied by many previous works [30], [31].

B. Computation Offloading in a UDN

To ease the presentation, we consider a typical but simplified UDN with one conventional high-power macro-cell base station (MBS) and N low-power SBSs (e.g., picocells, microcells, and femtocells), as shown in Fig. 1. Particularly, only one EgSvrs is deployed at an SBS. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of EgSvrs. M MUs, represented by $\mathcal{M} = \{1, 2, \ldots, M\}$, are geographically distributed in this UDN. The MUs' access to the MBS and the SBSs are coordinated by adopting the orthogonal frequency division multiple access (OFDMA) technique. Besides, we also consider employing a channel assignment strategy to allocate available radio channels for the communications between MUs and the MBS/SBSs, in order to avoid the same channel interference.

An EgSvr generally has a limited available computation resource that is a set of VMs. Specifically, we consider that EgSvr n has V_n VMs. Let $\mathcal{V}_n = \{1, 2, \ldots, V_n\}$ denote the set of VMs. Hence, we have $\mathbf{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cdots \cup \mathcal{V}_N$ to represent the available VMs provided by N EgSvrs. VM v_n, $v_n \in \mathcal{V}_n$, is defined by the CPU frequency f_{v_n}. In practice, an EgSvr assigns one VM to a mobile device that offloads a computation task, and a VM can only be allocated to one task at a time.

To enable an MU to run a computation-intensive mobile application efficiently, the MU needs to schedule the place of executing computation tasks, i.e., determining whether a computation task to be executed locally or remotely. More importantly, due to the ultra-dense deployment, an MU may be surrounded by multiple SBSs, and hence can offload computations to multiple corresponding EgSvrs at a time. Thus, the MU needs to further determine which EgSvr to execute a task, i.e., finding the association between a VM belonged to an EgSvr and a task. We define a binary variable x_{c_m,v_n} to indicate whether MU m's task c_m, $\forall c_m \in \mathcal{C}_m$ and $\forall m \in \mathcal{M}$, is assigned to a VM v_n belonged to EgSvr n, $\forall v_n \in \mathcal{V}_n$ and $\forall n \in \mathcal{N}$. As a result, we can have $\sum_{c_m \in \mathbf{T}} x_{c_m,v_n} \leq 1, \forall v_n \in \mathbf{V}$, as a VM can only be associated with one task, and $\sum_{v_n \in \mathcal{V}_n} x_{c_m,v_n} \leq 1, \forall c_m \in \mathbf{T}$, as a task can only be assigned to one VM.

When a task is scheduled to be executed by an EgSvr, the MU needs to transmit the task to the EgSvr via a wireless connection. We consider that the wireless connection follows a Rayleigh fading channel model. The signal-to-interference and noise rate (SINR) of a wireless connection between MU m and SBS n can be obtained by

$$\gamma_{c_m,n} = \frac{p_{c_m,n} g_{m,n}}{I_{m,n} + N_0 B_0}, \tag{1}$$

where $I_{m,n}$ is the co-channel interference of the link from MU m to SBS n, $p_{c_m,n}$ is the transmission power, B_0 is the transmission bandwidth, $g_{m,n}$ is the channel gain of the wireless connection between MU m and SBS n, and N_0 is the power spectrum density of noise. Particularly, to conduct reliable communications, the SINR needs to satisfy

$$\gamma_{c_m,n} \geq S_{\text{th}}, \forall c_m \in \mathbf{T}, n \in \mathcal{N}, \tag{2}$$

where S_{th} is the pre-defined threshold. According to the Shanon theory, the transmission rate is

$$R_{c_m,n} = B_0 \log_2(1 + \gamma_{c_m,n}). \tag{3}$$

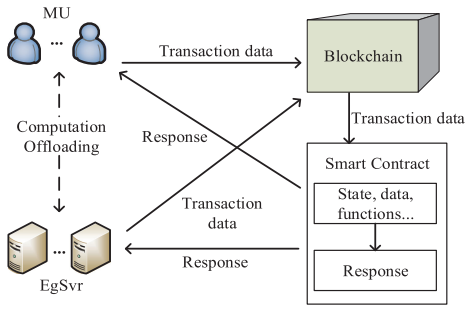

Fig. 2. The blockchain platform for matching MUs and EgSvrs.

IV. Decentralized Coordination on Blockchain for Computation Offloading

In this section, we first present the developed blockchain platform for matching MUs and EgSvrs, then the designed task-VM matching algorithm, and finally the smart matching contract implementation.

A. The Blockchain Platform for Matching MUs and EgSvrs

Our developed blockchain platform features a decentralized matching between MUs and EgSvrs to conduct computation offloading in UDNs, as shown in Fig. 2. Specifically, we use the blockchain, which is backed up by miners, to replace the traditional centralized task assignment platform. EgSvrs are sellers who are willing to sell their available computation resources (i.e., VMs) to MUs for profits, while MUs are buyers who need to buy EgSvrs' available computation resources to conduct remote computations. Particularly, both MUs and EgSvrs communicate with the blockchain platform directly, instead of among themselves. The blockchain platform enables MUs and EgSvrs to make transactions by conducting the task-VM matching. In particular, the matching is implemented by employing smart contracts. In addition, the developed blockchain platform supports two types of actions: on-chain actions that are recorded into the blockchain (i.e., represented by the solid lines in the figure) and off-chain actions that are executed without blockchain (i.e., represented by the dotted lines in the figure).

To enable the matching, EgSvrs and MUs submit to the blockchain platform with their computation provisioning service requests and computation offloading service requests, respectively. A computation provisioning service request includes the description of the number of available VMs and its price, and a computation offloading service request contains the description of all computation tasks owned by the MU and the information of radio channels between the MU and nearby SBSs. The matching is implemented by a smart contract which is actually a program to automate service transactions (i.e., the task-VM matching) by using the data submitted by MUs and EgSvrs to the blockchain. A transaction can generally trigger the execution of predefined functions in a smart contract and change the state of the smart contract (i.e., the state of the matching process). Miners store and execute the predefined functions, and matching results are packed as transactions to be added into the blockchain.

B. Stable Task-VM Matching for Computation Offloading

1) Problem Formulation: To conduct the task-VM matching, tasks and VMs are virtual buyers and sellers, respectively. We formally define task-VM matching as follows.

Definition 1 (A task-VM matching): Given the set of sellers $\mathbf{V}$ and the set of buyers $\mathbf{T}$, a task-VM matching is a mapping function f: $\mathbf{V} \cup \mathbf{T} \rightarrow 2^{\mathbf{T}} \cup 2^{\mathbf{V}}$, such that:

- For $\forall c_m \in \mathbf{T}$, $f(c_m) \in \mathbf{V}$.
- For $\forall v_n \in \mathbf{V}$, $f(v_n) \in \mathbf{T}$.
- For $\forall c_m$ and $\forall v_n$, $f(c_m) = v_n$ if and only if $f(v_n) = c_m$.

However, performing task-VM matching in practical UDNs is subject to some constraints. First, due to reliable wireless transmissions, the SINR of a channel between an MU and an SBS needs to meet its threshold S_{th}. Second, a task can be assigned to at most one VM, rather than multiple VMs. Thus, we can define a feasible task-VM matching as follows.

Definition 2 (A feasible task-VM matching): A task-VM matching f is feasible, if:

- For $\forall c_m \in \mathbf{T}$, each of its matched seller $v_n = f(c_m)$ should satisfy $\gamma_{c_m,v_n} \geq S_{th}$.
- For $\forall c_m \in \mathbf{T}$, there exists at most one matched seller, i.e., $|\{f(c_m)\}| \leq 1$.

2) The Utility Function: We consider the utility as a metric to construct matching preferences of the buyers and sellers. Recall that a task needs to be executed at the expense of energy consumption while satisfying the required task execution deadline. Thus, we consider energy consumption and task execution time in the utility function. The energy consumed for executing task c_m locally can be derived by $e^{\mathrm{l}}_{c_m} = \kappa l_{c_m} f^2_m$ [8], [36], where κ is a coefficient related to the chip architecture and f_m is the frequency clock of MU m. Likewise, the energy consumed for executing task c_m remotely by VM v_n is given by $e^{\mathrm{vm}}_{c_m,v_n} = \kappa l_{c_m} f^2_{v_n}$, where f_{v_n} is the frequency clock of the VM. Moreover, we can find the task computing time $t^{\mathrm{l}}_{c_m}$ and $t^{\mathrm{vm}}_{c_m,v_n}$ by $t^{\mathrm{l}}_{c_m} = l_{c_m}/f_m$ and $t^{\mathrm{vm}}_{c_m,v_n} = l_{c_m}/f_{v_n}$, respectively. When task c_m is executed by VM v_n, the energy is consumed by the offloading transmission, and the extra time is required for the transmission delay. Hence, we can obtain the the amount of energy consumption and the transmission delay by $e^{\mathrm{o}}_{c_m,n} = p_{c_m,n} d_{c_m}/R_{c_m,n}$ and $t^{\mathrm{o}}_{c_m,n} = d_{c_m}/R_{c_m,n}$, respectively.

We consider that the buyers' and sellers' utility function consists of two parts: the immediate cost due to task exeuction time and energy consumption and the immediate revenue due to completing the required computations. Hence, the utility function is defined as

$$\mathrm{U} = \alpha_l \mathrm{R}_l(l) - \alpha_t \mathrm{C}_t(t) - \alpha_e \mathrm{C}_e(e), \tag{4}$$

where α_l, α_t, and α_e ($\alpha_l, \alpha_t, \alpha_e \geq 0$, and $\alpha_l + \alpha_t + \alpha_c = 1$) are coefficients, $\mathrm{R}_l(l)$ is a revenue function that is related to a task's workload l, $\mathrm{C}_t(t)$ is a cost function that depends on the task execution time t, and $\mathrm{C}_e(e)$ is a cost function that is related to the amount of consumed energy e.

Specifically, we have the buyer's utility function of assigning task c_m to VM v_n as follows.

$$\mathrm{U}^{\mathrm{b}}_{c_m,v_n} = -\alpha_t \mathrm{C}_t(t^{\mathrm{o}}_{c_m,v_n}) - \alpha_e \mathrm{C}_e(e^{\mathrm{o}}_{c_m,v_n}), \tag{5}$$

where $\mathrm{C}^{\mathrm{b}}_t(t^{\mathrm{o}}_{c_m,v_n})$ and $\mathrm{C}_e(e^{\mathrm{o}}_{c_m,v_n})$ are the costs received by buyer c_m. It is noteworthy that since no computations are conducted at the buyer side in this case, $\mathrm{R}_l(l_{c_m}) = 0$. $\mathrm{C}_t(t^{\mathrm{o}}_{c_m,v_n})$ is obtained by

$$\mathrm{C}_t(t^{\mathrm{o}}_{c_m,v_n}) = \ln(1 + t^{\mathrm{o}}_{c_m,v_n}) / \ln(1 + t^{\mathrm{mx}}_{c_m}), \tag{6}$$

where $t^{\mathrm{mx}}_{c_m} = \max\{t^{\mathrm{l}}_{c_m}, t^{\mathrm{o}}_{c_m,v_n} | \forall v_n \in \mathcal{A}_{c_m}\}$ (Here, $\mathcal{A}_{c_m}$ is defined as the set of buyer c_m's feasible matching sellers). Likewise, $\mathrm{C}_t(t^{\mathrm{o}}_{c_m,v_n})$ is given by

$$\mathrm{C}_t(t^{\mathrm{o}}_{c_m,v_n}) = \ln(1 + e^{\mathrm{o}}_{c_m}) / \ln(1 + e^{\mathrm{mx}}_{c_m}), \tag{7}$$

where $e^{\mathrm{mx}}_{c_m} = \max\{e^{\mathrm{l}}_{c_m}, e^{\mathrm{o}}_{c_m,v_n} | \forall v_n \in \mathcal{A}_{c_m}\}$.

Similarly, we can have the buyer's utility function of executing task c_m locally as follows.

$$\mathrm{U}^{\mathrm{b}}_{c_m,0} = \alpha_l \mathrm{R}_l(l_{c_m}) - \alpha_t \mathrm{C}_t(t^{\mathrm{l}}_{c_m,0}) - \alpha_e \mathrm{C}_e(e^{\mathrm{l}}_{c_m,0}), \tag{8}$$

where $\mathrm{R}(l^{\mathrm{l}}_{c_m})$, $\mathrm{C}_t(t^{\mathrm{l}}_{c_m,0})$, and $\mathrm{C}_e(e^{\mathrm{l}}_{c_m,0})$ are the received revenue and costs. $\mathrm{R}_l(l_{c_m})$ is defined as

$$\mathrm{R}_l(l_{c_m}) = \ln(1 + l_{c_m}) / \ln(1 + l^{\mathrm{mx}}), \tag{9}$$

where $l^{\mathrm{mx}} = \max\{l_{c_m} | \forall c_m \in \mathcal{B}_{v_n}\}$.

In addition, we can also have the seller's utility function accordingly as

$$\mathrm{U}^{\mathrm{s}}_{c_m,v_n} = \alpha_l \mathrm{R}_l(t_{c_m}) - \alpha_t \mathrm{C}_t(l_{c_m}) - \alpha_e \mathrm{C}_e(e^{\mathrm{vm}}_{c_m,v_n}), \tag{10}$$

where $\mathrm{R}_l(l_{c_m})$, $\mathrm{C}_t(t^{\mathrm{vm}}_{c_m,v_n})$, and $\mathrm{C}_e(e^{\mathrm{vm}}_{c_m,v_n})$ are the revenue and costs, respectively, received by seller v_n. Similarly, we have $\mathrm{C}_t(t^{\mathrm{vm}}_{c_m,v_n}) = \ln(1 + t^{\mathrm{vm}}_{c_m,v_n}) / \ln(1 + \tau^{\mathrm{mx}}_{v_n}), \forall v_n \in \mathbf{V}$, where $\tau^{\mathrm{mx}}_{v_n} = \max\{t^{\mathrm{vm}}_{c_m,v_n} | \forall c_m \in \mathcal{B}_{v_n}\}$ (Here $\mathcal{B}_{v_n}$ is the set of seller v_n's feasible matching buyers). Likewise, we have $\mathrm{C}_e(e^{\mathrm{vm}}_{c_m,v_n}) = \ln(1 + e^{\mathrm{vm}}_{c_m,v_n}) / \ln(1 + e^{\mathrm{mx}}_{v_n})$, where $e^{\mathrm{mx}}_{v_n} = \max\{e^{\mathrm{vm}}_{c_m,v_n} | \forall c_m \in \mathcal{B}_{v_n}\}$.

According to the aggregated transaction information, i.e., task profile, wireless connection conditions, and available computational resources, on the blockchain platform, buyer c_m and seller v_n ($\forall c_m \in \mathbf{T}$ and $\forall v_n \in \mathbf{V}$) can independently construct preference lists, respectively, by sorting the utilities in a descending order. Denote by $\Upsilon_{c_m} = des[\{\mathrm{U}^{\mathrm{b}}_{c_m,v_n}, \mathrm{U}^{\mathrm{b}}_{c_m,0} | \forall v_n \in \mathcal{A}_{c_m}\}]$ and $\Delta_{v_n} = des[\{\mathrm{U}^{\mathrm{s}}_{c_m,v_n} | \forall c_m \in \mathcal{B}_{v_n}\}]$ the descending utility lists of buyer c_m and seller v_n ($\forall c_m \in \mathbf{T}, \forall v_n \in \mathbf{V}$), respectively, where $des[\cdot]$ is to sort the given list in a descending order, $\mathrm{U}^{\mathrm{b}}_{c_m,v_n}$ and $\mathrm{U}^{\mathrm{s}}_{v_n,c_m}$ are the buyer's and seller's utilities between buyer c_m and seller v_n, respectively. The corresponding sellers and buyers in Υ_{c_m} and Δ_{v_n} construct preference lists $\mathcal{P}^{\mathrm{b}}_{c_m}$ and $\mathcal{P}^{\mathrm{s}}_{v_n}$.

3) The Matching Algorithm Design: We base the deferred acceptance algorithm to design a task-VM matching algorithm. In particular, due to the feasibility and stability, we enhance the deferred acceptance matching algorithm to enable the designed task-VM matching algorithm to find a stable matching result. In the following, we describe the designed matching algorithm.

At the initialization step, buyers and sellers choose feasible matching candidates satisfying the conditions, i.e., $\gamma_{c_m,v_n} \geq S_{\mathrm{th}}$ given by Definition 2. Denote by $\mathcal{A}_{c_m}$ and $\mathcal{B}_{v_n}$ the feasible matching candidates of buyer c_m and seller v_n, respectively. Based on both candidate lists, buyers and sellers then construct their preference lists ($\mathcal{P}^{\mathrm{b}}_{c_m}$ and $\mathcal{P}^{\mathrm{s}}_{v_n}$), respectively. Afterwards, they initialize their rejection lists as empty sets. Denote by $\mathcal{R}^{\mathrm{b}}_{c_m}$ and $\mathcal{R}^{\mathrm{s}}_{v_n}$ the rejection lists of buyer c_m and seller v_n, respecitvely. Hence, we have $\mathcal{R}^{\mathrm{b}}_{c_m} = \emptyset$, and $\mathcal{R}^{\mathrm{s}}_{v_n} = \emptyset$, for $\forall c_m \in \mathbf{T}$, $v_n \in \mathbf{V}$.

At the main iteration step, the matching process is conducted. Specifically, $\mathcal{P}^{\mathrm{b}}_{c_m}$, $\mathcal{P}^{\mathrm{s}}_{v_n}$, $\mathcal{R}^{\mathrm{b}}_{c_m}$, $\mathcal{R}^{\mathrm{s}}_{v_n}$, and x_{c_m,v_m}, for $\forall c_m \in \mathbf{T}, \forall v_n \in \mathbf{V}$, are updated at each iteration round. Moreover, buyer $c_m \in \mathbf{T}$ applies to its favorite seller v_{n*} in $\mathcal{P}^{\mathrm{b}}_{c_m}$, i.e., $\mathrm{U}^{\mathrm{b}}_{c_m,v_{n*}} > \mathrm{U}^{\mathrm{b}}_{c_m,v_n}, \forall v_n \in \mathcal{P}^{\mathrm{b}}_{c_m}, v_n \neq v_{n*}$, or determines to execute the task locally if $\mathrm{U}^{\mathrm{b}}_{c_m,0} > \mathrm{U}^{\mathrm{b}}_{c_m,v_n}, \forall v_n \in \mathcal{P}^{\mathrm{b}}_{c_m}$. On the other hand, every seller $v_n \in \mathbf{V}$ receives a set of requests from buyers. These requests are assigned to a waiting queue, which is represented by $\mathcal{W}_{v_n}$. Note that all waiting queues will be initialized to be empty at the beginning of each matching round. Seller v_n accepts buyer c_{m*} if the buyer has the highest utility in $\mathcal{W}_{v_n}$, i.e., $\mathrm{U}^{\mathrm{s}}_{v_n,c_{m*}} > \mathrm{U}^{\mathrm{s}}_{v_n,c_m}, \forall c_m \in \mathcal{W}_{v_n}, c_m \neq c_{m*}$, and rejects the other buyers in $\mathcal{W}_{v_n}$, i.e., $x_{c_{m*},v_n} = 1$. Specifically, seller v_n will remove its rejected buyers from $\mathcal{P}^{\mathrm{s}}_{v_n}$ into $\mathcal{R}^{\mathrm{s}}_{v_n}$. The rejected buyers will also move the sellers who reject them from their preference lists into their rejection lists, i.e., $x_{c_m,v_n} = 0, \forall c_m \in \mathcal{R}^{\mathrm{s}}_{v_n}$. As a result, parts of EgSvrs receive one or more requests at each iteration round. The above process continues until no more buyers are rejected.

We summarize the designed matching algorithm in Algorithm 1.

4) The Stability of the Task-VM Matching: It is of great importance to show whether the task-VM matching is stable or not. To explore the stability of the task-VM matching, we first give two definitions as follows.

Definition 3 (Individual Rationality): A task-VM matching f is individually rational if:

- Every buyer is matched to a seller to obtain the utility which is greater than that being obtained for executing the buyer's task locally, i.e., $\mathrm{U}^{\mathrm{b}}_{c_m,f(c_m)} > \mathrm{U}^{\mathrm{b}}_{c_m,0}$.
- For every buyer c_m, $f(c_m)$ should satisfy the conditions in Definition 2, i.e., $\mathrm{U}^{\mathrm{b}}_{c_m,f(c_m)} = -\infty$ if the conditions in Definition 2 are not met.

For a stable task-VM matching, being individually rational is a requisite condition. It guarantees that a buyer prefers executing tasks by utilizing EgSvrs that can maximize its utility as long as the basic requirements of conducting computation offloading need to be satisfied for ensuring reliable communications.

In addition to the individual rationality, a stable task-VM matching also needs to satisfy another necessary condition called blocking pair, i.e., no blocking pair exists. Let $p \succ_k q$ denote participant k preferring p to q because k can obtain the higher utility for matching with p than that of matching with q. We define the blocking pair as follows.

Algorithm 1: The Designed Task-VM Matching Algorithm

Initialization
(1) Round $t = 0$;
(2) $\mathcal{A}_{c_m}, \mathcal{B}_{v_n}, \forall c_m \in \mathbf{T}, \forall v_n \in \mathbf{V}$;
(3) $\mathcal{P}^{\mathrm{b}}_{c_m}, \mathcal{P}^{\mathrm{s}}_{v_n}, \forall c_m \in \mathbf{T}, \forall v_n \in \mathbf{V}$;
(4) $\mathcal{R}^{\mathrm{b}}_{c_m}, \mathcal{R}^{\mathrm{s}}_{v_n}, \mathcal{W}_{v_n} = \emptyset, \forall c_m \in \mathbf{T}, \forall v_n \in \mathbf{V}$.
while true **do**
  for all $c_m \in \mathbf{T}$ **do**
    Find the first element v_{n*} in $\mathcal{P}^{\mathrm{b}}_{c_m}$, the utility of which is $\mathrm{U}^{\mathrm{b}}_{c_m,v_{n*}}$;
    if $v_{n*} = 0$ **then**
      $x_{c_m,v_n} = 0, \forall v_n \in \mathcal{P}^{\mathrm{b}}_{c_m}$;
    else
      c_m applies to v_{n*}, and v_{n*} moves c_m into $\mathcal{W}_{v_{n*}}$;
    end if
  end for
  for all $v_n \in \mathbf{V}$ **do**
    Find the favorite buyer c_{m*} in $\mathcal{W}_{v_n}$ according to $\mathcal{P}^{\mathrm{s}}_{v_n}$;
    $x_{c_{m*},v_n} = 1$;
    $x_{c_m{}',v_n} = 0, \forall c_m{}' \in \mathcal{W}_{v_n}, c_m{}' \neq c_{m*}$;
    Move the buyers in $\mathcal{W}_{v_n}$ (except for c_{m*}) from $\mathcal{P}^{\mathrm{s}}_{v_n}$ into $\mathcal{R}^{\mathrm{s}}_{v_n}$;
    The rejected buyer $c_m{}' \in \mathcal{R}^{\mathrm{s}}_{v_n}$ puts v_n from $\mathcal{P}^{\mathrm{b}}_{c_m{}'}$ into $\mathcal{R}^{\mathrm{b}}_{c_m{}'}$;
  end for
  if No buyers are rejected at round t **then**
    Break;
  else
    $t \longleftarrow t + 1$;
    $\mathcal{W}_{v_n} = \emptyset, \forall v_n \in \mathbf{V}$;
  end if
end while

Definition 4 (Blocking Pair): For every buyer c_m/seller v_n ($c_m \in \mathbf{T}, v_n \in \mathbf{V}$), which are matched with $f(c_m)/f(v_n)$, there exists a blocking pair if there is a pair of buyer p and seller q satisfying $q \succ_{c_m} f(c_m)$ and $p \succ_{v_n} f(v_n)$.

Particularly, the definition of the blocking pair can be presented mathematically as follows.

$$(\forall c_m \in \mathbf{T}, v_n \in \mathbf{V})\ ((c_m, f(c_m)), (v_n, f(v_n))) \Rightarrow$$
$$(\exists p \in \mathbf{T}, q \in \mathbf{V})\ (q \succ_{c_m} f(c_m)\ and\ p \succ_{v_n} f(v_n)).$$

Blocking pairs can obtain higher utilities. This means the matched participants have a strong willingness to deviate from the matching result, i.e., the matching is not stable.

In the following, we define the stability of the task-VM matching based on the definitions of individual rationality and blocking pairs.

Definition 5 (Stability): A task-VM matching f is stable if it is individually rational and contains no blocking pairs.

As a result, we can arrive at Proposition 1 about the stability of the task-VM matching achieved by Algorithm 1.

Proposition 1: The matching result derived by applying Algorithm 1 is stable.

Proof: To show the stability of the task-VM matching, we first prove that the algorithm is individually rational. Specifically, (5) and (8) are used in the algorithm to compare the utilities received by executing a task remotely and locally, which thus leads to a higher utility received by a matched seller. Moreover, the initialization of Algorithm 1 can also ensure the feasibility of matching results. Therefore, we can conclude that individual rationality is satisfied.

Then, we show the nonexistence of blocking pairs. Assume there is a blocking pair (p, q), $p \in \mathbf{T}, q \in \mathbf{V}$. According to Definition 4, (p, q) satisfies $q \succ_p f(p)$ and $p \succ_q f(q)$. According to Algorithm 1, if buyer p prefers q to $f(p)$, q should have a greater utility than that obtained by $f(p)$. As a result, buyer p applies to q before applying to $f(p)$. However, p fails to be matched with q. The reason is only that q has rejected p, in order to accept another buyer who has a higher utility. This contradicts with $p \succ_q f(q)$. As a result, there is no blocking pair after obtaining the matching result by applying Algorithm 1, which thus concludes the proof. ■

5) Computational Complexity: Algorithm 1 derives the matching result iteratively. Based on the algorithm, we can notice that the number of iterations is highly related to the number of candidates to be matched at the sides of both buyers and sellers. Let M_{c_m} and N_{v_n} denote the number of candidates to be matched at the sides of buyer c_m and seller v_n, respectively.

To analyze the computational complexity, we first consider the best case where every buyer prefers the seller who also prefers the corresponding buyer in the first-matching round. The matching process is completed within the one-round iteration, which takes the computational complexity of $\mathcal{O}(1)$. Then, we consider a worst-case where a buyer is finally matched to a seller after both the buyer and seller search all their candidates (i.e., Υ_{c_m} and Δ_{v_n}), respectively. Particularly, buyers have matching collisions to one another. In this case, the number of iterations at the side of buyers is $\sum_m M_{c_m}$, and $\sum_n N_{v_n}$ at the side of the sellers. Therefore, we can find that the matching process requires the computational complexity of $\mathcal{O}(\sum_m M_{c_m})$ (or $\mathcal{O}(\sum_n N_{v_n})$). From this result, we can conclude that the computational complexity required by Algorithm 1 is in polynomial time, which theoretically shows that our designed matching algorithm can efficiently find a matching result.

C. Smart Matching Contract Implementation

We employ a smart contract-based approach to implement the task-VM matching on the blockchain. The smart contract allows buyers and sellers to define and execute contracts on the blockchain. Specifically, a smart contract is a piece of a computer program that consists of functions (i.e., the executable units of code within a contract) and data (i.e., the states of the smart contract). The functions can only be triggered by specific roles (i.e., sellers or buyers) in the contract, and the data is the information used for performing task-VM matching.

After a smart matching contract is deployed on the blockchain, buyers and sellers can participate in the matching process. The contract will first accept and store action data from the buyers and sellers and then automatically execute the designed task-VM matching algorithm. The detailed operations are summarized as follows.

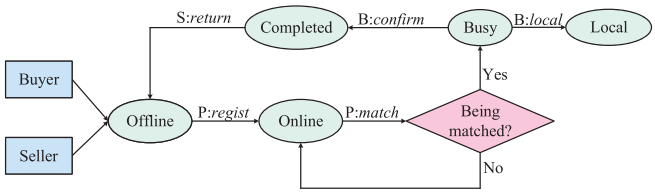

Fig. 3. The state transition diagram of smart matching contract implementation.

- A new smart contract is created on the blockchain.
- The buyers advertise the task information and wireless connection conditions, while the sellers advertise the VM information.
- The smart contract executes the designed matching algorithm to determine a matching result.
- The buyers and sellers confirm the transactions of task-VM matchings.
- The smart contract finalizes the market clearing.

Fig. 3 demonstrates the process of a smart matching contract, involving the buyers' and sellers' state transitions in the contract. Specifically, "B", "S", and "P" in the figure represent a buyer, a seller, and a participant (a buyer or a seller), respectively. A buyer has five states: "Offline", "Online", "Busy", "Completed", and "Local", while a seller has four states: "Offline", "Online", "Busy", and "Completed".

A state transition is triggered by executing a function in the contract. The text format of a function is "R: f", showing that only the role R can trigger the function f. We describe the principal functions as follows.

- $regist$: $regist$ is a function that triggers a buyer or seller who wants to participate in matching to register to the blockchain. The buyer or seller is in the "Offline" state before executing this function and can advertise its matching information, e.g., task profile, wireless connection conditions, and available computational resources. Moreover, to prevent violations, the buyer or seller must pay the refundable deposit once successfully registering to the blockchain, which encourages participants to follow the smart contract's rules faithfully. Since the payment strategy is widely studied by many previous works [37], [38], we ignore the description here. The registered buyer or seller will be transited from the "Offline" state to the "Online" state after executing this function.
- $match$: The function $match$ is automatically executed by smart contract when buyers and sellers are in the "Online" state. It first checks the integrity of the matching information and the deposition and then runs Algorithm 1 to output a matching result. In the meantime, the matched buyers and sellers will be transited from the "Online" state to a "Busy" state after obtaining the matching result, while the unmatched buyers and sellers will remain in the "Online" state.
- $local$ and $confirm$: In the "Busy" state, buyers determine to offload tasks or to execute tasks locally according to the matching result. If the buyers determine to execute tasks locally, the function $local$ is executed to trigger state transitioning from "Busy" to "Local". As a result, the deposit is refunded, and the buyer is removed from the contract. If a buyer is matched with a seller, it sends data to the matched seller who first receives and executes the offloaded computation task to obtain a computing result (off-chain actions). When buyers receive computing results from their corresponding sellers (off-chain actions), they send confirmation messages to the smart contract and pay the payment to their matched sellers on the blockchain (on-chain actions). The sellers also send confirmation messages to the smart contract after receiving payments (on-chain actions). The function $confirm$ is used by the confirmed buyers and sellers to trigger the state transitioning from "Busy" to "Completed" and get deposits and payments back from the contract (on-chain).
- $return$: The function $return$ is executed to directly trigger state transitioning from "Completed" to "Offline", if a seller in the "Completed" state wants to sell its spare computational resources for profits. Then, the seller performs the registration and pays the required deposit again to transit the "Offline" state to the "Online" state.

V. Simulation Results

In this section, we present the performance of the designed matching algorithm by conducting thorough simulations.

A. The Simulation Settings

We evaluate the performance of our designed matching algorithm by conducting extensive simulations. Specifically, the network scenario of the simulations is described in Fig. 1, where MUs are randomly moving within a given area with multiple SBSs and EgSvrs. The deployment of SBSs follows a widely-used Poisson point process [6]. The number of MUs and SBSs are set to be $\{50, 70, 100\}$ and $N = \{50, 70\}$, respectively.

The wireless connections between MUs and SBSs are considered as Rayleigh fading channels. Hence, we adopt the path loss model as $38.46 + 20 \log_{10}(l)$ dB [39], where l is in meter. We set the noise power spectrum density at receivers as -174 $\mathrm{dBm/Hz}$, and the maximum transmission power of MUs as 26 dBm (or 0.398 W) [39]–[41]. The communication bandwidth of a wireless connection is randomly drawn from the range $[10, 20]$ MHz.

For each EgSvr, the number of its available VMs is randomly chosen from $\{4, 6, 8, 10\}$, and the CPU frequency of a VM is set to be a number randomly drawn from the set $\{1.25, 1.5, 1.75, 2\}$ GHz. For each MU, its CPU frequency cycle is set to 1 GHz, and the number of its computation tasks is randomly chosen from $\{5, 10\}$. For each task, the workload is a random number within the range $(0, 1.6]$ Mbs, the corresponding required CPU cycle is set to be a value randomly drawn from $(0, 0.1]$ GHz, and the execution deadline is randomly selected from $[0.3, 1]$ s. The initial weights of utility

TABLE I
THE PARAMETER SETTINGS USED IN THE SIMULATION EXAMPLES

Parameter	Value
N	$\{50, 70\}$
M	$\{50, 70, 100\}$
C_m	randomly chosen from $\{5, 10\}$
V_n	randomly chosen from $\{4, 6, 8, 10\}$
l_{c_m}	randomly chosen from $(0, 0.1]$ GHz
d_{c_m}	randomly chosen from $(0, 1.6]$ Mbs
τ_{c_m}	randomly chosen from $[0.3, 1]$ s
f_m	randomly chosen from $[0.2, 1]$ GHz
f_{v_n}	randomly chosen from $\{1.25, 1.5, 1.75, 2\}$ GHz
κ	1×10^{-26} GHz
$P_m^{\max}$	26 dBm
B_0	$[10, 20]$ MHz
$\{\alpha_l, \alpha_t, \alpha_e\}$	$\{1/3, 1/3, 1/3\}$
Path loss model	$38.46 + 20 \log_{10}(l)$ dB
N_0	-174 dBm/Hz
S_{th}	10 dB

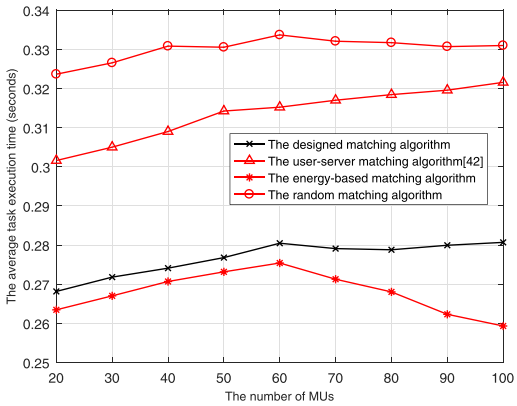

Fig. 4. The comparison of the average task execution time.

functions are set to be $\alpha_l = \alpha_t = \alpha_e = 1/3$. We employ the Monte Carlo simulation to conduct the simulations over 100 times to find the average performance results.

We summarize the detailed parameter settings in Table I.

B. The Performance Improvement of the Designed Matching Algorithm

To validate our designed matching algorithm's efficiency, we show the average task execution time and the average consumed energy and compare them with that using a decentralized user-EgSvr matching algorithm [42], an energy-based matching algorithm, and a random matching algorithm. To analyze the two performance metrics, we change the number of MUs in the simulation example. Besides, we set N to be 50.

Fig. 4 reveals that the designed matching algorithm can significantly decrease the average task execution time. We observe that the average task execution time achieved by the designed matching algorithm is lower than that of the user-server matching algorithm and the random matching algorithm, but higher than that of the energy-based matching algorithm. For example, when the number of MUs in the simulation example is 80, the MU spends about 0.28 s, 0.319 s, 0.268 s, and 0.332 s for executing a task by using the designed matching algorithm, the user-server matching algorithm, the energy-based matching algorithm, and the random matching algorithm, respectively. This is because the designed matching algorithm takes into account both execution time and energy consumption, while others do not. Besides, we can find that the MUs complete the required computations very quickly. In addition, we notice that the average task execution time is approximately linear regarding the number of MUs, which shows that designing a matching algorithm for computation offloading is in dire need.

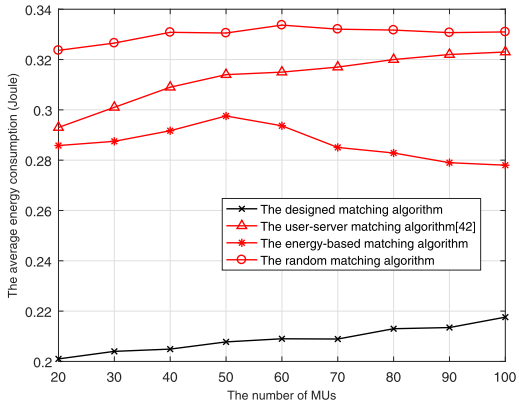

Fig. 5. The comparison of the average execution energy consumption.

Fig. 5 compares the average consumed energy achieved by the designed matching algorithm with that by the decentralized user-EgSvr matching, energy-based matching, and random matching algorithms. We find that the average energy consumption achieved by the designed matching algorithm is the lowest. For example, when the number of MUs in the simulation example is 60, the MU consumes the energy about 0.21 J, 0.315 J, 0.293 J, and 0.336 J for executing a task by using the designed matching algorithm, the user-server matching algorithm, the energy-based matching algorithm, and the random matching algorithm, respectively. The primary reason is that the designed matching algorithm allows MUs to execute tasks locally, which can avoid the case that offloading transmission costs much higher energy consumption. Besides, we observe that the average energy consumption increases with the increase in the number of MUs. This is because more computation tasks compete for the limited computational resources, resulting in a higher probability that some tasks are scheduled to EgSvrs with higher energy consumption.

The performance results shown in these figures demonstrate that the average task execution time and energy consumption can be improved by jointly taking task execution time and energy consumption into account. This implies that we need to balance task execution time and energy consumption when designing a matching algorithm for computation offloading.

C. The Convergence Performance

To evaluate the convergence performance of the designed matching algorithm, we show the matching rate, i.e., the ratio of the number of matched buyers to the number of all buyers, and compare it with different settings of the number of MUs and EgSvrs. In this simulation example, we set $\alpha_t = 0.5$.

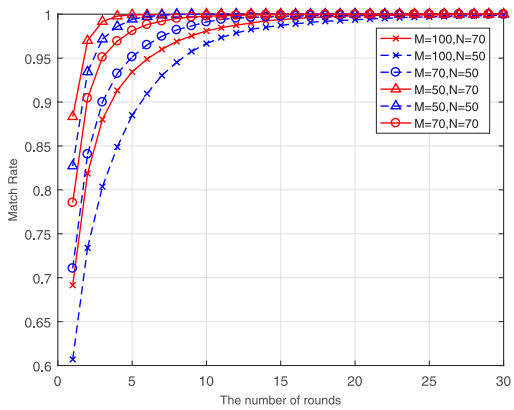

Fig. 6. The convergence of the designed matching algorithm.

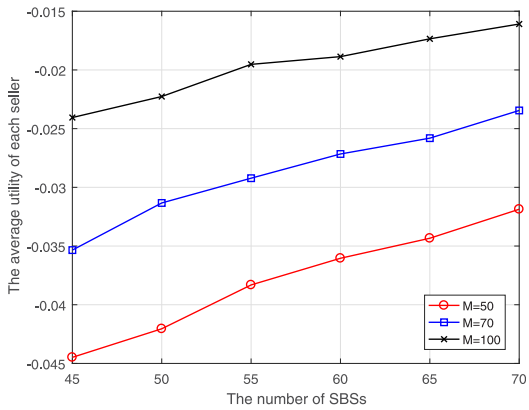

Fig. 8. The average utility received by each seller.

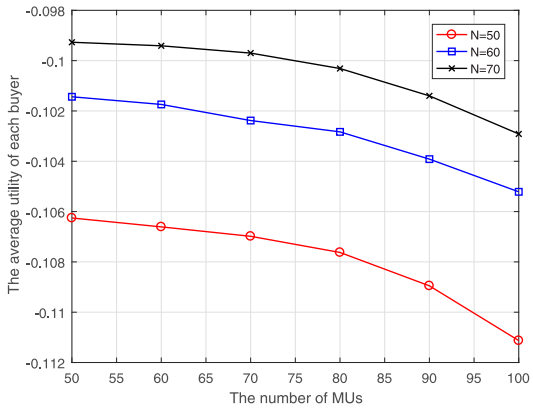

Fig. 7. The average utility received by each buyer.

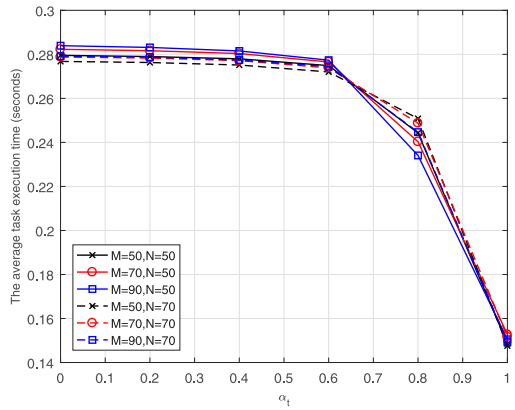

Fig. 9. The impact of α_t on the average task execution time.

Fig. 6 shows the convergence performance of the designed matching algorithm. We observe that the designed matching algorithm can converge very quickly. For example, the number of iteration rounds spent for reaching a stable state is 7 when $M = 50$ and $N = 70$, and 17 when $M = 70$ and $N = 50$. This is consistent with the theoretical analysis of the computational complexity. Additionally, we find that the convergence speed is faster when N/M is higher. For instance, the number of iteration rounds spent for reaching a stable state is 9 when $M = 50$ and $N = 50$, and 23 when $M = 100$ and $N = 50$. This is because more EgSvrs can provide more matching choices for MUs. The performance results imply that the designed matching algorithm is very efficient.

D. The Received Utility

Figs. 7 and 8 show the average utilities received by a buyer and a seller, respectively, when the task-VM matching is stable. Specifically, we compare the average utilities received by buyers and sellers by varying the number of MUs M and EgSvrs N, respectively. From Fig. 7, we observe that a buyer's average utility decreases as the number of MUs increases, while it increases with the increase in the number of EgSvrs. Increasing the number of buyers means that more buyers need to compete for limited resources, resulting in lower opportunities to obtain high utilities. In contrast to the MUs, increasing the number of EgSvrs implies more chances of getting high utilities by the buyers, which leads to higher average utility. From Fig. 8, we find that each seller's average utility increases with the increase in the number of MUs. The reason is that more MUs can provide more matching candidates that can enable sellers to obtain high utilities. Similarly, we can also see the growing utilities when increasing the number of EgSvrs. The reason is the same as that of providing more MUs.

E. Effects of the Utility Functions' Coefficient

To explore the impact of the utility functions' coefficient, we show the change of the average task execution time and the average consumed energy when varying the coefficient α_t from 0 to 1. Note that, as $\alpha_l + \alpha_t + \alpha_e = 1$, the effects of the coefficient α_l and α_e on the performance results can be shown by the figures obtained by changing α_t.

Specifically, Figs. 9 and 10 reveal that the average task execution time and the average energy consumption, respectively, as the coefficient α_t increases. When the coefficient of the utility function is small (e.g., $\alpha_t < 0.6$), the changes in the performance results are minor. The curves then are changing sharply when the coefficient is increasing. Particularly, when the coefficient α_t is larger than 0.8, the curves even change more sharply. This is because when the coefficient is small,

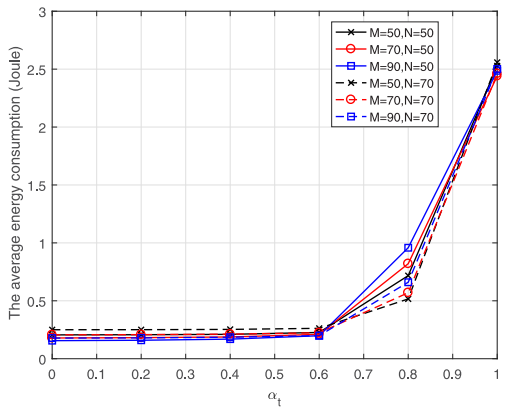

Fig. 10. The impact of α_t on the average energy consumption.

the utility related to the average energy consumption plays the leading role in the received utility. On the contrary, the cost related to the average task execution time plays an important role in the received utility when the coefficient is large. This is also the reason that as we increase the coefficient α_t, the curve of the average task execution time is decreasing, while the curve of the average energy consumption is increasing.

VI. Conclusions

In this paper, we have investigated the coordination problem among MUs and EgSvrs for computation offloading in MEC-enabled UDNs. To address this issue, we have proposed to take advantage of blockchain technology to design a decentralized coordination scheme to orchestrate MUs and EgSvrs for scheduling computation tasks to be offloaded. To schedule MUs' computations to the right EgSvrs, we have jointly considered task execution time and energy consumption to develop an efficient task-VM matching algorithm that has been proved to achieve a stable matching. Besides, we have developed a novel smart matching contract to implement the developed task-VM matching algorithm on the blockchain to perform matching on the blockchain without trusted third parties. To validate the efficacy of the designed matching algorithm, we have conducted extensive and thorough simulations. Simulation results demonstrate that our decentralized coordination scheme can significantly improve the performance and converge to a stable state quickly.

References

[1] C. Wang *et al.*, "Cellular architecture and key technologies for 5 G wireless communication networks," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 122–130, Feb. 2014.
[2] J. Ling and D. Chizhik, "Capacity scaling of indoor pico-cellular networks via reuse," *IEEE Commun. Lett.*, vol. 16, no. 2, pp. 231–233, Feb. 2012.
[3] C. Luo, S. Guo, S. Guo, L. T. Yang, G. Min, and X. Xie, "Green communication in energy renewable wireless mesh networks: Routing, rate control, and power allocation," *IEEE Trans. Paral. Distr. Syst.*, vol. 25, pp. 3211–3220, Dec. 2014.
[4] V. Chandrasekhar, J. G. Andrews, and A. Gatherer, "Femtocell networks: A survey," *IEEE Commun. Mag.*, vol. 46, no. 9, pp. 59–67, Sep. 2008.
[5] C. Galiotto, N. Marchetti, and L. Doyle, "The role of the total transmit power on the linear area spectral efficiency gain of cell-splitting," *IEEE Commun. Lett.*, vol. 17, no. 12, pp. 2256–2259, Dec. 2013.
[6] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: A survey," *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2522–2545, Oct.–Dec. 2016.
[7] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multitask mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.
[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.
[9] Z. Chen *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. IEEE/ACM SEC'17*, Oct. 2017, pp. 1–14.
[10] "Amazon AWS Instances," [Online] Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html.
[11] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5 G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, Sep. 2015.
[12] G. Min, Y. Wu, and A. Y. Al-Dubai, "Performance modelling and analysis of cognitive mesh networks," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1474–1478, Jun. 2012.
[13] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.
[14] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Select. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[15] C. Luo, G. Min, F. R. Yu, M. Chen, L. T. Yang, and V. C. M. Leung, "Energy-efficient distributed relay and power control in cognitive radio cooperative communications," *IEEE J. Select. Areas Commun.*, vol. 31, no. 11, pp. 2442–2452, Nov. 2013.
[16] T. Wang, G. Zhang, A. Liu, Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4831–4843, Jun. 2019.
[17] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving, and secure design," in *Proc. IEEE Bigdata'18*, (Seattle, WA USA), Dec. 2018, pp. 1178–1187.
[18] A. A. Omar, Z. A. Bhuiyan, A. Basu, S. Kiyomoto, and M. S. Rahman, "Privacy-friendly platform for healthcare data in cloud-based on blockchain environment," *Future Gener. Comput. Syst.*, vol. 95, pp. 511–521, Jun. 2019.
[19] S. Seng, X. Li, C. Luo, H. Ji, and H. Zhang, "A D2D-assisted MEC computation offloading in the blockchain-based framework for udns," in *Proc. IEEE ICC'19*, May 2019, pp. 1–6.
[20] K. Kotobi and S. G. Bilen, "Secure blockchains for dynamic spectrum access: A decentralized database in moving cognitive radio networks enhances security and user access," *IEEE Veh. Technol. Mag.*, vol. 13, no. 1, pp. 32–39, Mar. 2018.
[21] S. Raju, S. Boddepalli, S. Gampa, Q. Yan, and J. S. Deogun, "Identity management using blockchain for cognitive cellular networks," in *Proc. IEEE ICC'17*, May 2017, pp. 1–6.
[22] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Netw.*, vol. 32, no. 3, pp. 78–83, May 2018.
[23] D. Lin and Y. Tang, "Blockchain consensus based user access strategies in D2D networks for data-intensive applications," *IEEE Access*, vol. 6, pp. 72683–72690, Nov. 2018.
[24] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, and K. R. Choo, "Blockchain-enabled authentication handover with efficient privacy protection in SDN-based 5 G networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1120–1132, Apr.–Jun. 2021.
[25] S. Guo, X. Hu, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: A distributed and trusted authentication system," *IEEE Trans. Ind. Informatics*, vol. 16, no. 3, pp. 1972–1983, Mar. 2020.
[26] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5 G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, May 2019.
[27] G. Qiao, S. Leng, H. Chai, A. Asadi, and Y. Zhang, "Blockchain empowered resource trading in mobile edge computing and networks," in *Proc. IEEE ICC'19*, May 2019, pp. 1–6.
[28] D. B. Rawat and A. Alshaikhi, "Leveraging distributed blockchain-based scheme for wireless network virtualization with security and QoS constraints," in *Proc. IEEE ICNC'18*, Mar. 2018, pp. 332–336.

[29] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Mathe. Monthly*, vol. 69, no. 1, pp. 9–15, Jan. 1962.
[30] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.
[31] W. Zhang and Y. Wen, "Energy-efficient task execution for application as a general topology in mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 708–719, Jul. 2018.
[32] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Distributed consensus protocols and algorithms." book chapter in *Blockchain for Distributed Systems Security*, Hoboken, NJ, USA: (John) Wiley & (Sons, Inc.), 2019.
[33] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," [Online Available] Bitcoin.–URL: https://bitcoin.org/bitcoin.pdf, 2008.
[34] W. Li, S. Andreina, J. M. Bohli, and G. Karame, *Securing proof-of-stake blockchain protocols*. Springer-Verlag, New York; Berlin, Germany; Vienna, Austria, 2017.
[35] J. Kang, Z. Xiong, D. Niyato, P. Wang, D. Ye, and D. I. Kim, "Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks," *IEEE Wireless Comm. Lett.*, vol. 8, no. 1, pp. 157–160, Feb. 2019.
[36] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.
[37] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao, "A blockchain based witness model for trustworthy cloud service level agreement enforcement," in *Proc. IEEE INFOCOM'19*, Apr. 2019, pp. 1567–1575.
[38] F. Shi, Z. Qin, D. Wu, and J. McCann, "MPCSToken: Smart contract enabled fault-tolerant incentivisation for mobile P2P crowd services," in *Proc. IEEE ICDCS'18*, Jul. 2018, pp. 961–971.
[39] 3GPP, "3GPP TR 36.814 V9.0.0-evolved universal terrestrial radio access (E-UTRA); further advancements for e-utra physical layer aspects," *3rd Generation Partnership Project (3GPP)*, Tech. Rep. (TR) 36.814, Version 9.0.0, Mar. 2010.
[40] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.
[41] N. Nouri, J. Abouei, M. Jaseemuddin, and A. Anpalagan, "Joint access and resource allocation in ultra-dense mmwave noma networks with mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1531–1547, Feb. 2020.
[42] Q. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75868–75885, 2018.

Shuming Seng is currently working toward the Ph.D. degree at the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), China. Prior to this, he received the B.E. degree majoring in telecommunication engineering from Zhengzhou University (ZZU), China, in 2016. He is a Visiting Student with the Department of Computer Science at Virginia Commonwealth University, USA. His current research interests include radio resource management, blockchain, and mobile edge computing in wireless communications and networks.

Changqing Luo (Member, IEEE) is an Assistant Professor with the Department of Computer Science at Virginia Commonwealth University. He received the Ph.D. degrees from Case Western Reserve University and the Beijing University of Posts and Telecommunications in 2018 and 2011, respectively, and the M.E. and B.E. degrees from the Chongqing University of Posts and Telecommunications in 2007 and 2004, respectively. His research interests include cybersecurity, big data, and complex networks.

Xi Li received the B.E. and Ph.D. degrees in communication and information system from the Beijing University of Posts and Telecommunications (BUPT), in 2005 and 2010, respectively. In 2018, she was a Visiting Scholar with The University of British Columbia, Vancouver, BC, Canada. She is currently a Professor with the School of Information and Communication Engineering of BUPT. She has published more than 100 papers in international journals and conferences. Her current research interests include resource management and intelligent networking in next generation networks, the Internet of Things, and cloud computing. She has also served as a TPC Member of IEEE WCNC 2012/2014/ 2015/2016/ 2019/2020, PIMRC 2012/2017/2018/2019/2020, GLOBECOM 2015/2017/ 2018/2019, ICC 2015/2016/2017/2018/2019, Infocom 2018/2019, and CloudCom 2013/2014/2015, the Chair of Special Track on cognitive testbed in CHINACOM 2011, the Workshop Chair of IEEE GreenCom 2019, and a Peer Reviewer of many academic journals.

Heli Zhang received the B.S. degree in communication engineering from Central South University, China, in 2009, and the PH.D. Degree in communication and information System from Beijing University of Posts and Telecommunications (BUPT), China, in 2014. From 2014 to 2018, she was the Lecturer in the School of Information and Communication Engineering at BUPT. From 2018, she has been the associate professor in the School of Information and Communication Engineering at BUPT.

She has been the reviewer for Journals of IEEE Wireless Communications, *IEEE Communication Magazine*, IEEE Transactions on Vehicular Technology, IEEE Communication Letters and IEEE Transactions on Networking. She participated in many National projects funded by National Science and Technology Major Project, National 863 High-tech and National Natural Science Foundation of China, and cooperated with many Corporations in research. Her research interests include heterogeneous networks, long-term evolution/fifth generation and Internet of Things.

Hong Ji (Senior Member, IEEE) received the B.S. degree in communications engineering and the M.S. and Ph.D. degrees in information and communications engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1989, 1992, and 2002, respectively. In 2006, she was a Visiting Scholar with The University of British Columbia, Vancouver, BC, Canada. She is currently a Professor with BUPT. She has authored more than 300 journal/conference papers. Several of her papers had been selected for best paper. Her research interests include wireless networks and mobile systems, including cloud computing, machine learning, intelligent networks, green communications, radio access, ICT applications, system architectures, management algorithms, and performance evaluations.

She has guest-edited *International Journal of Communication Systems, (Wiley)* Special Issue on Mobile Internet: Content, Security and Terminal. She has served as the Co-Chair for Chinacom'11, and a member of the Technical Program Committee of WCNC'19/15/14/12, Globecom'17/16/15/14/13/12/ 11/10, ISCIT'17, CITS'16/15/12, WCSP'15, ICC'20/13/12/11, ICCC'13/12, PIMRC'12/11, IEEE VTC'12S, and Mobi-World'11. She is serving on the Editorial Boards of the IEEE Transactions on Green Communications and Networking, *Wiley International Journal of Communication Systems*.