Matching with Externalities for Device Assignment in IoT Sharding Blockchain

Jingrou Wu and Jin Zhang(✉)

Southern University of Science and Technology, Shenzhen 518055, China
11960005@mail.sustech.edu.cn, zhangj4@sustech.edu.cn

Abstract. Blockchain technology provides a powerful platform to record and manage Internet-of-Things (IoT) data. To support massive amounts of IoT data, sharding protocols are applied to enlarge the blockchain system scalability and increase efficiency. It divides the IoT devices into several committees (also called shards) so that data of different IoT devices can be processed at the same time in various shards. Random device assignment is the most popular way in IoT sharding blockchain system. Such assignment does not consider the features of different IoT devices and relationship among each IoT device. Hence, the random device assignment might lower the data processing speed. In this paper, we address this issue by modeling the IoT device assignment problem as a many-to-one matching model. Devices are assigned to only one shard while a shard can house many devices. Due to dynamic preference lists of devices, we consider the matching model with externalities. We propose an algorithm to find a stable matching and prove its convergence and stability. The simulation results indicate that the algorithm converges efficiently. Besides, the proposed algorithm has better performance than random assignment.

Keywords: Internet of Things (IoT) · Sharding blockchain · Matching theory

1 Introduction

Blockchain takes an important role in the Internet of Things (IoT) scenario [5] because it is an anonymous, traceable and secure decentralized system. However the low throughput and poor scalability becomes the bottleneck for blockchain systems. For example, Bitcoin [14] can only process 7 transactions per second (tps) and Ethereum [18] can only achieve a low speed of 13 tps [2]. The low efficiency becomes a huge challenge in IoT blockchain applications.

To address the above issue, sharding [11] technique is proposed. It divides the whole blockchain nodes into smaller groups, called committees or shards. Transactions are split into several sets and assigned to different shards. Each shard processes transactions in disjoint sets at parallel, hence increasing the throughput. As more nodes join in the sharding blockchain system, the number of shards

K. Wu et al. (Eds.): ICECI 2021, LNICST 437, pp. 46–60, 2022.
https://doi.org/10.1007/978-3-031-04231-7_4

increases accordingly which improves scalability. With the help of sharding, these new systems such as *OmniLedger* [9], *RapidChain* [19] and *Zilliqa* [1] are able to achieve processing speed thousands times higher than traditional blockchain systems.

In an IoT sharding blockchain system, blockchain nodes are firstly divided into different shards and then IoT devices, i.e., blockchain accounts, are assigned into these shard. IoT would generate data locally or transfer data to other devices. We call the recorded data as transactions in the IoT blockchain system. There are two kinds of transactions, intra-shard transactions whose input and output are in the same shard and inter-shard transactions whose input and output are in different shards. Inter-shard transactions cost more than intra-shard transactions because they require more verification and communications between involved shards. They become the main bottleneck of sharding blockchain efficiency. A sophisticated devices assignment can reduce the amount of inter-shard transactions. However, most sharding blockchain systems just simply use the random devices assignment. The random assignment algorithm does not consider the relationship between accounts. For example, it is better to assign the pair of accounts which share more transactions than others into the same shard to decrease an amount of inter-shard transactions. On the other hand, it does not take preferences of accounts over shards into consideration, leaving no room for devices to select. For example, shard size is an important factor to affect preferences of accounts. A large shard size indicates higher security than a small one when the number of malicious nodes are constant. But the large shard size decreases processing efficiency due to more communications and verification. Some accounts prefer security than efficiency while others think highly of efficiency.

The work [15] consider the transaction placement in a sharding blockchain system. It proposes a graph partition algorithm to solve transactions placement problem. However, it considers the unspent transaction outputs (UTXO) model of the sharding systems which is different from the account model we used in the IoT sharding blockchain system. UTXO model is not suitable for an IoT system because there is no such thing as the unspent transactions in the IoT system.

In this paper, we propose a new account assignment algorithm based on matching theory in IoT sharding blockchain systems which considers the features of the shards and accounts. Matching is applied in a distributed system where participants have preference over others. In our account assignment problem, both accounts and shards have preference over each other. They also want to make their decisions instead of a centralized algorithm because they have private preference information. The nature of account assignment imply that accounts are only able to select one shard but shards can hold many accounts. Besides, accounts' preferences over shards change with other accounts' choices which specify the feature of externalities. Therefore, we model such account assignment problem as a many-to-one matching game with externalities. Our contributions are as follows.

- We model the accounts assignment problem as a many-to-one matching game with externalities. This matching model also consider several constraints from the view of accounts, such as the shard size and the processing efficiency of shards.
- Then we propose an iterative algorithm to find a stable matching in which no one would like to change their actions. We also prove the convergence and stability of the algorithm.
- Our simulation results show that the proposed algorithm reach convergence quickly. Moreover, it performs better than the random assignment algorithm in reducing inter-shard transactions and achieving higher social welfare.

The rest of the paper is organized as follows. We introduce the background knowledge of sharding blockchain systems and fundamentals of matching in Sect. 2. Section 3 presents the system model and problem formulation. In Sect. 4, the algorithm for stable matching and relative proof are given. Experiment results are shown in Sect. 5, followed by related work in Sect. 6. Section 7 summarizes the work.

2 Background

In this section, we describe background knowledge of sharding blockchain and matching theory.

2.1 Sharding Blockchain

The blockchain is a decentralized system where each nodes keep a replica of the state of network. The key idea of sharding is to partition a large amount of objects into smaller subsets. In a sharding blockchain, there are three layers of sharding.

- **Network sharding.** The blockchain nodes are divided into several groups called shards or committees. Nodes would communicate with others to confirm colleagues in the same shard. Each shard is responsible to generate their own block. The block of a particular shard only need verification of nodes in the same shard instead of all participants in the system. Therefore the shard size significantly affects performance. More nodes in the same shard lead to more communication cost, therefore slowing transaction processing speed. On the other hand, a large shard size requires more malicious nodes to initiate valid affect which makes the shard much securer.
- **State sharding.** State sharding is storage partition. That is, nodes in shard A only have information about shard A instead of the whole blockchain. Not all sharding blockchains apply it such as [11]. But state sharding alleviates nodes' storage shortage and therefore improve scalability.
- **Transaction/Account sharding.** Transactions or accounts need partition as well so that they can be processed in different shards parallel. This generates two types of transactions, inter-shard transactions and intra-shard transactions. The inputs and outputs of Inter-shard transactions are in different

shards, while the inputs and outputs of intra-shard transactions are in the same shard. It is simply to process an intra-shard transaction just in the way of blockchain systems without sharding. However, it needs more design to deal with an inter-shard transactions, especially in a state sharding blockchain system. When an inter-shard transaction is sent to the shard A, usually the shard contains the outputs of the transaction, the shard A firstly find the shard B recording inputs of the transaction. Then shard A requires shard B to send verification. Only after receiving verification of shard B, shard A is able to verify and record the transaction. Hence, the inter-shard transaction incurs routing cost and communication overhead. Even worse, there are more than 95% inter-shard transactions [9,19] with the increase of blockchain size.
To further improve sharding blockchain systems efficiency, inter-shard transactions should be reduced. However, most sharding blockchain systems just assign transactions and accounts randomly.

To propose a new assignment algorithm, we need to specify the record-keep model. There are two kinds of record-keep models.

- **UTXO model.** UTXO model is applied in earlier blockchain systems such as BitCoin [14]. Instead of the user's balance, the UTXO model tracks all unspent transactions of the user. The user's balance is the sum of all unspent transactions. Such complex model allows parallel transactions cause the user could use several different UTXO simultaneously. It also provides higher privacy-protection.
- **Account model.** Account model is much simpler. It is similar with bank account which only records the account's balance. Thanks to simplicity, this model is widely used in blockchain systems with smart contract such as Ethereum [18]. On the other hand, it also improve the verification efficiency cause there is no need to follow all UTXO. Blockchain applications [10,13,17] usually use smart contract to implement more functions.

The assignment algorithms are different in this two record-keep models. In the UTXO model, transactions are assigned into shards one by one in streaming. Hence, the assignment algorithm would run at each time when a new transaction is initiated. However, in the account model, old accounts are assigned into different shards together with the process of shard formulation. The algorithm only run when a new account is created which has lower frequency than new transactions. That is, the assignment algorithm in account model has better efficiency than the algorithm in UTXO model.

2.2 Matching Theory

Matching theory is firstly proposed in [7]. It includes single-sided matching and two-sided matching. Single-sided matching studies the mapping relationship from a set to itself. For example, the roommates assignment in schools is a single-sided matching where students match with other students. Two-sided matching focuses on the mapping relationship between two disjoint sets such

as males and females, students and colleges, students and courses. The participants have their own preferences list over potential matching agents, i.e., other participants in single-sided matching models or the members of the opposite set in two-sided matching models. For example, students have preferences over colleges which might base on the colleges' reputation, the major and other factors. Similarly, colleges also rank students in their preferences.

We introduce more details of two-sided matching models following because it is the model we used in this paper. We call the two-sided matching as "matching" for short in following paper. According to the mapping relationships, there are three kinds of matching models.

- **One-to-one Matching.** Each agent of the set is able to match one agent of the other set. Marriage is a typical application of one-to-one matching where one person is supposed to marry another one.
- **Many-to-one Matching.** The members of the one set are allowed to have multiple relationships with agents of the other set while the participants of the other set only matches with the only one agents of the set. The matching between students and colleges is a many-to-one matching where a student can be admitted into only one college while a college has many students.
- **Many-to-Many Matching.** Agents of each set can hold many participants of the other set. For example, a student can choose many courses while the course is open to many students as well.

We could also divide the matching model into two types based on participants' preference lists.

- **Matching without externalities.** In this matching model, participants' preference lists keep same for all time. They would not change their preference lists for any reason.
- **Matching with externalities.** Different from the above model, the preference list of each participant would change due to externalities. For example, a student might be more likely to go to a college when her best friend also applies to it. Hence, the preference list of this student is dynamic over her best friend's choice.

Stable matching takes an important part in matching models. The stable matching means that there does not exists a pair of agents who prefer each other than their current selections. Hence, they would not leave current pair and the matching remains same. The deferred acceptance (DA) algorithm is proposed in [7] to find a stable matching. It is useful for matching without externalities. However, in a matching with externalities model, the DA algorithm is not always able to find a stable matching due to dynamic preference lists. A weaker stable matching, i.e. two-sided exchange stable or swap stable, is defined in [4].

3 System Model and Problem Formulation

We consider an account mode instead of UTXO mode sharding blockchain in our system because account mode is more widely used. We regard the account assignment as a one-shot process at the beginning of each epoch.

3.1 Shards and Accounts Model

The blockchain system is comprised of N shards denoted by $S = \{s_1, s_2, ..., s_N\}$. Each shard s_i has m_i miners. The total transaction capacity of each shard is different, presented by c_i. We assume all miners are rational and they share the same rewards as long as they are colleagues. Therefore, the miners in the identical shard would always make the consistent decision to maximize their rewards. In this way, We can regard the shard as a rational participant instead of a set of miners for simplicity.

We assume M accounts $A = \{a_1, a_2, ..., a_M\}$ need assigned at the time. An account belongs to only one shard while a shard can hold several accounts in an epoch. Each account a_j has n_j transactions to be processed in the following epoch and the average transaction fee is r_j. A graph $G = (V, E, W)$ is used to model the detail information of transactions where node set $V = \{a_1, a_2, ..., a_M\}$ is the set of accounts and the edge set $E = \{(a_{j_1}, a_{j_2})|j_1 \neq j_2\}$ is the set of transactions. The number of transactions between each accounts is denoted by $W = \{w_{j_1,j_2}|j_1 \neq j_2\}$ where w_{j_1,j_2} is the transaction amount between the account j_1 and j_2. For simplify, we assume the graph is an undirected graph which means there are total w_{j_1,j_2} transactions between a_{j_1} and a_{j_2}. The edge (a_{j_1}, a_{j_2}) and (a_{j_2}, a_{j_1}) are identical and $w_{j_1,j_2} = w_{j_2,j_1}$. We also define the set of accounts who have transactions with the account a_j as $T(a_j) = \{a_k|w_{j,k} > 0\}$. Then the relative accounts in the same shard of the account a_j is $T(a_j, s_i) = \{a_k|w_{jk} > 0 \, and \, a_k \in s_i\}$ where $a_k \in s_i$ means the account a_k is assigned to the shard s_i. So the inter-shard relative accounts of the account a_j are in the set $T(a_j) \backslash T(a_j, s_i)$.

3.2 Problem Formulation

Given the shards and accounts model, we formulate the problem as an optimal accounts assignment. We introduce the constraints of the system and give the formulation detail in this subsection.

Accounts Preference. Accounts rank the shard from three aspects as follows.

- **Shard size.** Each account a_j has the minimum and maximum number of the shard's miners, denoted by $m_{j_{min}}$ and $m_{j_{max}}$ because the shard size affects the consensus efficiency and security. If the shard size is not in this range, the account would not consider the shard. The preference of the account a_j on the shard s_i size, denoted by $\pi_s^{a_j}(m_i)$.
- **Transaction processing ability.** Transaction processing ability is relative with the remain transaction capacity. Accounts are more willing to join in a shard with sufficient room for transactions because it is more likely to have transactions serviced. The effect of remain capacity $\pi_c^{a_j}(c_i)$ is an increasing function with the remain capacity c_i of the shard s_i.
- **Neighbour accounts.** Accounts prefer intra-shard transactions than inter-shard transactions and therefore, they would like to stay in a shard with more relative accounts. We use $\pi_n^{a_j}(T(a_j, s_i))$ to describe such preference.

To give a shard an overall rank, the account a_j also has weight parameters $\omega_{1j}, \omega_{2j}, \omega_{3j}$ to combine these three preferences together. Accounts select shards by overall rank consisting of the shard's size, transaction processing speed and accounts in the same shard. Different accounts have different thoughts on these aspects as well. Hence, the utility of the account a_j to choose the shard s_i is defined as follows:

$$\begin{aligned} u_{a_j}(s_i) &= \omega_{1_j}\pi_s^{a_j}(m_i) + \omega_{2_j}\pi_p^{a_j}(c_i) + \omega_{3_j}\pi_n^{a_j}(T(a_j, s_i)) \\ &= \omega_{1_j}(m_i - x_j)^2 + \omega_{2_j}c_i \\ &+ \omega_{3_j}\sum_{a_k \in T(a_j, s_i)} w_{a_j, a_k}. \end{aligned} \tag{1}$$

The weights $\omega_{1_j}, \omega_{2_j}, \omega_{3_j}$ can be different for various accounts.

Shards Preference. Shards only consider the benefit brought by the account. The utility is composed of transaction fees $R(A_i) = \sum_{a_j \in A_i} n_j r_j$ and processing cost $C(A_i)$ where A_i is the account set of the shard s_i. The cost of dealing with these transactions is divided into two parts, the intra-shard cost and the inter-shard cost. We define the cost of each intra-shard transaction as c_i and the cost of each inter-shard transaction as c_c. As mentioned above, intra-shard transactions cost less than inter-shard transactions. Hence we have $c_i < c_c$. The total cost of the set A_i is the sum of all intra-shard transaction cost and inter-shard transaction cost,

$$\begin{aligned} C(A_i) = &\frac{1}{2}\sum_{a_j \in A_i}\sum_{a_{k_1} \in V(a_j, s_i)} c_i w_{jk_1} \\ &+ \frac{1}{2}\sum_{a_j \in A_i}\sum_{a_{k_2} \in \bigcup_{k \neq i} V(a_j, s_k)} c_c w_{jk_2}. \end{aligned} \tag{2}$$

Therefore the utility of the shard s_i containing the set of accounts A_i is

$$u_{s_j}(A_i) = R(A_i) - C(S_i). \tag{3}$$

Accounts Assignment Problem. Given the preferences of accounts and shards, we aim to maximize shards utility with accounts requirements as follows:

$$\max_{A_i} \quad u_{s_j}(A_i) = R(A_i) - C(S_i) \tag{4a}$$

$$s.t. \quad x_{j_{min}} \leq m_i \leq x_{j_{max}}, \forall a_j \in S_i \tag{4b}$$

$$v_j \leq v_i(S_i), \forall a_j \in S_i \tag{4c}$$

where constraint 4b means that the account a_j is only willing to choose a shard with proper number of miners and constraint 4c indicates that the process speed of the shard i must satisfies all accounts' requirements.

4 Many-to-One Matching for Accounts Assignment

In this section, we firstly give the formal definition of many-to-one matching for accounts assignment and then propose an algorithm to solve this matching problem.

4.1 Accounts Assignment Matching Model

We apply the many-to-one matching model in the shards and accounts where a shard holds many accounts and an account belongs to the only one shard. The formal definition is given as follows.

Definition 1 (Many-to-one matching). *Given the set of accounts* $A = \{a_1, a_2, ..., a_M\}$ *and the set of shards* $S = \{s_1, s_2, ..., s_N\}$*, the many-to-one matching* μ *is a mapping:* $A \cup S \cup \{\emptyset\} \rightarrow 2^A \cup (S \times 2^A \cup \{\emptyset, \emptyset\})$*, satisfying following conditions:*

1. $\mu(a_j) \in S \times A_{a_j} \cup \{\emptyset, \emptyset\}$*, for all* $a_j \in A$ *where* A_{a_j} *is the subset of* A *containing* a_j*.*
2. $\mu(s_i) \in 2^A$*, for all* $s_i \in S$*.*
3. $\mu(\emptyset) = \emptyset$*.*
4. *if* $a_j \in \mu(s_i)$*, then* $\mu(a_j) = (s_i, \mu(s_i))$*.*
5. *if* $\mu(a_j) = (s_i, A^{'})$*, then* $\mu(s_i) = A^{'}$*.*

The first three conditions indicate the matching results of accounts and shards. The matching result of an account is a combination of the selected shard and neighbouring accounts, i.e., accounts in the same shard, as the condition 1 shows. It also means that the account is assigned to only one shard. The special case here is $\mu(a_j) = \{\emptyset, \emptyset\}$ which means that the account a_j is not matched to any shard. The condition 2 describes the matching result of a shard where the shard contains a subset of accounts. As for the empty set, the condition 3 define the mapping result as the empty set as shows. The last two conditions 4 and 5 specify the properties of matching. If an account is assigned to a shard, then the shard must contain this account and vice versa. For simplification, we use $\mu(a_j) = s$ where $a_j \in A$ and $s \in S \cup \{\emptyset\}$ to represent $\mu(a_j) = (s, \mu(s))$.

In a matching model, preferences take an important role. Both shards and accounts make decisions based on their preferences. The utilities measure the preferences of shards and accounts as the Eq. (4a) and Eq. (1) show. We define the reflexive, transitive and complete binary relationship $\succ$ between preference relationship. The preference over accounts of the shard s_i is denoted by $\succ_{s_i}$ and similarly, the preference over shards of the account a_j is presented by $\succ_{a_j}$.

Definition 2 (Preference over accounts of shards). *For all* $A_{a_i}, A_{a_i'}(a_i \neq a_{i'}) \in 2^A$ *and* $s_i \in S$*,* $A_i \succ_{s_i} A_i^{'}$ *if and only if* $u_{s_i}(A_{a_i}) > u_{s_i}(A_{a_{i'}})$*.*

Definition 3 (Preference over shards of accounts). *For all* $s_i, s_{i'}(s_i \neq s_{i'}) \in S$ *and* $a_j \in A$*,* $(s_i, \mu(s_i)) \succ_{a_j} (s_{i'}, \mu(s_{i'}))$ *if and only if* $u_{a_j}(s_i) > u_{a_j}(s_{i'})$*.*

Shards and accounts are able to establish the preference list by ranking the utility. Shards do not consider matching results of other shards while accounts care about their neighbouring accounts in the same shard. Hence, it becomes a many-to-one matching model with externalities. That is, the preference lists of accounts are dynamic which changes with other accounts states.

Due to externalities, the traditional stable matching is hard to find. Instead, the concept of two-sided exchange-stable [4] is applied in such matching model. The two-sided exchange-stable matching bases on the concept of swap matching.

Definition 4 (Swap matching). *We define two kinds of swap matching as follows:*

- ***Two-sided exchange.*** *Given the matching μ, for any $a_j, a_{j'} \in A$ with $s_i = \mu(a_j) \neq s_i^{'} = \mu(a_{j'})$, the swap matching $\mu_{a_j}^{a_{j'}}$ of the original matching μ with two accounts $a_j, a_{j'}$ is defined as $\mu_{a_j}^{a_{j'}} = \{\mu \backslash \{(a_j, s_i), (a_{j'}, s_i^{'})\}\} \cup \{(a_j, s_i^{'}), (a_{j'}, s_i)\}$.*
- ***Account-vacancy exchange.*** *Given the original matching μ and any account $a_j \in A$, we have $\mu(a_j) = s_i$. For any $s \in S, s \neq s_i$, the swap matching $\mu_{a_j}^{\emptyset}$ is defined as $\mu_{a_j}^{s} = \{\mu \backslash \{a_j, s_i\}\} \cup \{a_j, s\}$.*

Two-sided exchange defines the exchange between two accounts. In this type of swap, only two accounts in different shards exchange their positions while other accounts stay in the original shards. The second kind of swap allows an exchange between an account and an available vacancy in other shards.

Accounts and shards are not always glad to swap. We define the acceptable swap operation as swap-blocking pair.

Definition 5 (Swap-blocking pair). *According to different types of swap matching, we define two swap-blocking pair as follows.*

- ***Two-sided pair.*** *In a matching μ, the swap-blocking pair $(a_j, a_{j'})$ for any $a_j, a_{j'} \in A$ with $s_i = \mu(a_j) \neq s_{i'} = \mu(a_{j'})$ satisfies following three conditions.*
  1. $u_a^{\mu_{a_j}^{a_{j'}}} \geq u_a^{\mu}$, *for both accounts* $a \in \{a_j, a_{j'}\}$.
  2. $u_{s_i}^{\mu_{a_j}^{a_{j'}}} + u_{s_{i'}}^{\mu_{a_j}^{a_{j'}}} \geq u_{s_i}^{\mu}$.
  3. *Either one of two inequalities is strict.*

  where u_x^{μ} is the utility of the player $x \in A \cup S$ in the matching μ.
- ***Account-vacancy pair.*** *In a matching μ, the swap-blocking pair (a_j, s) for any $a_j \in A$ with $s_i \neq \mu(a_j) \neq s$ satisfies following three conditions.*
  1. $u_{a_j}^{\mu_{a_j}^{s}} \geq u_{a_j}^{\mu}$.
  2. $u_{s_i}^{\mu_{a_j}^{s}} + u_s^{\mu_{a_j}^{s}} \geq u_{s_i}^{\mu} + u_s^{\mu}$.
  3. *Either one of above inequalities is strict.*

The first condition requires non-decrease accounts utility after swap. The second condition means non-decrease total utilities of concerned shards. The last condition indicates either the accounts utility or the shards utility should increase with the swap. These three conditions are reasonable because both accounts and shards are individually rational. Accounts refuse the swap when they found the utility become less after the swap. As for shards, they care about not only themselves but also the whole blockchain system. Therefore, they would like to swap when the total utilities increase.

Then, we define two-sided exchange-stable (also called swap-stable) matching.

Definition 6 (Two-sided exchange-stable matching). *A matching μ is a two-sided exchange-stable matching if and only if no swap-blocking pair exists.*

The stable matching is important. Otherwise, either accounts or shards would like to change their current decisions.

4.2 Accounts Assignment Algorithm

In this section, we propose a naive iterative accounts assignment algorithm to find an swap-stable many-to-one matching.

- Firstly, we initiate a random matching μ_0 between accounts and shards.
- Then, we search for a swap-blocking pair in this step. Once a swap-blocking pair $(a_j, a_{j'})$ or (a_j, s) is found, we allow the swap operation. Hence, the matching becomes $\mu^{a_{j'}}_{a_j}$ or $\mu^{s}_{a_j}$ respectively.
- Repeat the above step until we find a matching μ^* without any swap-blocking pair.

We then prove the convergence and stability of the Algorithm 1.

Theorem 1 (Convergence). *The Algorithm 1 is able to converge to the matching μ^* finally.*

Proof. From the definition of the swap-blocking pair, we know that each swap operation increases either involved accounts' utilities or the total utilities of involved shards. We set $us(\mu) = \sum_{s \in S} u_s(\mu)$. In a swap operation, utilities of all shards remain stable except for involved shards. Then sum of utilities of involved shards is non-decreasing. Hence, we have $us(\mu^{'}) \geq us(\mu)$ where $\mu^{'}$ is the swap matching of μ. We also define $ua(\mu^{'}) = \sum_{a \in A} u_a(\mu)$.

- If $us(\mu^{'})$ is always greater than $us(\mu)$ after each swap, then convergence is reached when $us(\mu^{'}) = us(\mu)$. Because $us(\mu)$ has maximum value, $us(\mu^{'})$ will be equal to $us(\mu)$ in limited rounds.
- If $us(\mu^{'}) = us(\mu)$ before convergence, the two-sided exchange must happen because account-vacancy exchange must lead to $us(\mu^{'}) > us(\mu)$. In this two-sided exchange, only the accounts in two-sided exchange pair change

Algorithm 1. Accounts Assignment Algorithm

Input: Accounts set A,shards set S, transactions graph G
Output: The swap-stable matching μ^*

```
Randomly initialize μ
repeat
    for a_j ∈ A do
        for s_i ∈ S do
            if (a_j, s_i) is a swap-blocking pair then
                μ* = μ^{s_i}_{a_j}
            end if
        end for
    end for
    for a_j, a_j' ∈ A, j ≠ j' do
        if (a_j, a_j') is a swap-blocking pair then
            μ* = μ^{a_j'}_{a_j}
        end if
    end for
until No swap-blocking pair exists
```

(improve) their utility while other accounts and shards remain the same utilities. Therefore, this two-sided exchange does not bring externalities and hence our algorithm would never go back to any previous state. Due to limited state space size, the algorithm would converge finally.

The Algorithm 1 would converge in both two conditions, and therefore it is able to converge to μ^* eventually.

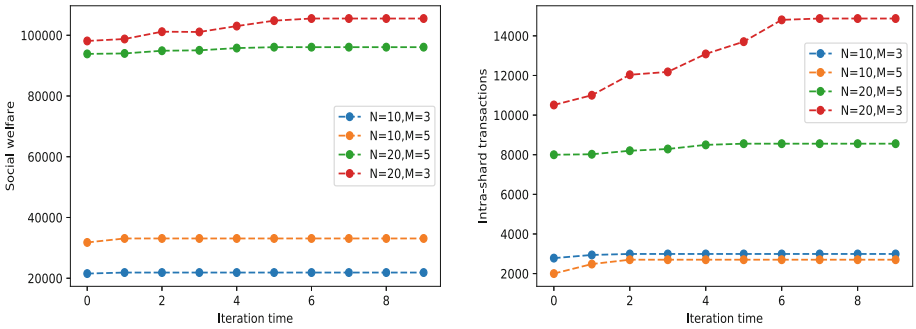
Theorem 2 (Two-sided stability). *The matching μ^* obtained from the Algorithm 1 is two-sided stable.*

Proof. We prove this theorem by contradiction. Assuming the final matching μ^* is not two-sided stable, then there must exist a swap-blocking pair as definition 6 defines. If so, the Algorithm 1 would not terminate due to the existence of the swap-blocking pair. Hence, the matching μ^* is not the final matching from the Algorithm 1 which is contradict to the assumption.

Therefore, the Algorithm 1 output μ^* is a two-sided stable matching.

5 Experimental Results

In this section, we conduct numerical simulations of the proposed many-to-one matching model. We firstly show the efficiency of the Algorithm 1. We use simulation results to present the Theorem 1 that the Algorithm 1 convergences in limited round. Then we compare the total utilities and social welfare of the many-to-one matching model with the random algorithm.



(a) Social welfare

(b) Intra-shard transactions

Fig. 1. Social welfare and the number of intra-shard transactions of iterations

5.1 Simulation Setup

Our simulation setting is as follows, unless otherwise noted.

We conduct the simulation with $N = 10$ accounts and $M = 5$ shards. The shards' capacity is randomly assigned. The number of miners in each shard is distributed from the uniform distribution $U \sim [4, 10]$. Accounts have their own perfect shard's size $x_a, a \in A$ which is drawn from the uniform distribution $U \sim [3, 11]$. Accounts accept the shard's size lying on $[x_a - \epsilon, x_a + \epsilon]$ where ϵ is set to be 2. Accounts' preference weights are generated randomly.

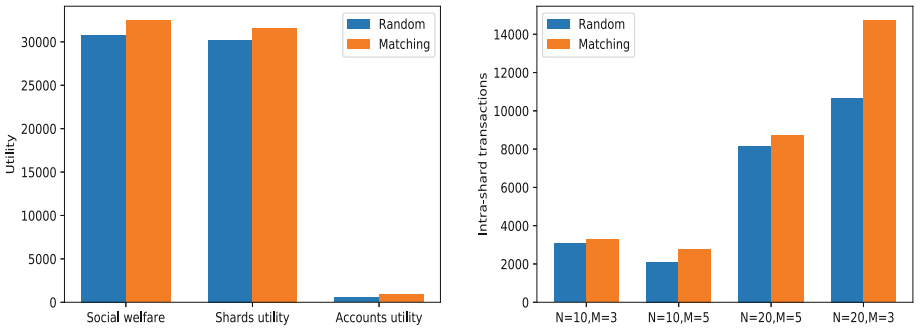
As for the transaction graph, we firstly specify the degree of busyness for each account. Based on the busyness, we create transactions and build a graph. We assume in the next epoch, accounts would not initiate more than 100 transactions.

5.2 Convergence Efficiency

In Fig. 1(a) and 1(b), we show the convergence of the Algorithm 1 with different amount of accounts and shards, i.e. $N = 10$ accounts with $M = 3$ shards, $N = 10$ accounts with $M = 5$ shards, $N = 20$ accounts with $M = 3$ shards and $N = 20$ accounts with $M = 5$ shards.

As the Theorem 1 and Theorem 6 claimed, the proposed algorithm converges in limited rounds. The convergence takes palace in several rounds. There are only 2 iterations in $N = 10, M = 3$ and 3 in $N = 10, M = 5$. With $N = 20$ accounts and $M = 5$ shards, the algorithm reaches the convergence in 5 rounds. It costs 7 rounds to converge with $N = 20$ accounts and $M = 3$ shard. The iteration times increases with both number of shards and accounts because more shards and accounts bring more possible assignments.

Figure 1(a) shows the change of social welfare with iterations. We noticed that the social welfare does not always increase with iterations. Because the swap operation only considers utilities of concerned parties, other accounts might have



(a) Utility (b) Intra-shard transactions

Fig. 2. Utility and intra-shard transactions in the matching algorithm and random assignment

utility decreased after swap. But we could see only one swap operation at the first iteration would increase social welfare significantly.

In Fig. 1(b), we could see the change of number of intra-shard transactions with iterations. Different from the social welfare, the intra-shard transactions in non-decreasing with iteration times. Because the swap operation would not bring more transaction fees for all transactions. It just transfer from one shard to another shard. Due to non-decreasing utilities of involved shards, the cost decreases while the income remains same. That is, there are more intra-shard transactions and less inter-shard transactions compared with the previous matching.

5.3 Compare with Random Assignment

We measure performance of the proposed algorithm in both utility improvement and intra-shard increase by comparing with the random assignment.

Figure 2(a) shows the improvement of utility with $N = 10$ accounts and $M = 5$ shards. The utilities of both shards and accounts increase by applying the proposed algorithm. Therefore, the social welfare is improved accordingly.

Figure 2(b) compare the number of intra-shard transactions in four settings. In each setting, the proposed algorithm induce more intra-shard transactions than the random assignment.

6 Related Work

Shading protocols aim to improve scalability and enhance throughput of traditional blockchain systems. Most sharding-relative works focus on sharding system design [6,9,11,19]. The concept of sharding is firstly applied in a permissioned blockchain system of the centrally bank [6]. *ELASTICO* [11] is the first perminssionless blockchain system using sharding technique. The shards are formed randomly according to miners' solutions to the hash puzzle and then

accounts are assigned to shards in the same way. Similar random accounts assignment are applied in *OmniLedger* [9] and *RapidChain* [19]. Compared with the first work in sharding perminssionless blockchain system, they provide stage sharding protocols for further performance improvement.

Among these works, shards formulation and accounts assignment are introduced briefly without any sophisticated design and analysis. Random shards formulation and accounts assignment are most popular because they are easy for implementation. Shards formulation is studied in [3]. A coalition game is proposed to model the problem of reputation-based shards formulation. The new shards formulation algorithm increases the system's security and throughput. *OptiShard* [8] invests the optimal shard size in a sharding blockchain systems based on two features: performance and security. The work [15] focus on transactions placement in an UTXO sharding blockchain model. It models the transaction relationship as a graph and proposes a graph partition algorithm *OptChain* for transactions assignment. *OptChain* algorithm decreases inter-shard transactions significantly. However it is not suitable in an account model.

There are also some other works providing participants' behaviour analysis from the view of game theory. The work [12] aims to maximize participants' engagement. It provide an incentive mechanism to motivate participants' cooperation. An evolutionary game [16] is used to analysis the change of participants population in sharding applications.

None of above works leverages the matching theory for accounts assignment problem.

7 Conclusion

We investigated the devices assignment problem in an IoT sharding blockchain system. We model the devices assignment as a many-to-one matching model with externalities. We come up with an iterative algorithm to find a two-sided exchange stable matching of accounts (IoT devices) and shards in this model. We give a complete proof of the convergence and stability of the proposed algorithm. At last, we evaluate the matching model and the algorithm to show its convergence efficiency and better performance. Compared with the random assignment, our mechanism increases total utilities of shards and social welfare.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61701216, Shenzhen Science, Technology and Innovation Commission Basic Research Project under Grant No. JCYJ20180507181527806, Guangdong Provincial Key Laboratory (Grant No. 2020B121201001) and "Guangdong Innovative and Entrepreneurial Research Team Program"(2016ZT06G587) and the"Shenzhen Sci-Tech Fund"(KYTDPT20181011104007).

References

1. The zilliqa project (2017). https://www.zilliqa.com
2. Scalability-bitcoin wiki (2018). https://en.bitcoin.it/wiki/Scalability

3. Asheralieva, A., Niyato, D.: Reputation-based coalition formation for secure self-organized and scalable sharding in IoT blockchains with mobile edge computing. IEEE Internet Things J. **PP**(99), 1 (2020)
4. Bodine-Baron, E., Lee, C., Chong, A., Hassibi, B., Wierman, A.: Peer effects and stability in matching markets. In: Persiano, G. (ed.) SAGT 2011. LNCS, vol. 6982, pp. 117–129. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24829-0_12
5. Conoscenti, M., Vetró, A., Martin, J.: Blockchain for the internet of things: a systematic literature review. In: The Third International Symposium on Internet of Things: Systems, Management and Security (IOTSMS-2016) (2016)
6. Danezis, G., Meiklejohn, S.: Centrally banked cryptocurrencies. arXiv preprint arXiv:1505.06895 (2015)
7. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Amer. Math. Monthly **69**(1), 9–15 (1962)
8. Kantesariya, S., Goswami, D.: Determining optimal shard size in a hierarchical blockchain architecture. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–3 (2020). https://doi.org/10.1109/ICBC48266.2020.9169448
9. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: OmniLedger: a secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE (2018)
10. Li, M., et al.: CrowdBC: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans. Parallel Distrib. Syst. **30**(6), 1251–1266 (2019). https://doi.org/10.1109/TPDS.2018.2881735
11. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)
12. Manshaei, M.H., Jadliwala, M., Maiti, A., Fooladgar, M.: A game-theoretic analysis of shard-based permissionless blockchains. IEEE Access **6**, 78100–78112 (2018)
13. Montes, J.M., Ramirez, C.E., Gutierrez, M.C., Larios, V.M.: Smart contracts for supply chain applicable to smart cities daily operations. In: 2019 IEEE International Smart Cities Conference (ISC2), pp. 565–570 (2019). https://doi.org/10.1109/ISC246665.2019.9071650
14. Nakamoto, S.: A peer-to-peer electronic cash system (2008). https://bitcoin.org
15. Nguyen, L.N., Nguyen, T.D.T., Dinh, T.N., Thai, M.T.: OptChain: optimal transactions placement for scalable blockchain sharding. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 525–535 (2019). https://doi.org/10.1109/ICDCS.2019.00059
16. Ni, Z., Wang, W., Kim, D.I., Wang, P., Niyato, D.: Evolutionary game for consensus provision in permissionless blockchain networks with shards. In: ICC 2019–2019 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2019)
17. Pee, S.J., Kang, E.S., Song, J.G., Jang, J.W.: Blockchain based smart energy trading platform using smart contract. In: 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 322–325 (2019). https://doi.org/10.1109/ICAIIC.2019.8668978
18. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014). https://ethereum.org
19. Zamani, M., Movahedi, M., Raykova, M.: RapidChain: scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948 (2018)