IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 10, 2020, accepted February 14, 2020, date of publication February 18, 2020, date of current version February 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974750

Smart Contract-Based Long-Term Auction for Mobile Blockchain Computation Offloading

TONGLAI LIU[1,2], JIGANG WU[1], LONG CHEN[1], YALAN WU[1], AND YINAN LI[1]
[1]School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China
[2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Jigang Wu (asjgwucn@outlook.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1003201, in part by the National Natural Science Foundation of China under Grant 61672171, in part by the Guangdong Natural Science Foundation under Grant 2018B030311007, Grant 2018B010107003, and Grant 2019B010121001, in part by the Guangxi Natural Science Foundation of China under Grant 2018GXNSFAA138082, and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201816.

ABSTRACT In a mobile blockchain network, many mobile devices have insufficient computational capacity to execute computation-intensive tasks locally. To tackle this problem, blockchain tasks can be offloaded to edge servers with the aid of auction. However, most auction mechanisms on mobile blockchain ignore the automatic parallel execution and long-term performance. This paper aims to solve the problem of computation offloading in a mobile blockchain network. We transform this problem into a multi-choice multi-dimensional knapsack problem which is NP-hard. To improve the total utility of auction participants, this paper proposes a smart-contract-based double auction mechanism, named long-term auction for mobile blockchain (LAMB). The subtasks can be offloaded from one mobile device to heterogeneous edge servers. Also, LAMB satisfies the economic properties of an auction mechanism. Experimental results demonstrate that, the utility and utilization ratio can be achieved by 130.55% higher and 138.64% higher, respectively, in comparison to the existing auction algorithm WBD. Furthermore, the proposed LAMB can guarantee long-term performance for task offloading, and it can achieve automatic execution in an autonomous and secure environment.

INDEX TERMS Mobile blockchain, smart contract, edge computing, offloading, resource allocation.

I. INTRODUCTION

The global mobile commerce market continues to grow rapidly in recent years. It is estimated that the annual growth rate will be 27% by 2020 [1]. In order to guarantee the reliable and consistent performance of mobile commerce transactions, some trusted centralized authorities are needed to provide computational resources. However, it leads to some issues including a single point of failure and additional fees. In 2008, a new p2p electronic payment system named Bitcoin was developed, which is the successful application of blockchain technologies. It can be widely used to solve the issues of the solution for centralization and double-spending [2]. As an underlying technology of Bitcoin, blockchain is an open, decentralized and tamper-proof ledger that can effectively save transactions among participants in a verifiable manner [3], [4]. Blockchain can facilitate authentication and authorization without dependence on a trusted third party [5]. It has been widely used in various mobile devices, such as vehicles, smartphones, and Internet of Things (IoT) devices.

To achieve the consistency and correctness of transactions among different peer nodes, the consensus mechanism is implemented in blockchain [6]. The pivotal model of the consensus mechanism proposed by Nakamoto is a computation-intensive protocol named Proof-of-Work (PoW) [7]. A blockchain user or miner needs to solve a pre-set PoW puzzle to add new blocks to the blockchain. The process of solving PoW puzzles is defined as mining [8]. It requires miners to generate a value that is less than a dynamic threshold by a hash function. The steps of a typical PoW process are summarized as follows. The consensus node uses an input-attribution function to validate a subset of unidentified transactions and merge them into new blocks. All the nodes calculate the PoW solution for the cipher puzzle constructed by the value of the new block. After the puzzle solution is acquired, a new block is broadcasted over the whole network.

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

Once the new block is successfully verified by most nodes in blockchain, it would be linked to the previous block [9]. Several designs provide rewards to miners, who find the block successfully, to maintain the effectiveness of the proof-of-x type consensus mechanism [2], [8]. However, it is difficult to obtain the puzzle solution, because the solution requires high computational capacities and energy. It brings storage burden to miners when the application data are large [10]. As a result, mobile devices or other resource-limited IoT devices cannot participate in the mining locally. Additionally, it is also difficult for these devices to handle other computation-intensive data processing tasks, especially the tasks of blockchain-based applications for IoT [11].

In order to facilitate blockchain applications in a mobile environment, multi-access edge computing (MEC) is considered as a promising solution for mobile blockchain applications. It provides the computing capabilities for mobile devices using edge servers [12]. Hence, in a mobile blockchain network with edge computing, the mining tasks and other computation-intensive tasks can be offloaded to the edge servers. They are close to mobile devices with edge computing or multi-access edge computing [13]. The more miners involved, the better the robustness of the network [14]. To encourage more miners to join the mining process, some incentive mechanisms are devised in edge computing enabled mobile blockchain network. A resource assignment mechanism based on an auction was presented in [15]. Their model considered allocative externalities due to the competition among miners. Their model can maximize the social welfare of the resource provider with edge computing. In order to supervise the trading market, a broker with sparse information was introduced in the existing work. The authors presented a trading protocol using the double auction scheme for incentive, to optimize the computing resources. Their algorithm also maximized social welfare while the privacies of the buyers/sellers were considered [16]. To stimulate the edge servers to share their resources for the applications in a mobile blockchain network, scholars devised a group-buying mechanism and proposed a three-stage auction to achieve resource assignment for mobile blockchain [17]. Double auction mechanisms were also proposed in previous works to maximize social welfare for the resource assignment in a mobile blockchain network with edge computing. [18]–[20].

In the aforementioned works, they focus on the auction incentive mechanism to optimize maximum social welfare for buyers/sellers while achieving the computation offloading. However, the long-term performance is ignored, and their auction mechanism depends on a trusted third platform. This paper investigates the edge computing enabled mobile blockchain network and design protocols for mobile devices to execute the mobile blockchain applications using the resources of edge servers. First of all, we try to solve the problem of computation offloading for the mobile blockchain tasks, to improve the total utility of buyers and sellers, and to guarantee efficiency and long-term performance. Then, we discuss the automatic execution of the algorithm in an autonomous environment to eliminate any third auction platform. Finally, we need to ensure the trading between sellers and buyers to acquire the desired benefits and protect the private information, such as the capacity, the computing ability, and the computing resource states. The major work of this paper is listed as follows.

- In a mobile blockchain network with edge computing, a double auction architecture is presented, which is based on smart contract without any trusted third auction platform. Moreover, a long-term auction mechanism with an approximation ratio of $1 + \epsilon$ is contributed to encourage edge servers to provide services and facilitate the execution of computation-intensive tasks on mobile devices.
- A platform with MEC is presented for computation offloading in a mobile blockchain network. Three ways of task offloading are implemented to execute applications. This platform can guarantee the computation efficiency and can maximize the total utility of buyers and sellers.
- We theoretically prove that the proposed double auction algorithm can satisfy computation efficiency, individual rationality, budget balance, and truthfulness. Experimental results demonstrate that LAMB can achieve 130.55% higher utility and 138.64% higher utilization ratio, compared with the existing heterogeneous task auction algorithm WBD. Besides, the proposed LAMB guarantees the long-term performance of the double auction.

The remaining structure of this paper is as follows. Section II describes related work. Section III outlines the system architecture, model and problem formulation. Furthermore, it also describes the design of smart contracts. Section IV describes the incentive mechanism. Section V shows the simulation and real-world experiments with computation offloading platforms. Section VI concludes our work.

II. RELATED WORK

A. COMPUTATION OFFLOADING

Some computation offloading works have been presented to facilitate more edge servers to provide services for mobile applications. The authors in [21] proposed an incentive mechanism with a bidding scheme and a resource allocation scheme to solve the mixed integer programming. By means of a group-buying mechanism, the authors in [22] presented a three-stage auction to maximize the social welfare of the whole platform. In their work, cloudlet deployment and resource assignments were combined. The Lyapunov optimization technique was adopted in the online incentive mechanism to obtain minimum provisioning costs of users [23]. The authors in [24] presented a framework mapping and translating ARM vector intrinsics to $\times 86$ vector intrinsics. With the help of this framework, any applications with ARM architecture can be executed directly on

the ×86 architecture. This framework reduces the execution time of compiled code offloading. In order to satisfy the heterogeneous requirements of users in MEC, the authors in [25] presented a combinational auction mechanism. And the authors in [26] presented a two-stage auction mechanism to obtain maximum total utility of the system. Nevertheless, tasks at one mobile device cannot be divided into subtasks in [22], [23], [25], to expedite the execution via offloading. In order to improve energy efficiency of smartphones, a collaborative mobile data offloading scheme was presented on the basis of WiFi [27]. Various offloading mechanisms were presented to boost task offloading efficiency in [28]–[31]. However, their successful deployment in practice depends on whether all buyers and sellers are willing to execute their algorithms in a cooperative way or not, which ignores the selfishness nature of buyers and sellers. Regardless of the total utility, buyers are often willing to obtain the better resources with minimum cost, while sellers are often willing to choose higher bid for their resources.

B. SMART CONTRACT

As described in [2], at the cryptocurrency level, the currency systems of the companies should be guaranteed to be running autonomously without any interventions in the whole trading process. The smart contract can meet this requirement. Once a smart contract is deployed, it can work automatically and autonomously [32].

Firstly developed by Nick Szabo [33], the smart contract is one of the critical components in blockchain. It is a collection of executable codes that runs on the distributed ledger technology platform, driven by the event and state [34], [35]. Ethereum is one of the most widespread decentralized platforms that support smart contracts successfully [36]. Fig. 1 shows the framework of smart contract in Ethereum. Ethereum Virtual Machine (EVM) provides an anonymous and a secure running environment for smart contracts [37].

Nowadays, many applications of smart contracts have been presented. Mccorry *et al.* [38] used a smart contract to construct a decentralized voting protocol without reliance on any trusted parties for the tally calculation and privacy protection. In [39], the authors presented a fair undeniable service provisioning scheme for Industrial Internet of Things (IIoT) scenarios where the blockchain is utilized as a service publisher and an evidence recorder. Smart contracts are used to solve disputes. In [38], a decentralized and self-adjustment voting network was established. Smart contracts were used to store voting rules and other data. The privacy of all voters can be controlled so that the fairness of voting was guaranteed. In particular, for auction, many authors have studied smart contracts. Wu *et al.* [40] proposed a new smart auction contract named CReam on the Ethereum blockchain. CReam replaces the centralized auctioneer, and rational buyers and sellers operate properly and safely without trusted third authorities. However, the process of auction and transaction written in smart contracts gives rise to more fees.

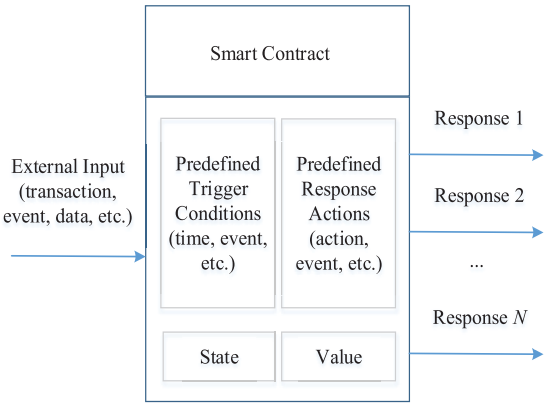


FIGURE 1. Framework of the Ethereum smart contract.

Wang *et al.* [34] proposed a negotiation for adaptive Qos-aware service composition based on smart contracts. Smart contracts can guarantee that the transactions are performed reliably and automatically. Additionally, smart contracts can identify troubled service providers.

Based on the smart contract, the authors of [41] introduced an auction mechanism implementing a Vickrey second price auction for energy transactions in Ethereum. However, there was only one winner among bidders, and bidder collusion may exist. The rules of smart contracts can be executed automatically so that it can reduce disputes. Therefore, it is an important part of the blockchain network. To my knowledge, this paper is the first work to combine the incentive mechanism with the smart contract to achieve decentralized long-term double auction for the computation offloading in an edge computing enabled mobile blockchain network, without reliance on a trusted third party.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. AUCTION ARCHITECTURE BASED ON SMART CONTRACT

The purpose of this paper is to offload the multiple tasks of mobile devices to multiple edge servers by the double auction mechanism based on a smart contract without any trusted third auction platform. Fig. 2 shows the proposed architecture where sellers and buyers can transact with the help of a trusted Ethereum smart contract. Following the agreed rules between sellers and buyers, the smart contract can achieve the automatic execution and guarantee the reliability of transactions. The proposed mechanism is distinct from a traditional e-auction system, composing of seller, buyer and third-platform.

In the proposed architecture, there are three key entities, i.e., mobile device, edge server, and smart contract. Mobile devices and edge servers run Ethereum blockchain, and they communicate with each other through the base station. The base station does not run the blockchain application. Proof-of-Stake (PoS) is used as a consensus mechanism.

The entities are detailed as follows.

- Edge Server: The edge server is the seller in our proposed double auction mechanism. It writes its resource

information and price information into the smart contract and receives the matching result from the auctioneer. Besides, it receives a reward from the buyer after completing the buyer's computation tasks.
- Mobile Device: The mobile device is the buyer in our proposed double auction mechanism. It writes its task information and price information into the smart contract and verifies the smart contract. Besides, it sends the tasks to the seller and pays for a reward to the seller after the tasks are finished.
- Smart Contract: The smart contract is the auctioneer in our proposed double auction mechanism. The smart contract accepts and stores action data from sellers and buyers. It runs algorithm LAMB and returns matching results to sellers and buyers.

All programmable computation in Ethereum needs fees, whose fundamental unit is gas [42]. Gas is consumed during the process that smart contracts are deployed and executed. To some extent, the amount of gas consumed by the processes of smart contracts represents the complexity of the process of the smart contract. The more complex the process of smart contract, the higher the value of gas consumption is [35]. Hence, for the process of smart contracts, the simpler the better. The proposed algorithm LAMB is computational efficiency and it runs in polynomial time that will be proved in the following section.

In the proposed system, a new contract is created on the blockchain in each time period. After the smart contract is created, sellers and buyers can join in the auction. The smart contract can accept and store action data from both of them, and the smart contract will run LAMB automatically. According to the optimal matching results, the transactions between sellers and buyers are initiated, sellers will provide computing resources for buyers, and then the buyers will provide resource coins (such as ether [42]) for sellers.

B. SYSTEM MODEL

Users with unexecuted mining tasks play the part of buyers, and edge servers plays the part of sellers. The smart contract play the part of an auctioneer. The whole assignment time T is divided into multiple time slots (i.e., multiple rounds). Auction is performed in each round. There are N buyers in the mobile blockchain network, denoted as $Y = \{y_1, y_2, \cdots, y_N\}$. The i-th buyer y_i requires resources to implement its computational tasks. Each task can be divided into K independent and heterogeneous subtasks, i.e., $y_i = \{y_{i_1}, y_{i_2}, \cdots, y_{i_K}\}$. The subtasks of one buyer can be executed by different servers. According to the report of Google data centers [43], the arrival intervals among tasks distribute exponentially. Therefore, our assumption is that the arrival process of subtasks follows a Poisson distribution in each round [44]. We divide the arrival process into Λ periods. In a period of time $\lambda \in \Lambda$, the number of resource requests from the i-th buyer is denoted by $l_{i^\lambda}^{(t)}$, where t is the round number. There are M sellers in the mobile blockchain network, denoted by $S = \{s_1, s_2, \cdots, s_M\}$. Let $o_j^{(t)}$ indicate the number of computing resources that the j-th seller s_j owned in the round t. The bid of the j-th seller $s_j \in S$ includes $o_j^{(t)}$. During each round, for the seller s_j, the unit price of its resources is denoted by $p_j^{(t)}$, its workload is denoted by $w_j^{(t)}$, the computation efficiency is denoted by $c_j^{(t)}$, and the data transmission efficiency is denoted by $e_j^{(t)}$ [45]. When the applications of buyer need to be offloaded, the buyer will bid to the auctioneer. During each round, for y_i, let $r_{i_k}^{(t)}$ indicate the resource demands of the k-th subtask, and $v_{ij}^{(t)}$ indicate the valuation of the buyer $y_i \in Y$ for the seller $s_j \in S$. Moreover, because each buyer is constrained by its budget, we utilize B_i to represent the buyer y_i's total budget in all rounds. Meanwhile, the auctioneer gets the bids from sellers. Then, the auctioneer allocates resources.

C. PROBLEM FORMULATION

In a mobile blockchain network, edge computing devices are closer to the mobile devices in a distributed geographical location. These mobile devices can obtain different qualities of services provided by the different edge servers. Besides, in light of sensing capability, CPU, memory, bandwidth, each buyer will make different valuations for different sellers according to their preferences for different computation capability in one round. The valuation is calculated as follows.

$$v_{ij}^{(t)} = a_i^{(t)} + \hat{k}_1 \frac{e_j^{(t)}}{r_i^{(t)}} + \hat{k}_2 \frac{c_j^{(t)}}{w_j^{(t)}} \tag{1}$$

where $a_i^{(t)}$ is the valuation of y_i's task in round t, $\forall t \in T$, $i = 1, 2, \cdots, N, j = 1, 2, \cdots, M$, and $\hat{k}_1, \hat{k}_2$ are two constant coefficients. Let $X^{(t)}$ represent an $M \times N$ assignment matrix in each round. The meaning of $x_{ij}^{(t)}$ is as follows.

$$x_{ij}^{(t)} = \begin{cases} 1, \text{ if } s_j^{(t)} \text{ serves for } y_i^{(t)} \text{ in the round } t, \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Definition 1: The utility of a buyer refers to the valuation minus the payment. The payment includes the demanding resource fee of the buyer, and the gas fee for the smart contract.

The final unit trading price of round t is denoted by $f_{ij}^{(t)}$, which is the unit price to pay for the obtained resources asked by sellers. When the buyer wins the auction, the utility $u_i^{(b)}$ of the buyer y_i can be calculated as follows.

$$u_i^{(b)} = \sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{k=1}^{K} x_{ij}^{(t)} (v_{ij}^{(t)} - r_{i_k}^{(t)} f_{ij}^{(t)}) - \hat{k}_3 \theta_i \tag{3}$$

where θ_i represents the gas fee of y_i, $i = 1, 2, \cdots, N$, and $\hat{k}_3$ is a constant coefficient. Therefore, the objective for the i-th buyer is to maximize its total utility in order to provide an incentive. The objective function is formalized as

$$F_1 = \max \sum_{i=1}^{N} u_i^{(b)} \tag{4}$$

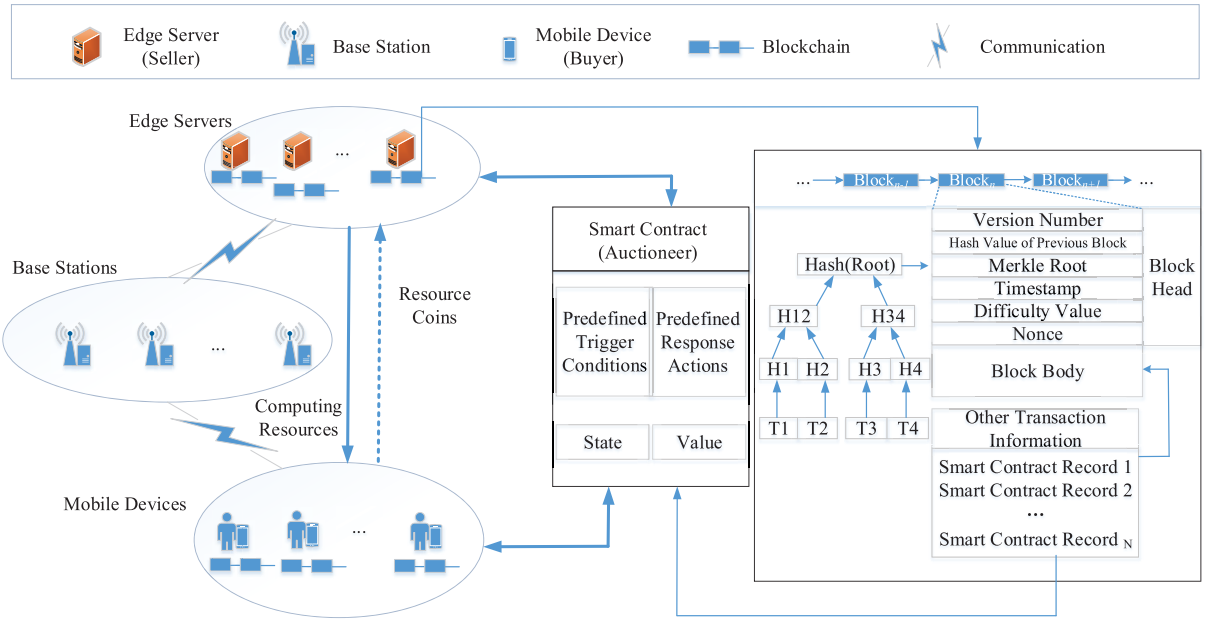


FIGURE 2. Double auction architecture based on smart contract in edge computing enabled mobile blockchain network.

Definition 2: The utility of a seller refers to the payoff minus the cost. The cost includes the providing resource fee of the seller, and gas fee for the smart contract.

And the utility $u_j^{(s)}$ that the seller $s_j \in S$ gains by selling its computing resources can be given as

$$u_j^{(s)} = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{\lambda=1}^{\Lambda} x_{ij}^{(t)} l_{i^\lambda}^{(t)} (f_{ij}^{(t)} - p_j^{(t)}) - \hat{k}_3 \theta_j \tag{5}$$

where θ_j represents the gas fee of s_j and $\hat{k}_3$ is a constant coefficient, $j = 1, 2, \cdots, M$. Thus, the seller's objective is to maximize its utility,

$$F_2 = \max \sum_{j=1}^{M} u_j^{(s)} \tag{6}$$

It is apparent that the above bi-objective problems can be transformed into a single objective problem as follows.

$$\max \quad \alpha F_1 + \beta F_2 \tag{7}$$

where $\alpha + \beta = 1$, α is the weight of the buyer's utility, β is the weight of the seller's utility. The solution is the supported solution under this weight trade-off. If the buyer's utility is of higher concern, we can set $\alpha > \beta$, where $\alpha > 0.5$. In a similar way, if we attach more importance to the seller's utility, we can set $\beta > 0.5$. When the two objectives are of equal importance, $\alpha = \beta = 0.5$. In this paper, we set $\alpha = \beta = 0.5$.

During the process of assignment, the buyers can obtain the resources from the sellers. But because the resources are limited, the number of used resources cannot surpass the number of resources that are provided by sellers. Thus, we can obtain the constraint as follows,

$$\sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ij}^{(t)} r_{ik}^{(t)} < \sum_{t=1}^{T} o_j^{(t)} \tag{8}$$

According to (3), we can know that the final payment to the seller cannot exceed the valuation that the buyer is willing to pay. Thus, we can obtain the constraint as follows,

$$\sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{\lambda=1}^{\Lambda} x_{ij}^{(t)} l_{i^\lambda}^{(t)} f_{ij}^{(t)} < \sum_{t=1}^{T} \sum_{j=1}^{M} x_{ij}^{(t)} v_{ij}^{(t)} \tag{9}$$

And the valuation of a buyer cannot surpass its budget B_i in T. Thus, we can obtain the constraint as follows,

$$\sum_{t=1}^{T} \sum_{j=1}^{M} x_{ij}^{(t)} v_{ij}^{(t)} \leq B_i \tag{10}$$

for $i = 1, 2, \cdots, N$.

Theorem 1: The proposed problem described by (1)-(10) is NP-hard.

Proof 1: It is well known that the multiple-choice multi-dimensional knapsack problem has been already proved to be NP-hard [46]. The objective of this paper is to maximize the $\sum_{i=1}^{N} v_i x_i$. The capacity of the knapsack problem corresponds to the resources that each seller owned in each round. The weight of items in the knapsack problem corresponds to the buyer's requests. Thus, the problem defined by (1)-(10) corresponds to a multiple-choice multi-dimensional knapsack problem. This concludes the theorem.

D. ECONOMIC PROPERTIES

An incentive mechanism is needed to solve the allocation problem as described in subsection C. Meanwhile, the

mechanism should satisfy the economic properties, such as truthfulness, individual rationality, budget balance, and computational efficiency [25], [47].

- Truthfulness: If an auction is regarded as a truthful transaction, the utility of all participants will be maximum for the truthful valuation of the bidder. No buyer y_i or seller s_j could increase his own utility by bidding untruthfully. In other words, for $y_i \in Y$, the true valuation equals to the cost that is required by executing the application itself. And the bidder cannot improve its utility by giving its untruthful valuation.
- Individual rationality: The utility of each winning buyer/seller is non-negative. Each winning buyer pays less than his bid and each winning seller obtains more than his ask. i.e.,

$$u_i^{(b)} > 0 \text{ and } u_j^{(s)} > 0, \text{ for } \forall y_i \in Y \text{ and } \forall s_j \in S$$

  For the entire incentive mechanism, this is the most basic condition for each participant.
- Budget balance: The final fees that buyers paid are greater than or equal to the total charge of sellers. Therefore, the remaining participants need not pay additional surplus and they can guarantee their non-negative utility.
- Computational efficiency: The algorithm terminates in polynomial-time.

E. SMART CONTRACT

Smart contracts are mutually agreed upon the prespecified rules to execute conditions such as 'IF-THEN' mechanism. They can be regarded as cryptographic autonomous boxes that are unlocked only when pre-defined conditions are satisfied. Smart contracts are capable of processing data, operating transactions and managing smart assets [48]. Here, in order to deploy and utilize a smart contract on the blockchain, we can write it by many high-level languages. Mainly three languages (e.g. Solidity, Serpent and LLL) are employed to write smart contracts in Ethereum. Solidity is considered the most popular and stable one [40]. The smart contract code in memory is held by contract accounts which store instructions, and the contract accounts are activated by external accounts or other contract accounts [49].

In an agreement, the contract executes when the time or event is triggered [50]. As shown in Fig. 2, the key events on the blockchain in each time period are given below.

- A new contract is deployed on the blockchain.
- All participants pay the deposits.
- Sellers and buyers submit prices to the auctioneer.
- The auctioneer executes LAMB to determine the optimal matching for sellers and buyers.
- The transaction is initiated. Sellers provides computing resource for buyers.
- After tasks are finished, buyers reward sellers with resource fees.
- All participants can withdraw the deposits.

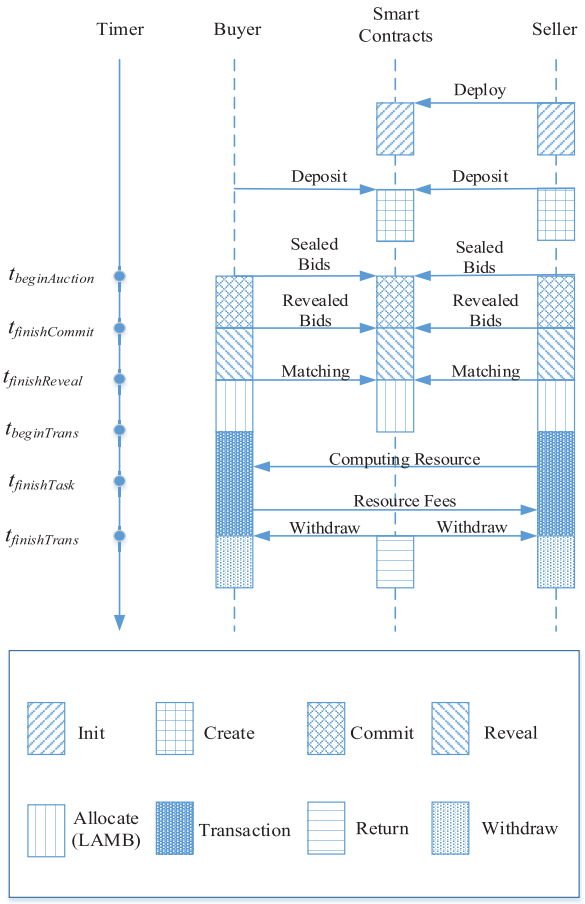


FIGURE 3. Process of a double auction based on smart contract.

In the proposed scheme, the prices from sellers or buyers are sealed. In other words, each one cannot view the prices of others until the prices are revealed. Additionally, due to the security of blockchain, all the trading data are stored on the blockchain. Fig. 3 shows the process of a double auction on the basis of the smart contract. Fig. 4 shows the overview of the functions of the proposed double auction smart contract. The main functions of the entire trading process include *Init*, *Create*, *CommitBid*, *RevealBid*, *Matching*, *Transaction*, *Withdraw*.

- Init: The function *Init* defines some variables used in a new smart contract and sets the initial value. *wl* denotes the seller's workload. f denotes the final unit clearing price. *bBids* denotes the list of bids from a buyer. *sBids* denotes the list of bides from a seller. *bBidders* denotes the list of buyers. *sBidders* denotes the list of sellers. When the time point $t_{beginAuction}$ comes, sellers and buyers can start to commit their bids. $t_{finishCommit}$ indicates the deadline for sellers and buyers to commit their sealed bids. $t_{finishReveal}$ indicates the deadline for sellers and buyers to reveal their sealed bids. $t_{beginTrans}$ is the time that sellers begin to provide services for the buyers. $t_{finishTask}$ indicates the moment the seller has finished the buyer's task. $t_{finishTrans}$ is the time that sellers have

Init: wl=0, f=0, bBids=[], sBids[], bBidders=[], sBidders=[], $t_{beginAuction=0}$, $t_{finishCommit=0}$, $t_{finishReveal=0}$, $t_{beginTrans=0}$

Create: INPUT: ($s.deposit$ $b.deposit$, $t_{beginAuction}$, $t_{finishCommit}$, $t_{finishReveal}$, $t_{beginTrans}$)
Verify $s.deposit$
Verify $b.deposit$

CommitBid: INPUT: (s, $hsbid$, $depost$, $nonce$), (b, $hbbid$, $depost$, $nonce$)
Verify $t_{beginAuction}<t<t_{finishCommit}$
Verify $s \notin sBids$, $b \notin bBids$
Verify $s.deposit \geq deposit$, $b.deposit \geq deposit$
bBids[b]=H(hbbid, nonce)
sBids[s]=H(hsbid, nonce)

RevealBid: INPUT: (s, $s.bid$, $s.nonce$), (b, $b.bid$, $b.nonce$)
Verify $t_{finishCommi}< t <t_{finishReveal}$
Verify $s.deposit$, $b.deposit$
Verify $s \in sBids$, $b \in bBids$
Verify H(s.bid, nonce)==s.hsbid
Verify H(b.bid, nonce)==b.hbbid

Matching: INPUT: (s, $s.bid$, b, $b.bid$)
Verify $t_{finishReveal}< t <t_{beginTrans}$
LAMB(sBidders, sBidders, s.bid, b.bid)

Transaction: INPUT: ()
Verify $t_{finishTask}< t <t_{finishTrans}$
*Send(s, wl*f)*

Withdraw: INPUT: ()
Verify $t>t_{finishTrans}$
Send(s, s.deposit)
Send(b, b.deposit)

FIGURE 4. Double auction smart contract.

completed services for the buyers and the buyers have completed payment.

- Create: The deployment of a new smart contract on the blockchain is executed by one of the sellers calling the function *Create*. Anyone of the participants can create a new smart contract. The address of the smart contract on the blockchain is obtained from this function. The entities can interact with the address in the future. After a smart contract is deployed, all the auction participants can access it. If a seller wants to initialize a new contract, he must call the function *Init*. Any one of the sellers or buyers can start a new smart contract. Hence, in order to prevent a vicious seller or a buyer from initializing mendacious auctions and then withdrawing from them, the seller and the buyer are required to pay $s.deposit$ and $b.deposit$ to the contract and their deposits will be verified.
- CommitBid: When the $t_{beginAuction}$ point comes, sellers and buyers can submit their sealed bids to the smart contract. At this time, they are not allowed to reveal their bids. The data on the blockchain is public, any bidders can observe others' information. Therefore, the sealed bids are needed to protect them from being observed by other sellers or buyers. The *commitBid* accepts the tuples $< b, hbbid, deposit, nonce >$ *and* $< s, hsbid, deposit, nonce >$, where $hbbid = H(b.bid, nonce)$ and $hsbid = H(s.bid, nonce)$, H denotes a trapdoor function (e.g., a cryptographic hash function), and *nonce* denotes a value generated randomly. *hsbid* and *hbbid* will be stored on the blockchain until the $t_{finishCommit}$ comes. The function *CommitBid* also requires that sellers and buyers transfer *deposit* to the smart contract to prevent them from submitting mendacious bids. Sellers and buyers can fully withdraw their deposits.
- RevealBid: After the point $t_{finishCommit}$, all the bidders including the sellers and the buyers must trigger the *RevealBid* to reveal their bids so that the smart contract can execute the matching operation to allocate resources. The *RevealBid* includes the tuple $< bid, nonce >$ with the same bid and nonce used in *CommitBid*. The bids can be verified to guarantee that the revealed bid equals the value in *CommitBid* by checking $hsbid == H(s.bid, nonce)$ and $hbbid == H(b.bid, nonce)$. This phase still verifies whether the deposits are sent to the smart contract or not. The bids with no deposit will be rejected for the security.
- Matching: After all the bids are revealed, the smart contract will run LAMB to determine the results of resource allocation and the clearing prices.
- Transaction: After the execution of LAMB, the matching results will be sent to sellers and buyers. The seller interacts with the corresponding buyer and provides the resource for the buyer. After the tasks are completed, the buyer will pay for the resource fees.
- Withdraw: After the transaction is completed ($t > t_{finishTrans}$), sellers and the buyers can withdraw their deposits. The deposit is designed to improve security. Hence, the deposits will be returned to the bidders.

IV. INCENTIVE MECHANISM

A. ASSIGNMENT ALGORITHM

During the round t, after subtasks arrive, the auctioneer calculates the unit price of buyers' resources. The unit price between y_i and s_j can be calculated as follows.

$$\hat{p}_{ij}^{(t)} = \frac{v_{ij}^{(t)}}{\sum_{\lambda=1}^{\Lambda} l_{i^\lambda}^{(t)}} \tag{11}$$

for $i = 1, 2, \cdots, N, j = 1, 2, \cdots, M$.

After getting each unit resource price $\hat{p}_{ij}^{(t)}$, the auctioneer calculates the difference between the unit price of buyers and the unit price of sellers, i.e., $\hat{p}_{ij}^{(t)} - p_j^{(t)}$. Next, the auctioneer sorts the list of differences in descending order. Next, the auctioneer matches buyers and sellers according to this list. For resource allocation, matching starts if the max difference in the list is positive. In the process of matching, the buyer y_i and the seller s_j with max difference matches first. This match succeeds if the number $r_{i_k}^{(t)}$ of resource requests of the

subtasks y_{i_k} does not exceed the resource number $o_j^{(t)}$, and the payment of the buyer does not surpass its valuation $v_{ij}^{(t)}$. Otherwise, the buyer y_i matches the next seller s_{j+1} in the list. At the same time, if the seller s_j cannot satisfy the current buyer y_i, it matches the next buyer y_{i+1} in the list. When total computing resources are not enough for buyers or all of the buyers' requirements are met, the process of matching exits and the next round auction begins.

When the buyer y_i and the seller s_j match successfully, similar to [51], the final unit clearing price f_{ij} is calculated as

$$f_{ij}^{(t)} = \frac{\hat{p}_{ij}^{(t)} + p_j^{(t)}}{2} \tag{12}$$

From a McAfee auction [52], the proposed algorithm LAMB can be described as in Algorithm 1.

Algorithm 1 LAMB: Long-Term Auction for Mobile Blockchain

Input: The number of rounds T buyers M, sellers N and subtasks K, the bid of buyers $v_{ij}^{(t)}$, $r_i^{(t)}$ and of sellers $o_j^{(t)}$, $p_j^{(t)}$, $c_j^{(t)}$,$e_j^{(t)}$, $w_j^{(t)}$

Output: The charges for buyers G and the payments to sellers P

for $t \leftarrow 1$ to T **do**
  $X_{N*M}^{(t)} \leftarrow \emptyset$;
  for $i \leftarrow 1$ to N **do**
    for $j \leftarrow 1$ to M **do**
      $\hat{p}_{ij}^{(t)} \leftarrow v_{ij}^{(t)} / \sum_{\lambda=1}^{\Lambda} l_{i^\lambda}^{(t)}$
    end for;
  end for;
  $D^{(t)} \leftarrow \emptyset$;
  for $i \leftarrow 1$ to N **do**
    for $j \leftarrow 1$ to M **do**
      $d_{ij}^{(t)} \leftarrow \hat{p}_{ij}^{(t)} - p_j^{(t)}$;
      $D^{(t)} \leftarrow D^{(t)} \cup d_{ij}^{(t)}$
    end for;
  end for;
  Sort $d_{ij}^{(t)}$ in $D^{(t)}$ in descending order;
  if $max(d_{ij}^{(t)}) < 0$ **then**
    exit
  end if;
  $H^{(t)} \leftarrow \emptyset$;
  for $\forall d_{ij}^{(t)} \in D^{(t)}$ **do**
    for $\lambda^{(t)} \leftarrow 1$ to $\Lambda^{(t)}$ **do**
      if $l_{i^\lambda}^{(t)} < o_j^{(t)}$ and $H_i^{(t)} + cost^{(t)} < v_{ij}^{(t)}$ and $B_i > 0$ **then**
        $o_j^{(t)} \leftarrow o_j^{(t)} - r_{i_k}^{(t)}$;
        $D_i^{(t)} \leftarrow D_i^{(t)} \cup cost^{(t)}$;
        $s_{i_k j}^{(t)} \leftarrow r_{i_k}^{(t)}$;
        $X_{ij}^{(t)} \leftarrow 1$;
        $B_i \leftarrow B_i - v_{ij}^{(t)}$
      end if;
    end for;
  end for;
end for;
$P \leftarrow \emptyset$;
for $s_j \in S$ **do**
  $p_j \leftarrow \sum_{t=1}^{T} \sum_{j=1}^{N} \sum_{\lambda=1}^{\Lambda} X_{ij}^{(t)} l_{i^\lambda}^{(t)} f_{ij}^{(t)}$;
  $P \leftarrow P \cup p_j$
end for;
$G \leftarrow \emptyset$
for $y_i \in Y$ **do**;
  $g_i \leftarrow \sum_{t=1}^{T} \sum_{i=1}^{M} \sum_{k=1}^{K} X_{ij}^{(t)} r_{i_k}^{(t)} f_{ij}^{(t)}$;
  $G \leftarrow G \cup g_j$
end for;
return P, G

B. THEORETICAL ANALYSIS

Theorem 2: Algorithm LAMB is truthful in the mobile blockchain network.

Proof 2: For the buyer $y_i \in Y$, it can change its bid by increasing or decreasing its valuation. But the untruthful bid will result in auction failure or utility decrease, i.e.,

$$u_i^{(b)} > max\{0, \tilde{u}_i^{(b)}\} \tag{13}$$

where $\tilde{u}_i^{(b)}$ is the utility of the i-th buyer's untruthful bid, $i = 1, 2, \cdots, N$.

For the seller $s_j \in S$, it can change its bid by increasing or decreasing its unit price of resources. If the bid is smaller than the truthful value, according to (11), the payment for their resources will decrease even lower than the truthful value. Therefore, the utility $u_j^{(s)}$ will decrease. If the bid is higher than the truthful value, the order of $s_j \in S$ in the list of differences will retreat and fail to auction.

$$u_j^{(s)} > max\{0, \tilde{u}_j^{(s)}\} \tag{14}$$

where $\tilde{u}_j^{(s)}$ is the utility of the j-th seller's untruthful bid, $j = 1, 2, \cdots, M$. Therefore, the algorithm LAMB is truthful.

Theorem 3: Algorithm LAMB is individual rational in the mobile blockchain network.

Proof 3: For the buyer $y_i \in Y$, when the match is successful, it needs to pay

$$g_i = \sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{k=1}^{K} X_{ij}^{(t)} r_{i_k}^{(t)} f_{ij}^{(t)} \tag{15}$$

The payment for the seller $s_j \in S$ is

$$p_j = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{\lambda=1}^{\Lambda} X_{ij} l_{i^\lambda}^{(t)} f_{ij}^{(t)} \tag{16}$$

As we know, a match can succeed if $\hat{p}_{ij} - p_j > 0$. Hence, from (11), we obtain

$$p_j^{(t)} \leq f_{ij}^{(t)} \leq \hat{p}_{ij}^{(t)} \tag{17}$$

We know that

$$u_j^{(s)} = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{\lambda=1}^{\Lambda} x_{ij}^{(t)} l_{i^\lambda}^{(t)} (f_{ij}^{(t)} - p_j^{(t)})$$

and

$$u_i^{(b)} = \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{k=1}^{K} x_{ij}^{(t)}(v_{ij}^{(t)} - r_{i_k}^{(t)} f_{ij}^{(t)})$$

where $v_{ij}^{(t)} = r_{i_k}\hat{p}_{ij}^{(t)}$, $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, M$, $t = 1, 2, \cdots, T$. Then according to (17) we can know that utility of both buyers and sellers are no-negative.

Theorem 4: Algorithm LAMB is budget balance in the mobile blockchain network.

Proof 4: When allocation succeeds, we have

$$\sum_{k=1}^{K} r_{i_k}^{(t)} = \sum_{\lambda=1}^{\Lambda} l_{i^\lambda}^{(t)} \tag{18}$$

for $i = 1, 2, \cdots, N$.

After substituting (18) into (15) and (16), we obtain $g_i = p_j$, for $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, M$, which concludes this theorem.

Theorem 5: Algorithm LAMB is computationally efficient in the mobile blockchain network.

Proof 5: During the process of sorting and the process of allocation (line 1 to 30), the time complexity is $O(NMT)$. Process of calculating the payment and charge (line 31 to 40) runs in $O(max\{N, M\})$. Hence, LAMB is computationally efficient.

Theorem 6: The approximation ratio of the solution of LAMB is $(1 + \epsilon)$.

Proof 6: We relax the indicator variable constraint (2) temporarily, $x_{ij}^{(t)} \in [0, 1]$. According to (3), (5), (8) and (9), we can obtain the optimal solution of total utility U_{opt} in round t as

$$U_{opt}^{(t)} = \sum_{i=1}^{N}\sum_{j=1}^{M}(x_{ij}^{(t)} v_{ij}^{(t)} - \sum_{k=1}^{K} s_{i_kj}^{(t)} p_j^{(t)}) \tag{19}$$

where s_{i_kj} is the number of resources that the k-th subtask of the i-th buyer y_i obtained from the j-th seller s_j. In LAMB, some tasks of buyers cannot be executed by the edge server due to the resource constraints, and the U_{opt} is the total utility of both sellers and buyers when all resources of servers are allocated.

It is noteworthy that the resource requests of buyers and the total resources that sellers owned in one round should satisfy

$$\sum_{j=1}^{M} o_j^{(t)} \le \sum_{i=1}^{N}\sum_{k=1}^{K} r_{i_k}^{(t)} \tag{20}$$

Let $\Delta^{(t)}$ be the utility. When tasks are assigned successfully, $\Delta^{(t)}$ is

$$\sum_{\lambda=1}^{\Lambda}(x_{ij}(v_{ij}^{(t)} - r_{i_k}^{(t)} f_{ij}^{(t)}) + l_{i^\lambda}^{(t)}(f_{ij}^{(t)}) - p_j^{(t)})) \tag{21}$$

Let $U_{opt}^{(t)} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K}\Delta + U_{rest}^{(t)}$, where

$$U_{rest}^{(t)} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K}(s_{i_kj}^{(t)} - x_{ij}^{(t)} r_{i_k}^{(t)})(\hat{p}_{ij}^{(t)} - p_j^{(t)}) \tag{22}$$

After substituting (20) and (22) into (19), then

$$\begin{aligned} &U_{opt}^{(t)} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K}(\Delta^{(t)} + (s_{i_kj}^{(t)} - x_{ij}^{(t)} r_{i_k}^{(t)})(\hat{p}_{ij}^{(t)} - p_j^{(t)})) \\ &\le \sum_{i=1}^{N}\sum_{j=1}^{M} o_j^{(t)}(\hat{p}_{ij}^{(t)} - p_j^{(t)}) \\ &\quad + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K}(s_{i_kj}^{(t)} - x_{ij}^{(t)} r_{i_k}^{(t)})(\hat{p}_{ij}^{(t)} - p_j^{(t)}) \le U^{(t)}(1+\epsilon) \end{aligned} \tag{23}$$

where $\epsilon = (\sum_{k=1}^{K}\sum_{i=1}^{N}(\sum_{j=1}^{M} s_{i_kj}^{(t)} - x_{ij}^{(t)} r_{i_k}^{(t)}))/\sum_{j=1}^{M} o_j^{(t)}$.

According to (8) and (9), then

$$\sum_{i=1}^{N}\sum_{k=1}^{K} r_{i_k} \le \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M} s_{i_kj} \le \sum_{j=1}^{M} o_j \tag{24}$$

followed by

$$\sum_{k=1}^{K}\sum_{i=1}^{N}(\sum_{j=1}^{M} s_{i_kj}^{(t)} - x_{ij}^{(t)} r_{i_k}^{(t)}) \le \sum_{j=1}^{M} o_j^{(t)} \tag{25}$$

where $\epsilon \in [0, 1]$. Consequently, the approximation ratio of the solution is $(1 + \epsilon)$.

V. EVALUATION

A. SIMULATION SETUP

Firstly, we assess the performance of our platform. Then, we conduct the long-term performance of the proposed algorithm.

Definition 3: Utility refers to the sum utility of buyers and sellers.

Definition 4: Satisfaction ratio $\bar{r}_s$ is the amount of offloaded subtasks $\bar{n}_p$ divided by the amount of all subtasks $\bar{n}_t$, i.e.,

$$\bar{r}_s = \frac{\bar{n}_p}{\bar{n}_t} \tag{26}$$

Definition 5: Utilization ratio $\bar{r}_u$ represents allocation efficiency. It is the number of used resources $\bar{n}_u$ divided by the number of all resources $\bar{n}_a$, i.e.,

$$\bar{r}_u = \frac{\bar{n}_u}{\bar{n}_a} \tag{27}$$

The criterias used in this paper are (1) *utility*, (2) *satisfaction ratio*, and (3) *utilization ratio*. To assess the performance of LAMB, the following algorithms are employed.

- WBD: This double auction scheme is presented in [53]. Heterogeneous tasks are offloaded to servers in accordance to different bid densities. The higher the bid density is, the earlier the bidder will be matched. With the completion of the matching, the utility is calculated according to the VCG mechanism [25].
- RAND: This scheme shows that all tasks on a mobile device are offloaded to servers randomly. For one mobile device, if the selected server cannot satisfy the task requests, the task will be executed by itself.
- MATCH: The scheme uses KM algorithm [54], tasks on mobile devices are offloaded in accordance to the value of their bids. The higher the value of its bid is, the earlier the bidder will be matched.

Due to the feature of a single round of the aforementioned three algorithms, LAMB is also executed for a single round. In order to verify the long-term performance of LAMB, LAMB is executed for multi-rounds in comparison to the multi-round auction algorithm POEM [55]. The experiments are implemented with Matlab 2018a. All data are the average results for 100 iterations.

Assume that 900 mining tasks are offloaded to 5 edge servers [47]. For the computational resources that consist of CPU memory, battery, etc, their amount is generated randomly between 0 and 5, and the value of their bids is generated randomly between 1 and 5.5. For each edge server, the value of resources is generated randomly between 100 and 1000, and the unit price is selected randomly between 0 and 1.

B. SIMULATION RESULTS

Fig. 5 shows the performance comparisons in terms of satisfaction ratio for $T = 100$. The satisfaction ratio decreases against the increasing number of tasks. With the increasing number of tasks, only some subtasks can be served due to limited resources, resulting in a decrease of the satisfaction ratio. When the number of tasks is small, the satisfaction ratio of POEM is much higher than that of the other algorithms, because POEM offloads the subtasks, instead of the whole task, to tackle the small requests. LAMB outperforms POEM because of the task disintegration in LAMB, although POEM outperforms LAMB under the circumstance there is no task disintegration. Especially, the downward trend is not obvious for the MATCH, RAND and WBD. The reason is that the buyer has run out of its budget in the first few rounds. Thus, the number of winning buyers is small in each round. In other words, the average percentage is low. Averagely, LAMB outperforms POEM by 5.66%, and POEM outperforms LAMB without subtasks by 0.58%. At the same time, LAMB outperforms WBD, MATCH and RAND by 71.22%, 111.86% and 120.24%, respectively, in light of satisfaction ratio.

The comparison results in total utility are exhibited in Fig. 6. It shows the impact of task number on total utility when $T = 100$. The total utility increases with the increasing number of tasks, due to the more executed tasks and subtasks.

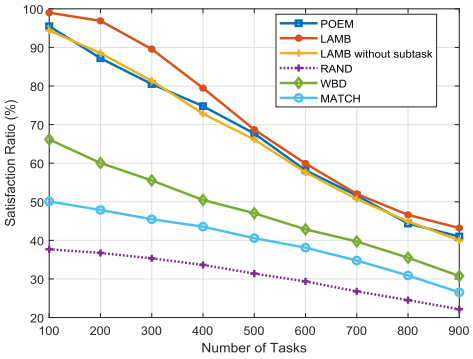


FIGURE 5. Comparisons in satisfaction ratio.

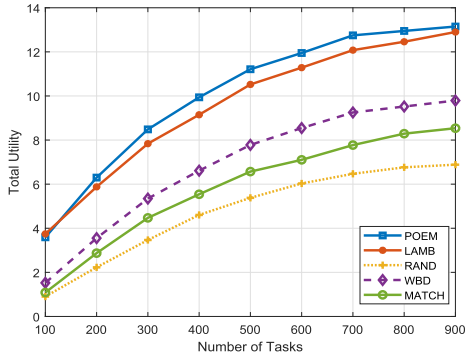


FIGURE 6. Comparisons in total utility.

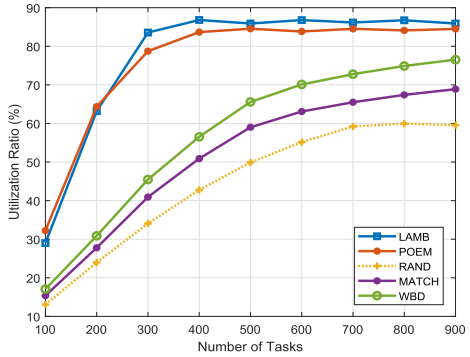


FIGURE 7. Comparisons in utilization ratio.

The total utilities are also improved. Moreover, both the proposed algorithms and compared ones achieve the same increasing speed. LAMB aims to maximize the utility of buyers, although POEM outperforms LAMB after $x = 200$. Note that, the more subtasks are executed, the more subtasks are served by devices with better computational capabilities. POEM can achieve the highest utility. Due to the insufficient resources of servers, total utility will reach a constant value when more tasks participate, as shown in Fig. 6. Averagely, POEM outperforms LAMB, WBD, MATCH and RAND by 5.51%, 36.9%, 72.3% and 122.3%, respectively.

From Fig. 7, the utilization ratio of each algorithm increases with the increasing number of tasks. It means that increasing the number of tasks can significantly improve the

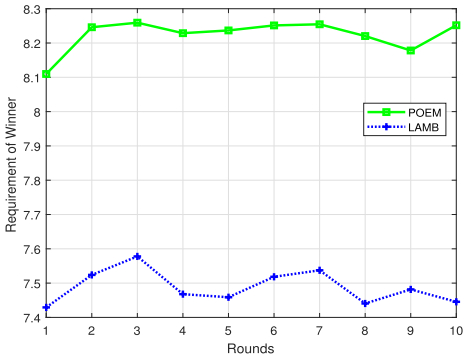


FIGURE 8. Requirement of winner.

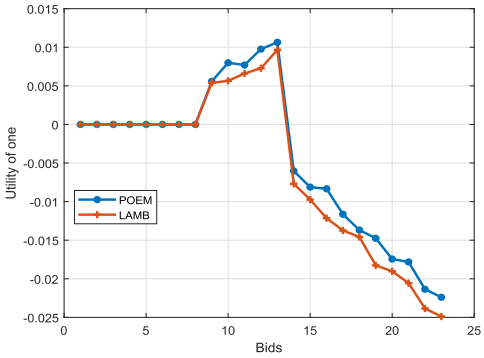


FIGURE 9. Truthfulness.

utilization ratio. When the number of tasks is less than 300, the utilization ratio of POEM and LAMB increases with the increasing number of tasks. Nevertheless, when the number of tasks is larger than 300, utilization ratios of both POEM and LAMB converge to a constant value. There still exist some resources which cannot be used for each task, which are expensive and insufficient. Averagely, LAMB outperforms POEM, WBD, MATCH and RAND by 2%, 41.43%, 81.71% and 82.92%, respectively. Fig. 8 shows the relation between the requirement of the winner and the number of rounds. After the previous rounds, averagely, the resource requests of a round winner in LAMB are smaller than that in POEM, due to the punishment mechanism in POEM.

We also evaluate the truthfulness, as shown in Fig. 9. It describes the utility of the buyer y_i versus its bid. The curve describes the change of utility of two compared schemes with untruthful bids of the i-th buyer from 1 to 23. For each y_i, the truthful bid is 8.3. While the untruthful bid is from 1 to 8.3, the bid is too small so that the transactions cannot successfully match. Therefore, the utility is 0. While the untruthful bid increases from 8.3 to 13.5, the averaged utility increases with the increasing bids. While untruthful bid increases from 13.5 to 23, the payment is more than the cost of y_i, resulting in the negative utility.

C. REAL-WORLD IMPLEMENTATION

For the proposed double auction mechanism, this paper implements a prototype system. The environment is depicted in Fig. 10. The experiments are performed on a Raspgerry Pi,

FIGURE 10. Desktops, raspberry pi and supercomputer.

a supercomputer and two desktops. The client applications are executed in one Raspberry Pi in order to simulate the mobile devices. Each client has the same CPU core as its processor. In Fig. 10, from Boxes 1 and 2, the screen of the computer terminal shows that the mining tasks are running on the host, i.e., the server device (Box 2). The communication between the Raspgerry Pi in Box 3 and the server utilizes the wireless network. The functions of major entities in our real-world experiments are as follows,

- *Client (Buyer)*: When mining applications in *client* need to be executed by other devices, *client* will bid to the auctioneer to get right of using *server*. After that, it will offload the tasks to *server* and wait for a return.
- *Server (Seller)*: The *Server* can bid to the auctioneer according to their computing resources and computational capabilities. When *server* gets applications, it will execute the applications and return the results to the *client* which applications belong to.
- *Smart Contract (Auctioneer)*: *Auctioneer* obtains bids from *servers* and *clients*. Then *Auctioneer* will match the *servers* and *clients* according to the different algorithms, i.e., LAMB, WBD, RAND and MATCH.

The basic steps can be implemented as follows. First, the buyers and the seller submit bids to the auctioneer, then the auctioneer calculates and matches double sides in accordance to their bids. When a buyer matches successfully, its tasks can be executed by the matched seller. In the real-world experiments, we constructed 105 tasks in the Raspgerry Pi and 3 servers in our supercomputer and desktops. The tasks contained 900Mb specific files including all kinds of files. These tasks can be offloaded to three servers. When servers obtain the root hash of the Merkle Tree based on these files, the results will be sent to client applications.

D. REAL-WORLD EXPERIMENT RESULTS

Due to the limited computational capability of our equipments, there is a total of 105 mining tasks to be offloaded in real-world experiments. When the number of mining tasks is larger than 105, our equipment will crash. We investigate the running time of computation offloading. There are three ways to execute the applications,

- Executing locally.
- Offloading part of subtasks.
- Offloading all subtasks.

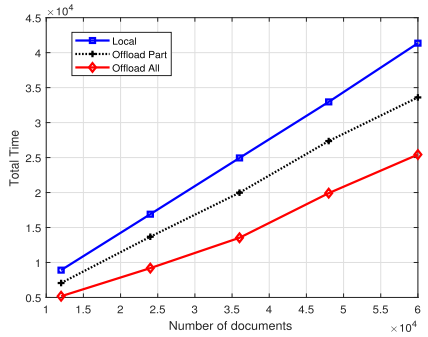


FIGURE 11. Running time.

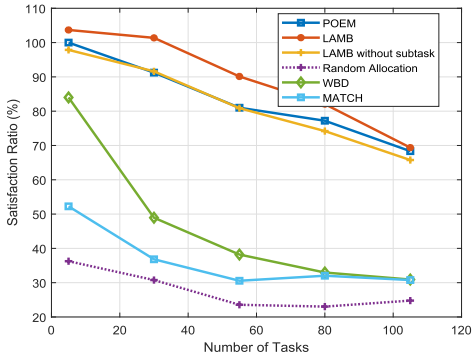


FIGURE 12. Satisfaction ratio of real-world experiments.

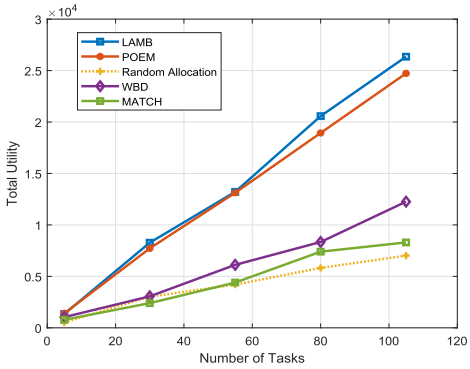


FIGURE 13. Total utility of real-world experiments.

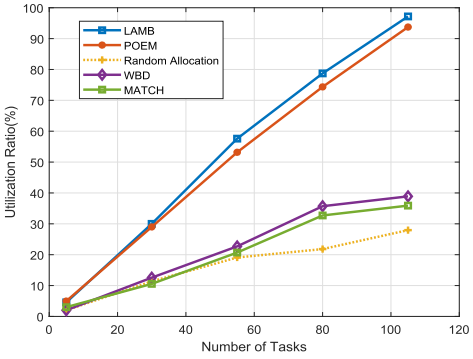


FIGURE 14. Utilization of real-world experiments.

Fig. 11 shows the running time tendency of three ways with the increasing number of documents. Offloading all documents is faster than part-offloading by 27.97%, and than executing locally by 41.44%, since the computational capability of server is much better than the local device.

Similar to Fig. 5, Fig. 12 shows the relation between satisfaction ratio and the number of applications. It is clear that the satisfaction ratio decreases following the increasing number of tasks both in Fig. 5 and in Fig. 12. Only some subtasks can obtain service from the edge server due to the limited resources. The performance of LAMB in real-world experiment is similar to that of the simulation in terms of satisfaction ratio. Averagely, LAMB outperforms POEM, WBD, MATCH and RAND by 13.01%, 77.65%, 128.91% and 202.03%, respectively.

Fig. 13 shows the total utility generated by the real-world experiments. It increases with the increasing number of tasks, similar to the tendency in Fig. 6. With the increasing number of mobile applications which are successfully offloaded, the total utilities of buyers are improved. As a result, the total utility continually increases in Fig. 13. Averagely, LAMB outperforms WBD, MATCH and RAND by 130.55%, 194.8% and 237.89%, respectively. Due to the limitation of computational capability of mobile devices, the number of demanding resources of buyers cannot surpass the number of providing resources of sellers.

Fig. 14 shows the relation between the utilization ratio and the number of tasks. Similar to Fig. 7, the utilization ratio of LAMB increases with the increasing number of tasks. When more applications can be offloaded successfully, more resources of edge servers are assigned to improve the utilization ratio. LAMB offloads the subtasks, instead of the whole task. The utilization ratio of LAMB is higher than that of other algorithms. Averagely, LAMB outperforms WBD, MATCH and RAND by 138.64%, 158.8% and 225.02%, respectively.

VI. CONCLUSION

In this paper, we have proposed a smart contract-based mobile blockchain architecture with edge computing for double auction without any trusted third platforms. Considering the incentive mechanism and computation offloading simultaneously, we have proposed a long-term auction mechanism with an approximation ratio of $1 + \epsilon$. We have built a task offloading environment on the basis of the proposed incentive mechanism so as to obtain the optimal completion time for the tasks. The incentive mechanism has addressed the task scheduling and heterogeneous preference of the computational capability of sellers. By the theoretical analysis, We have demonstrated that the proposed incentive mechanism satisfies properties such as individual rationality, budget balance, truthfulness, and computational efficiency. Our future work will concentrate on load balancing of the tasks under the proposed architecture.

REFERENCES

[1] K. Suankaewmanee, D. T. Hoang, D. Niyato, S. Sawadsitang, P. Wang, and Z. Han, "Performance analysis and application of mobile blockchain," in *Proc. Int. Conf. Comput., Netw. Commun.*, 2018, pp. 642–646.

[2] M. H. U. Rehman, K. Salah, E. Damiani, and D. Svetinovic, "Trust in blockchain cryptocurrency ecosystem," *IEEE Trans. Eng. Manag.*, to be published.

[3] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*, 2016, pp. 279–296.

[4] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Netw.*, vol. 32, no. 6, pp. 184–192, Nov./Dec. 2018.

[5] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Y. Zomaya, "Blockchain for smart communities: Applications, challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 144, pp. 13–48, Oct. 2019.

[6] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," 2017, *arXiv:1710.01567*. [Online]. Available: http://arxiv.org/abs/1710.01567

[8] R. Pass and E. Shi, "FruitChains: A fair blockchain," in *Proc. ACM Symp. Princ. Distrib. Comput. (PODC)*, 2017, pp. 315–324.

[9] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2015, pp. 281–310.

[10] A. Andrews, *Bitcoin: The Complete Guide to Understanding Bitcoin Cryptocurrency and Bitcoin Mining*. Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2018.

[11] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.

[12] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Decentralized resource allocation for video transcoding and delivery in blockchain-based system with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11169–11185, Nov. 2019.

[13] H. Wu, H. Zhang, L. Cui, and X. Wang, "CEPTM: A cross-edge model for diverse personalization service and topic migration in MEC," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–12, Aug. 2018.

[14] Z. Xiong, J. Kang, D. Niyato, P. Wang, and V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Trans. Services Comput.*, to be published.

[15] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[16] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for edge-cloud-assisted Internet of Things," *IEEE Trans Ind. Informat.*, vol. 15, no. 6, pp. 3661–3669, Jun. 2019.

[17] C. Xia, H. Chen, X. Liu, J. Wu, and L. Chen, "ETRA: Efficient three-stage resource allocation auction for mobile blockchain in edge computing," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 701–705.

[18] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[19] C. Chen, J. Wu, H. Lin, W. Chen, and Z. Zheng, "A secure and efficient blockchain-based data trading approach for Internet of vehicles," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9110–9121, Sep. 2019.

[20] Z. Li, Z. Yang, S. Xie, W. Chen, and K. Liu, "Credit-based payments for fast computing resource trading in edge-assisted Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6606–6617, Aug. 2019.

[21] Y. Li, J. Wu, and L. Chen, "Nestle: Incentive mechanism specialized for computation offloading in local edge community," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.* Cham, Switzerland: Springer, 2018, pp. 90–104.

[22] G. Zhou, J. Wu, L. Chen, G. Jiang, and S.-K. Lam, "Efficient three-stage auction schemes for cloudlets deployment in wireless access network," *Wireless Netw.*, vol. 25, no. 6, pp. 3335–3349, Aug. 2019.

[23] L. Lu, J. Yu, Y. Zhu, and M. Li, "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 720–733, Apr. 2018.

[24] J. Shuja, S. Mustafa, R. W. Ahmad, S. A. Madani, A. Gani, and M. Khurram Khan, "Analysis of vector code offloading framework in heterogeneous cloud and edge architectures," *IEEE Access*, vol. 5, pp. 24542–24554, 2017.

[25] L. Chen, J. Wu, X.-X. Zhang, and G. Zhou, "TARCO: Two-stage auction for D2D relay aided computation resource allocation in HetNet," *IEEE Trans. Services Comput.*, to be published.

[26] H. Zhang, F. Guo, H. Ji, and C. Zhu, "Combinational auction-based service provider selection in mobile edge computing networks," *IEEE Access*, vol. 5, pp. 13455–13464, 2017.

[27] A. Y. Ding, B. Han, Y. Xiao, P. Hui, A. Srinivasan, M. Kojo, and S. Tarkoma, "Enabling energy-aware collaborative mobile data offloading for smartphones," in *Proc. IEEE Int. Conf. Sens., Commun. Netw. (SECON)*, Jun. 2013, pp. 487–495.

[28] Z. Zhang, J. Wu, G. Jiang, L. Chen, and S.-K. Lam, "QoE-aware task offloading for time constraint mobile applications," in *Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN)*, Oct. 2017, pp. 510–513.

[29] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[30] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task allocation for mobile cloud computing in heterogeneous wireless networks," in *Proc. 24th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2015, pp. 1–9.

[31] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[32] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "ShadowEth: Private smart contract on public blockchain," *J. Comput. Sci. Technol.*, vol. 33, no. 3, pp. 542–556, May 2018.

[33] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY, J. Transhumanist Thought*, vol. 18, p. 2, 1996.

[34] P. Wang, J. Meng, J. Chen, T. Liu, Y. Zhan, W.-T. Tsai, and Z. Jin, "Smart AC negotiation for adaptive QoS-aware service composition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1403–1420, Jun. 2019.

[35] Z. Li, H. Guo, W. M. Wang, Y. Guan, A. V. Barenji, G. Q. Huang, K. S. McFall, and X. Chen, "A blockchain and autoML approach for open and automated customer service," *IEEE Trans Ind. Informat.*, vol. 15, no. 6, pp. 3642–3651, Jun. 2019.

[36] V. Buterin, "A next-generation smart contract and decentralized application platform," White Paper, 2014, pp. 1–36, vol. 3.

[37] Y. Hirai, "Defining the ethereum virtual machine for interactive theorem provers," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 520–535.

[38] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 357–375.

[39] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, "A blockchain-based nonrepudiation network computing service scheme for industrial IoT," *IEEE Trans Ind. Informat.*, vol. 15, no. 6, pp. 3632–3641, Jun. 2019.

[40] S. Wu, Y. Chen, Q. Wang, M. Li, C. Wang, and X. Luo, "CReam: A smart contract enabled collusion-resistant e-auction," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 7, pp. 1687–1701, Jul. 2019.

[41] A. Hahn, R. Singh, C.-C. Liu, and S. Chen, "Smart AC campus demonstration of decentralized transactive energy auctions," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. (ISGT)*, Apr. 2017, pp. 1–5.

[42] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

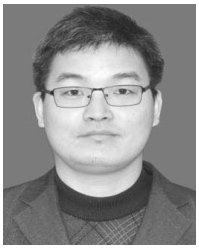
[43] C. Jiang, Y. Chen, Q. Wang, and K. J. R. Liu, "Data-driven stochastic scheduling and dynamic auction in IaaS," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[44] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.

[45] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, pp. 1–4, Jun. 2010.

[46] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in *Knapsack Problems*. Berlin, Germany: Springer, 2004, pp. 235–283.

[47] L. Chen, L. Huang, Z. Sun, H. Guo, and H. Xu, "Spectrum combinatorial double auction for cognitive radio network with ubiquitous network resource providers," *IET Commun.*, vol. 9, no. 17, pp. 2085–2094, Nov. 2015.
[48] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 254–269.
[49] C. Dannen, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Springer, 2017.
[50] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, "Blockchain based smart contract for bidding system," in *Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI)*, Apr. 2018, pp. 208–211.
[51] L. Li, Y.-A. Liu, K.-M. Liu, and Y. Ming, "Pricing in combinatorial double auction-based grid allocation model," *J. China Univ. Posts Telecommun.*, vol. 16, no. 3, pp. 59–65, 2009.
[52] R. P. McAfee, "A dominant strategy double auction," *J. Econ. Theory*, vol. 56, no. 2, pp. 434–450, Apr. 1992.
[53] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[54] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logistics*, vol. 52, no. 1, pp. 7–21, Feb. 2005.
[55] Q. Yu, J. Wu, and L. Chen, "POEM: Pricing longer for edge computing in the device cloud," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.* Cham, Switzerland: Springer, 2018, pp. 355–369.



TONGLAI LIU received the B.E. and M.E. degrees from the Guilin University of Electronic Technology, China, in 2007 and 2010, respectively. He is currently pursuing the Ph.D. degree in computer science and technology with the Guangdong University of Technology, China. He joined the Guilin University of Electronic Technology. His current research interests include blockchain technology, edge computing, networking communication, and database.

JIGANG WU received the Ph.D. degree from the University of Science and Technology of China, in 2000. He was with the Center for High Performance Embedded Systems, Nanyang Technological University, Singapore, from 2000 to 2010. He was a Tianjin Distinguished Professor and the Dean of the School of Computer Science and Software, Tianjin Polytechnic University, China, from 2010 to 2015. He is currently a Distinguished Professor with the Guangdong University of Technology. He has published more than 300 articles in IEEE TC, TPDS, TVLSI, TNNLS, TSMC JPDC, PARCO, JSA, and so on. His research interests include edge computing, high performance computing, and cyber security.

LONG CHEN received the Bachelor of Engineering degree in computer science from Anhui University, Anhui, China, in 2011, and the Ph.D. degree from the School of Computer Science and Technology, USTC, in 2016. He was admitted to the University of Science and Technology of China (USTC) exempted from national graduate school entrance examination. He is currently an Associate Professor with the Guangdong University of Technology. He has published 30 articles in IEEE TVT, TSC, JSA, and so on. His main research interests are cognitive radio networks, network economics, and mobile computing.

YALAN WU received B.Sc. degree from the Guangdong University of Technology (GDUT), China, in 2016, where she is currently pursuing the Ph.D. degree in computer science and technology. She was an Intern Student of Nanyang Technological University for the joint project of high performance architecture, from July 2017 to September 2017. Her research interests include vehicular networks, mobile computing, fault tolerant computing, and high performance architecture. She was awarded the Best Student with the School of Computer Science and Technology, GDUT.

YINAN LI received the B.Sc. degree from the Taiyuan University of Technology, China. He is currently pursuing the master's degree with the School of Computer Science and Technology, Guangdong University of Technology. His current research is on edge computing and blockchain.