IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 January 2024, accepted 29 January 2024, date of publication 13 February 2024, date of current version 21 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365510

METHODS

Blockchain Transaction Sharding Algorithm Based on Account-Weighted Graph

JIAO LI AND YUANHANG NING
School of Computer Science, Xi'an Shiyou University, Xi'an 710065, China
Corresponding author: Yuanhang Ning (yhning93@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61802301, and in part by the Postgraduate Innovation and Practice Ability Development Fund of Xi'an Shiyou University under Grant YCS23213168.

ABSTRACT Sharding technology is widely regarded as an effective way to solve blockchain scalability limitations, such as lower throughput and longer delay time. However, sharding encounters challenges related to high cross-shard transaction proportion and complex cross-shard transaction verification. Therefore, a transaction sharding algorithm based on account-weighted graph is proposed. Firstly, an account-based transaction sharding model is established from the viewpoint of data sharding. Secondly, based on this model, an account-weighted graph is constructed with accounts as nodes and transaction frequency between accounts as weights for the long-term accumulated transaction data of blockchain. The community discovery algorithm is adopted, and the accounts with the largest modularity gain are selected and merged according to the association relationship between the accounts, so a multi-shard blockchain is formed initially. Finally, the multi-shard blockchain is adjusted and rebuilt by splitting and merging. The proposed algorithm is compared with the modular sharding algorithm and the random sharding algorithm under the sharding granularity of 10. The cross-shard transaction proportion is reduced by 78.8% and 78.7%, the average cross-shard transaction delay of accounts is reduced by 36.1% and 44.5%, and the transaction throughput is increased by 93.1% and 122.1%. Under the other sharding granularities, the proposed algorithm also outperforms the two algorithms in terms of cross-shard transaction proportion and account transaction delay. In conclusion, this method effectively reduces the cross-shard transaction proportion and minimizes cross-shard transaction delay.

INDEX TERMS Scalability, transaction sharding, sharding granularity, cross-shard transaction proportion, transaction delay.

I. INTRODUCTION

Blockchain technology is a technical solution that does not rely on third parties and utilizes its own distributed nodes for the storage, verification, transmission, and communication of network data. Blockchain technology has the characteristics of decentralization, traceability, and transparency [1]. The diverse applications of blockchain technology will reshape traditional industries, create new business models, and play a crucial role in applications such as financial technology, product traceability, and privacy protection [2].

However, in the development of blockchain technology, the ''impossible triangle'' issue has always existed, as it is impossible to simultaneously achieve decentralization, security, and scalability [3]. Scalability is currently the biggest obstacle preventing blockchain systems from being suitable for real-life applications [4], [5]. TPS (Transactions Per Second) refers to the average number of transactions processed per second by a system. Currently, the transaction throughput of blockchain systems is relatively low. For example, in the Bitcoin network, a block is packaged every 10 minutes, with a maximum transaction throughput TPS of only about 7 TPS, while the Ethereum platform has around 30 TPS, and during the activity period of an internet shopping platform, it reaches around 100,000 TPS. Clearly, in practical applications, the low TPS and long transaction delay of blockchain systems make them unsuitable for internet-level applications, severely constraining the implementation of blockchain applications [6], [7].

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

To enhance the scalability of blockchain, researchers have proposed various blockchain scalability solutions [8]. Existing blockchain scalability optimization schemes include sharding technology [9], state channels [10], and directed acyclic graph [11], [12]. Parallel process is extensively used to increase processing speed in solving different scientific problems, which plays a great role in the scalability of blockchain technology. Parallelism is the main method to achieve the required processing power [13]. Elastico [14] introduced a concept called sharding, and since then, blockchain sharding technology has become a popular research direction [15]. As one of the mainstream methods of on-chain expansion, sharding technology is currently the most effective expansion solution to achieve high performance without sacrificing centralization [16]. It adopts a divide-and-conquer approach, dividing nodes into small groups called shards, which can process transactions in parallel and alleviate the storage overhead of each node [17].

Blockchain sharding technology is primarily divided into network sharding, transaction sharding, and state sharding [18]. Network sharding is a method of dividing blockchain network into different shards through a specific organizational approach, and these shards process transactions in parallel, aiming to improve network performance and processing capacity. Nodes in the network are allocated to different shards through verifiable random functions, ensuring fair node allocation while also considering the balance of computing power after sharding. Transaction sharding is a method to assign transactions to multiple shards, with each shard responsible for receiving, validating, and processing a portion of transactions. This helps reduce the burden on each shard, thereby transaction processing speed is improved. The primary challenge of transaction sharding lies in how to partition transactions reasonably to ensure that dependencies between transactions are met while maintaining efficient processing. State sharding is the bottleneck of sharding technology, making the verification process of cross-shard transactions extremely difficult. Different shards require transaction transfer or frequent cross-shard communication in a certain way due to the storage of different ledgers. The complexity of state sharding constrains the development of sharding technology. Network sharding, as the foundation of sharding technology, first focuses on the security issues of the system. Existing network sharding strategies adopt the method of randomly selecting nodes to partition sub-chains, avoiding the problem of malicious node aggregation after sharding, thereby ensuring the security of blockchain network [19]. How to better implement transaction sharding is currently a noteworthy issue.

Several studies have applied sharding technology to improve the overall throughput of blockchain systems, such as Zilliqa [20], Omniledger [21], Monoxide [22], and Ethereum 2.0 [23], [24]. These are classical mainstream sharding schemes on blockchain, which use different sharding strategies based on the account balance model [25] and the UTXO model [26]. To ensure user randomness, these projects use the modular sharding algorithm (MSA) and the random sharding algorithm (RSA), which often ignore the relationship between accounts. In these sharding strategies, there is a significant increase in cross-shard transactions, which may lead to delays in verifying and synchronizing transaction states, network congestion, and poor performance on blockchain. In extreme cases, all transactions on blockchain are cross-shard transactions, which will require additional communication, data synchronization, and coordination between multiple shards. Some scholars have considered the relationship between accounts. For example, BrokerChain proposes an account-splitting mechanism that stores account states among multiple shards to facilitate coordination for participating in multiple cross-shard transactions [27]. In FBTS [28], the frequency of transactions between accounts is taken into account, and accounts with high transaction frequencies are recursively placed in the same shard. It can thus be seen, that the relationships between accounts and the frequency of transactions directly impact the number of cross-shard transactions on blockchain. To reduce the cross-shard transaction proportion, a starting point could be focusing on accounts since the complexity of transactions varies between different accounts. Therefore, designing better sharding strategies can effectively mitigate the negative impact of cross-shard transactions when employing sharding technology on blockchain.

In this paper, we propose a blockchain transaction sharding algorithm based on account-weighted graph (AWSA) that takes into account the relationships between accounts from a global perspective. This method extracts transaction relationships between accounts from historical transaction data and abstracts account relationships as weighted graphs. By reflecting account relationships intuitively in the account-weighted graph, accounts can be assigned to different shards to reduce the occurrence of cross-shard transaction proportion after sharding.

The following is a summary of the main contributions of this paper:

- From the perspective of transaction sharding, we propose an account-based transaction sharding model. This model changes the strategy for processing transaction data in traditional blockchains, aiming to partition transaction data from the perspective of sharding to achieve parallel verification and processing, thereby improving processing efficiency.
- The blockchain transaction sharding algorithm based on account-weighted graph is proposed. Starting from the association between accounts, we extract transaction frequency information between accounts, use this frequency as the weight to construct an account-weighted graph, and place strong-correlated accounts in the same shard. Therefore, a large number transactions occur in the single shard and avoid the cross-shard operation.
- The performance of the proposed blockchain transaction sharding algorithm based on account-weighted graph is verified through simulation experiments. The final

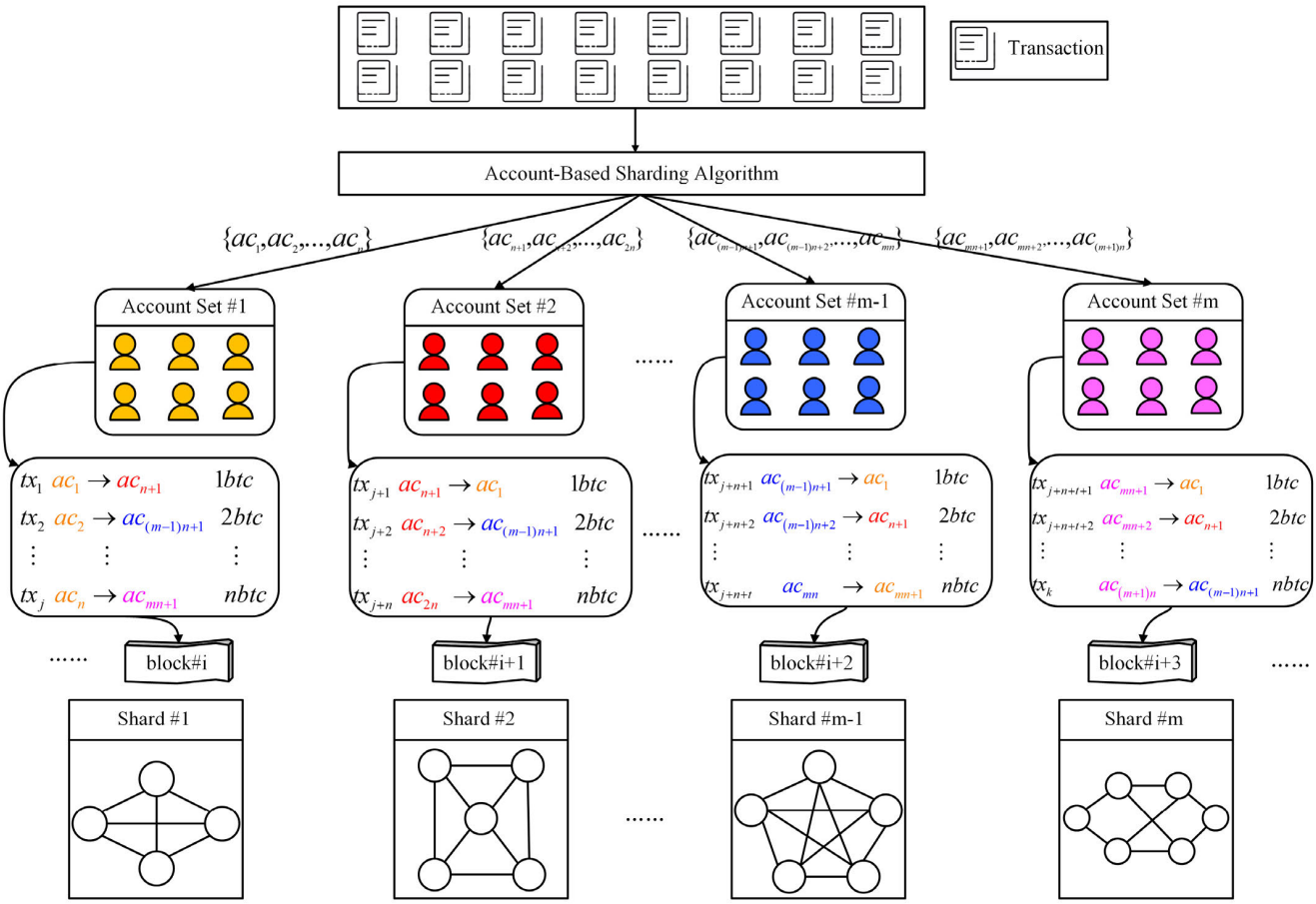

FIGURE 1. Account-based transaction sharding model.

simulation results show that, compared to existing algorithms, this algorithm can effectively reduce the cross-shard transaction proportion and minimize cross-shard transaction delay.

The remainder of this paper is structured as follows: the related works are reviewed and discussed in Section II, the system model is presented in Section III, the blockchain transaction sharding algorithm based on account-weighted graph is presented in Section IV, the evaluation of our experimental study is presented in Section V and the study is concluded in Section VI.

II. RELATED WORK

The concept of blockchain sharding originates from the traditional database sharding concept [7], [29]. By transactions' sharding on blockchain, the shards collaborate, allowing multiple tasks to be completed in parallel at the same time. This characteristic, which linearly improves scalability with the increase in the number of shards, significantly enhances the throughput and performance of the entire blockchain [30].

In the Zilliqa project [20], the state information of each account is uniquely accessed through the corresponding account address, and each transaction in the network is randomly mapped to a working shard for verification based on the sender's address. It was known as the random sharding algorithm.

OmniLedger [21] is an innovative decentralized ledger solution known for its high performance and long-term security. It replaces PoW for selecting validator grouping with a cryptographic lottery protocol and then uses a distributed unbiased random number generation scheme to assign validators to different shards. It employs an anti-predictive public randomness protocol to select large shards for transaction processing to ensure these shards are statistically representative, known as the random sharding algorithm.

The Monoxide solution [22] replaces a single chain with a concurrent multi-chain system, where each single chain is called a consensus group that can work in parallel, sharing the network's throughput, computing, and storage pressure. The selection of consensus groups is based on the first k bits of the user's address to determine which consensus group the user belongs to, known as the modular sharding algorithm.

In Ethereum 2.0 [23], [24], the core is the beacon chain, which acts as a central hub connecting various shards. Sharding allows the network to process transactions in parallel, enhancing overall processing capability. The beacon chain generates random numbers regularly to allocate transactions randomly to shards. The beacon chain is responsible for

managing validators, maintaining consensus and time, and coordinating cross-shard transaction verification. However, the beacon chain may face performance bottlenecks when handling a large number of cross-shard transactions. It was known as the random sharding algorithm.

Brokerchain [27] uses broker accounts to store account states in multiple shards and participate in the coordination of several cross-shard transactions. These divided accounts have the same account address, and the coordination mechanism of brokers can reduce cross-shard communication based on transfer-type transactions, thus increasing the throughput of blockchain and reducing the average confirmation delay of transactions. The main purpose of Brokerchain is to accelerate the confirmation of cross-shard transactions.

The FBTS algorithm [28] considers the frequency between accounts and selects transactions with high frequencies to be placed in one shard to reduce cross-shard transaction proportion. However, the intricate transaction relationships between accounts are overlooked, and the account is not optimized for sharding as a whole.

LB-Chain proposes an account migration strategy to adjust the load balance between shards [31]. The account migration strategy is mainly divided into account allocation and account prediction. Account allocation predicts the transaction volume of each account based on the results of the random sharding strategy used by Ethereum, using historical transaction data. Then, the account allocation strategy is used to migrate accounts. Table 1 shows a summary of related studies.

TABLE 1. Existing research on sharding methods.

Sharding projects	Year	Shard method	Shard class
Zilliqa[20]	2017	Random mapping based on account address	Random sharding algorithm
OmniLedger[21]	2018	RandHound+ VRF-based	Random sharding algorithm
Monoxide[22]	2019	First k bit of address	Modular sharding algorithm
Ethereum2.0[23], [24]	2021	RANDAO+VDF	Random sharding algorithm
Brokerchain[27]	2022	Broker	Transaction characteristics
FBTS[28]	2023	Frequency of transaction	Transaction characteristics
LB-Chain[31]	2023	Static: random	Random sharding algorithm
		Dynamics: periodic forecasts	Machine learning

The mainstream sharding schemes on blockchain have two ways of account and transaction allocation. Zilliqa [20], OmniLedger [21], Ethereum2.0 [23], [24], use the random sharding algorithm, while Monoxide [22] uses the modular sharding algorithm. Although [28] considers the transaction frequency relationship of accounts, the analysis is not thorough, and it does not analyze overall. Brokerchain [27] uses the modular sharding algorithm used by the Monoxide project as a comparison, but its main purpose is to accelerate the confirmation of cross-shard transactions by storing account states as shards. LB-Chain [31] proposes a dynamic solution for account migration based on the random sharding algorithm used by Ethereum as a historical result.

It can be seen that in the long-term historical transaction data migration, static algorithms do not optimize the sharding of accounts from an overall perspective, ignoring the complex transaction relationships between accounts, leading to a high cross-shard transaction proportion. All accounts and their transactions on blockchain constitute a social network, and the relationships between accounts can be seen from the intricate network. However, simple sharding strategies can lead to a large number of cross-shard transactions and imbalanced transaction volumes. In a multi-shard blockchain network, each shard is similar to a community, containing different accounts. A good community relationship ensures that the vast majority of transactions are completed within the community, greatly reducing the number of cross-shard transactions. The accounts within each community need to communicate constantly, and the relationship between accounts coincides with the graph structure in the data structures. Therefore, Graph structure is an effective way to represent relational relationships [32]. Based on the above, we abstract the relationship between accounts as a weighted graph to analyze the relationship between accounts more directly and propose sharding algorithms to finish mapping each account to a different shard.

III. SYSTEM MODEL

In the real transaction system of blockchain, the mainstream models include the UTXO model used by Bitcoin [25] and the account balance model used by Ethereum [26]. These models facilitate real-time packaging of transactions for on-chain processing. However, when migrating historical transaction data, the consideration shifts towards static sharding models to reduce storage costs and enhance blockchain's capability for parallel transaction processing. Concurrently, the features of historical transactions show better consistency with the features of future transactions, and historical transactions are representative in terms of transactions' features. Hence, an analysis of transactions' features is imperative. On blockchain, a transaction includes the account information of both parties, transaction time, transaction location, transaction amount, time intervals, and so on. Based on these elements, a deeper understanding and analysis of transactions can be achieved. In real transactions, the account of distribution salaries and loan repayments are relatively fixed, and transactions are launched at the appointed time. It reflects the account aggregation of transactions from a time perspective. Similarly, in real transactions, the cooperative relationships and frequent transactions between accounts can reflect the frequency aggregation and location aggregation of accounts. The transaction patterns and behavioral characteristic trends in a period are often relatively stable, and predictions based on short-term historical transaction data typically can obtain relatively accurate results. However, transactions sharded by time and transactions by location also can result in traffic overload between shards. The reasons are that they fail to

consider the correlation between accounts, leading to uncertainty in transaction data storage, and the issue of high cross-shard transaction proportion. Therefore, transactions sharded by account are superior to transactions sharded by time and transactions by location in reducing cross-shard transaction proportion. Based on the above, an analysis of historical transaction data is conducted, taking into account various transaction characteristics. An account-based transaction sharding model is proposed. This model aims to better address the intricate and complex interrelationships among accounts within a sophisticated transaction network, ultimately enhancing the efficiency and security of transaction processing.

Fig. 1 illustrates the account-based transaction sharding model. On blockchain, Ac represents the set of accounts, $Ac = \{ac_1, ac_2, \ldots, ac_n\}$, Tx denotes the set of transactions, $Tx = \{tx_1, tx_2, \ldots, tx_k\}$, $tx_k = \{ac_i, ac_j, value\}$. The account ac_i serves as the sender of the transaction and is denoted as $tx_i.sender = ac_i$, Account ac_j serves as the receiver of the transaction and is denoted as $tx_i.receiver = ac_i$, where $value$ signifies the transaction amount between sender ac_i and receiver ac_j. The blockchain stores accumulated transaction data over the years, and transaction sharding involves distributing historical data to shards from the perspective of storage optimization. The rules for allocating data to shards involve transaction sharding strategies. The account-based sharding model first extracts accounts from historical transactions, then allocates the accounts to shards, and subsequently maps the transactions to the working shards based on the senders' addresses. For any shard in a multi-shards blockchain, nodes obtain the right to record through the internal shard consensus protocol, and the transactions are packaged into the block, according to specific block sizes, and complete the on-chain process. Due to the unique account addresses, the data in each shard consists of the collection of all transactions initiated by the accounts in this shard. Therefore, the same transaction will only be allocated to one shard, and transactions initiated by the same account will be assigned to the same shard. For each transaction, its allocation in a particular shard is deterministic and unique, without redundant storage across shards. However, a specific block is assigned to shards, each node completes transaction verification through consensus protocols. Nodes within the shard save on-chain transaction data through redundant accounting. Therefore, the on-chain process on a multi-shards system is similar to that of traditional blockchains. Thus, a multi-shard blockchain network can operate in parallel. The evaluation of transaction sharding strategies includes metrics such as cross-shard transaction proportion, account proportion with different cross-shard number, account transaction delay with different cross-shard number, average cross-shard transaction delay, average cross-shard number, and transaction throughput. The metrics hold significant importance in the evaluation of sharding strategies. When designing transaction sharding strategies, efforts should be made to minimize the occurrence of cross-shard transactions, reduce the additional storage, and shorten querying costs for transactions.

IV. THE PROPOSED METHOD

Based on the account-based transaction sharding model, this section proposes a transaction sharding algorithm based on account-weighted graph and provides the algorithm steps.

A. ALGORITHM DIAGRAM

Fig. 2 illustrates the schematic diagram of blockchain transaction sharding algorithm based on account-weighted graph. On blockchain, the association and frequency between accounts precisely align with the weighted graph in data structures. Therefore, we abstract the account relationships into a weighted graph. Fig. 2(a) represents an abstract depiction of real-world transaction accounts, where circles represent transaction accounts, and the weights on the edges between accounts indicate the transaction frequency between the parties. In real-life scenarios, a transaction is directional, meaning that transactions from Alice to Bob and from Bob to Alice are distinct. However, when designing transaction sharding strategies, the verification and processing of a transaction only require consideration of the association between accounts, without regard to the directionality of the transactions. For the sake of research convenience, the account association is abstracted as an undirected weighted graph. Fig. 2(b) illustrates the preferred shard discovery and shard merge. In this stage, a community detection algorithm is applied to the account-weighted graph for preferred shard selection and merging [33]. First, according to the account-weighted graph, for any account, the modularity of the accounts associated with it is calculated, and the associated accounts with the largest modularity are selected and merged to form an account set. Through multi-round operations, the partitions and aggregation of accounts can be achieved to form multi-shards. Subsequently, the process enters the shard merge phase, compressing all accounts within each shard into a single account, with the weights between shards being transformed into weights between the new accounts. Perform preferred shard discovery until the shards to which all accounts belong no longer change. Fig. 2(c) depicts the generated shard results, producing a multi-shard blockchain network. According to the complexity of the transaction, the account is divided into different shards. Each shard is independent of the other and is responsible for processing the transactions of its respective shard and maintaining the shard status. Transaction information is verified and processed in parallel between each shard, thereby significantly improving the load-balancing capability and resource utilization efficiency of the entire blockchain system. Thus, starting from the relationships between accounts aids in optimizing the shard system.

B. RELATED DEFINITIONS

We define our algorithm based on the account-based transaction sharding model. Construct the account-weighted graph

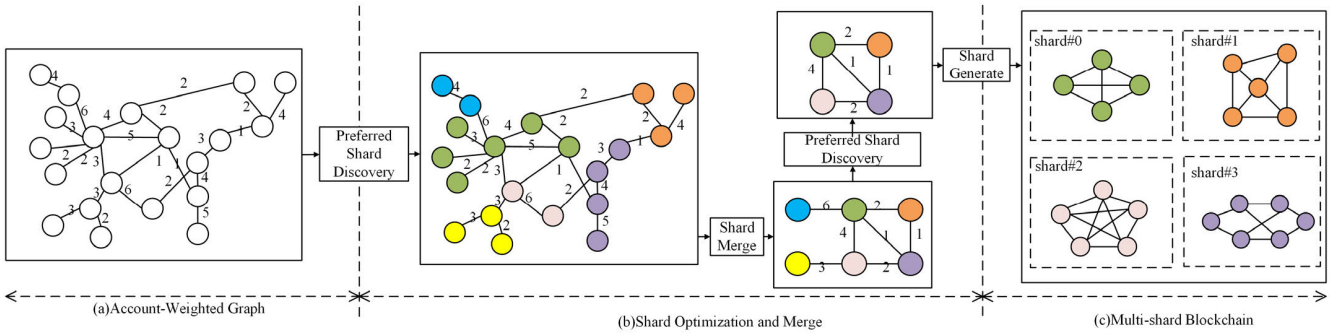

FIGURE 2. Schematic diagram of blockchain transaction sharding algorithm based on account-weighted graph.

$G = (Ac, Tn)$, where Tn represents the set of transaction frequencies on blockchain, $Tn = \{tn_1, tn_2, \ldots, tn_s\}$, $tn_i = \{ac_i, ac_j, Num(ac_i, ac_j)\}$, $tn_i \in Tn$, and calculate the cumulative frequencies of transactions where ac_i is as a sender, and ac_j is as a receiver, and denote it as $Num(ac_i, ac_j)$. The account-weighted graph G is denoted as M, $M = \frac{1}{2}\sum_{i,j} Num\left(ac_i, ac_j\right)$. A multi-shard network is denoted as $Shard$, $Shard = \{shard_1, shard_2, \ldots, shard_m\}$, where $shard_i$ represents a set of accounts, $shard_i = \{ac_1, ac_2, \ldots, ac_n\}$. For any given $shard_i$, the sum weights of $shard_i$ is denoted as $w_i = \sum_{i,j} Num(ac_i, ac_j)$, where $ac_i \in shard_i$, $ac_j \in shard_i$. For any given $shard_i$, the sum of weights connected to $shard_i$ is denoted as w_{shard_i}, $w_{shard_i} = \sum_{i,j} Num(ac_i, ac_j)$, where $ac_i \in shard_i$, $ac_j \notin shard_i$. Moreover, the weight between $shard_i$ and $shard_j$ is denoted as $w_{shard_{i,j}}$, $w_{shard_{i,j}} = \sum_{i,j} Num(ac_i, ac_j)$, where $ac_i \in shard_i$, $ac_j \in shard_j$. For any given $shard_i$, the associated shards of $shard_i$ are denoted as R_{shard_i}, where $R_{shard_i} = \cup shard_j$ and $w_{shard_{i,j}} \neq 0$.

According to community discovery algorithm concepts [33], the formula for the gain obtained from $shard_i$ and $shard_j$ is:

$$\Delta Q_{i,j} = \frac{1}{2M}\left(w_{shard_{i,j}} - \frac{w_{shard_j} \times w_{shard_i}}{M}\right) \tag{1}$$

Table 2 gives a brief description of the symbols of formula (1).

At different shard granularities m, For $\forall tx_k, tx_k \in Tx$, $tx_k = \{ac_i, ac_j, value\}$, if $ac_i \in shard_k$, $ac_i \in shard_n$, and $shard_k \neq shard_n$, then tx_k is referred to as a cross-shard transaction. The total number of transactions is denoted as $|Tx|$, the cross-shard transactions on blockchain are denoted as Tx_{cross}, and the number of cross-shard transactions is denoted as $|Tx_{cross}|$. Therefore, the cross-shard transaction proportion is denoted as C_r.

$$C_r = \frac{|Tx_{cross}|}{|Tx|} \tag{2}$$

For a multi-shard blockchain with the sharding granularity of m, the set of cross-shard number is denoted as N, $N = \{0, 1, \ldots, m-1\}$. For any account ac_i, the cross-shard number is denoted as n_{ac_i}, the set of accounts with the cross-shard number n is denoted as An, $An = \{ac_i \,|n_{ac_i} = n\}$, $ac_i \in Ac$.

TABLE 2. A brief description of the symbols in formula 1.

Symbol	Description
$\Delta Q_{i,j}$	The gain obtained from the combination of $shard_i$ and $shard_j$
M	The weight of the account-weighted graph G
$w_{shard_{i,j}}$	The weight between $shard_i$ and $shard_j$
w_{shard_j}	The sum of weights connected to $shard_j$
w_{shard_i}	The sum of weights connected to $shard_i$

The proportion of account with the cross-shard number n is denoted as A_{r_n}.

$$A_{r_n} = \frac{|An|}{|Ac|} \tag{3}$$

For account ac_i under the cross-shard number of n, the cross-shard transaction delay is denoted as $t_{n_{ac_i}}$, and the average cross-shard transaction delay under the cross-shard number of n is denoted as t_n.

$$t_n = \frac{\sum\left\{t_{n_{ac_i}} \,|n_{ac_i} = n\right\}}{|An|} \tag{4}$$

For blockchain, the average cross-shard delay of accounts is denoted as t_{avg}.

$$t_{avg} = \frac{\sum_{i=1}^{i=|Ac|} t_{n_{ac_i}}}{|Ac|} \tag{5}$$

For blockchain, the average cross-shard number of accounts is denoted as n_{avg}.

$$n_{avg} = \frac{\sum_{i=1}^{i=|Ac|} n_{ac_i}}{|Ac|} \tag{6}$$

The total number of cross-shard transactions for ac_i is denoted as $\left|ac_{i_{cross}}\right|$, and the total number of transactions for ac_i is denoted as $|ac_i|$, the cross-shard transaction proportion for ac_i is denoted as c_{r_i}.

$$c_{r_i} = \frac{\left|ac_{i_{cross}}\right|}{|ac_i|} \tag{7}$$

From a qualitative point of view, for account ac_i, it contains cross-shard transactions and non-cross-shard transactions. For non-cross-shard transactions, they only need to complete the verification of the transaction in a single shard, the delay of an account that does not involve cross-shard transactions is denoted as $t_{0_{ac_i}}$; For cross-shard transactions, the transactions need several shards to collaborate to validate the transactions, so we introduce the account weighted average cross-shard delay t_{ac_i}.

$$t_{ac_i} = t_{n_{ac_i}} \times c_{r_i} + t_{0_{ac_i}} \times (1 - c_{r_i}) \tag{8}$$

On blockchain, the weighted average cross-shard transaction delay for all accounts is denoted as t.

$$t = \frac{\sum_{i=1}^{i=|Ac|} t_{ac_i}}{|Ac|} \tag{9}$$

For blockchain networks, the transaction throughput is denoted as T.

$$T = \frac{\sum_{i=1}^{i=k} tx_i}{t} \tag{10}$$

C. ALGORITHM DESCRIPTION

Sharding technology has to some extent improved blockchain performance, but it results in the problems of higher cross-shard transactions, transaction validation difficulty, and longer transaction delays on blockchain. The inter-account transaction frequency is a primary factor affecting whether cross-shard transactions occur frequently, so this paper proposes a blockchain transaction sharding algorithm based on account-weighted graph. Using the idea of community discovery [33], the algorithm organizes accounts into clusters through multiple rounds of preferred shard selection and merges shards based on the association relationship between accounts. The method encompasses two phases: (1) the preferred shard discovery and shard merge; (2) the shard granularity adjustment.

In the preferred shard discovery and shard merge, Algorithm 1 is executed. This algorithm analyzes the frequency of transactions among transaction accounts and takes the account-weighted graph as input. Firstly, it calculates the weight M of the account-weighted graph. Secondly, it treats each account as a shard and calculates $w_{shard_{i,j}}$, w_{shard_i} w_{shard_j} and R_{shard_i}. It proceeds to iterate through R_{shard_i}, calculates the modularity change $\Delta Q_{i,j}$ of the relevant shards, and records the shard with the maximum modularity change. If there exists a shard with a modularity greater than 0, the account is added to that shard; otherwise, it remains unchanged. Then, this process is repeated until no further changes occur in the shards to which all accounts belong. In the shard merge phase, it compresses the accounts within each shard into a single account. For accounts in the internal shard, the weights between accounts within a shard are updated to the weights of the new account ring, while for accounts in the inter-shards, weights between shards are updated to the weights between the new accounts. Finally, re-enter the preferred shard discovery and repeat the process until no further changes occur in the shards to which all accounts belong. After completing the preferred shard discovery and shard merge, the ideal state of the shard results is generated. Algorithm 1 provides the pseudocode for the preferred shard discovery and shard merge phase.

Algorithm 1 Preferred Shard Discovery and Shard Merge

Input: Account-weighted graph $G = (Ac, Tn)$, account set $Ac = \{ac_1, ac_2, \ldots, ac_n\}$, frequency set $Tn = \{tn_1, tn_2, \ldots, tn_s\}$, $tn_k = \{ac_i, ac_j, Num\left(ac_i, ac_j\right)\}$
Output: $Shard = \{shard_1, shard_2, \ldots, shard_s\}$

$Shard = \emptyset$, $w_{shard_i} = 0$, $R_{shard_i} = \emptyset$
$flag\_init = True$, $flag\_re = Flase$
$M = \frac{1}{2}\sum_{i,j} Num\left(ac_i, ac_j\right)$
for $(i = 1; i \le n; i++)$
  $shard_i = shard_i \cup ac_i$
end for
while *True*
  $flag\_init = Flase$
  for $(i = 1; i \le n; i++)$
    Calculate $w_{shard_{i,j}}$, w_{shard_i}, w_{shard_j}, R_{shard_i}
    $Max = -\infty$, $q = 0$
    for $\left(j = 1; j \le \left|R_{shard_i}\right|; j++\right)$
      $\Delta Q_{i,j} = \frac{1}{2M}\left(w_{shard_{i,j}} - \frac{w_{shard_j} \times w_{shard_i}}{M}\right)$
      if $\Delta Q_{i,j} > Max$ **then**
        $Max = \Delta Q_{i,j}$, $q = j$
      end if
    end for
    if $Max > 0$
      $shard_q = shard_q \cup shard_i$, $shard_i = \emptyset$
      $flag\_init = True$, $flag\_re = True$
    end if
  end for
  if $flag\_init = Flase$
    break
  end if
  $Shard = \{shard_1, shard_2, \ldots, shard_s\}$
  Restructure graph $G = (Ac, Tn)$, $Ac = Shard$, $Tn = \{tn_1, \ldots, tn_k, \ldots, tn_s\}$, $tn_k = \{shard_i, shard_j, w_{shard_{i,j}}\}$
  if $flag\_re = True$
    go to 1
  end if
  return $Shard = \{shard_1, shard_2, \ldots, shard_s\}$

Algorithm 1 generates a multi-shard blockchain network without any limitations on the number of shards that can be used. Therefore, to better balance the number of shards, we have specified a shard granularity to adjust the number of shards on a multi-shard blockchain network. In the shard granularity adjustment, Algorithm 2 takes the execution results of algorithm 1 and the specified shard granularity as input. If the shard number after algorithm 1 processing is greater than the specified shard granularity, the algorithm iterates through the shard result and merges the two shards with the smallest number of accounts into one shard. If the shard result is smaller than the specified shard granularity, the algorithm iterates through the shard result and splits the shard with the largest number of accounts into a new shard. It continues this process of iterating through the shard result

Algorithm 2 Shard Granularity Adjustment

Input: $Shard = \{shard_1, shard_2, \ldots, shard_s\}$, sharding granularity m
Output: $Shard = \{shard_1, shard_2, \ldots, shard_m\}$
The accounts' number of $shard_i$ is denoted as $|shard_i|$, Sort $Shard$ in ascending order of $|shard_i|$, $Shard = \{shard_1, shard_2, \ldots, shard_s\}$
if $s > m$ **then**
  for $(i = 1; i \leq s - m; i + +)$
  $shard_{i+1} = shard_{i+1} \cup shard_i$
  Sort $Shard$ in ascending order of $|shard_i|$, obtain $Shard = \{shard_{i+1}, shard_{i+2}, \ldots, shard_s\}$
  end for
end if
if $s < m$ **then**
  for $(i = 1; i \leq m - s; i + +)$
  $shard_{s+i} = shard_s/2$
  $shard_s = shard_s - shard_{s+i}$
  Sort $Shard$ in ascending order of $|shard_i|$, obtain $Shard = \{shard_1, shard_2, \ldots, shard_{s+i}\}$
  end for
end if
return $Shard = \{shard_1, shard_2, \ldots, shard_m\}$

until the number of shards reaches the specified shard granularity and then outputs the shard result. Algorithm 2 provides the pseudocode for adjusting the shard granularity.

V. EXPERIMENTAL STUDY

Through the analysis in Section II of this paper, the current classical mainstream sharding schemes on blockchain were investigated, which use the MSA algorithm and the RSA algorithm. Therefore, we selected the MSA algorithm and the RSA algorithm for comparison with the proposed AWSA algorithm. To verify the effectiveness of the algorithm, we conducted simulation experiments of three algorithms based on the proposed transaction sharding model and analyzed the experimental results under different sharding granularity. Three algorithms are evaluated in terms of performance metrics such as cross-shard transaction proportion, account proportion with different cross-shard number, account transaction delay with different cross-shard number, average cross-shard transaction delay of, average cross-shard number, and transaction throughput.

The dataset used for our study was extracted from the IEEE CSCloud 2023 with a total of 1,048,575 transactions and 93,624 accounts. These algorithms were implemented using the VSCode experimental platform, programmed in python language, and executed on a 64-bit Windows operating system with an intel core i5-13500HX 2.50 GHz central processing unit (CPU) and 16 GB of memory.

A. CROSS-SHARD TRANSACTION PROPORTION

From an overall qualitative perspective, the cross-shard transaction proportion is introduced to evaluate whether transactions require cross-shard operations to be completed. Fig. 3 illustrates the cross-shard transaction proportion using three different sharding algorithms MSA, RSA, and AWSA under different shard granularities. Experiment results indicate that the cross-shard transaction proportion for all three sharding algorithms increases with the increment of shard granularity. However, the AWSA algorithm exhibits an overall lower cross-shard transaction proportion compared to the MSA algorithm and the RSA algorithm. Specifically, at a shard granularity of 5, the cross-shard transaction proportion for the MSA algorithm and the RSA algorithm are 80% and 81%, respectively, while that of the AWSA algorithm is 18%. At a shard granularity of 10, the cross-shard transaction proportion for the MSA algorithm and the RSA algorithm are 90.27% and 89.57%, respectively, while that of the AWSA algorithm is 19.17%. At a shard granularity of 20, the cross-shard transaction proportion for the MSA algorithm and the RSA algorithm reaches 93%, whereas that of the AWSA algorithm is 20%. Overall, the AWSA algorithm reduces the cross-shard transaction proportion by 62 to 73 percentage points compared to the MSA algorithm and the RSA algorithm. Moreover, it can be observed that the cross-shard transaction proportion for the AWSA algorithm hardly increases with the increase in shard granularity, owing to its consideration of the relationship between the two parties involved in a transaction. Therefore, the AWSA algorithm can effectively reduce the cross-shard transaction proportion.

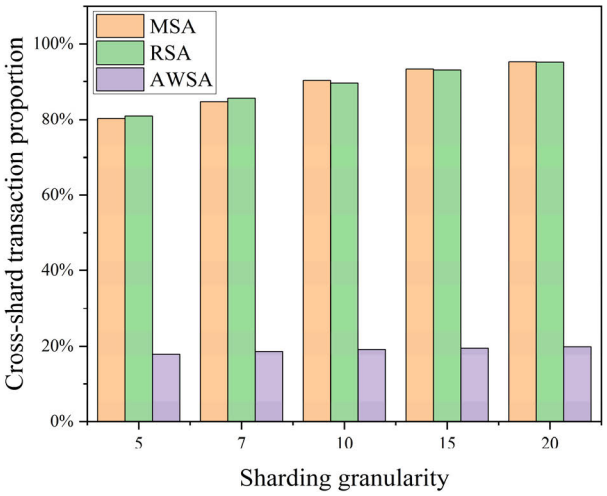

FIGURE 3. Cross-shard transaction proportion.

B. ACCOUNT PROPORTION WITH DIFFERENT CROSS-SHARD NUMBER

The cross-shard transaction proportion is to determine whether the transaction needs to be completed across shards or not. However, for each account, the proportion of transactions that require to be completed within a single shard and the proportion that require to be completed between cross-shard change. Therefore, the introduction of the account proportion with different cross-shard number serves to quantitatively measure transaction complexity. The higher the cross-shard number, the higher the transaction costs between the accounts. If a majority of accounts in the system frequently

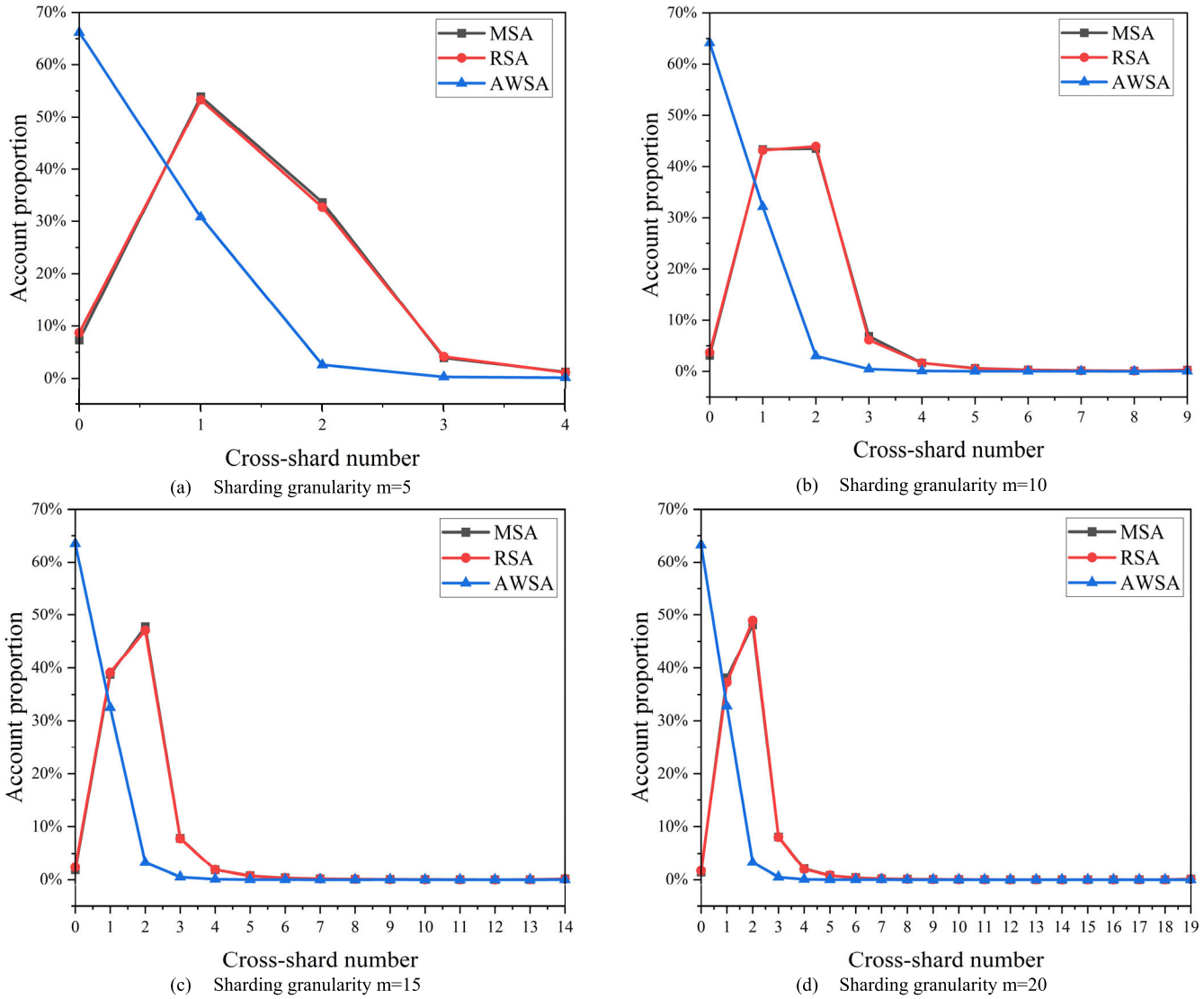

FIGURE 4. Account proportion with different cross-shard number.

require cross-shard operation, the delay time for transaction verification and completion might increase, which potentially degrades the benefits of parallel processing transactions brought by sharding technology. Fig. 4 shows the account proportion with different cross-shard number using the three sharding algorithms MSA, RSA, and AWSA under different sharding granularities. The results indicate that the account proportion for the MSA algorithm and the RSA algorithm are concentrated in scenarios where the shards are 1 and 2, accounting for over 80% of the total account transactions. This implies that more than half of the accounts involve only 2-3 shards in transaction verification. For the AWSA algorithm, at a cross-shard number of 0, the account proportion is around 65%, indicating that more than half of the accounts can complete transactions within a single shard. At a cross-shard number of 1, the account proportion is approximately 35%, suggesting that one-third of the accounts only cross one shard to complete their transactions. Consequently, the shards constructed by the AWSA algorithm rarely have multiple cross-shard transactions. As the shard granularity increases, the cross-shard number of 1 for the MSA algorithm and the RSA algorithm decreases gradually from 54% to 38%, while for AWSA algorithm, the account proportion increases slightly from 31% to 33%. This indicates that the distribution of accounts remains relatively stable under different shard granularities using the AWSA algorithm. In summary, compared to the RSA algorithm and the MSA algorithm, the proposed sharding algorithm effectively reduces the cross-shard number for accounts and minimizes the impact of high cross-shard transactions on blockchain performance.

C. ACCOUNT TRANSACTION DELAY WITH DIFFERENT CROSS-SHARD NUMBER

In practical applications, users are more concerned about how quickly their transactions can be committed to the blockchain.

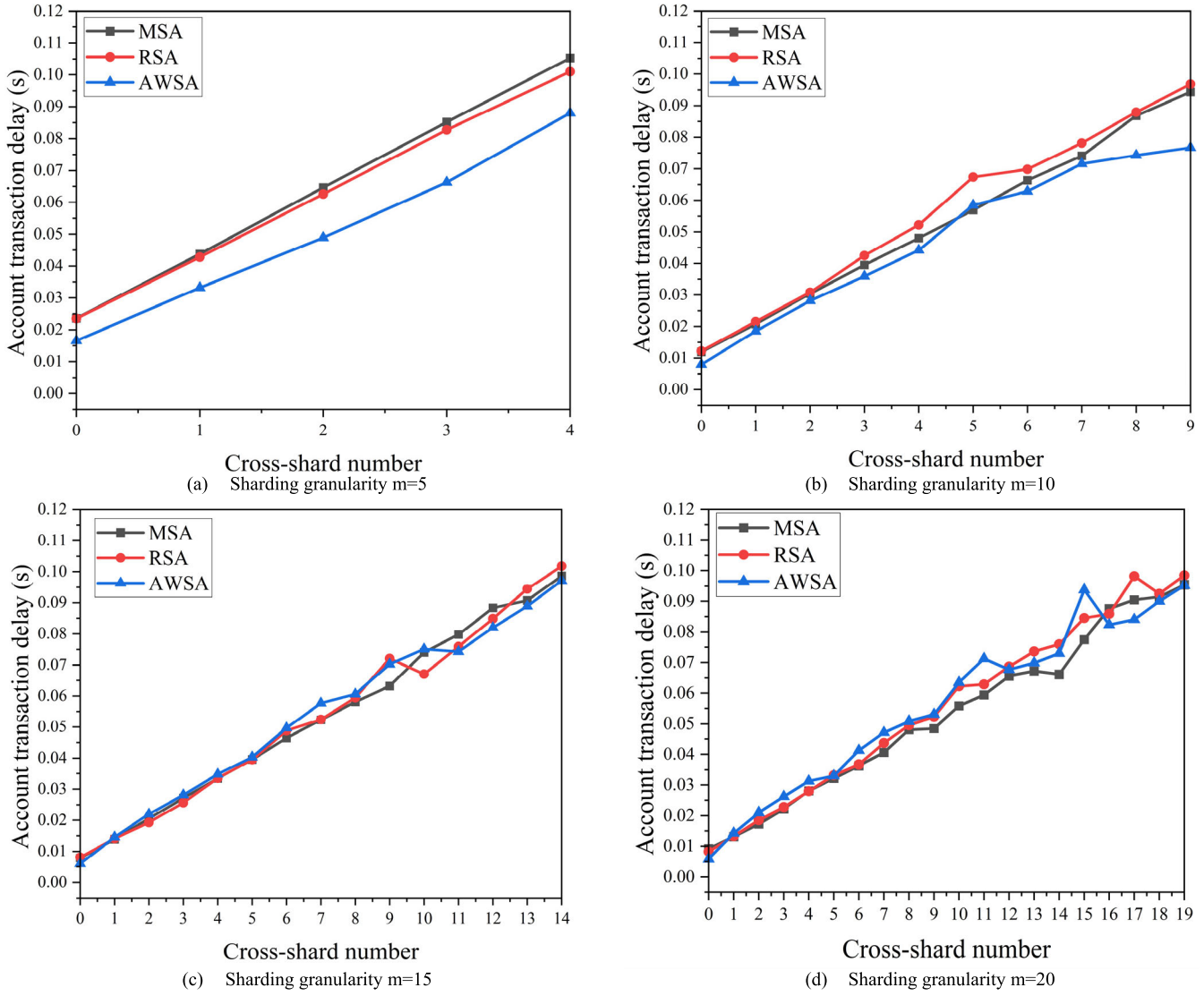

FIGURE 5. Account transaction delay with different cross-shard number.

This is an important metric. For any account, there is a certain correlation between the account transaction delay and its cross-shard number. Therefore, we introduce the account transaction delay with different cross-shard number. Fig. 5 shows the account transaction delay with different cross-shard number using the MSA algorithm, the RSA algorithm, and the AWSA algorithm with different shard granularities. As depicted in Fig. 5, the cross-shard transaction delay for all three algorithms undergoes an overall increasing trend with the increasing cross-shard number. According to Fig. 5(a) and Fig. 5(b), it is evident that the transaction delay of the AWSA algorithm is consistently lower than that of the MSA algorithm and the RSA algorithm under different shard granularities. Transaction delay is not only related to the cross-shard numbers but also to the transaction volume. Table 3 presents the maximum transaction volume proportion, the minimum transaction volume proportion, and the variance of transaction volume of the three sharding algorithms under different shard granularities. These values indicate a correlation between transaction variances and account transaction delay illustrated in Table 3. At shard granularities of 15 and 20, as depicted in Fig. 5(c) and Fig. 5(d), the account transaction delay under the AWSA algorithm exhibits greater fluctuations and often surpasses the values of the other two algorithms. The reason is revealed in Table 3 is that there are significant disparities in transaction data proportion and notable variances at shard granularities of 15 and 20.

Table 4 presents the transaction volume proportion of each shard at a shard granularity of 15. The data in Table 4 demonstrates that all three algorithms manifest an uneven distribution of transaction volumes among shards. However, the distribution of transaction volumes among shards under the AWSA algorithm is more imbalanced compared to the other two algorithms, resulting in a more pronounced occurrence of

TABLE 3. Transaction volume analysis with different sharding granularities.

Sharding granularity	MSA			RSA			AWSA		
	Max	Min	Variance	Max	Min	Variance	Max	Min	Variance
5	31.30%	9.80%	0.0058	35.07%	9.19%	0.0141	28.43%	7.03%	0.0064
10	18.44%	4.69%	0.0026	25.96%	4.33%	0.0410	19.16%	2.10%	0.0033
15	15.60%	2.82%	0.0020	15.15%	2.65%	0.0018	17.49%	1.08%	0.0026
20	16.70%	1.74%	0.0020	15.41%	1.65%	0.0020	17.67%	0.53%	0.0022

hot shard phenomena. This phenomenon implies an uneven distribution of transaction volumes among shards, which leads to different access delays for accessing different shards. Nonetheless, under a high cross-shard number, the account transaction delay under the AWSA algorithm is lower than that under the other two algorithms. Hence, the AWSA algorithm effectively reduces the account transaction delay with different cross-shard number.

TABLE 4. The distribution of transaction volume under each algorithm with the sharding granularity of 15.

Shard index	MSA	RSA	AWSA
1	3.65%	2.65%	11.16%
2	6.09%	3.64%	13.49%
3	4.16%	6.36%	1.20%
4	4.03%	5.32%	6.94%
5	5.38%	12.24%	2.77%
6	3.88%	2.72%	6.86%
7	2.86%	15.15%	17.49%
8	5.79%	11.86%	2.28%
9	2.95%	3.31%	1.08%
10	14.70%	4.43%	10.13%
11	15.60%	3.46%	6.14%
12	13.31%	4.19%	13.34%
13	3.57%	14.24%	2.10%
14	2.82%	6.96%	1.75%
15	11.22%	3.48%	3.28%

D. AVERAGE CROSS-SHARD TRANSACTION DELAY AND AVERAGE CROSS-SHARD NUMBER

The account proportion with different cross-shard number and the account transaction delay with different cross-shard number are analyzed concerning a specific shard granularity. However, in practical application, we are more concerned about the average delay on blockchain, rather than individual account's performance, such as the account transaction delay and the account cross-chain number. Therefore, we introduce the concepts of average cross-shard transaction delay and average cross-shard number. These metrics provide an intuitive understanding of the delay and cross-shard number involved in completing transactions for an average account under the sharding strategy.

The bar chart in Fig. 6 depicts the average cross-shard transaction delay under different shard granularities, while the scatter plot with connecting lines illustrates the average cross-shard number under different shard granularities. A certain correlation exists between them. Overall, no matter how shard granularity changes, the AWSA algorithm consistently maintains a lower average cross-shard transaction delay and average cross-shard number compared to the MSA algorithm and the RSA algorithm. At a shard granularity of 3, compared with the MSA algorithm and the RSA algorithm, the average transaction delay of accounts of the AWSA algorithm is reduced by 56% and 44%, so the difference in transaction delay is the most obvious. At this granularity, the average cross-shard number of accounts for both the MSA algorithm and the RSA algorithm reaches 1, while for the AWSA algorithm, it is 0.27. At a shard granularity of 10, compared with the MSA algorithm and the RSA algorithm, the average transaction delay of the AWSA algorithm is reduced by 36.41% and 44.5%. At a shard granularity of 20, the average transaction delay of accounts stabilizes, and AWSA demonstrates a reduction of 45% and 47% compared to the MSA algorithm and the RSA algorithm. At this stage, the average cross-shard number for the MSA algorithm and the RSA algorithm increases to 1.84, while for the AWSA algorithm, it is 0.43.

With the shard granularity increases, the average cross-shard transaction delay of accounts decreases for all three algorithms, and the change trends are essentially consistent. This is primarily attributed to the larger shard granularity, leading to more dispersed transactions with processed transaction data stored in various shards. Transaction verification only requires processing within the shards where account transactions exist, resulting in faster transaction validation between accounts. However, this also leads to an increase in the average cross-shard number of accounts. From the change of the average cross-shard number of accounts, with the increase of the shard granularity, the average cross-shard number of accounts of the three sharding algorithms is gradually increasing. The AWSA algorithm consistently maintains a lower average cross-shard number of accounts compared to the MSA algorithm and the RSA algorithm, and the gap becomes more obvious in higher shard granularity. Simultaneously, it is evident that the increasing trend in the average cross-shard number of accounts for the MSA algorithm and the RSA algorithm gradually slows down, while the average cross-shard number of accounts for the AWSA algorithm shows no significant change after a shard granularity of 5. In summary, the AWSA algorithm demonstrates a better ability to reduce both the average cross-shard transaction delay of accounts and the average cross-shard number of accounts.

E. TRANSACTION THROUGHPUT

Although sharding improves the throughput of blockchain, due to the complexity and difficulties in cross-shard

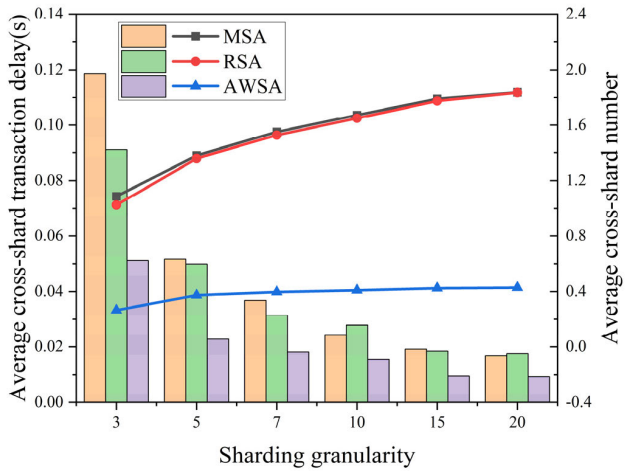

FIGURE 6. Average cross-shard transaction delay and average cross-shard number.

interaction, there still exists a significant gap between existing sharding protocols and the practical throughput requirement. Ideally, transaction throughput should increase proportionally with the shard granularity. To better validate the relationship between shard granularity and transaction throughput, we present a comparison of transaction throughputs under different shard granularities. Fig. 7 illustrates transaction throughput under different shard granularities.

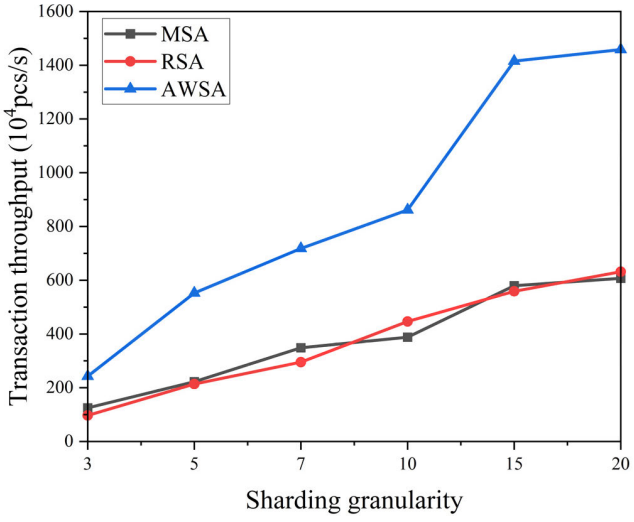

FIGURE 7. Transaction throughput.

The results indicate that, with an increase in shard granularity, the transaction throughput of all three algorithms exhibits an increasing trend, with the AWSA algorithm consistently outperforming the other two algorithms. At a shard granularity of 3, there is little difference in transaction throughput. Compared with the MSA algorithm and the RSA algorithm, the AWSA algorithm improves the transaction throughput by 94% and 150%. At a shard granularity of 10, compared with the MSA algorithm and the RSA algorithm, the AWSA algorithm improves the transaction throughput by 93% and 122%. As the transaction throughput stabilizes, specifically at a shard granularity of 20, the AWSA algorithm improves transaction throughput by 140% and 131% compared to the MSA algorithm and the RSA algorithm. This demonstrates a significant enhancement in transaction throughput with the AWSA algorithm. For the AWSA algorithm, there is a notable improvement in transaction throughput as the shard granularity increases from 10 to 15. However, from 15 to 20, there is no further improvement in transaction throughput. This suggests that the intricate relationship among accounts has approached a stable state, limiting the improvement in blockchain performance. It is evident that as transaction throughput reaches a certain level, the increasing trend in transaction throughput becomes less pronounced. This phenomenon is attributed to the diminishing benefits of parallel processing due to the increased complexity of cross-shard transactions with higher shard granularity. Experimental results indicate that the AWSA algorithm effectively enhances transaction throughput and improves blockchain performance.

Our algorithm is suitable for scenarios where historical transaction data accumulated over the long term is used for data migration. The application of the community discovery algorithm is a multi-round, step-by-step process of discovering and merging to form account sets. Although the execution process is complex, it only needs to be performed once. The time and space complexity sacrifice are aimed at reducing the cross-shard transaction proportion that is vital to impact sharding performance.

VI. CONCLUSION

Sharding technology is widely employed to address the scalability issues of blockchain, and minimizing cross-shard transactions is a crucial aspect to improve sharding performance. This paper proposed a blockchain transaction sharding algorithm based on account-weighted graph to tackle the issue of high cross-shard transaction proportion resulting from the adoption of sharding technology on blockchain. Firstly, the account-based transaction sharding model was established, and transaction data was divided from the perspective of transaction sharding to realize sharding verification and processing. Secondly, the blockchain transaction sharding algorithm based on account-weighted graph is proposed. The account-weighted graph, which takes the transaction frequency between accounts as the weight of the edge, can indirectly show the correlation between accounts, and the weighted graph is analyzed to place the strong-correlation accounts in the same shard. We compare the proposed AWSA algorithm with the classical sharding approaches, namely the MSA algorithm and the RSA algorithm. The proposed algorithm outperforms existing methods in various performance metrics, including cross-shard transaction proportion, account proportion with different cross-shard number, account transaction delay with different cross-shard number, average cross-shard transaction delay, average

cross-shard number, and transaction throughput. Experimental results demonstrate that the AWSA algorithm can effectively decrease the cross-shard transaction proportion and reduce cross-shard transaction delay.

In future work, we will focus on the problem of hot data aggregation after sharding, and consider the balance of sharding transaction volume and dynamic sharding strategy to improve sharding performance.

REFERENCES

[1] H. M. Aboalsamh, L. T. Khrais, and S. A. Albahussain, "Pioneering perception of green fintech in promoting sustainable digital services application within smart cities," *Sustainability*, vol. 15, no. 14, p. 11440, Jul. 2023.
[2] L. T. Khrais, "Comparison study of blockchain technology and IOTA technology," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Palladam, India, Oct. 2020, pp. 42–47.
[3] J. Kan, S. Chen, and X. Huang, "Improve blockchain performance using graph data structure and parallel mining," in *Proc. 1st IEEE Int. Conf. Hot Inf.-Centric Netw. (HotICN)*, Shenzhen, China, Aug. 2018, pp. 173–178.
[4] Y. Liu, X. Xing, H. Cheng, D. Li, Z. Guan, J. Liu, and Q. Wu, "A flexible sharding blockchain protocol based on cross-shard Byzantine fault tolerance," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2276–2291, 2023.
[5] C. Pan, Z. Liu, and Z. Liu, "Research on scalability of blockchain technology: Problems and methods," *J. Comput. Res. Develop.*, vol. 55, no. 10, pp. 2099–2110, Oct. 2018.
[6] R. Nourmohammadi and K. Zhang, "Sharding and its impact on fork probability," in *Proc. IEEE 1st Global Emerg. Technol. Blockchain Forum, Blockchain Beyond (iGETblockchain)*, Irvine, CA, USA, Nov. 2022, pp. 1–6.
[7] Z. Cai, J. Liang, W. Chen, Z. Hong, H.-N. Dai, J. Zhang, and Z. Zheng, "Benzene: Scaling blockchain with cooperation-based sharding," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 2, pp. 639–654, Feb. 2023.
[8] Z. Jin, "Research on scaling technology of Bitcoin cash blockchain," *Electron. World*, vol. 5, pp. 122–123, Mar. 2018.
[9] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," 2018, *arXiv:1804.00399*.
[10] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2018, pp. 949–966.
[11] F. Wang, L. Li, and N. Tian, "Multiple rounds of PBFT verification scheme to improve scale and validity of sharding," *Comput. Eng. Appl.*, vol. 56, no. 24, pp. 102–108, Dec. 2020.
[12] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. Accessed: Jan. 14, 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[13] H. S. Shahhoseini, E. Saleh Kandzi, and M. Mollajafari, "Nonflat surface level pyramid: A high connectivity multidimensional interconnection network," *J. Supercomput.*, vol. 67, no. 1, pp. 31–46, Jan. 2014.
[14] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 17–30.
[15] H. Huang, W. Kong, X. Peng, and Z. Zheng, "Survey on blockchain sharding technology," *Comput. Eng.*, vol. 48, no. 6, pp. 1–10, May 2022.
[16] C. Li, H. Huang, Y. Zhao, X. Peng, R. Yang, Z. Zheng, and S. Guo, "Achieving scalability and load balance across blockchain shards for state sharding," in *Proc. 41st Int. Symp. Reliable Distrib. Syst. (SRDS)*, Vienna, Austria, Sep. 2022, pp. 284–294.
[17] L. Li, Y. Wu, Z. Yang, and Y. Peng, "Medical electronic medical record sharing scheme based on partitioned blockchain," *J. Comput. Appl.*, vol. 42, no. 1, pp. 183–190, Jun. 2022.
[18] H. Huang, Y. Zhao, and Z. Zheng, "TMPT: Reconfiguration across blockchain shards via trimmed Merkle patricia trie," in *Proc. IEEE/ACM 31st Int. Symp. Quality Service (IWQoS)*, Orlando, FL, USA, Jun. 2023, pp. 1–10.
[19] J. Li, Y. Wang, and Y. Gao, "An anti-collusion attack network sharding algorithm for blockchain," *Comput. Appl. Res.*, vol. 40, no. 1, pp. 28–32, Sep. 2023.
[20] K. Aiyar, M. N. Halgamuge, and A. Mohammad, "Probability distribution model to analyze the trade-off between scalability and security of sharding-based blockchain networks," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2021, pp. 1–6.
[21] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2018, pp. 583–598.
[22] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX. Symp. Net. Systems. Design. Impl. (NSDI'19)*, Feb. 2019, pp. 95–112.
[23] M. Cortes-Goicoechea, L. Franceschini, and L. Bautista-Gomez, "Resource analysis of Ethereum 2.0 clients," in *Proc. 3rd Conf. Blockchain Res. Appl. Innov. Netw. Services (BRAINS)*, Paris, France, Sep. 2021, pp. 1–8.
[24] F. Cassez, J. Fuller, and A. Asgaonkar, "Formal verification of the Ethereum 2.0 Beacon chain," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer Int. Publishing, 2022, pp. 167–182.
[25] L. Brünjes and M. J. Gabbay, "Utxo-vs account-based smart contract blockchain programming paradigms," in *Lever. Applications of Formal Methods, Verification and Validation: Applications*. Berlin, Germany: Springer Int. Publishing, 2020, pp. 73–88.
[26] S. Nakamoto. (2008). *Bitcoin: A Peer-to-peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[27] H. Huang, X. Peng, J. Zhan, S. Zhang, Y. Lin, Z. Zheng, and S. Guo, "BrokerChain: A cross-shard blockchain protocol for account/balance-based state sharding," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, London, United Kingdom, May 2022, pp. 1968–1977.
[28] X. Zhang, "Research on blockchain sharding method based on transaction feature analysis," M.S. thesis, Dept. Computer, Shiyou Univ., Xian, China, 2023.
[29] Z. Wang, "A combined micro-block chain truncation attack on Bitcoin-NG," in *Information Security and Privacy*. Berlin, Germany: Springer Int. Publishing, 2019, pp. 322–339.
[30] M. Agarwal, S. Biswas, and S. Nandi, "Discrete event system framework for fault diagnosis with measurement inconsistency: Case study of rogue DHCP attack," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 789–806, May 2019.
[31] M. Li, W. Wang, and J. Zhang, "LB-Chain: Load-balanced and low-latency blockchain sharding via account migration," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 10, pp. 2797–2810, Oct. 2023.
[32] S. M. Mohtavipour, M. Mollajafari, and A. Naseri, "A novel packet exchanging strategy for preventing hol-blocking in fat-trees," *Cluster Comput.*, vol. 23, no. 2, pp. 461–482, Jun. 2020.
[33] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech. Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

JIAO LI received the Ph.D. degree from Northwestern Polytechnical University, China, in 2017. She is currently an Associate Professor with the School of Computer Science, Xi'an Shiyou University, China. Her current research interests include blockchain communication and blockchain scalability.

YUANHANG NING was born in Henan, China. He received the bachelor's degree in the Internet of Things engineering. He is currently pursuing the master's degree in computer science and technology with Xi'an Shiyou University. His research interests include blockchain sharding technology.