Edge Computing and Lightning Network Empowered Secure Food Supply Management

Keke Gai, *Senior Member, IEEE,* Zhengkang Fang, Ruili Wang, Liehuang Zhu, *Member, IEEE,* Peng Jiang, Kim-Kwang Raymond Choo, *Senior Member, IEEE*

Abstract—The recent COVID-19 pandemic pandemic has highlighted the importance of food safety and supply chain governance. In other words, we need to ensure traceability along the supply chain, support high-frequency transactions and effective data collections, etc. Thus, we posit the potential of using lightning network, which is a decentralized traceable paradigm for achieving high-frequency transactions in blockchain-based systems. In addition, we also utilize edge computing to help facilitate data collection. However, a key challenge in securing food supplies is determining the optimal global transaction path in lightning network, while achieving efficiency and meeting the dynamic nature of food supply management. Thus, we propose a blockchain-edge scheme that utilizes our proposed dynamic programming to produce optimal solutions to selecting global transaction paths. Specifically, our scheme optimizes routing fees under existing constraints (e.g., transmission cost, computing resource consumption, and lightning network balance). Findings from our evaluations demonstrate that the utility of our proposed approach in facilitating food safety management.

Index Terms—Blockchain, lightning network, dynamic programming, edge computing, food supply management, optimization

1 Introduction

The significance of *Food Safety Management* (FSM) and food supply chain is self-evident, particularly during the recent COVID-19 pandemic [1]–[3]. Typically, the FSM consists of a broad range of stakeholders ranging from food producers to logistical operators to retailers to regulators to consumers, and so on [4], [5]. One of the challenges in FSM and food supply chain is the lack of or inefficient administration / coordination, which resulted in information asymmetry for both upstream and downstream providers. This is, perhaps, unsurprising due to information silos between the different stakeholders. For example, service providers in the supply chain generally have their own or use different datacenters, maintain their own systems that do not share information with each other, etc. Hence, there is a risk of data tampering, data provenance, and so on [6]–[8].

Given the recent interest in blockchain [9]–[11], it is no surprise that blockchain-based FSM systems have also been explored in the literature [12]–[14]. For example, due to the decentralized nature of blockchain it has been used to facilitate data sharing and ensure reliability [15]–[18]. However, a key challenge of blockchain-based (and many other) systems is to ensure the reliability of the data source. In addition, FSM systems generally need to support high volume data flow, for example for data gathered from different sensing nodes and systems, and real-time data analysis. Existing blockchain-based systems are generally incapable of supporting such high throughput. For example, in the Bitcoin network, the average block time is around 10 minutes and about 7 TPS (transaction per second). Clearly, this is not suitable for deployment in FSM setting [19]. Even though consortium blockchain does not rely on mining, which minimizes the consensus time [20]–[22], optimization on throughput capacity is still needed.

Lightning network is a technology that can be used to carry out real-time transactions in the off-chain channel without the need to upload data to the blockchain. The cross-node transaction paths in lightning network generate the routing fee, which is proportional to the number of channels involved in the transaction path and fee rate. Optimizing path selection can effectively reduce the associated lightning network fees, but establishing transaction chan-

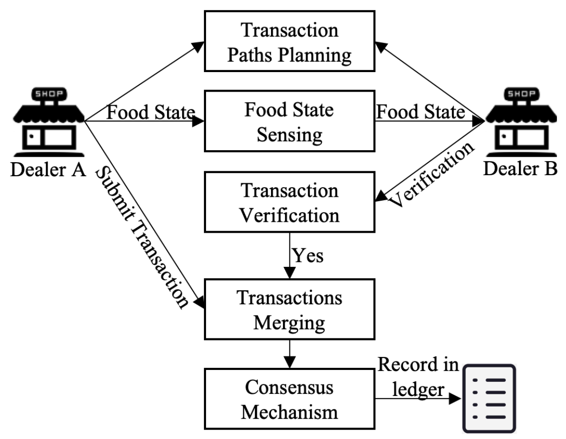


Fig. 1: Architecture of our proposed model.

- *K. Gai (first author) Z. Fang, L. Zhu and P. Jiang are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081; K. Gai, L. Zhu and P. Jiang are also with the School of Cyberspace Security, Beijing Institute of Technology, Beijing, China, {gaikeke, 1120161858, liehuangz, pengjiang}@bit.edu.cn.*
- *R. Wang is with the School of Natural and Computational Sciences, Massey University, Auckland, New Zealand. Email: Ruili.wang@massey.ac.nz.*
- *K.-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, 78249, USA, raymond.choo@fulbrightmail.org.*
- *Corresponding Author: Ruili Wang (email: Ruili.wang@massey.ac.nz)*
- *Co-corresponding Author: Liehuang Zhu (email: liehuangz@bit.edu.cn)*

nels between each pair of nodes is technically challenging due to the topology radiation [23]. In general, let us assume that there is a large number of transaction paths passing through a central node. The balance of lightning network will be affected when the channel's capacity fails to meet the transaction demand. Hence, optimizing path selections is an alternative method for increasing lightning network performance for blockchain-based systems.

Hence, in this paper we propose a blockchain-based FSM system that uses lightning network to meet the exacting demands (e.g., high efficiency). As shown in Fig. 1, our proposed FSM optimization blockchain (*FSMOB*) model comprises edge sensing network nodes that facilitate both authorization and data collection. We use RFID chips to uniquely label the food items in the edge sensing nodes, which will allow us to collect real-time information of food states. Thus, this avoids the need to manually input food state data and reduce the possibility of source data tampering or error. Moreover, each edge sensing node functions as both the lightning network node and the blockchain network node, which completes real-time transactions over the lightning network and constructs a consensus between both ends of the channel. The integrated transaction data will not be uploaded to the blockchain by the channel established by nodes until the transaction channel in lightning network is closed.

To explain our problem formulation, we use simple paths on weighted directed graph to represent the transaction path and edges on the graph denote channels. We also propose a dynamic programming algorithm to optimize path selections in the lightning networks, and the lightning network routing fees can be optimized under minimal constraints (e.g., transmission cost, computing resource consumption, and lightning network balance). The proposed problem has been proven to be a NP-hard problem (see Section 3).

The main contributions of this paper are as follows:

1) Design of an edge-blockchain model that uses lightning networks to facilitate high efficiency in FSM. Specifically, the proposed blockchain system relies on food state sensing and lightning network transactions, which ensures data trustworthiness and efficiency.
2) Design of a dynamic programming algorithm to solve the lightning network transaction path selection problem (i.e., multi-dimensional optimization problem). The proposed algorithm optimizes the lightning network routing fees under the constraints of transmission cost, computing resource consumption, and lightning network balance, with the aim of obtaining the optimal transaction path.

The rest of the work is organized in the following order. We briefly summarize the related literature in Section 2. Next, Sections 3 and 4 describe the proposed model and algorithms, respectively. The evaluation setup and findings are presented in Section 5. Finally, Section 6 concludes the work.

2 Related Work

As discussed in the preceding section, a number of blockchain-based FSM and food supply chain systems have been proposed in the literature [24]–[26]. For example, Lin *et al.* [27] proposed to use the Electronic Product Code Information Services *EPCIS* and blockchain in the FSM setting. Specifically, they sought to ensure data traceability and tamper-resistance. Our work differs from this approach in the sense that we seek to improve the lightning network technology to achieve higher-level throughput for the blockchain-based systems.

Salah *et al.* [28] proposed an approach to effectively execute business transactions using blockchain and smart contracts, in the context of tracking and tracing soybeans throughout the agricultural supply chain. Such an approach omits the need for trusted centralized authorized agencies and intermediaries, and the blockchain transaction records facilitate food traceability. However, the approach does not consider the low throughput of blockchain-based systems. The blockchain-based approach of Hua *et al.* [29] does not consider the scalability of their proposed blockchain-based system.

Similar to our work, other researchers have utilized both blockchain and edge computing [30]–[32]. For example, Fitwi *et al.* [33] proposed a lightweight privacy protection scheme that uses edge monitoring cameras deployed on the blockchain. The system assumes that the security video monitoring system does not capture personal or private data during the monitoring process, which is clearly impractical.

To maximize optimization of blockchain throughput, a number of researchers introduced solutions that utilize Internet of Things (IoT) devices [34]–[36]. Unlike most of these existing approaches, we use the lightning network to build real-time transaction off-chain channels for achieving higher-level of blockchain throughput.

There have also been approaches based on lightning network. For example, Armani *et al.* [37] proposed a method combining Bitcoin blockchain with lightning network to achieve fast and secure transactions. However, their work does not address optimization of the lightning network. The approach of [38] focused on optimization and included a simple cost structure to achieve reasonable short path length and overall balanced channel capacity. A more recent work [39] utilizes the deep Q-network (DQN)-based technique for task migration in mobile edge computing system. Differing from prior attempts, our approach is designed to obtain an optimal solution to resource management in lightning networks in a blockchain-edge FSM system.

3 Proposed Model

3.1 Model Design

The major goal of the model is faithfully and efficiently recording food transfer process. Fig. 2 illustrates the workflow of the proposed model. As food safety regulatory department, government issues and verifies the certificate to participants. The proposed model records the whole process of food transaction process and food states in food supply chain for food source traceability. The general process of the

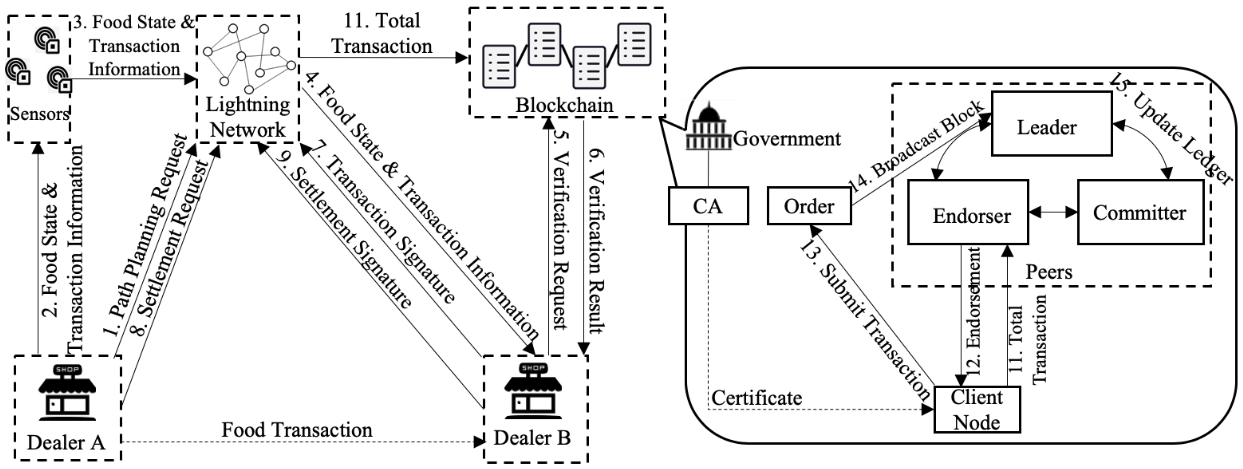


Fig. 2: Workflow of proposed FSMOB model.

model is divided into following phases, including transaction paths planning, food state sensing, transaction verification, transactions merging, and uploading to blockchain.

3.1.1 Transaction Paths Planning

Due to the introduction of lightning network, the proposed model needs to execute transactions in channels of lightning network. During the settlement, the total transaction is uploaded to blockchain for recording. We need to generate a cross nodes transaction path before transaction as channels cannot directly be established between two nodes. The transaction sender proposes transaction path planning request to lightning network; then, lightning network performs the transaction path planning process.

The core of lightning network is to build a point-to-point transaction off-chain channel for intra channel transactions. However, the complexity of large-scale distributed network inevitably leads to the impossibility of establishing direct transaction channels among all nodes. Therefore, constructing cross nodes transaction paths by successively connecting transaction channels is necessary. The path of cross nodes transaction between two nodes is not unique. Different paths selection may bring different transaction performance because the network environment and channel capacity of each transaction channel are different The path selection problem is an optimization problem under multi-dimensional constraints (routing fees, transmission costs, computing resource consumptions and network balances). Our approach proposes a dynamic programming algorithm to deal with the transaction path planning problem. The model needs to plan transaction path before executing transaction to obtain the global optimal path selection. Specific solution is given in Section 3.3.

The transaction paths selection scheme aims to obtain the optimal paths satisfying the constraints. Transmission cost are considered to be constraints, especially in large-scale distributed networks, which are embodied in transmission latency of transaction channel.

We also set the network balance constraint to avoid redundant transaction paths. When there are overload cross node transaction path through the central node, cross node transaction failure may occur, unless the transaction paths are re-planned. In our approach, we consider setting network balance constraint a great improvement of real-time transaction in many emergency situations, e.g., preventing pandemic.

3.1.2 Food State Sensing

Collecting food state and transfer data is necessary before executing the transaction. Client forms transaction request and submit the transaction request to lightning network for processing. Food state sensing collects real-time food state data through sensors and reads food ID in RFID tags. The food state data are relayed to receiver of transaction.

The process of the food recorded transaction needs to be uniquely identified by RFID passive tag. RFID reader reads RFID tags to obtain unique codes. Sensors automatically obtain the food state of the transaction sender after transaction paths planning. Food state information collected by sensor is recorded, ($State$). Food code is recorded as ID.

Moreover, we suggest that collected food state data be encrypted before transmiting to lightning network. Signed and encrypted food codes ($E(Sign(ID))$) are stored in the RFID passive tag. The collected data are denoted by $\{State, E(Sign(ID))\}$. RFID reader decrypts $E(Sign(ID))$ and verifies the signature.

When the ID is verified, $E(Sign(State|ID))$ will be transmitted to the lightning network server. The lightning network server relays $E(Sign(State|ID))$ to transaction receiver. The transaction receiver decrypts the message and verifies whether it has been tampered. In this phase, we use sensors and RFID reader to read the food state of the food to be traded and transmit it after encryption to avoid manual input, which increases the technical cost of tampering with food state source data and improves the reading rate. Within the controllable cost range, it provides support for tamper-resistant off-chain data.

3.1.3 Transaction Verification

After receiving food state data, the transaction receiver submits a verification request to blockchain, queries records of the food in blockchain ledger, and verifies whether the food is traded to meet the food safety standard/ requirement. The verification result will be used to make a decision on whether the transaction is executed.

The transaction receiver constructs the received food state, $\{State, ID\}$, as validation request V. All nodes on blockchain are readable to blockchain ledger. The participating nodes of blockchain can execute verification contract in smart contract by submitting the verification request V. The contract uses ID as a key to read the value in ledger and compare it with the state in V for assessing whether food transaction meets the food safety standards. Finally, the verification result V_r is sent back to transaction receiver.

The validation process of transactions can effectively implement food management norms. Comparing with traditional methods, it is more effective to reject all transactions that do not meet food safety standards before implementation. The verification process verifies each transaction, based on a partially decentralized consortium blockchain, to make verification process more transparent and credible. Logically speaking, this pre-transaction validated FSM method cuts off the circulation of COVID-19 virus carrying food before virus may be transmitted through food. The management is more fine-grained.

3.1.4 Transactions Merging

Cross nodes transactions need to execute atomic transactions in each transaction channel in turn. It means that the cross nodes transaction is completed when all atomic transactions are completed. After the completion of the transaction, lightning network will not immediately submit the transaction to blockchain but continues to wait for other transactions. Each transaction in channel updates the food quantity allocation scheme in channel. Finally, in the channel settlement, a combined total transaction is constructed according to the final allocation scheme of channel prestored food.

The transaction signed by sender is blocked in lightning network and wait for the signature confirmation of transaction receiver when a transaction is initiated. $Sign_s$ and $Sign_r$ are the signature of sender and receiver of transaction, respectively. Once the transaction T is signed by both parties as $Sign_r(Sign_s(\mathrm{T}))$, the system updates channel food balance state C_b to complete a transaction in channel, according to the transaction. The channel executes multiple transactions before settlement and each transaction updates channel food balance state C_b as described above. Cross nodes transaction needs to update food balance state C_b in all channels of transaction path. When either party of the transaction initiates a settlement request, both parties need to sign and confirm the channel food balance state in final settlement within a specified time period. After channel food balance state C_b is signed as $Sign_r(Sign_s(C_b))$, channel constructs the final balance state into a total transaction and submits it to blockchain.

Lightning network carries out real-time transactions in an off-chain channel by merging multiple transactions into a total transaction and uploading them to blockchain. This avoids a consensus process for every transaction. In the scenario of FSM, such throughput cannot meet needs of high-frequency food trading. The proposed method of merging transactions with lightning network can significantly reduce the trigger frequency of consensus process, which is the most important factor to improve the throughput of model. The data security of lightning network is guaranteed by signature of both parties.

As described in the above workflow, when lightning network processes transactions in channel, both sides of transaction need to sign and confirm the each transaction's balance update for final channel balance state confirmation. This ensures that the total transaction submitted to blockchain is the result of consensus made by both parties.

3.1.5 Uploading to Blockchain

The total transaction of channels is submitted by lightning network to blockchain for consensus and finally is recorded in blockchain ledger. We divide the consensus mechanism into three steps, namely, endorsement, order, and confirmation. Endorsement is to simulate the execution of transaction and sign the simulation results to construct consensus of the transaction execution results. Ordering is to make the order of multiple submitted transactions for consensus. Confirmation is to verify whether the packaged block meets the consensus result. The committer nodes update their own ledgers when the confirmation is verified.

Based on our application scenario, there are *Certificate Authority* (CA) nodes in FSMOB that are acted by regulators, and CA nodes provide access permission for blockchain participants. On the premise that the identity of participants is clear, We assume that the probability of nodes becoming malicious nodes is low. Therefore, based on the consideration of consensus efficiency of blockchain, we choose partially decentralized consortium blockchain for deployment.

First, the client node submits the transaction proposal to endorsement nodes. The endorsement nodes simulate the transaction according to the transaction information in the proposal and sign the simulated execution result back to the client. The simulation execution result is the reading and writing sets of the ledger state database. Byzantine fault tolerance requires at least $(3f+1)$ nodes to reach a consensus when the number of malicious nodes in the blockchain is f. When the client receives $2f$ identical endorsement results (add itself is $2f+1$), it can be considered that a consensus has been reached on the simulated execution results of the transaction. The client submits the endorsement result together with the transaction request to the order nodes network. After *Practical Byzantine Fault Tolerance* (PBFT) is applied to complete the four processes of pre-prepare, prepare, commit and reply, we conclude that all nodes have reached a consensus on the sequencing of multiple transactions. Finally, the order nodes pack the ordered transactions into a block and send it to the leader node; the leader node confirms the consensus result and forwards the block to the committer nodes. Committer nodes update the transaction states to its own ledger. Finally, consumers and regulator submit query requests to ledger in FSMOB through the client nodes.

The consensus mechanism ensures the consistency of each committer node. All blockchain participants share all transaction data recorded in the ledger of food supply chain. In COVID-19 epidemic situation, the value of epidemic prevention is to avoid forging food transaction records to cheat the management system, resulting in the circulation of virus carrying wild animals and food in supply chain.

3.2 Problem Formulation

Performance of the transaction path selection scheme is directly related to lightning network's efficiency. Distinct schemes in different applications will result in different routing fees, transmission costs, computing resource consumptions, and network balances. In this paper, the proposed problem is finding out an optimal global transaction paths scheme in lightning network. We define the problem entitled *Optimal Fee Transaction Path* (OFTP) problem in Def. (1).

Definition 1. **Optimal Fee Transaction Path** (OFTP) problem: **Inputs:** the transaction channel set $\{Ch_i\}$ $(0 \leq i \leq n)$ of all the existing nodes in the lightning network; establish the set $\{P_j\}$ $(0 \leq j \leq m)$ of all the transaction paths, establish the matrix P-matrix of all the paths alternatives, and the variable quintuples of each path to be selected recorded in the P-matrix (F_{P_j}, T_{P_j}, C_{P_j}, $RMax_{P_j}$, $RMin_{P_j}$), the upper limit of transmission cost constraint is T^c, the upper limit of computing resource consumption constraint is C^c, and the upper limit of network balance constraint is B^c. We define a binary function $p(P_j)$. A value of 1 indicates that the transaction path P_j is selected, and a value of 0 indicates that the transaction path P_j is not selected. **Output:** find a transaction paths selection scheme with the lowest routing fee F. This problem can be described as follows:

$$F = Min[\sum_{p(P_j)=1} F_{P_j}]. \tag{1}$$

$$T = \sum_{p(P_j)=1} T_{P_j}. \tag{2}$$

$$C = \sum_{p(P_j)=1} C_{P_j}. \tag{3}$$

$$B = \underset{p(P_j)=1}{Max}(RMax_{P_j}) - \underset{p(P_j)=1}{Min}(RMin_{P_j}). \tag{4}$$

Find the value of F, under constraint $0 \leq T \leq T^c$, $0 \leq C \leq C^c$, $0 \leq B \leq B^c$.

In Eq. (1), $Min[\sum_{p(p_j)=1} F_{p_j}]$ refers to the minimum value of routing fee of transaction paths scheme in all transaction paths schemes. $RMax_{P_j}$ and $RMin_{P_j}$ refer to the maximum and minimum residual capacity of each channel in a transaction path. $\underset{p(P_j)=1}{Max}(RMax_{P_j})$ refers to the maximum channel residual capacity of each transaction path after $RMax_{P_j}$ comparison. Similarly, $\underset{p(P_j)=1}{Min}(RMin_{P_j})$ refers to the minimum value of the minimum channel residual capacity of each trading path after $RMin_{P_j}$ comparison. We use the difference between the maximum value and the minimum value of the global channel residual capacity to evaluate the network balance. The value of the difference has a negative relationship with the performance of the network balance. The network balance requires that our problem meet the constraint of $\underset{p(P_j)=1}{Max}(RMax_{P_j}) - \underset{p(P_j)=1}{Min}(RMin_{P_j})$.

In the process of generating a sequence of channels to be selected for a transaction path, we define the $s(Ch_i)$ binary function. If the transaction channel Ch_i is selected, then the binary function value $s(Ch_i)$ is 1, otherwise 0. The routing fee of each channel is f_{Ch_i}, the transmission cost is t_{Ch_i}, the computing resource consumption is c_{Ch_i}, and the residual capacity of the channel is r_{Ch_i}. F_{P_j} is the total routing cost of transaction path P_j.

$$F_{p_j} = \sum_{s(Ch_i)=1} f_{Ch_i}. \tag{5}$$

T_{P_j} is the total transmission cost of transaction path P_j.

$$T_{p_j} = \sum_{s(Ch_i)=1} t_{Ch_i}. \tag{6}$$

C_{P_j} is the total computing resource consumption of transaction path P_j. when the channel is not selected for transaction path construction, its computing resource consumption is very small and can be ignored.

$$C_{p_j} = \sum_{s(Ch_i)=1} c_{Ch_i}. \tag{7}$$

$RMax_{P_j}$ is the maximum of the residual capacity of each transaction channels in transaction path P_j, and $RMin_{P_j}$ is the minimum of the residual capacity of each transaction channels in transaction path P_j. $Max_{s(Ch_i)=1}()$ and $Min_{s(Ch_i)=1}()$ refer to the maximum and minimum residual capacity of each channel in a transaction path, respectively. From this we can get:

$$RMax_{P_j} = \underset{s(Ch_i)=1}{Max}(r_{Ch_i}). \tag{8}$$

$$RMin_{P_j} = \underset{s(Ch_i)=1}{Min}(r_{Ch_i}). \tag{9}$$

In conclusion, the OFTP problem can be expressed as follows:

$$F = min[\sum_{p(P_j)=1} \sum_{s(Ch_i)=1} f_{Ch_i}]. \tag{10}$$

$$T = \sum_{p(P_j)=1} \sum_{s(Ch_i)=1} t_{Ch_i}. \tag{11}$$

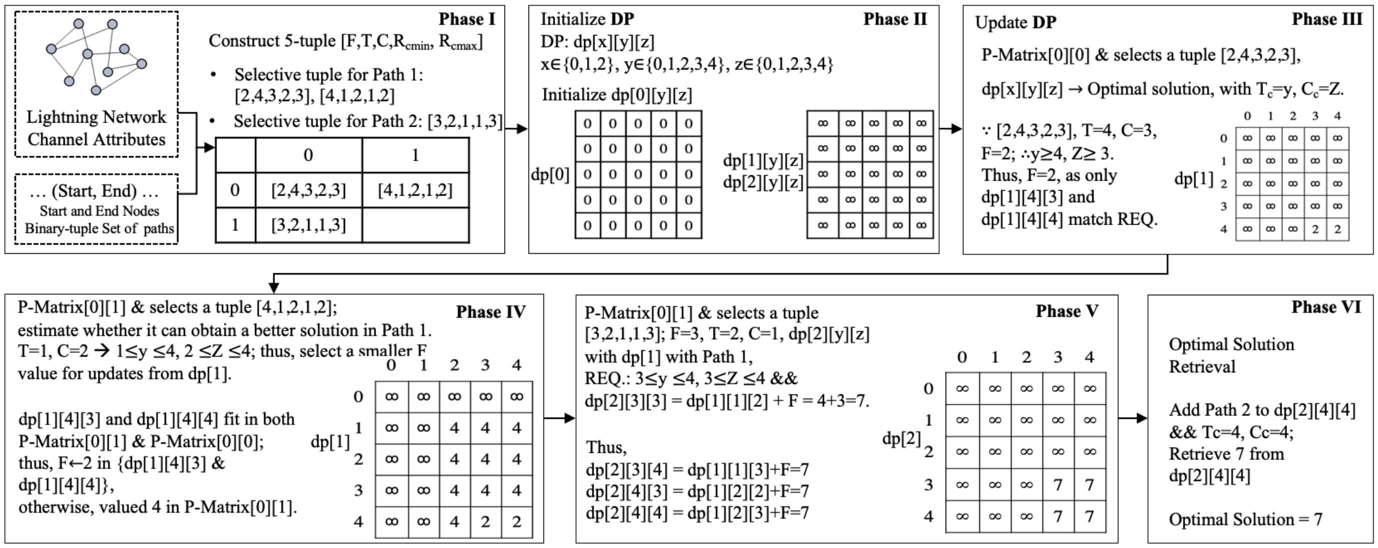
$$C = \sum_{p(P_j)=1} \sum_{s(Ch_i)=1} c_{Ch_i}. \tag{12}$$

$$B = \underset{p(P_j)=1}{Max}(\underset{s(Ch_i)=1}{Max}(r_{Ch_i})) - \underset{p(P_j)=1}{Min}(\underset{s(Ch_i)=1}{Min}(r_{Ch_i})). \tag{13}$$

Under the constraints of $0 \leq T \leq T^c$, $0 \leq C \leq C^c$, $0 \leq B \leq B^c$, the transaction paths scheme with the value of F is obtained.

Theorem 1 and Proof 1 demonstrate the proposed problem is an NP-hard problem.

Theorem 1. $\exists$ variables f_{Ch_i}, t_{Ch_i}, c_{Ch_i}, r_{Ch_i} and binary functions $p(P_j)$, $s(Ch_i)$. It is known that the OFTP problem is defined as:
$F = min[\sum_{p(P_j)=1} \sum_{s(Ch_i)=1} f_{Ch_i}]$,


Phase I
Lightning Network Channel Attributes
… (Start, End) … Start and End Nodes Binary-tuple Set of paths
Phase II
Phase III
dp[1]
Phase IV
Phase V
dp[2]
Phase VI

Construct 5-tuple [F,T,C,R_{cmin}, R_{cmax}]

- Selective tuple for Path 1: [2,4,3,2,3], [4,1,2,1,2]
- Selective tuple for Path 2: [3,2,1,1,3]

	0	1
0	[2,4,3,2,3]	[4,1,2,1,2]
1	[3,2,1,1,3]	

Initialize **DP**
DP: dp[x][y][z]
x∈{0,1,2}, y∈{0,1,2,3,4}, z∈{0,1,2,3,4}
Initialize dp[0][y][z]

dp[0]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

dp[1][y][z]
dp[2][y][z]

∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞

Update **DP**
P-Matrix[0][0] & selects a tuple [2,4,3,2,3],
dp[x][y][z] → Optimal solution, with T_c=y, C_c=Z.
∵ [2,4,3,2,3], T=4, C=3, F=2; ∴y≥4, Z≥ 3.
Thus, F=2, as only dp[1][4][3] and dp[1][4][4] match REQ.

	0	1	2	3	4
0	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞
4	∞	∞	∞	2	2

P-Matrix[0][1] & selects a tuple [4,1,2,1,2];
estimate whether it can obtain a better solution in Path 1.
T=1, C=2 → 1≤y ≤4, 2 ≤Z ≤4; thus, select a smaller F value for updates from dp[1].

dp[1][4][3] and dp[1][4][4] fit in both P-Matrix[0][1] & P-Matrix[0][0];
thus, F←2 in {dp[1][4][3] & dp[1][4][4]},
otherwise, valued 4 in P-Matrix[0][1].

	0	1	2	3	4
0	∞	∞	∞	∞	∞
1	∞	∞	4	4	4
2	∞	∞	4	4	4
3	∞	∞	4	4	4
4	∞	∞	4	2	2

P-Matrix[0][1] & selects a tuple [3,2,1,1,3]; F=3, T=2, C=1, dp[2][y][z] with dp[1] with Path 1,
REQ.: 3≤y ≤4, 3≤Z ≤4 &&
dp[2][3][3] = dp[1][1][2] + F = 4+3=7.

Thus,
dp[2][3][4] = dp[1][1][3]+F=7
dp[2][4][3] = dp[1][2][2]+F=7
dp[2][4][4] = dp[1][2][3]+F=7

	0	1	2	3	4
0	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	7	7
4	∞	∞	∞	7	7

Optimal Solution Retrieval

Add Path 2 to dp[2][4][4] && Tc=4, Cc=4;
Retrieve 7 from dp[2][4][4]

Optimal Solution = 7

Fig. 3: A motivational example of the Optimal Fee model's work flow.

$$T = \sum_{p(P_j)=1} \sum_{s(Ch_i)=1} t_{Ch_i},$$
$$C = \sum_{p(P_j)=1} \sum_{s(Ch_i)=1} c_{Ch_i},$$
$$B = \underset{p(P_j)=1}{Max}\ (\underset{s(Ch_i)=1}{Max}\ (r_{Ch_i})) - \underset{p(P_j)=1}{Min}\ (\underset{s(Ch_i)=1}{Min}\ (r_{Ch_i})).$$

The constraints of this problem are $0 \leq T \leq T^c$, $0 \leq C \leq C^c$ and $0 \leq B \leq B^c$. Then this OFTP problem is a NP-hard problem.

Proof 1. The ∃ variable N_n represents the total number of nodes in the lightning network. Under the definition of Theorem 1, the F value of global optimal transaction path scheme is obtained. If the traversal algorithm is used, the number of all transaction paths to be selected and $N_n!$ are in direct proportion, the time complexity of the optimal solution of all transaction path schemes is $O(((N_n)!)!)$. This problem can be reduced to a three-dimensional bin-packing (*3D-BP*) problem. F is regarded as the minimum value of the goods in the bin. Constraints T^c, C^c, B^c can be regarded as the length, width and height of the bin. The selected transaction path is the loaded goods. We can prove that *OFTP* problem can be reduced to three-dimensional bin-packing problem, *OFTP* $\leq$ 3D-BP. Since the problem is a NP-hard problem, *OFTP* problem is also a NP-hard problem.

3.3 Optimal Fee Model

The input of this model includes the attribute values of each transaction channel in the lightning network and the start and end nodes binary-tuple set of all transaction paths to be established. Transaction channel attributes include routing fee, transmission cost, computing resource consumption and channel residual capacity. Each binary-tuple *(start, end)* corresponds to a transaction path between start node and end node to be established. Fig. 3 illustrates a motivational example of the work flow for our proposed scheme.

Based on the input, we use a *Path Weight Configuration* (PWC) algorithm to search all accessible simple paths from the start node to the end node of each binary-tuple, and calculate their corresponding weights quintuple (F_{P_j}, T_{P_j}, C_{P_j}, $RMax_{P_j}$, $RMin_{p_j}$). All the options of the transaction path to be established corresponding to each binary-tuple are the quintuple of all the simple paths calculated in the previous step. All options for the transaction path to be established form the P-Matrix.

P-Matrix is the input of *ptimal Transaction Paths Scheme* (OTPS) algorithm. In this paper, we use the idea of dynamic programming to deal with the Optional transaction path schemes in P-matrix. Then *OTPS* algorithm can get the global optimal transaction path schemes to be established, and output the optimal transaction path scheme as S-Matrix. Because the optimal solution of i path to be established is *dp [i]*, then the optimal solution of the first *i + 1* path to be established is *dp [i + 1] = DP [i] + value [i]*. According to the properties of the optimal substructure, we can construct the optimal solution from bottom to top. In the end, we can get the optimal transaction path scheme which makes the fee minimum and satisfies the constraints.

4 Algorithms

Path Weight Configuration (PWC) algorithm is a transaction path weight calculation algorithm running on lightning network. The main function of PWC algorithm is to preprocess channel attributes and transaction paths to be established information before executing Alg. (2), and generates weight quintuple of path to be selected as the input of Alg. (2). The input of *PWC* algorithm is the adjacency table G which stores the topological structure and attributes of established channel, and the table P which stores (start,end) pairs of transaction paths to be established. Process each transaction path to be established in sequence.

First, the next node satisfying the simple path condition is found in adjacency domain of the starting point, and the node is added to the path to form a shorter simple path. Then, the attributes quintuple of the current path is updated, and the new node is used as the starting point to continue to find the next node. In this way, new nodes are added until

Algorithm 1 Path Weight Configuration Algorithm

```
Require: G, P
Ensure: P-Matrix
 1: for i ← 0 to n do
 2:     u ← P[i].start, v ← P[i].end
 3:     cnt ← 0
 4:     function $Find\_Way$(G, u, v, found)
 5:         if u == v then
 6:             u ← P[i].start
 7:             for j ← 0 to found do
 8:                 P-Matrix[i].path[cnt].way[j] ← way[j]
 9:                 f ← f + G.attr[u][way[j]].f
10:                 t ← t + G.attr[u][way[j]].t
11:                 c ← c + G.attr[u][way[j]].c
12:                 if (G.attr[u][way[j]].rc − P[i].volume) > rcmax then
13:                     rcmax ← G.attr[u][way[j]].rc − P[i].volume)
14:                 end if
15:                 if (G.attr[u][way[j]].rc − P[i].volume) < rcmin then
16:                     rcmin ← G.attr[u][way[j]].rc − P[i].volume)
17:                 end if
18:             end for
19:             P-Matrix[i].path[cnt].f ← f
20:             P-Matrix[i].path[cnt].t ← t
21:             P-Matrix[i].path[cnt].c ← c
22:             P-Matrix[i].path[cnt].rcmax ← rcmax
23:             P-Matrix[i].path[cnt].rcmin ← rcmin
24:             P-Matrix[i].size ← cnt
25:             cnt++
26:         end if
27:         for w ← G.nodes[u].firstArc; w; w ← w.nextArc do
28:             if !$In\_the\_way$(G.nodes[w.agjvex].data) then
29:                 way[found] ← G.nodes[w.agjvex].data
30:                 $Find\_Way$(G, w.agjvex, v, found +1)
31:             end if
32:         end for
33:     end function
34: end for
35: return P-Matrix
```

the end of the path is added to the path. After adding the end, add the final quintuple of path to be established to the P-matrix. Then it returns to the node of the previous joining path, and then selects next node in its adjacency domain to join the path. The *Depth First Search* (DFS) process is repeatedly executed in the above recursive manner. Finally, we can get the algorithm output P-Matrix, which stores all the optional path attributes quintuples of all the paths to be selected. We need to set five temporary variables *f*, *t*, *c*, *rcmax* and *rcmin* when updating the attributes quintuple of found simple path. According to the problem definition in Section 3, *f*, *t*, *c* are updated by accumulating the corresponding attribute values of channels. The temporary variables *rcmax* and *rcmin* are updated by selecting the maximum and minimum channel residual capacity on the current path to assign *rcmax* and *rcmin* respectively. The key stages of the algorithm are as follows.

1) The next node joining path is found in adjacency domain of the starting point. The judgment that satisfies the condition is to judge whether the node has been added to the path. If a node has been added, it may form a ring, which does not meet the requirements of a simple path. Therefore, if the condition is not satisfied, it returns to the previous node directly.
2) Judge whether to join the next node in adjacency domain of the previous node. If the condition is satisfied, the node is added to the path and the next node that can join the path is found in its adjacency domain. When a node is added to path, five temporary variables that record attributes quintuple of current simple path need to be updated.
3) When the end of path is added to path, the value of temporary variables (final attributes quintuple of path to be selected) is assigned to P-Matrix.
4) The above process is repeated to depth first search for G which stores channel attributes and the channel topology. Get the P-Matrix which holds all attributes of path to be established and output it as the input of Alg. (2).

The output P-Matrix of Alg. (1) is the input of Alg. (2), so Alg. (1) is the basis of Alg. (2). Alg. (2) filters all possible solutions in Algorithm 1 and gets the optimal solution.

OTPS algorithm is a transaction paths selection algorithm running on Lightning network. The optimal solution is established to obtain the minimum routing fee F under the constraints of transmission cost T_c, computing resource consumption C_c and network balance B_c. Therefore, the input includes all quintuple matrix P-Matrix and constraints generated by Alg. (1), T_c, C_c, B_c.

First, the algorithm will judge each path three rounds under possible constraints. The first round is to determine whether the maximum difference of channel residual capacity on the selected path is less than the network balance constraint. If it is less than the constraint, the path does not meet the constraint condition; otherwise, the subsequent judgment is made. In the second round, judge whether a better solution can be obtained comparing the first round's path. update the dp array and $S-Matrix$ if a better solution can be obtained. The third round is to judge whether a better path exists after the group already selected the path. Update the dp array and $S-Matrix$ if a better path can be found. The optimal solution can be obtained by circularly processing all the paths to be selected. Next, we introduce main phases of OTPS algorithm.

1) After traversing the candidate path of each path to be established, quintuples of path attributes stored in P-Matrix are extracted for each path to be established; the routing fee F transmission cost T and resource consumption C are recorded; $Rcmin$ and $Rcmax$ are updated with the current channel minimum remaining capacity and maximum remaining capacity.
2) In different constraint ranges, we judge whether network balance constraint condition is satisfied; the

Algorithm 2 Optimal Transaction Paths Scheme Algorithm

```
Require: P-Matrix
Ensure: S-Matrix
 1: RCmax ← 0, RCmin ← $\infty$
 2: Set all cells in dp as 0
 3: for i ← 0 to n do
 4:     for j ← 0 to P-Matrix[i].size do
 5:         if P-Matrix[i].path[j].rcmax > RCmax then
 6:             Rcmax ← P-Matrix[i].path[j].rcmax
 7:         end if
 8:         if P-Matrix[i].path[s].rcmin < RCmin then
 9:             Rcmin ← P-Matrix[i].path[j].rcmin
10:         end if
11:         F ← P-Matrix[i].path[j].f
12:         T ← P-Matrix[i].path[j].t
13:         C ← P-Matrix[i].path[j].c
14:         B ← RCmax − RCmin
15:         for k ← Tc to 0 do
16:             for s ← Cc to 0 do
17:                 if B ≤ Bc then
18:                     if dp[i][k - T][s - C] != $\infty$ then
19:                         if dp[i][k][s] > dp[i][k -T][s - C] + F then
20:                             dp[i][k][s] = dp[i][k - T][s - C] + F
21:                             Update j to S-Matrix
22:                         end if
23:                     end if
24:                     if dp[i - 1][k - T][s - C] != $\infty$ then
25:                         if dp[i][k][s] > dp[i - 1][k -T][s - C] + F then
26:                             dp[i][k][s] = dp[i - 1][k - T][s - C] + F
27:                             Add j to S-Matrix
28:                         end if
29:                     end if
30:                 end if
31:             end for
32:         end for
33:     end for
34: end for
35: return S-Matrix
```

range of constraints conditions are $[0, T_c]$, $[0, C_c]$. Then, it is determined whether the maximum difference of channel residual capacity on the selected path $Rcmax - Rcmin$ is less than the network balance constraint B_c. If $Rcmax - Rcmin > B_c$, the path to be selected does not meet the constraint condition; otherwise, the subsequent judgment is made.

3) The solution satisfying all constraints is judged twice. The first step is to determine whether a better solution can be obtained when the group chooses the path for the first time. If $dp[i][k][s] > dp[i][k-T][s-c] + F$, we can obtain a better solution to update the dp array and $S - Matrix$. The second step is to determine whether the better solution can be obtained by using this path replacement after the group has selected the path. If $dp[i][k][s] > dp[i-1][k-T][s-C] + F$, we can obtain a better solution to update the dp array and $S - Matrix$.

4) Loop through the above processes until all paths to be selected are processed. Finally, the optimal solution is obtained.

Theorem 2. Global transaction paths scheme processed by OTPS algorithm is the optimal solution to obtain the minimum routing fee under constraints.

Proof 2. Step 1: When there is only one path to be established, the path with the lowest routing fee in P-Matrix [0] is the optimal solution. Step 2: Assumes that S-Matrix [i] is the optimal solution of the first i paths to be established. S-Matrix [i + 1] consists of paths with the lowest fee in S-Matrix [i] and P-Matrix [i + 1], and because paths with the lowest fee in S-Matrix [i] and P-Matrix [i + 1] are all the optimal solutions, S-Matrix [i + 1] is the optimal solution of i + 1 paths to be established. Theorem 2 is proved.

Time complexity analysis: we make a brief analysis of the time complexity of the algorithm, assuming that there are n paths to be established, with N nodes in total. Two dimensional constraints are T^c and C^c. In order to calculate the optimal solution, we need to deal with the path establishment n times. The number of times we need to calculate is directly proportional to T^c and C^c for each path to be established. The number of paths to be selected for each path to be established is proportional to N^2. Therefore, the time complexity is $O(NT^cC^cN^2)$.

5 Evaluation Setup and Findings

In this section, we will describe our setup and present our findings (in comparison to other three schemes).

5.1 Experiment Configuration

In our evaluation comparison, we used five groups of performance analysis with different parameters (see Table 1). In each group, the performance of OTPS algorithm and other three algorithms were compared under the same parameter configuration.

The parameter configuration is as follows. We defined the number of paths to be established as N_p, which denotes the total number of paths to be established by the algorithm. The constraint value of transmission cost (T_c) represents the maximum total transmission cost that the system could tolerate after all paths are established, the computing resource consumption constraint value (C_c) denotes the maximum consumption of computing resources that the system could carry, and the network balance constraint value (B_c) represents the maximum difference of channel capacity.

The three other competing approaches used in the evaluations are as follows: the heuristic algorithm method proposed by Stasi and Avallone [40], the Greedy 1 algorithm (that uses a *Less-Goes-First* policy to create the path, and the *Greedy 2* algorithm (that uses the quotient of fee divided by the upper bound of constraints as the target of the greedy). In the *Greedy 2* algorithm, it obtains the greedy

Groups	N_p	T_c	C_c	B_c
Setting1	4	40	40	20
Setting2	7	70	70	20
Setting3	10	100	100	20
Setting4	4	40	40	15
Setting5	7	50	70	20

TABLE 1: Parameters Settings in the Evaluations.

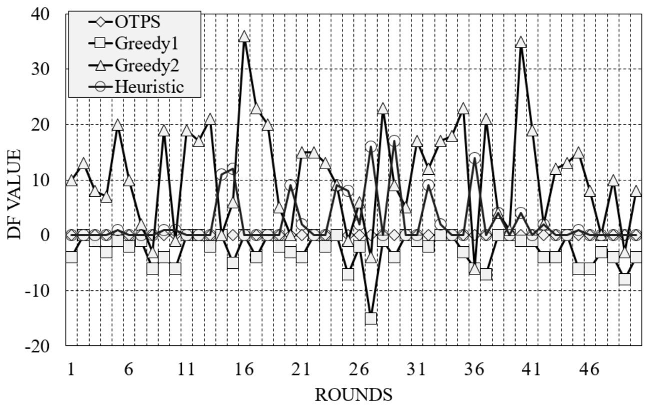


Fig. 4: DF values under setting 1: A comparative summary.

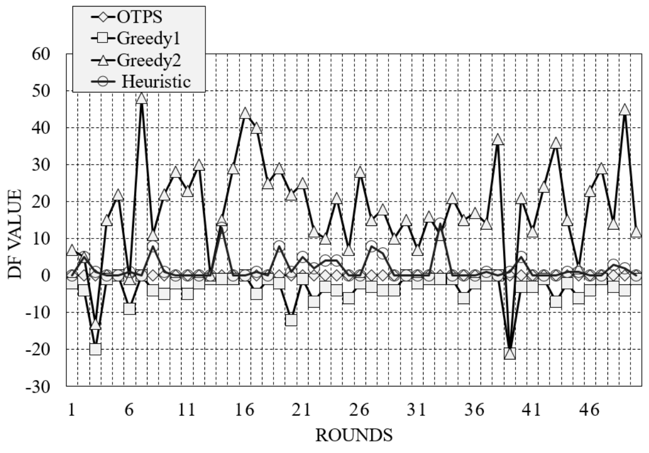


Fig. 5: DF values under setting 2: A comparative summary.

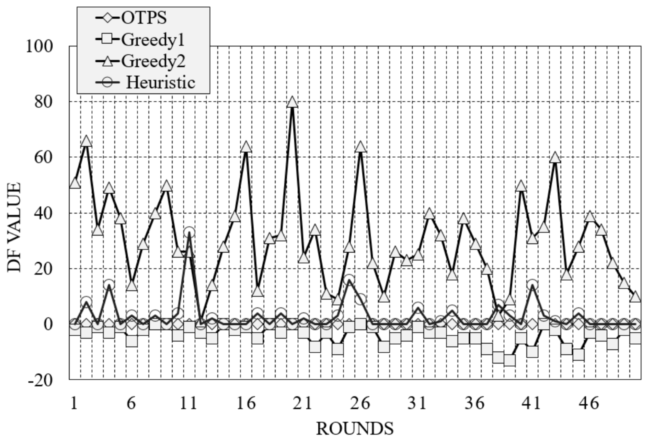


Fig. 6: DF values under setting 3: A comparative summary.

choice process of each path to be established, calculates the quotient of fee divided by constraints of all the optional paths, and selects the optional paths with the lowest value. These processes are carried out to calculate all paths to be established and form the final paths selection scheme. We compared the difference between the final output route fee and the optimal route fee (*DF*) obtained from all four approaches. The actual runtime (*RT*) of each scheme with the same input processing was then compared.

The experiment was performed on a personal computer (PC) with i5-6400 CPU and 8G memory, and running Windows 10. We used Python 3.6 to write the simulation program. First, we generated the input matrix of the test algorithm and used random numbers to generate the values of the variables in the input matrix. Next, we inserted as input the same matrix for each comparison algorithm. Finally, the algorithm path scheme with the corresponding routing fee value and the actual execution time were produced as output. We conducted 50 rounds of tests for each set of parameter configuration.

In our empirical study, we statistically analyzed the optimal route fee hit rate (*HR*) and compared the specific *HR* value of the four algorithms. In order to comprehensively evaluate the performance of the four algorithms, we set the evaluation criteria *EC*, derived from $EC = DF/HR$. Our evaluations focused on the differences between routing and optimal routing fees. In other words, the higher hit rate of routing fee, the better the optimization effect (i.e., an optimal scheme has a lower *EC* value).

5.2 Findings

From Fig. 4 to Fig. 6, we observed that our approach has relatively stable performance, when the number of needed paths (N_p) increases. The findings also demonstrated that our approach's DF values are 0 under all settings, so that the optimal values are obtained at all times. For the other schemes, the fluctuation range of deviation tends to increase as N_p increases. The degree of deviation from the optimal value of Greedy 1 algorithm is much lower than that of Greedy 2. In other words, the results clearly shows that our approach acheives optimal solutions regardless of the N_p value.

As shown in Fig. 4 and Fig. 7, B_c values differ under settings 1 and 4, where other parameters remain consistent. We found that the deviation of the optimal value of the Heuristic algorithm increases under a stricter B_c constraint. The deviation degree of two greedy algorithms slightly changes.

Results shown in Fig. 5 and Fig. 8 suggests that T_c value reduces from settings 2 to 5. In line with the presentations of T_c and C_c given in Sections 3 and 4, we concluded that the impacts of T_c and C_c were similar. In addition, the deviation degrees of both greedy algorithms significantly increases, but not the Heuristic algorithm.

From Fig. 9 to Fig. 11, one can observe that the runtime of each scheme are consistent around a certain value under the same setting, shown by the smooth waves. Both greedy algorithms are fast and their runtime could be ignored. Our approach's runtime is shorter than that of the Heuristic

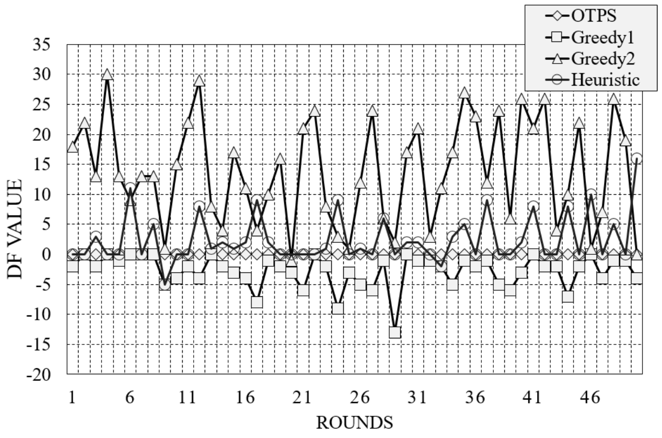


Fig. 7: DF values under setting 4: A comparative summary.

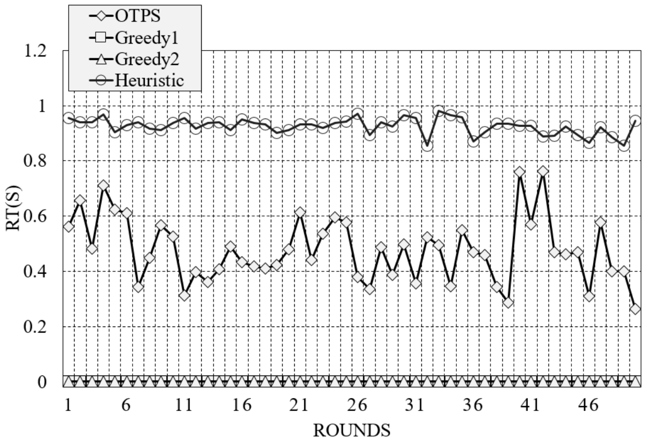


Fig. 9: Runtime comparisons under setting 1.

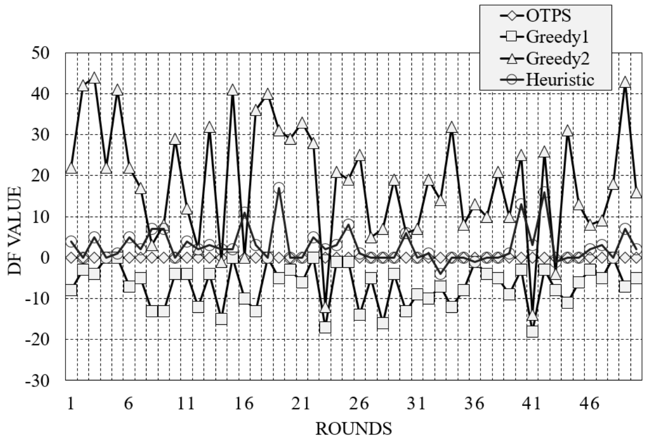


Fig. 8: DF values under setting 5: A comparative summary.

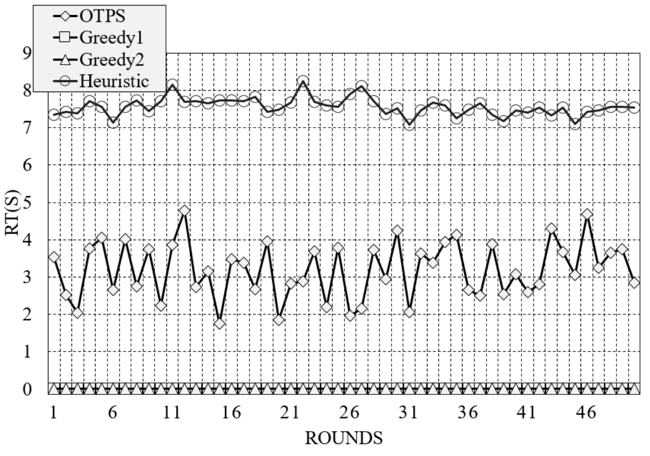


Fig. 10: Runtime comparisons under setting 2.

algorithm. The execution time increases when N_p increases in all settings. The growth rate of the execution time with the increase of N_p follows the order below. *Heuristic* $\gg$ OTPS $\gg$ *Greedy 1* $\approx$ *Greedy 2*.

Fig. 14 shows the optimal values of hit rates in all settings. When the routing fee output is equal to the optimal value, the hit time increments by 1. Finally, the ratio of hit times to the total number of rounds in a group was taken as the *HR* value.

Moreover, we compared the impact of B_c on the execution time (*RT*). As shown in Fig. 9 and Fig. 12, both greedy algorithms are not affected by the decrease of B_c. The execution time of Heuristic algorithm is greatly reduced, and the *RT* value of our approach also significantly decreases and the extent of decline is more pronounced in the Heuristic algorithm. We also evaluated the influence of T_c(equivalent to C_c) on *RT* values. As depicted in Fig. 10 and Fig. 13, both greedy algorithms are not affected by T_c. The *RT* values of Heuristic algorithm and our approach greatly reduce and the change rate of RT value of OTPS is much greater than that of the Heuristic algorithm.

As shown in Table 2 and Fig. 14, our approach has a stable hit rate of 100% but the other schemes decrease with an increasing N_p value under settings 1, 2, and 3. Based on the *HR* value comparisons under settings 1 and 4, we found that the reduction of B_c reduces the hit rate of all schemes, with the exception of our approach. Next, the decrease of T_c and C_c results in the decrease of hit rate of heuristic algorithm, from the *HR* value comparisons under settings 2 and 5. It was also observed that the hit rates follow a progressive order of Greedy 2, Greedy 1, Heuristic, and our approach.

We evaluated the standard EC from $EC = DF/HR$, where EC is obtained from dividing the average of DF values by HR. We observed that a lower-level of the routing fee deviated from the optimal value is associated with a higher hit rate and a smaller *EC* value. Thus, smaller *EC* value implies more optimal output – see Fig. 15.

Furthermore, as shown in Table 3 and Fig. 15, as N_p increases, the performance of the three schemes degrades (*EC* increases). Based on the comparison results of *EC* under settings 1 and 4, a decreasing B_c impacts on the performance on the three schemes. From findings of settings 2 and 5, T_c and C_c have similar impacts. Hence, our approach outperforms these three schemes.

5.3 Discussion

We observed that our approach outperforms the other schemes in all given settings, due to the stable optimal

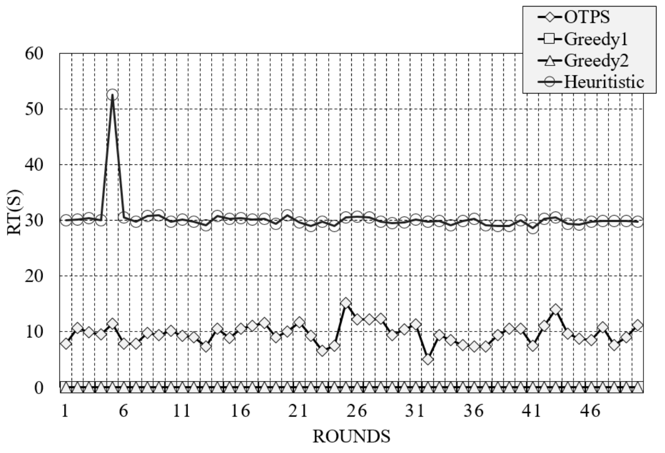


Fig. 11: Runtime comparisons under setting 3.

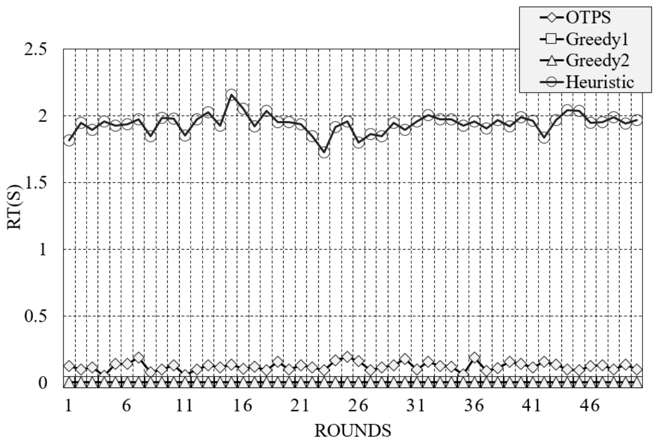


Fig. 13: Runtime comparisons under setting 5.

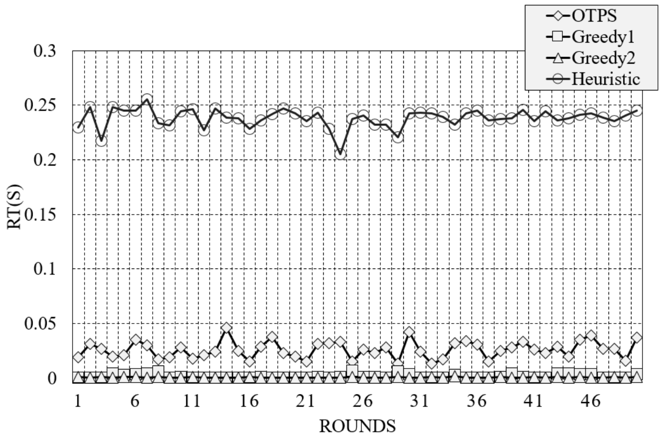


Fig. 12: Runtime comparisons under setting 4.

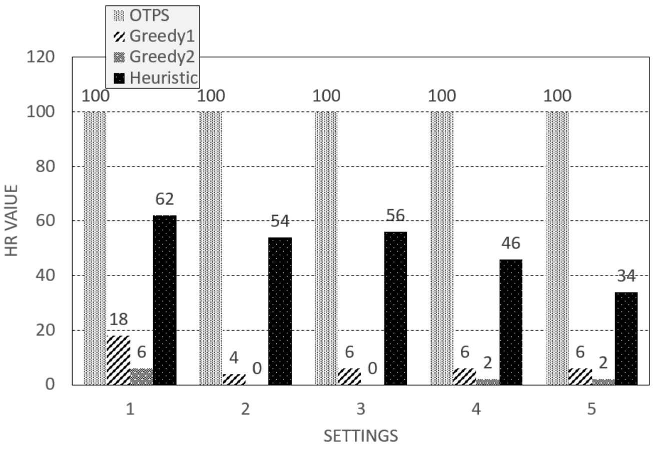


Fig. 14: HR value comparisons under three settings.

solution outputs. For example, the hit rates are not affected by parameters settings, and this implies that our approach consistently optimizes the planning of lightning network path selections. The findings support our deductions in Sections 3 and 4.

Our approach has a shorter execution time, as well as obtaining a stable optimal solution. Especially, the execution time has a huge advantage in the case of strict constraints, which would improve the real-time performance of the model. In other words, the *OTPS* algorithm has a better performance compared to the three other approaches in real-time environment (smaller T_c), more stringent computing resource constraints (smaller C_c), and higher network balance requirements (smaller B_c).

An factor on real-time performance in our proposed model is the capacity of off-chain transactions in lightning network. As discussed in prior sections, lightning networks significantly reduce time latency (shorter time period of transaction uploads), as off-chain transactions and consensus frequency are merged. In our evaluation, we configured the time cost of each transaction through consensus mechanism to upload to the blockchain as t_B, and transaction execution time in lightning network channel is denoted as t_L. We assumed that the capacity of channel could execute n transactions continuously. Let T_B represents the total time consumption when n transactions are not processed through the lightning network, and T_L represents the total time consumption when n transactions are processed through the lightning network. Therefore, we arrive at the following mathematical expressions:

$$T_B = nt_B$$

$$T_L = \sum_{n=1}^{n} \frac{it_L + t_B}{n}$$

$$\overline{T_B} = t_B$$

$$\overline{T_L} = \frac{(n+1)t_L}{2n} + \frac{t_B}{n}$$

Since $t_B \gg t_L$, we arrive at $\overline{T_B} \gg \overline{T_L}$. In other words, the use of lightning network can greatly improve the real-time performance of transactions.

6 Conclusion

The importance of food safety in our society and ensuring sufficient food supply will become more pronounced as our population increases. Seeking to address limitations in existing approaches, we proposed a blockchain-based FSM system that also utilizes edge computing and lightning

Settings	OTPS	Greedy1	Greedy2	Heuristic
Setting 1	100	18	6	62
Setting 2	100	4	0	54
Setting 3	100	6	0	56
Setting 4	100	6	2	46
Setting 5	100	6	2	34

TABLE 2: HR value of algorithms in different parameters settings.

Settings	OTPS	Greedy1	Greedy2	Heuristic
Setting 1	0	0.15	1.83	0.04
Setting 2	0	0.2	3.04	0.031
Setting 3	0	0.22	5.1	0.048
Setting 4	0	2.76	13.82	2.48
Setting 5	0	6.68	18.56	2.84

TABLE 3: EC value of algorithms in different parameters settings.

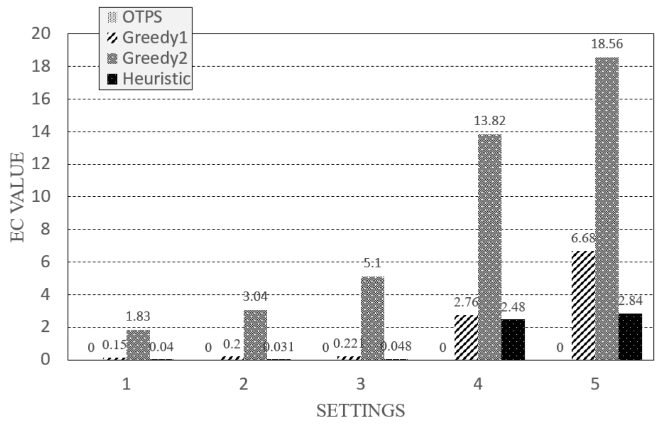


Fig. 15: EC value comparison of four algorithms under five parameters configurations.

network. Specifically, our proposed system sends supply chain transactions through the lightning network channels off-chain and uploads them to the blockchain for record-keeping. In addition, we use dynamic programming to optimize transaction path of the lightning network to improve the performance of the system, which is more practical than directly using blockchain to record transaction information.

Future research includes implementing a prototype of our proposed system, within a real-world food supply ecosystem to evaluate its performance and identify any potential shortcomings.

Acknowledgment

The authors would like to thank all anonymous reviewers for their valuable comments and suggestions to improve this paper. This work is partially supported by Natural Science Foundation of Beijing Municipality (Grant No. 4204111, 4202068), National Natural Science Foundation of China (Grant No. 61972034, 61902025), Natural Science Foundation of Shandong Province (Grant No. ZR2019ZD10), Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201803), Henan Key Laboratory of Network Cryptography Technology (Grant No. LNCT2019-A08), Beijing Institute of Technology Research Fund Program for Young Scholars (Keke Gai, Peng Jiang), and the Cloud Technology Endowed Professorship (Kim-Kwang Raymond Choo).

References

[1] M. Haghani, M. Bliemer, F. Goerlandt, and J. Li, "The scientific literature on coronaviruses, COVID-19 and its associated safety-related research dimensions: A scientometric analysis and scoping review," *Safety Science*, p. 104806, 2020.

[2] C. Galanakis, "The food systems in the era of the coronavirus (COVID-19) pandemic crisis," *Foods*, vol. 9, no. 4, p. 523, 2020.

[3] M. Rizou, I. Galanakis, T. Aldawoud, and C. Galanakis, "Safety of foods, food supply chain and environment within the COVID-19 pandemic," *Trends in Food Science & Technology*, vol. PP, no. 99, p. 1, 2020.

[4] X. Xu, F. Rahman, B. Shakya, A. Vassilev, D. Forte, and M. Tehranipoor, "Electronics supply chain integrity enabled by blockchain," *ACM Transactions on Design Automation of Electronic Systems*, vol. 24, no. 3, pp. 1–25, 2019.

[5] M. Du, Q. Chen, J. Xiao, H. Yang, and X. Ma, "Supply chain finance innovation using blockchain," *IEEE Transactions on Engineering Management*, vol. PP, no. 99, p. 1, 2020.

[6] Y. Kang, I. Park, J. Rhee, and Y. Lee, "Mongodb-based repository design for iot-generated rfid/sensor big data," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 485–497, 2016.

[7] S. Liu, H. Zheng, H. Meng, H. Hu *et al.*, "Study on quality safety traceability systems for cereal and oil products," in *WRI World Congress on Software Engineering*, vol. 1, 2009, pp. 163–166.

[8] M. G. C. A. Cimino, B. Lazzerini, F. Marcelloni, and A. Tomasi, "Cerere: an information system supporting traceability in the food supply chain," in *Seventh IEEE International Conference on E-Commerce Technology Workshops*, 2005, pp. 90–98.

[9] J. Bao, D. He, M. Luo, and K.-K. R. Choo, "A survey of blockchain applications in the energy sector," *IEEE Systems Journal*, 2020.

[10] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 62–75, 2019.

[11] L. Soltanisehat, R. Alizadeh, H. Hao, and K.-K. R. Choo, "Technical, temporal, and spatial research challenges and opportunities in blockchain-based healthcare: A systematic literature review," *IEEE Transactions on Engineering Management*, 2020.

[12] T. Feng, "An agri-food supply chain traceability system for china based on rfid blockchain technology," in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, 2016, pp. 1–6.

[13] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," in *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, 2018, pp. 1–4.

[14] D. Tse, B. Zhang, Y. Yang, C. Cheng, and H. Mu, "Blockchain application in food supply information security," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management*, 2017, pp. 1357–1361.

[15] D. Tse, B. Zhang, Y. Yang, C. Cheng, and H. Mu, "Blockchain application in food supply information security," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2017, pp. 1357–1361.

[16] L. Zhu, Y. Wu, K. Gai, and K. Choo, "Controllable and trustworthy blockchain-based cloud data management," *Future Generation Computer Systems*, vol. 91, pp. 527–535, 2019.

[17] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3548–3558, 2019.

[18] W. Dai, C. Dai, K. R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 725–737, 2020.

[19] M. Chow, Z. Lai, C. Liu, E. Lo, and Y. Zhao, "Sharding blockchain," in *IEEE International Conference on Internet of Things*, 2018, pp. 1665–1665.

[20] R. Han, V. Gramoli, and X. Xu, "Evaluating blockchains for iot," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, pp. 1–5.

[21] M. M. Jalalzai, C. Busch, and G. G. Richard, "Proteus: A scalable bft consensus protocol for blockchains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 308–313.

[22] S. Yu, K. Lv, Z. Shao, Y. Guo, J. Zou, and B. Zhang, "A high performance blockchain platform for intelligent devices," in *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2018, pp. 260–261.

[23] E. Erdin, M. Cebe, K. Akkaya, E. Bulut, and A. S. Uluagac, "A heuristic-based private bitcoin payment network formation using off-chain links," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 294–301.

[24] M. Zhaofeng, W. Xiaochang, D. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2013–2021, 2019.

[25] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in DWSNs," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, p. 1, 2020.

[26] C. Wang, J. Shen, J. Lai, and J. Liu, "B-TSCA: Blockchain assisted trustworthiness scalable computation for v2i authentication in VANETs," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, p. 1, 2020.

[27] Q. Lin, H. Wang, X. Pei, and J. Wang, "Food safety traceability system based on blockchain and epcis," *IEEE Access*, vol. 7, pp. 20 698–20 707, 2019.

[28] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar, "Blockchain-based soybean traceability in agricultural supply chain," *IEEE Access*, vol. 7, pp. 73 295–73 305, 2019.

[29] J. Hua, X. Wang, M. Kang, H. Wang, and F. Wang, "Blockchain based provenance for agricultural products: A distributed platform with duplicated and shared bookkeeping," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 97–101.

[30] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, p. 1, 2020.

[31] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *2018 IEEE International Conference on Communications*, 2018, pp. 1–6.

[32] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of IoT devices using blockchain-enabled fog nodes," in *IEEE/ACS 15th International Conference on Computer Systems and Applications*, 2018, pp. 1–8.

[33] A. Fitwi, Y. Chen, and S. Zhu, "A lightweight blockchain-based privacy protection for smart surveillance at the edge," in *2019 IEEE International Conference on Blockchain*, 2019, pp. 552–555.

[34] K. Gai, Y. Wu, L. Zhu, Z. Zhang, and M. Qiu, "Differential privacy-based blockchain for industrial Internet-of-Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4156–4165, 2019.

[35] Feng Tian, "A supply chain traceability system for food safety based on haccp, blockchain internet of things," in *2017 International Conference on Service Systems and Service Management*, 2017, pp. 1–6.

[36] X. Min, Q. Li, L. Liu, and L. Cui, "A permissioned blockchain framework for supporting instant transaction and dynamic block size," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 90–96.

[37] F. B. Armani, F. Grimaccia, S. Leva, and M. Mussetta, "Seamless grid: an off-chain model proposal for scalable p2p electricity markets and grids management," in *2019 IEEE Milan PowerTech*, 2019, pp. 1–6.

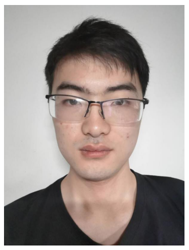
[38] A. H. Jun Ren, L. Feng, S. A. Cheong, and R. S. Mong Goh, "Optimal fee structure for efficient lightning networks," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems*, 2018, pp. 980–985.

[39] C. Zhang and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Generation Computer Systems*, vol. 96, no. JUL., pp. 111–118, 2019.

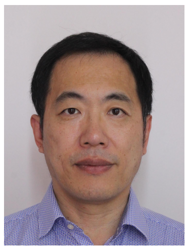
[40] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing payments on the lightning network," in *IEEE International Conference on Internet of Things*, July 2018, pp. 1161–1170.

Keke Gai [SM 20'] received the B.Eng. degree majored in automation, from Nanjing University of Science and Technology, Nanjing, China, in 2004, the M.E.T. (Master's of Educational Technology) degree in educational technology from the University of British Columbia, Vancouver, BC, Canada, in 2010, the MBA degree in business Administration, in 2009, M.S. degree in information technology, in 2014, from the Lawrence Technological University, Southfield, MI, USA, and the Ph.D. degree in computer science from Pace University, New York, NY, USA.

He is currently an Associate Professor at the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He has published 2 books and more than 130 peer-reviewed journal/conference papers. He has been granted 5 IEEE Best Paper awards (e.g. TrustCom 18' and HPCC 18') and 2 IEEE Best Student Paper awards in recent 5 years. His research interests include cyber security, blockchain, edge computing, cloud computing, and reinforcement learning.



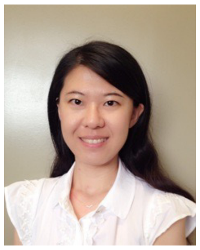
Zhengkang Fang received his Bachelor's degree in Internet of things engineering from Beijing Institute of Technology, China, in 2020. His research interests include blockchain, cloud computing and network security.



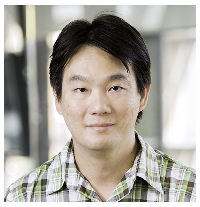
Ruili Wang rreceived the Ph.D. degree in Computer Science from Dublin City University, Dublin, Ireland. Currently, he is the Professor of Artificial Intelligence and the Chair of Research with the School of Natural and Computational Sciences, Massey University, Auckland, New Zealand, where he is the Director of the Centre of Language and Speech Processing. His current research interests include data processing, speech and natural language processing, image and video processing, and intelligent systems.

Liehuang Zhu received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004, the M.E. (Master of Engineering) degree and B.E. (Bachelor of Engineering) degree from Wuhan University, Wuhan, China, in 2001 and 1998, respectively.

He is currently a professor at School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China. He has published more than 100 peer-reviewed journal or conference papers, including 10+ IEEE/ACM Transactions papers (IEEE TIFS, IEEE TII, IEEE TVT, IEEE TSG, Information Sciences, IEEE Network, Computer & Security, etc.). He has been granted a number of IEEE Best Paper Awards, including IWQoS 17', TrustCom 18'. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing.



Peng Jiang received her Ph.D. degree from Beijing University of Posts and Telecommunications in 2017. She is currently an Associate Professor of the School of Computer Science and Technology, Beijing Institute of Technology. Previously, she was a visiting student in University of Wollongong, Australia, and a Postdoctoral Fellow in The Hong Kong Polytechnic University, Hong Kong. Her research interests include cryptography, information security, and blockchain. She has published more than 30 papers in the area of cybersecurity and has served as a program committee member in many international conferences. Her current research interests include cyber security, blockchain and cryptography.



Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2015, he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, British Computer Society's 2019 Wilkes Award Runner-up, 2019 EURASIP JWCN Best Paper Award, Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, Inscrypt 2019 Best Student Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008.