THEORETICAL DISCUSSIONS ON BLOCKCHAIN-BASED SYSTEMS: DESIGN, EXTENSION, AND APPLICATION

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by

Suhan Jiang
May, 2022

Examining Committee Members:

Dr. Jie Wu, Dept. of Computer and Information Sciences (Advisor)
Dr. Chiu C. Tan, Dept. of Computer and Information Sciences
Dr. Justin Y. Shi, Dept. of Computer and Information Sciences
Dr. Xiaojiang Du, Stevens Institute of Technology (External)

ABSTRACT

Blockchain techniques have received an outburst of interest both in academia and industry, with numerous benefits such as decentralization, persistency, anonymity and auditability. There has been widespread adoption of blockchain in various fields ranging from cryptocurrency, financial services, risk management, internet of things (IoT) to public and social services. As a distributed ledger, blockchain records data in the form of linked blocks secured by cryptography. Instead of being controlled by a central third party like a bank, this ledger is maintained by a P2P network. To work properly, it is essential that the nodes in the network have the same local view of the ledger. A consensus mechanism is the core of a blockchain system since it regulates how to achieve the same states among the nodes in a distributed fashion. Specifically, it grants a qualified miner the right of appending the ledger in a period of time.

Despite its potential as a disruptive technology, blockchain is currently facing multiple challenges and problems, which incurs some issues on the performance, security, and fairness of blockchain-based systems. There are many underlying reasons for those challenges and problems, such as an inefficient networking protocol, a time-consuming consensus algorithm, a strategic mining node and so on.

In this dissertation, we summarize our theoretical studies on the above-mentioned topic from three different aspects. Details are as follows:

1. Design: A major weakness of the current blockchain systems is the low transaction throughput, which is resulted from the small block size, the long block generation time, and the delayed block propagation. To shorten the block propagation delay, we design a neighbor selection algorithm, improving the communication speed in the P2P network of the blockchain. This design is an attempt on topology optimization and reduces the frequency of blockchain forking. Fork-

ing is a situation where the blockchain temporarily diverges into two or more branches, which wastes mining power and causes security issues.

2. Extension: Although a consensus mechanism aims to regulate mining nodes' behavior, it also can be manipulate by strategic miners within legal limits due to their profit-driven nature. In the dissertation, we analyze strategies that miners apply for utility maximization under different consensus mechanisms from the perspective of game theory. And then we propose distributed algorithms to optimize resource allocation among miners.

3. Application: We also propose a blockchain-based market where we believe the application of blockchain will enhance the corresponding applications. It is a blockchain-powered data market that allows multi-user cooperative search. We also integrate smart contract into these applications.

All discussions in this dissertation are theoretical explorations. The theoretical results may not be realistically practical since we hold strong assumptions. We hope that, we could fill the gap between theory and practice in the future.

ACKNOWLEDGEMENTS

I would like to thank my advisor Jie Wu for bringing me into the field of networking, for his guidance and constant encouragement. He has taught me how to find research problems from practice, conduct solid research with real world impact, and present research results to different audiences. Dr. Wu offered me with a lot of opportunities to attend conferences and introduced me to many world-class scientists in the field. In addition to always being accessible and providing detailed help whenever I need them, I also learnt that good research projects should not only be intellectually exciting, but also be practical and useful. He always encourages me to continue exploring one interesting idea and brainstorms with me to develop this small idea to a big one. Being a student of his is one of the luckiest experiences. I would also like to thank Dr. Chiu C. Tan, Dr. Justin Y. Shi, and Dr. Xiaojiang Du for being my committee members. I sincerely appreciate their valuable comments and help to complete my dissertation.

I enjoyed my time at Temple University. I thank previous and current our research group members: Wei Chang, Dawei Li, Huanyang Zheng, Ning Wang, Jiacheng Shang, Rajorshi Biswas, Yang Chen, Yubin Duan, Nadia Niknami, and Abdalaziz Sawwan, for their inspiring discussions and many thoughtful comments on papers and practice talks. I also thank Shuaibing Lu, Biyu Zhou, Longhua Guo, Xiaoyu Zhu, and Juan Li for giving me great advice on how to be a good researcher over the years. I also thank all my other friends at United States for all the delicious food and fun social events we had together: Lu Pang, Sijia Yu, Xinyi Li, Yihan Fu, Mingyang Yin, Zhijia Chen and Fengjiao Li.

Above all, I would like to thank my parents, Xiqing Jiang and Xingjuan Wu, for their enduring love and support. I dedicate this dissertation to them.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

Recently, cryptocurrency [110, 69, 83] has attracted extensive attentions from both industry and academia. Blockchain is the underlying technology of a number of digital cryptocurrencies. Blockchain is a chain of blocks that store information with digital signatures in a decentralized and distributed network. The features of blockchain, including decentralization, immutability, transparency and auditability, make transactions more secure and tamper proof. Apart from cryptocurrency, blockchain technology can be used in financial and social services, risk management, healthcare facilities, and so on. A number of research studies focus on the opportunity that blockchain provides in various application domains. This chapter presents a brief introduction of blockchain, explaining the taxonomy and architecture of blockchain, describing the underlying networking, providing a comparison among different consensus mechanisms, and discussing challenges, including scalability, security, fairness and regulatory issues. In addition, this chapter also summarizes our research on the above mentioned topics.

1.1 Blockchain Architecture

Blockchain is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger [84]. Fig. 1.1 illustrates an example of a blockchain. With a previous block hash contained in the block header, a block has only one parent block. It is worth noting that uncle blocks (children of the block's ancestors) hashes would also be stored in ethereum blockchain [30]. The first block

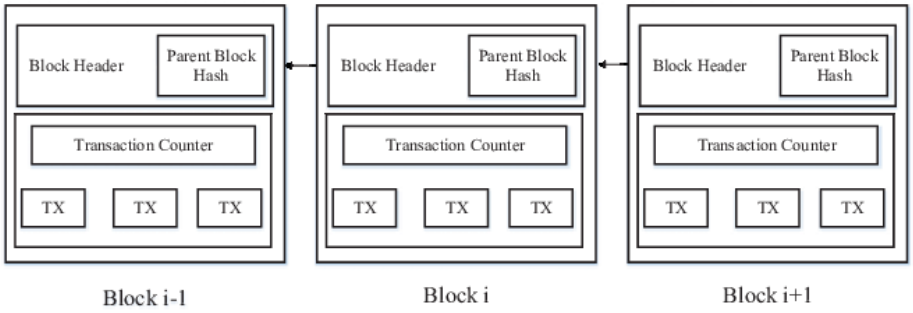


Figure 1.1: An example of blockchain which consists of a continuous sequence of blocks.

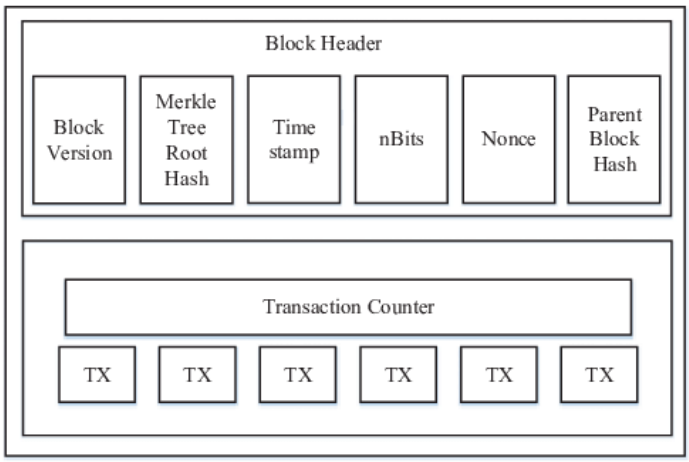


Figure 1.2: Block structure.

of a blockchain is called genesis block which has no parent block. We then explain the internals of blockchain in details.

1.1.1 Block

A block consists of the block header and the block body as shown in Fig. 1.2. In particular, the block header includes the following:

1. Block version: indicates which set of block validation rules to follow.

2. Merkle tree root hash: the hash value of all the transactions in the block.

3. Timestamp: current time as seconds in universal time since January 1, 1970.

4. nBits: target threshold of a valid block hash.

5. Nonce: an 4-byte field, which usually starts with 0 and increases for every hash calculation.

6. Parent block hash: a 256-bit hash value that points to the previous block.

The block body is composed of a transaction counter and transactions. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. Blockchain uses an asymmetric cryptography mechanism to validate the authentication of transactions [25]. Digital signature based on asymmetric cryptography is used in an untrustworthy environment. We next briefly illustrate digital signature.

1.1.2 Digital Signature

Each user owns a pair of private key and public key. The private key that shall be kept in confidentiality is used to sign the transactions. The digital signed transactions are broadcasted throughout the whole network. The typical digital signature is involved with two phases: signing phase and verification phase. For instance, an user Alice wants to send another user Bob a message. (1) In the signing phase, Alice encrypts her data with her private key and sends Bob the encrypted result and original data. (2) In the verification phase, Bob validates the value with Alice's public key. In that way, Bob could easily check if the data has been tampered or not. The typical digital signature algorithm used in blockchains is the elliptic curve digital signature algorithm (ECDSA) [62].

1.1.3 Key Characteristics of Blockchain

In summary, blockchain has following key characteristics.

- Decentralization. In conventional centralized transaction systems, each transaction needs to be validated through the central trusted agency (e.g., the central

bank), inevitably resulting to the cost and the performance bottlenecks at the central servers. Contrast to the centralized mode, third party is no longer needed in blockchain. Consensus algorithms in blockchain are used to maintain data consistency in distributed network.

- Persistency. Transactions can be validated quickly and invalid transactions would not be admitted by honest miners. It is nearly impossible to delete or rollback transactions once they are included in the blockchain. Blocks that contain invalid transactions could be discovered immediately.

- Auditability. Bitcoin blockchain stores data about user balances based on the Unspent Transaction Output (UTXO) model [2]: Any transaction has to refer to some previous unspent transactions. Once the current transaction is recorded into the blockchain, the state of those referred unspent transactions switch from unspent to spent. So transactions could be easily verified and tracked.

1.2 Blockchain Underlying Networking

The network of the Bitcoin blockchain is based on an unstructured P2P network. Nodes in the Bitcoin network are identified by their IP addresses. Each node has a list of IP addresses of potential peers. The list is bootstrapped through a DNS server, and additional addresses are exchanged between peers. From his list, a node randomly selects 8 reachable peers, with which it forms long-lived outgoing connections. A node can be recognized as reachable or non-reachable, depending on whether or not to accept an incoming connection. Outgoing connections and incoming connections are functionally-equal. The only difference is that, a node's outgoing connections are initiated by himself, while his incoming connections are unsolicited. Reachable nodes can additionally accept up to 117 unsolicited connections from other nodes. This chapter only considers a Bitcoin network composed of all reachable nodes. Thus, The

total number of connections a node can have is 125 by default. In the following, we first give a brief introduction to how a node decides his neighbor set.

1.2.1 Current Neighbor Selection in Bitcoin

When and how a node selects his outgoing neighbors, 8 in total, is described as follows. New outgoing connections are selected if a node boosts or if an outgoing connection is dropped by the network. A node with $\omega \in [0, 7]$ outgoing connections selects the $(\omega+1)$-th connection as follows: first, he decides whether to select from a tried table (nodes that he has connected to before) or a new table (nodes that are provided by the current neighbors but never contacted). The default algorithm makes tried addresses more likely to be selected when there are few outgoing connections or the tried table is large. Second, select a random address from the chosen table, with a bias towards addresses with fresher timestamps. After that, the node attempts to connect to the address. If connection fails, repeat the above two steps. As a node also receives incoming connecting requests from other nodes, he accepts all those unsolicited connections until reaching the upper bound, *i.e.*, 117 connections. A Bitcoin node never deliberately drops a connection, except when a blacklisting condition is met.

1.2.2 Transaction and Block Dissemination

Nodes use a simple gossip protocol to propagate messages containing transactions and blocks to update and synchronize their ledger replicas. To minimize the information spread in the network, Bitcoin uses an advertisement-based request management system. More specifically, if node A receives information about a new Bitcoin object (e.g., a transaction or a block) from another node, A will advertise this object to its other connections (e.g., node V) by sending them an inv message. These messages are much smaller in size than the actual objects, because they only contain the hash

and the type of object that is advertised. Only if node V has not previously received the object advertised by the inv message, will V request the object from A with a getdata request. Following the Bitcoin protocol, node A will subsequently respond with a Bitcoin object (e.g., the contents of a transaction or a block).

By doing so, inventory messages limit the amount of data broadcast in the network. Note that in case the object advertised corresponds to a block, neighbor V first requests the block header before the actual block data. Here, when a block header is advertised via a headers message, the receiving node internally stores the highest block known by the sending peer. The receiving node also validates any received header by verifying its corresponding PoW. Transactions, on the other hand, are propagated directly following the reception of the corresponding transaction inv message.

1.3 Consensus Algorithms

In blockchain, how to reach consensus among the untrustworthy nodes is a challenge as the blockchain network is distributed, *i.e.*, there is no central node that ensures ledgers on distributed nodes are all the same. Some protocols are needed to ensure ledgers in different nodes are consistent. We next present several common approaches to reach a consensus in blockchain.

- PoW (Proof of work) [81]: a decentralized consensus mechanism that requires members of a network to expend effort solving an arbitrary mathematical puzzle to prevent anybody from gaming the system.

- PoC (Proof of capacity) [8]: a consensus mechanism algorithm used in blockchains that allows for mining devices in the network to use their available hard drive space to decide mining rights and validate transactions.

- PoS (Proof of stake) [66]: a consensus mechanism algorithm used to validate cryptocurrency transactions. With this system, owners of the cryptocurrency

can stake their coins, which gives them the right to check new blocks of transactions and add them to the blockchain.

- PBFT (Practical byzantine fault tolerance) [33]: a replication algorithm to tolerate byzantine faults. Some blockchains utilize the PBFT as consensus algorithm since PBFT could handle up to 1/3 malicious byzantine replicas. A new block is determined in a round. In each round, a primary would be selected according to some rules. And it is responsible for ordering the transaction. The whole process could be divided into three phase: pre-prepared, prepared and commit. In each phase, a node would enter next phase if it has received votes from over 2/3 of all nodes. So PBFT requires that every node is known to the network.

1.4 Challenges and Problems

Despite the great potential of blockchain, it faces numerous challenges, which limit the wide usage of blockchain.

Scalability

With the amount of transactions increasing day by day, the blockchain becomes bulky. Each node has to store all transactions to validate them on the blockchain because they have to check if the source of the current transaction is unspent or not. Besides, due to the original restriction of block size and the time interval used to generate a new block, the Bitcoin blockchain can only process nearly 7 transactions per second, which cannot fulfill the requirement of processing millions of transactions in real-time fashion. Meanwhile, as the capacity of blocks is very small, many small transactions might be delayed since miners prefer those transactions with high transaction fee.

Privacy Leakage

Blockchain can preserve a certain amount of privacy through the public key and private key. Users transact with their private key and public key without any real identity exposure. However, it is shown in [40], [5] that blockchain cannot guarantee the transactional privacy since the values of all transactions and balances for each public key are publicly visible. Besides, the recent study [41] has shown that a user's Bitcoin transactions can be linked to reveal user's information. Moreover, Biryukov et al. [11] presented an method to link user pseudonyms to IP addresses even when users are behind Network Address Translation (NAT) or firewalls. In [11], each client can be uniquely identified by a set of nodes it connects to. However, this set can be learned and used to find the origin of a transaction.

Selfish Mining

Blockchain is susceptible to attacks of colluding selfish miners. In particular, Eyal and Sirer [10] showed that the network is vulnerable even if only a small portion of the hashing power is used to cheat. In selfish mining strategy, selfish miners keep their mined blocks without broadcasting and the private branch would be revealed to the public only if some requirements are satisfied. As the private branch is longer than the current public chain, it would be admitted by all miners. Before the private blockchain publishment, honest miners are wasting their resources on an useless branch while selfish miners are mining their private chain without competitors. So selfish miners tend to get more revenue. Based on selfish mining, many other attacks have been proposed to show that blockchain is not so secure. For example, stubborn mining.

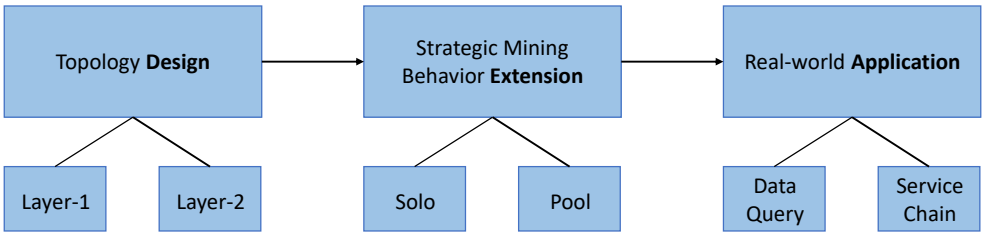


Figure 1.3: Research Overview.

1.5 Contributions

Fig. 1.3 shows an overview of my research during my PHD study. In this dissertation, we summarize our study on the above-mentioned topic from three different aspects. Details are as follows:

1. Design: A major weakness of the current blockchain systems is the low transaction throughput, which is resulted from the small block size, the long block generation time, and the delayed block propagation. To shorten the block propagation delay, we design a neighbor selection algorithm, improving the communication speed in the P2P network of the blockchain. This design is an attempt on topology optimization and reduces the frequency of blockchain forking. Forking is a situation where the blockchain temporarily diverges into two or more branches, which wastes mining power and causes security issues.

2. Extension: Although a consensus mechanism aims to regulate mining nodes' behavior, it also can be manipulate by strategic miners within legal limits due to their profit-driven nature. In the dissertation, we analyze strategies that miners apply for utility maximization under different consensus mechanisms from the perspective of game theory. Then we propose distributed algorithms to optimize resource allocation among miners. Analysis based on game theory incurs strong assumptions in this part, and leads to some theoretical conclusions only serving as guidance for reality.

3. Application: We also propose a blockchain-based market where we believe the application of blockchain will enhance the corresponding applications. It is a blockchain-powered data market that allows multi-user cooperative search. We also integrate smart contract into these applications.

CHAPTER 2

BLOCKCHAIN TOPOLOGY DESIGN

Bitcoin builds upon an unstructured peer-to-peer overlay network to disseminate transactions and blocks. Broadcast in such a network is slow and brings inconsistencies, *i.e.*, peers have different views of the system state. Due to the delayed block propagation and the competition of mining, forking, *i.e.*, the blockchain temporarily diverges into two or more branches, occurs frequently, which wastes computation power and causes security issues. This chapter proposes an autonomous and distributed topology optimization mechanism to reduce block propagation delay and hence reduce the occurrence of blockchain forks. In the proposed mechanism, a node can autonomously update his neighbor set using the information provided by his current neighbors, since each neighbor will recommend a peer from his own neighbor set, *i.e.*, a neighbor's neighbor, to this node. Each recommendation is based on a peer's propagation ability, which is characterized as a criteria function obtained through a combination of empirical analysis and machine learning. We further propose some metrics to evaluate a Bitcoin network topology. Experiment results reflect the effectiveness of the proposed mechanism and also indicate the correlation between block propagation time and fork rate. Thus, we analyze the relation between block propagation time and fork rate by applying an epidemic model to capture the block propagation process. We prove that a Bitcoin network topology produces a lower fork rate than another as long as its average block propagation time to 84% of the entire network is shorter.

Algorithm 1 Outgoing Neighbor Set Filling

Output: a neighbor set $N_i = \{j\}, j = 1, \cdots, 8$

if i's possible neighbor list is empty **then**
    Initiate a potential neighbor list from DNS servers
while i has $\omega \in [0, x-1]$ nearby neighbors **do**
    Pick j of highest S_j from nearby-neighbor list
    if i successfully connects to j **then**
        Add j to N_i
        $\omega = \omega + 1$
while i has $\omega \in [0, 7-x]$ middle neighbors **do**
    Pick j of highest S_j from middle-neighbor list
    if i successfully connects to j **then**
        Add j to N_i
        $\omega = \omega + 1$
Return N_i

2.1 Recommendation-based Neighbor Selection

Previous studies on P2P network optimization prove that using the proximity neighbor selection technique can improve the propagation performance in P2P networks. Meanwhile, existing research also shows that, some influential nodes with strong propagation power can accelerate information propagation in large-scale complex networks. Thus, a node in the Bitcoin network, *i.e.*, a large-scale complex P2P network, when selecting his neighbors, should take two factors into consideration. One is a peer's proximity and the other a peer's propagation ability in the network. Due to the specificity of the Bitcoin network, we should define suitable measurements to measure these two factors so that a node can determine each known peer's suitability to be a neighbor. Traditionally, the proximity of two nodes in networks can be captured by their geographical distance. In this chapter, we apply the round trip time, which can be easily obtained through a ping message, to describe the proximity between two nodes. Lots of methods also have been proposed to rank a node's propagation ability in the network, such as betweenness centrality, eigenvalue centrality, or k-shell. Most of them require a global view of the network topology, which is

Algorithm 2 Outgoing Neighbor Set Update

Output: $N_i = \{j\}$ where $j = 1, \cdots, 8$

```
Get 8 peers recommended by current neighbors
Classify 16 peers as nearby or middle peers
for all nearby peers do
    Rank peers based on S_j
    Pick top x connectable peers and update N_i
    Record the remaining peers in the nearby-neighbor list
for all middle peers do
    Rank peers based on S_j
    Pick top (8 − x) connectable peers and update N_i
    Record the remaining peers in the middle-neighbor list
Return N_i
```

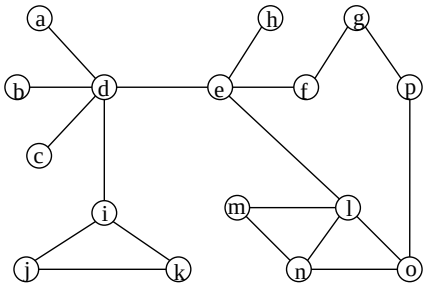
unrealistic for the Bitcoin network. In this chapter, we formulate a criteria function to quantify a peer's propagation ability using local features.

Currently, a node obtains network information from DNS servers and his connected neighbors. Those information are provided in the form of a long list of potential peers' IP addresses. This large-volume but not informative list makes it hard for a node to efficiently select suitable peers. In fact, if a node gets more useful information from his neighbor and utilizes those information to set up his neighbor set, he definitely can connect to nearby peers with better propagation abilities. As each node could improve his block propagation and receiving time, we believe it definitely leads to a better global topology for the Bitcoin network. Putting all considerations mentioned above together, we propose a recommendation-based neighbor selection mechanism. Our proposed algorithm is a combination of recommendations from the existing neighbors and self-measurement with local information. The key insight of our research is that efficient neighbor selection maps to feature selection and criteria function fitting in the field of machine learning. The following part of this section focuses on describing how a node propose performs neighbor selection using the proposed algorithm. And details on how to measure a peer's propagation ability are explained in Section 2.2.

2.1.1 Proposed Neighbor Selection Algorithm

We want to form a network where nodes are connected in an more efficient way for block propagation, while the network is still relatively random to prevent centralization. Thus, in our mechanism, a node only uses the proposed algorithm to determines his outgoing neighbor set, and always accept all incoming requests within the limitation of 117 connections. Besides, we want our algorithm not only to be applicable for Bitcoin. It should be suitable for a new Bitcoin-like network's construction as well as for any existing Bitcoin-like network's reorganization. Thus, our neighbor selection algorithm consists of two parts: one is a Neighbor Finding algorithm, as is shown in Algorithm 1, designed for any node of which the outgoing neighbors are fewer than 8 to fill/refill its neighbor set, and the other is a Neighbor Update algorithm, as is shown in Algorithm 2, used by a node with 8 outgoing neighbors to periodically refine its neighbor set.

Generally, a node i determines whether a peer j is suitable as a neighbor based on two factors: (1) j's propagation ability, calculated with the criteria function, *i.e.*, S_j, (details on the criteria function will be shown in the next section) and (2) the proximity between i and j, measured by the round-trip-time (RTT), denoted by t_{ij}. The proximity plays two conflicting roles here. The suitability of j can be shaded by its long distance from i, even if j is a propagation-well node. The link latency makes the direct connection between i and j replaceable by several intermediate relays starting from one of i's current neighbors. This case gives a guidance that a node should balance the propagation ability and the proximity when selecting a neighbor. However, a small t_{ij} is not always a preferred choice since it implies i and j may be located in the same 'social hub', and therefore, connecting to j helps little if i wants his block to go beyond this hub and spread the whole network effectively. Based on the analysis above and also inspired by the prior work indicating that networks with small-world topology can spread information faster than lattice networks [101], we



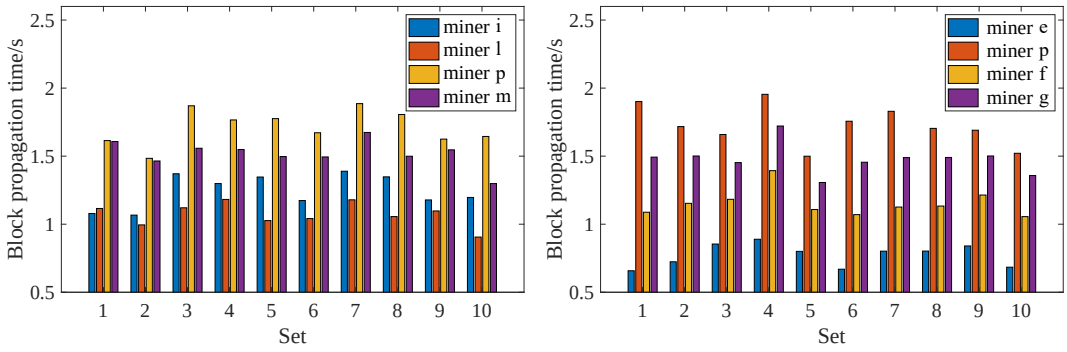
Miner	Power (%)
d, e	12
i	7
l	9
rest	5

Figure 2.1: A mining network of 16 nodes.

design our algorithm in a proximity-aware method. Node i will classify a peer j's proximity as near, middle, or far. A far j will not be attempted by i even if its S_j is big. Since i should keep 8 connections, it can choose x nearby and $(8 - x)$ in the middle region based on peer's propagation ability. Here, x is a predefined parameter. In terms of the peer-proximity classification standards and the value of x, we find they are influenced by the geographical distribution of all nodes in the network, and we determine them appropriately in our experiment.

2.1.2 Discussion on Incentive behind Recommendation

Any recommendation system has its underlying financial incentive whether it is in the form of direct or indirect reward. In our algorithm, nodes also has incentive to make recommendation to their neighbors. First, recommendation is bidirectional, *i.e.*, a node sows as well as reaps. Each node works toward shorter block propagation time as well as receiving time. This decreases the time wasted on extending stale blocks, and thereby saving its electricity cost. Also, its probability of winning block rewards will be improved, which leads to a higher profit in the long run. Second, individual improvement leads to better communication in the network, which will decrease the whole fork rate, making Bitcoin a more robust and reliable system. In fact, Bitcoin market price is affected by the security and performance of the system itself, where a healthy network is necessary.



(a) $D_i = 3$ and $D_l = 4$ (b) $D_e = 4$ and $D_p = 2$

Figure 2.2: Degree impact on a miner and its neighbor(s).

2.2 Feature Selection and Function Fitting

We are aiming to select a small set of features which are easy to calculate for a node using local information, but can accurately reflect a peer's propagation ability. All those features contribute to a criteria function, which helps a node determine the suitability of another node if selected as his neighbor. The propagation ability should be measured in two perspectives: (1) how well a neighbor can spread the node's block to the rest of the network, and (2) how fast a neighbor can notify the node of the newest blocks from the rest of the network. We propose several candidate features and apply empirical analysis to study their impacts. We start with two simple topologies of a mining network with 16 nodes, one is shown in Fig. 2.1 and the other is a completely-connected graph where each node has 15 edges. We control network parameters, *e.g.* upload/download bandwidth, link latency, in different experiments for comparison. In the simulation, we treat the process of block generation and propagation for 100 rounds as a set and we repeat 10 sets in each experiment. Corresponding results and analysis are detailed in the following.

2.2.1 Impact of a Neighbor's Degree

In this part, we analyze the impact of degree on the block propagation time and receiving time. To rule out the impact caused by nodes' different network environments,

we make every connection between any two nodes with the same upload/download bandwidth and link latency in this experiment. The first comparison node pair is (j, m) and their corresponding neighbor pair is (i, l). Fig. 2.2(a) reflects two facts: (1) a higher-degree node itself tends to have a shorter block propagation time (by comparing node l of $D_l = 4$ and node i of $D_i = 3$), and (2) a higher-degree node can shorten its neighbor's block propagation time by comparing node m of $D_m = 2$ and node j of $D_j = 2$. Similar results can be obtained from Fig. 2.2(b) by choosing comparison node pair (f, g) and their corresponding neighbor pair (d, p). Thus, we conclude, a higher-degree neighbor has a better block propagation ability.

2.2.2 Impact of a Neighbor's Local Clustering Coefficient

Local clustering coefficient, denoted as C_i and expressed in Eq. 2.2-1, measures how well a node's neighbors are connected to each other, namely how close they are to being a clique.

$$C_i = \frac{|\{e_{j,k} | \forall j, k \in N_i\}|}{\frac{1}{2} D_i (D_i - 1)}, \tag{2.2-1}$$

where $\frac{1}{2} D_i (D_i - 1)$ represents the maximum possible number of edges among all node i's neighbors and $\{e_{j,k} | \forall j, k \in N_i\}$ is the set of edges connecting two of i's neighbors. The local clustering has remarkable impacts on network structure and functionality. Studying the effects of clustering coefficient on the network evolving can provide insights into the understanding of the growing mechanism and further help us design a better criteria function and explain the observation on information spreading through the Bitcoin network. Some literature showed that the clustering has negative correlation with degree in undirected networks and our experiments reach the same conclusion.

As is shown in Fig. 2.3(a), node d has a higher local clustering coefficient compared with node e and its own block propagation time is longer than that of node e. Meanwhile, node d's neighbor, node i, also has a longer block propagation time

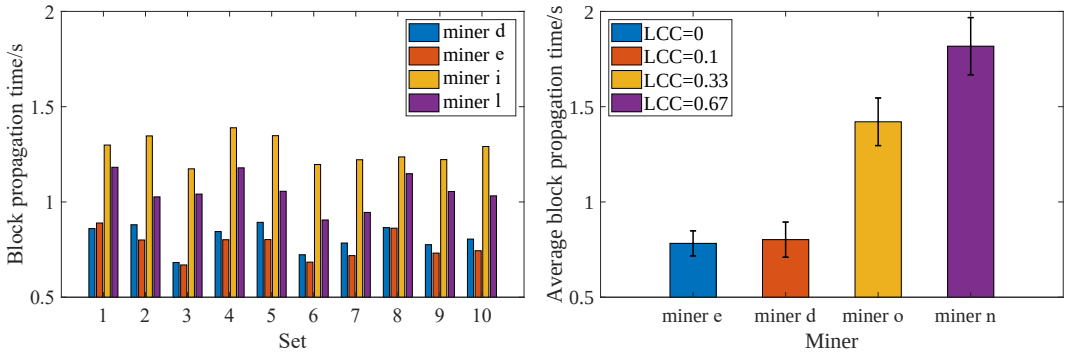
compared to node e's neighbor, node l. Besides, by comparing node d of $D_d = 5$ and node e of $D_e = 4$, we can also be guided that, local clustering coefficient seems of more significance than degree. Fig. 2.3(b) also provides an intuitive sense that local clustering coefficient is negatively related to a node's propagation ability. Thus, we consider that a neighbor with a lower local clustering coefficient should be more suitable.

2.2.3 Impact of a Neighbor's Mining Power

In [78], the authors hold the view that there exists a small set influential nodes that skew broadcast fairness. According to their analysis, nodes with more mining power receive a block more efficiently than others. Inspired by their results, we also consider that a peer's mining power may also be a feature to reflect his propagation ability. We pick nodes f and g for comparison. According to the network topology, f and g are directly connected and have the identical mining power. However, f's neighbor e has a higher mining power than that of g's neighbor p. Fig. 2.5 presents a comparison of the propagation time for nodes f and g. Obviously, f's block generally propagates faster than g's. This means a neighbor's propagation ability has a positive correlation with its mining power. In particular, once a block is relayed by a node, it means a portion of mining power supports this block. The more mining power extends on this block, the higher possibility this block has to be accepted by the network, if there exist competing blocks.

2.2.4 Criteria Function Fitting

Based on the empirical observation, we want to figure out a criteria function, taking as input a node's feature set value and generating as output a score to reflect this node's propagation ability. Such a criteria function allows a node to quantify a peer's propagation ability to determine the suitability of being a neighbor. Mathematically,



(a) $C_d = 0.1$ and $C_e = 0$ (b) $D_e = 4$ and $D_p = 2$

Figure 2.3: Local clustering coefficient impact on a miner and its neighbor(s).

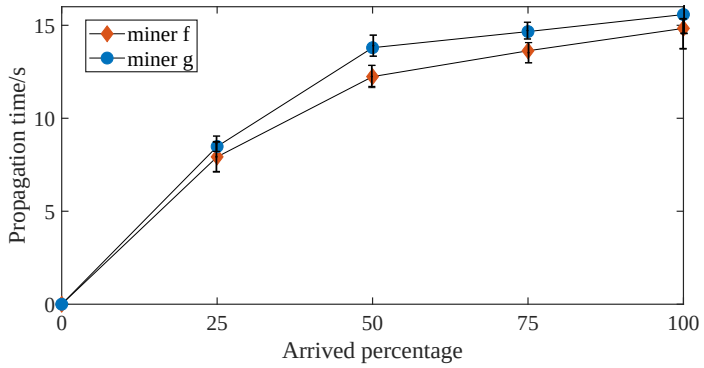


Figure 2.4: Neighbors' mining powers impact a node's block propagation time.

a node's propagation ability, denoted by S_i, is scored by the criteria function defined in the Eq. (2.2-2).

$$S_i = g(C_i) \sum_{j \in N_i} (D_j + 1) + m_i \tag{2.2-2}$$

Obviously, $g(C_i)$ accounts for the effect of i's local clustering and plays a negative role in propagation. Inspired by the result from [34], we make two attempts here by adopting $g(C_i)$ as either an exponential function, *i.e.*, $g(C_i) = k \cdot \alpha^{-C_i}$, or a power function, *i.e.*, $g(C_i) = k \cdot C_i^{\alpha}$. To find a best fitting, we use machine learning to figure out the value of k and α for both attempts. Our result shows that a simple exponential function $g(C_i) = 10^{-C_i}$ is enough for S_i since complicate functions or parameter values add little meaning to score nodes but make the analysis more complicated. Indeed, the perspective and results of this chapter are not limited by a very specific $g(C_i)$, as

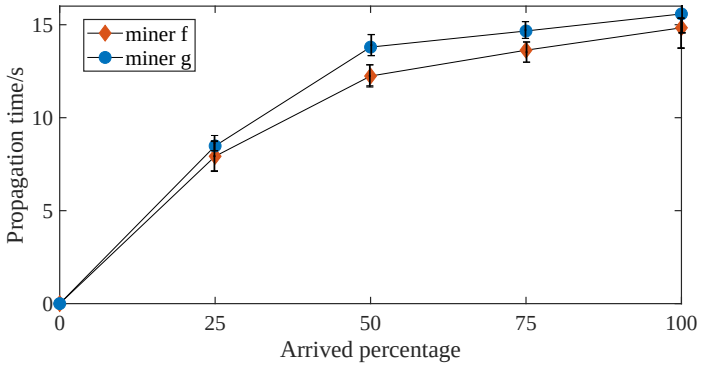


Figure 2.5: Neighbors' mining powers impact a node's block propagation time.

long as it is a decreasing function.

2.3 Metrics for Bitcoin-like Network Topology

Since our proposed algorithm, as well as other existing works, will definitely generate different Bitcoin network topologies, another challenge is how to give a comprehensive and objective evaluation of a topology instantiation. In the following, we detail novel metrics by which a Bitcoin/Bitcoin-like topology can be evaluated. These metrics are designed relying on the premise of the PoW consensus mechanism.

Block Propagation Time in the Bitcoin Network

We take as an important metric the time required to deliver a block from an originator to X percentage of nodes in the network, which is evaluated by most Bitcoin-like networks.

Consensus Delay

Since the network layer is to serve the consensus layer, we utilize consensus delay, proposed in [46], as another metric to quantify a Bitcoin network topology. As is defined in its original paper, consensus delay is that, for a specific execution and time, how long nodes have to look to find a point where they agree on the state.

Blockchain Fork Rate

The blockchain fork rate is another important metric, which is defined as the ratio of the number of blocks that are not included in the longest chain against the chain length. This metric indicates the effectively-utilized mining power in the current network and also indicates how much electricity waste of useless mining.

Fairness

As we claimed previously, it is unfair that some miners are still mining on the stale block while some have already started a new mining round, as it is an information-asymmetric situation. In this chapter, we want to quantify fairness. We calculate the average block receiving time for each miner and the fairness is obtained using the maximal average block receiving time difference in the network. Optimally, the fairness is 0, *i.e.*, in the long run, any miner should wait for an identical time to receive a block if he is not the block creator.

2.4 Blockchain Simulator

We evaluate Bitcoin/Bitcoin-like network topologies with a blockchain simulator running on the PoW consensus. Our simulator is implemented based on existing systems [32, 50]. Our modification enables different topologies for the simulator. The inputs are node information, topology information, and block information. There are two node types, *i.e.*, relay nodes and miners, and you need to specify the relay number and the miner number, as well as miners' mining power. In terms of topology, it is optional, *i.e.*, you can specify your own topology, or leave it empty. Then the simulator will automatically create a topology based on real-world information collected from bitcoin network and the neighbor selection algorithm you implemented in the simulator. For block information, you can specify how many blocks to generate

in total and how large your block size is. More details are given in the below.

Types of Nodes

In our simulator, we distinguish between two node types, *i.e.*, relay nodes and miners, by attributing a particular non-zero mining power to each miner. A relay node has functionalities of verifying blocks, and then relaying valid blocks to his neighbors. A miner, besides block verification and relay, can generate new blocks according to the PoW consensus. Bitcoin rewards miners who successfully append the blockchain, which is not implemented in our simulator, as we think this factor has negligible impact on the network performance evaluation.

Geographical Distribution of Nodes

For full nodes, we model their geographical locations using information provided by Bitnodes [5], and accordingly distributed full nodes to the respective region. For miners, we retrieve the mining pool distribution from blockchain.info [7], and accordingly distributed the mining pool's public nodes to respective regions.

PoW Consensus

The PoW consensus is applied by miners. Under this mechanism, the time it takes a miner to find a solution follows a geometric probability distribution, which can be approximated as an exponential distribution due to the improbability of a success in each guess and the rate of guessing. In our experiments, we replace the proof of work mechanism with a scheduler that triggers block generation at different miners with exponentially distributed intervals. Thus, every 10 minutes, a new block is then attributed to a miner. Besides, forks are inherently resolved by the longest chain rule.

Mining Power Distribution

In our simulator, we design two methods: (1) all miners are attributed to identical mining power, and (2) all miners are attributed to mining power following the distribution from blockchain.info [7].

Network Environment

We use the round-trip-time (RTT) between two nodes to quantify their proximity. The corresponding RTT between any two nodes is calculated and assigned in advance, based on their geographical distance using the function shown in [89]. To simulate a node's block verification time, we add a data processing latency T_p processing at each hop whenever a node needs to forward a block. For simplicity, we choose an empirical value of 45ms instead of modeling T_p as a function varying on the block size. We also tackle the bandwidth in a simple way by assuming that each node equally allocates its bandwidth to each connection and each connection bandwidth only varies on regions.

2.5 Evaluation

In this section, we evaluate the performance of various Bitcoin topology instantiations by leveraging our metrics and a blockchain simulator [50]. We set the value n to be 2 since it is the best configuration after extensive experiments. We consider a peer as nearby if the RTT is no larger than 100 ms, and consider a peer as middle if the RTT is between 100 ms and 450 ms, otherwise, he is a far peer. For comparison purposes, we implement our proposed algorithm and another 4 algorithms, which are described as followings:

- Default: Randomly pick nodes from the known peer list to satisfy the 8-neighbor requirement.
- RTT based scoring (RTTS): This algorithm [26] allows each node to score a

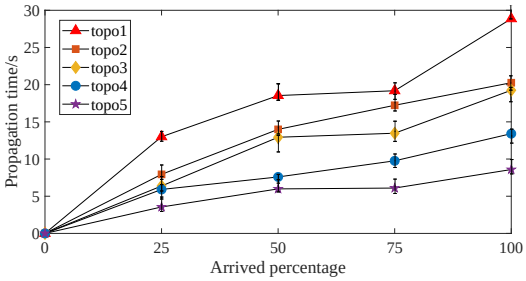


Figure 2.6: Miners with different powers.

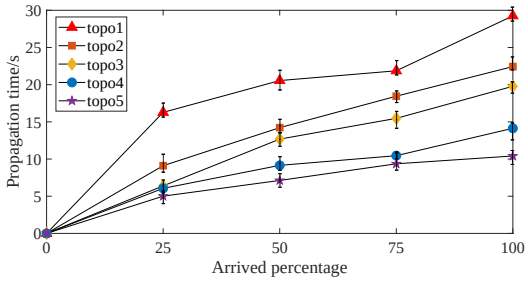


Figure 2.7: Miners with identical power.

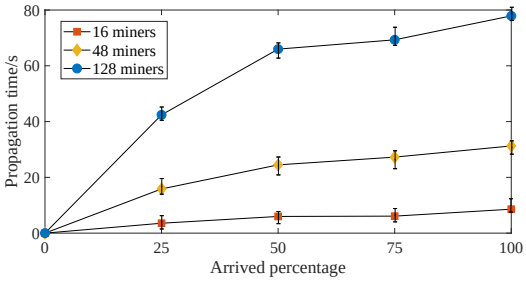


Figure 2.8: Different numbers of miners.

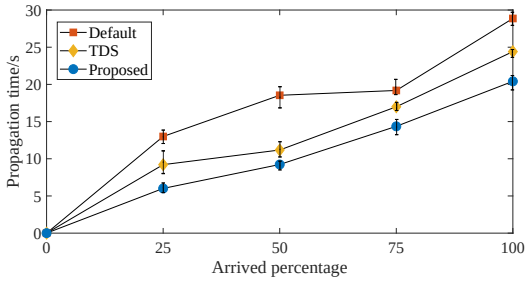


Figure 2.9: Topology reorganization.

peer based on the round-trip-time between them and then decide its outgoing connection priority.

- Geographical distance based scoring (GDS): This algorithm [48] allows each node to score a peer based on the physical distance between them and then decide its outgoing connection priority.

- Time difference based scoring (TDS): This algorithm [21] allows each node to score a peer based on the time difference between block generation and receipt of this peer's INV message and then decide its outgoing connection priority.

2.5.1 Static Environments

We first study the effectiveness of all algorithms in a static P2P environment, where nodes do not join and leave.

Performance of New Network Constructions

Given a network from scratch, different algorithms can produce different topologies. In this part, we first define the number for each node type, and their geographical locations will be determined by the blockchain simulator. After that, we use each algorithm to construct a topology and evaluate the algorithm's effectiveness through the topology performance. In this experiment, we set the number of miners and the number of relay nodes to be 16 and 256, respectively.

Fig. 2.6 shows block propagation time used to reach a certain percentage of nodes under different network topologies. Obviously, the topology generated by our proposed algorithm performs best on this metric. In Table 2.1, we show the results of other metrics and we can see our proposed algorithm performs well except fairness, which means a relatively big difference between the worst receiving time and the best receiving time. The reason behind this big gap may come from the reason that some node with a bad propagation ability is ignored by the entire network. Next, we equally distribute mining power among all miners, which was envisioned in the Bitcoin original whitepaper. According to Fig. 2.7, this slight change causes the propagation delay increases no matter what algorithm is applied to generate the network topology. This result further confirms the correctness that we consider a node's mining power as a feature of his propagation ability. We keep the relay node number unchanged while increasing the miner number. Obviously, the propagation delay becomes longer since the total number of nodes increases. However, Fig. 2.8 shows the delay increase is non-linear with the miner number increase, which indicates the network is still under-saturation.

Performance of Existing Network Reorganizations

As we stressed before, our neighbor selection algorithm is backward compatible, *i.e.*, it also improves an existing network topology after all joined nodes adopt our

Algorithm	Median	Broadcast	Consensus	Fork rate	Fairness
Default	13.04	19.93	755	1.61%	3.94
RTT	10.33	17.35	670	1.52%	3.82
GDS	8.79	14.19	607	1.12%	3.12
TDS	6.40	1.00	579	1.02%	3.24
Proposed	4.67	7.8	511	0.78%	3.14

Table 2.1: Averaged median delay, broadcast delay, consensus delay, and worst-best receiving time among all 16 miners, and fork rate in the bitcoin network using various algorithms.

Topology	25%	50%	75%	85%	100%	fork rate
1	19.87	31.33	31.63	32.93	34.34	1.52%
2	19.81	30.71	31.27	33.38	35.77	1.82%
3	17.70	30.48	31.33	35.51	37.07	2%
4	17.06	29.99	32.10	36.44	38.50	2.22%
5	19.33	29.63	30.60	37.75	39.09	2.38%
6	18.35	27.66	34.04	38.46	39.98	2.56%

Table 2.2: All-miner networks optimized by our proposed algorithm.

algorithm. To see how effective our algorithm is in improving an existing network, we first use the default algorithm to build the original topology and then optimize it with the proposed algorithm. As only TDS can update the network in a static environment, we show the propagation time of the original topology as well as the topologies generated by TDS and our proposed algorithm in Fig. 2.9. As mentioned in Table 2.1, the original topology leads to a fork rate of 1.61%. After reorganization, TDS and our proposed algorithm can lower the fork rate to 1.54% and 1.18%, respectively. However, reorganization cannot reach the performance achieved by new-construction, which indicates the importance of designing a good topology formation mechanism for a Bitcoin/Bitcoin-like network.

We further consider an extreme cases, where all nodes are miners. We set the node number, *i.e.*, the miner number, as 48, and apply the default algorithm to generate 6 different random topologies. For each random topology, we run our proposed algorithm to optimize it. When we measure those optimized topologies, we obtain an interesting observation. Our previous experiment results confirm that the block

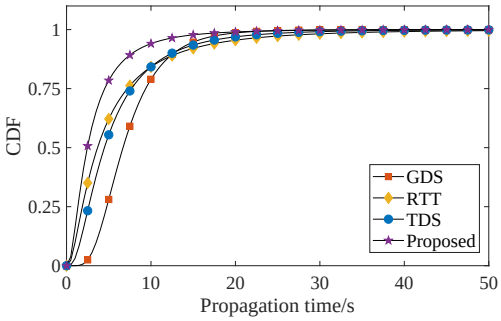


Figure 2.10: Environment with churns.

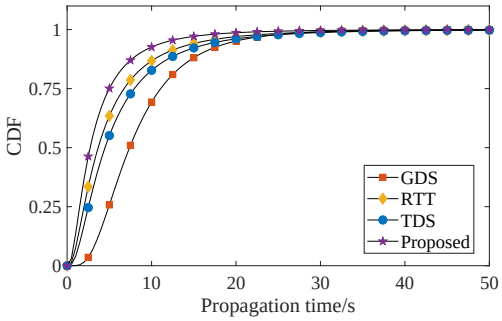


Figure 2.11: Churns and link failures.

broadcast is positively-correlated to the fork rate. However, it seems that, the positive correlation already happens at a certain point where even if not all miners are informed. According to Table 2.2, we find if a topology's average block propagation time to 85% miners is shorter, then its fork rate is also lower, compared with a topology with a longer average 85% block propagation time.

2.5.2 Dynamic Environments

We further evaluate the network performance in a dynamic environment, where nodes and connections are changing. In the first setting, we add the joining and leaving of nodes, where the churn rate is modeled according to the distribution in [59]. In fact, the first three algorithms take effect only when the number of nodes changes. In the second setting, we simulate the connection failure between nodes to fully capture network dynamics and make our evaluation more sound. We plot the cumulative distribution function for each network topology and the corresponding results are shown in Fig. 2.10 and Fig. 2.11. It is clear that our proposed algorithm achieves high effectiveness compared with others.

2.6 Discussion and Future Work

There still exist some defects in our proposed algorithm. First, our algorithm targets on information propagation acceleration, which helps alleviate forking due to

network delay, while fails to handle forks caused by network partition. Second, in our evaluation, all algorithms used for comparison are statics, meaning that neighbors are unchanged once selected, unless they meet blacklisting conditions. However, in reality, a node's neighbor set are dynamic with lots of update. In this case, our proposed algorithm can just probabilistically outperform. These two problems are motivated us as our future direction.

2.7 Related Work

Bitcoin network aims on information propagation while suffering from a significant delay. As Bitcoin contains two distinct types of information, *i.e.*, transactions and blocks, some works [86] focus on accelerating transaction propagation, while we are mainly concerned with block propagation. There are also many works in development to speed up block propagation. The resulting solutions can be roughly divided into three categories: (1) block compression to limit the amount of data that needs to be propagated [36, 98, 87, 53, 28] , (2) third-party relay networks for fast inter-miner communication [35, 12, 13], and (3) network protocol design for topology optimization. There are a few works designing different network protocols for Bitcoin nodes [40]. [64] proposes a tree-based topology where a node should join to a tree as a leaf while ensuring the tree is as balanced as possible. [48] suggests a cluster-based topology and then their proposed algorithm requires a node chooses its closest neighbors according to their physical distances. [26] designs a similar algorithm while the difference lies in the definition of closeness, which is captured by the round-trip-time between two nodes. However, these works are designed for a node to choose suitable neighbors when it first enters this network or for constructing a topology from the scratch. They are more applicable for constructing a new Bitcoin-like topology instead of optimizing the existing Bitcoin network.

Our proposed mechanism can be used for a new Bitcoin-like network construction

as well as an existing Bitcoin-like reorganization. [21] also proposes a topology reorganization algorithm, which allows a node to periodically update its outbound neighbor set and each neighbor is ranked by the difference between a block generation time and the receipt time of the block sent by this neighbor. Our algorithm also allows a node to update suitable neighbors based on scores. However, the measurement we use to rank a candidate neighbor is different from that in [21]. Meanwhile, we also introduce recommendation from current neighbors, providing a node with a better view of the global topology.

2.8 Conclusion

In this chapter, we propose an autonomous topology optimization mechanism for the Bitcoin network. The main part of the mechanism is a recommendation-based neighbor selection algorithm, which allows miners to update their neighbor sets in a distributed fashion using information provided by the current neighbors. A criteria function is designed for miners to make recommendation and selection. Two metrics, *i.e.*, block propagation delay and blockchain fork rate, are used to quantify the performance of a Bitcoin network topology. Simulations show a good rate of decrease in block propagation delays (both average and maximum) and fork rates, compared to classic algorithms, and also prove the validation of our proposed propagation model.

CHAPTER 3

EXTENSION: BLOCK SIZE MANIPULATION

In a Bitcoin market, miners participate in blockchain mining with an aim to make profits. Until reaching consensus, PoW-valid blocks (including validated transactions and proper PoW solutions) can be viewed as being successfully mined and are rewarded. There are two types of rewards for miners: fixed *block subsidies* and time-varying *transaction fees*. Block subsidies, predetermined by design, are the current major revenue source. Transaction fees, offered by Bitcoin transaction senders to accelerate their transactions, heavily depend on the corresponding transaction size. Thus, a larger-size block tends to contain higher transaction fees and hence more rewards. However, the probability of a miner to successfully mine a block diminishes as the block size increases, since a larger-size block takes a longer time to reach consensus. Thus, the reward included in the block is vitally affected by its size, which is independently decided by a miner. In this chapter, we use a game-theoretic approach to study how a miner's payoff, *i.e.*, expected profits, is determined by his block size. More specifically, we derive an expression to characterize the relation between the miner's payoff and block sizes. Besides, we use game theory to analyze how profit-driven miners will manipulate their block sizes to optimize payoff instead of adopting the default block size. We conduct numerical experiments on real-world data collected from Bitcoin to find peaceful equilibrium where miners have no incentive to misbehave. The achieved block sizes thereby give guidelines on the default block size, in order to deter miners from misbehaving. Our analysis suggests a block size of 4 MB.

3.1 System Model

In this section, we introduce a realistic Bitcoin mining model. As commonly done in blockchain-related analysis, we assume the whole system is in a quasi-static state [50, 49]. That means no miners join or leave, existing miners maintain their behavior, and the system reaches equilibrium. Therefore, in our model, the system comprises a fixed set of miners associated with their mining power.

Suppose there are n miners starting to mine a new block on the top the same block, *i.e.*, there is no fork at the beginning time. All players(miners) are asked to solve the proof-of-work puzzle in order to mine a block. The puzzle can be solved only by using a trial and error strategy, and the occurrence of solving this problem can be well approximated by a random variable following a Poission process. The time for the whole system to find a valid block is exponentially distributed with a fixed rate parameter. Time used to find the first block by any of the miners is the minimum of all finding times by all different miners. The value of the rate parameter is determined by the consensus protocol, such that the expected block time interval is of a constant value. The rate parameter represents the difficulty of the proof-of-work puzzle, and we use the terms difficulty and rate interchangeably. The total mining power affects the value of the rate parameter. The difficulty parameter value is adjusted by the whole system to decrease (or increase) the rate of each miner. In an equilibrium, the rate parameter is fixed as a constant. Currently, the difficulty of finding a block is dynamically adjusted so that it takes $T = 600$ seconds on average. Thus, the mining Poission process has a fixed parameter $1/T$ for the whole network.

We assume that the difficulty of all puzzles is the same. In fact, the difficulty of puzzles at each time is proportional to the total mining power in the Bitcoin network of that time. It is reasonable because in our quasi-static model, the miners and their mining power are fixed, thus there is a stable total mining power. There exists more than one valid solution for each puzzle. It is possible that two or even more miners

solve their puzzles for a same-height block, which will result in a blockchain fork. Then, the rest miners have to choose only one to continue mining on. The branch accepted by the majority survives, and the corresponding miner would be rewarded. Thus, we could conclude that, miners who solve a puzzle are not necessarily rewarded, and only the first to make his solved block reach consensus will obtain rewards.

The reward of a mined block comes from two aspects: the fixed block subsidy and the extra fee from transactions included in this block. The block subsidy can be referred to as base reward, which is relatively fixed over time. This reward is comprised of the minting of new currency with the creation of each block. Transaction fees come from the aggregation of newly introduced transactions in the system. This reward is time-dependent. As the time progresses, there are more pending transactions in the system, and the potential fees grow. We follow the assumption made in [32] that transactions (and their associated fees) arrive at a constant and continuous rate. To be more precise, during any time interval t, the sum of fees in the announced transactions is ct, where c is a specific constant. As is emphasized in [32], this assumption helps to simplify analysis on the effects of transaction fees, although there is no guarantee it holds in practice. Thus, the transaction fee density of unverified transactions is also constant.

According to the Bitcoin mining protocol, each miner can decide what and how many transactions to include in their block. Following the assumption in [32], if there are m transaction fees available, a miner can choose to include any real-valued number of transaction fees between 0 and m in his block. That is, a miner can selectively choose a set of transactions whose fees are very close to whatever real-valued target he wants. It is a reasonable approximation due to the large number of transactions per block. Thus, the amount of transaction fees included in a block is proportional to its size. Besides, the set of transactions chosen by each miner has no effects on the time and chance to solve hid puzzle. However, it matters during the block broadcast

time. Once a miner finds a block, he needs to broadcast it to the rest of the Bitcoin network. In order to be added permanently to the blockchain, this block must be accepted by the majority. If we take the block transfer delay into consideration, the time used to make a block reach consensus is heavily dependent on its size and hence, the set of transactions in it.

Once a block is mined, all miners move on to find the next block. This process is repeated indefinitely. The profit of a miner for each block is the difference between his total expenses and his total reward. Rational miners strive to maximize their profits, giving rise to a game.

3.2 Distribution in the Block Size Game

The repeated search for the blocks becomes a series of independent one-shot competitions, and in each competition, only one miner gets the reward but all miners pay expenses. To analyze the expected revenues, rather than considering the individual iterations, we consider a one-shot game played by the miners. A player's strategy is the choice of his block size. The choice of block sizes are made a-priori by all the players.

To find the payoff of each player, we start by analyzing the block finding time probability distribution. This is a function of the players' selection of block sizes. We model the block finding time as a random variable denoted X with cumulative distribution function (CDF) and probability density function (PDF) denoted $F_X(t; B, \lambda)$ and $f_X(t; B, \lambda)$, respectively. The corresponding notations are listed in Table 3.1.

3.2.1 Distribution Analysis

The first step towards analyzing the system is to derive an expression for the distribution, namely $F_X(t; B, \lambda)$ and $f_X(t; B, \lambda)$. We begin with the distribution of a single player i with mining rate λ_i. Assume i's block size is B_i, thus, his propagation

Table 3.1: Summary of Parameters.

Symbol	Description
T	Average block arriving interval
R	Block subsidy
α	Transaction fee density
β	Block propagation time per unit
n	Number of miners or players
λ_i	Player i's mining rate
h_i	Player i's mining power where $h_i = {}^{\lambda_i}/_{T}$
B_i	Block size decided by player i
P_i	Payoff for player i to mine a block

time is $p_i = \beta B_i$. Denote the time this player requires for successfully mining a block as a random variable X_i. The probability density function (PDF) of X_i is

$$f_{X_i}(t; B_i, \lambda_i) = \begin{cases} 0 & t < p_i \\ \lambda_i e^{-\lambda_i (t - p_i)} & t \geq p_i \end{cases}, \tag{3.2-1}$$

which describes the probability of player i, whose mining rate is λ_i, to successfully mines a block of size B_i at time t. And the corresponding cumulative density function (CDF) is

$$F_{X_i}(t; B_i, \lambda_i) = \begin{cases} 0 & t < p_i \\ 1 - \ e^{-\lambda_i (t - p_i)} & t \geq p_i \end{cases}, \tag{3.2-2}$$

defining i's accumulated winning probability until time t.

As $F_{X_i}(t; B_i, \lambda_i) = Pr(X_i \leq t) = 1 - Pr(X_i \geq t)$, we can obtain that

$$Pr(X_i \geq t) = \begin{cases} 1 & t < p_i \\ e^{-\lambda_i (t - p_i)} & t \geq p_i \end{cases}. \tag{3.2-3}$$

Since all the players are competing on mining the next block, any player with the minimal value of X_i is the first one to find the next block. Therefore, the time required for finding the next block is $X = \min_{i \in \{1, 2, \cdots, n\}} X_i$.

We use a boolean variable $active_i(t)$ to capture a player i's winnablility at time t,

which is expressed in the below:

$$active_i(t) = \begin{cases} 0 & t < p_i \\ & \\ 1 & t \geq p_i \end{cases}. \tag{3.2-4}$$

It is obvious that, i has zero winnablility before time p_i, even if he could solve his PoW puzzle at $t = 0$. After p_i, i starts to hold a probability to win. Besides, we define $active(t)$ as the set of any player who is likely to win at time t. That is, $active(t) = \{i \mid active_i(t) = 1,\ \forall i\}$.

The probability that none of the players have found a block by time t, $Pr(X > t)$, is the product of $Pr(X_i > t)$ for all i (as players are independent of each other), shown in Eq.(3.2-5).

$$\begin{aligned} Pr(X > t) &= \bigcap_{i \in \{1,2,\cdots,n\}} Pr(X_i > t) = \prod_{i=1}^{n} Pr(X_i > t) \\ &= e^{\sum_{i \in active(t)}[-\lambda_i(t-p_i)]} \end{aligned} \tag{3.2-5}$$

Thus, X's corresponding CDF and PDF are shown below,

$$\begin{aligned} F_X(t; B, \lambda) &= 1 - Pr(X > t) \\ &= 1 - e^{\sum_{i \in active(t)}[-\lambda_i(t-p_i)]}, \\ f_X(t; B, \lambda) &= (\sum_{i \in active(t)} \lambda_i) \cdot e^{\sum_{i \in active(t)}[-\lambda_i(t-p_i)]}. \end{aligned} \tag{3.2-6}$$

3.2.2 Proof of A Valid PDF

Theorem 1. $f_X(t; B, \lambda)$ is a valid probability density function to express the probability of finding a block as time passes in the whole blockchain mining network.

Proof. We present the full verification process in the below by checking that $\int_{-\infty}^{+\infty} f_X(t; B, \lambda)\mathrm{d}t = 1$ holds.

$$\begin{aligned} \int_{-\infty}^{+\infty} f_X(t; B, \lambda)\mathrm{d}t &= \sum_{l=1}^{l=n} \int_{p_l}^{p_{l+1}} f_X(t; B, \lambda)\mathrm{d}t \\ &= \sum_{l=1}^{l=n} \int_{p_l}^{p_{l+1}} \lambda |active(p_l)| e^{\sum_{j \in active(p_l)}[\lambda_j(t-p_l)]}\mathrm{d}t \end{aligned} \tag{3.2-7}$$

$$
\begin{aligned}
&= \sum_{l=1}^{l=n} \int_{p_l}^{p_{l+1}} \lambda x_l e^{-x_l \lambda t} e^{\sum_{j \in active(p_l)} p_j} \mathrm{d}t \\
&= e^{-\lambda} \sum [e^{\sum_{j \in active(p_l)} (p_l - p_j)} - e^{\sum_{l \in active(p_l)} (p_{l+1} - p_j)}] \\
&= e^{-\lambda} [e^{\sum_{j \in active(p_1)} (p_1 - p_j)} - e^{\sum_{j \in active(p_\infty)} (p_\infty - p_j)}] \\
&= e^{-\lambda}(e^0 - e^{+\infty}) = 1
\end{aligned}
$$

□

Thus, the PDF we use is valid, hence our model is as well.

3.3 Payoff in the Block Size Game

3.3.1 Payoff Analysis

The payoff is defined as the expected profit of player i. We use the variable $profit_i$ to represent i's profit and hence at time t, i's expected profit is denoted as $E(profit_i|X = t)$. We model the profit of a block consisting of the fixed block subsidy and transaction fees inside that block, which is proportional to the block size. Thus, for a specific player i, the total available profit is $R + \alpha B_i$. Recall that, in expectation, the probability that a specific active player will find a block is his mining power divided by the total mining power owned by all the active players. Thus, if a block was found at time t, then the expected profit of player i is shown below.

$$
E(profit_i|X = t) = \frac{active_i(t) \cdot \lambda_i}{\sum_{j \in active(t)} \lambda_j} (R + \alpha B_i) \tag{3.3-8}
$$

Since we define the player i's payoff as the expectation of his profit, we express it in Eq.(3.3-9),

$$
\begin{aligned}
P_i &= E(profit_i) = E(E(profit_i|X = t)) \\
&= \int_{-\infty}^{+\infty} E(profit_i|X = t) \cdot f_X(t; B, \lambda) \mathrm{d}t \\
&= \lambda_i (R + \alpha B_i) \sum_{l=i}^{n} \frac{e^{\sum \lambda_j (p_j - p_l)} - e^{\sum \lambda_j (p_j - p_{l+1})}}{\sum \lambda_j}
\end{aligned} \tag{3.3-9}
$$

where $j \in active(p_l)$ for all valid l.

Impacts of Individual Block Size on Self-payoff

A player can improve his expected payoff by two means - increasing either (i) his expected reward or (ii) his chance of being rewarded. Although both of them are implemented by adjusting the block size, they are in conflicting directions. When a player chooses a big block size, he prefers to increase his potential transaction fee reward by including more transactions, at the cost of lowering his chance to be rewarded (since a bigger block incurs a longer propagation time). When a player chooses a small block size, he prefers to increase his chance for receiving a reward by shortening the propagation time of his block (therefore prolonging his mining time), at the cost of decreasing his reward amount from transaction fees.

Impacts of Individual Block Size on Others' Payoffs

Theorem 2. A player indirectly increases each of his rivals' payoff by increasing his own block size.

Proof. We calculate the first-order derivatives of player k's payoff over B_i:

$$\frac{\partial P_k}{\partial B_i} = \beta\lambda_i\lambda_k(R+\alpha B_k)\sum_{l=max\{i,k\}}^{n}\frac{e^{\sum\lambda_j(p_i-p_l)}-e^{\sum\lambda_j(p_j-p_{l+1})}}{\sum\lambda_j}. \tag{3.3-10}$$

where $k \neq i$ and $j \in active(p_l)$ for all valid l.

Obviously, ${}^{\partial P_k}/_{\partial B_i} \geq 0$ always holds. This result can be interrupted as follows. When any player increases his own block size, it brings external benefits to other players. This is because the player lengthens his own propagation time, allowing others to mine for a longer time. This increases their probability of finding a valid PoW solution. □

3.3.2 Optimal Block Size and Mining Power

According to Eq.(3.3-9), a player's expected payoff is related to the block sizes selected by all the players, as well as the mining power distribution in the whole Bitcoin network. Now, we are interested in finding out how a player's mining power would affect his decision on the block size. Intuitively,

Theorem 3. A player's optimal block size is positively related to his mining power.

Proof. We assume two heterogeneous players: player 1 with lower mining power and player 2 with higher mining power. Besides, we assume there is no bound on the block size. Thus, players are allowed to put as many transactions as they want in the block. We define player 1's mining power as h_1 and player 2's mining power as h_2 , respectively. Given $h_1 + h_2 = 1$ and $h_1 < h_2$, then we can see $\lambda_1 = {}^{h_1}/_{T}$ and $\lambda_2 = {}^{h_2}/_{T}$. We analyze these two players' payoffs under two possible conditions: (1) $B_1 < B_2$ and (2) $B_1 > B_2$, respectively.

(1) $B_1 < B_2$: This means player 1 with lower mining power would choose a smaller block size than player 2. Then, each player's expected payoff can be expressed as

$$\begin{aligned} P_1 &= (R + \alpha B_1)\left[1 - (1 - h_1)e^{-h_1\beta(B_2 - B_1)/T}\right] \\ P_2 &= (R + \alpha B_2)h_2 e^{-(1-h_2)\beta(B_2 - B_1)/T}. \end{aligned} \tag{3.3-11}$$

To figure out each player's optimal block size, we calculate the first-order derivative of P_j over B_j in Eq.(3.3-12).

$$\begin{aligned} \frac{\partial P_1}{\partial B_1} &= \alpha\left[1 - h_2 e^{\frac{-h_1\beta(B_2-B_1)}{T}}\right] - \frac{h_1 h_2}{T}\beta(R + \alpha B_1)e^{\frac{-h_1\beta(B_2-B_1)}{T}} \\ \frac{\partial P_2}{\partial B_2} &= \alpha h_2 e^{\frac{-h_1\beta(B_2-B_1)}{T}} - \frac{h_1}{T}\beta(R + \alpha B_2)h_2 e^{\frac{-h_1\beta(B_2-B_1)}{T}} \end{aligned} \tag{3.3-12}$$

Let ${}^{\partial P_2}/_{\partial B_2} = 0$, then $B_2 = {}^{T}/_{\beta(1-h_2)} - {}^{R}/_{\alpha}$. Let $B_2^* = \underset{B_2 \geq 0}{\operatorname{argmax}} P_2$, thus we conclude that

$$B_2^* = \begin{cases} 0 & \text{if } \frac{R}{\alpha} \geq \frac{T}{\beta(1-h_2)} \quad \text{case (a)} \\ \frac{T}{\beta(1-h_2)} - \frac{R}{\alpha} & \text{otherwise} \qquad\quad \text{case (b)} \end{cases}$$

Now we discuss the optimal $B_1^* = \operatorname{argmax}_{0 \leq B_1 \leq B_2^*} P_1$. B_1^* is dependent on B_2^*. Given player 2's dominant strategy, player 1 should choose his best response. In case (a),

$B_2^* = 0$, then

$$\frac{\partial P_1}{\partial B_1} = \alpha \left[1 - h_2 e^{{}^{h_1 \beta B_1}/_{T}}\right] - \frac{h_1 h_2}{T} \beta (R + \alpha B_1) e^{{}^{h_1 \beta B_1}/_{T}}.$$

For any $B_1 \geq 0$, ${}^{\partial P_1}/_{\partial B_1} \leq 0$ always holds. Thus, $B_1^* = 0$. In the case (b), $B_2^* = {}^{T}/_{\beta(1-h_2)} - {}^{R}/_{\alpha}$, and the payoff function P_1 for player 1 is concave in B_1 since ${}^{\partial^2 P_1}/_{(\partial B_1)^2} < 0$ always holds. As ${}^{\partial P_1}/_{\partial B_1}|_{B_1=B_2} < 0$ holds if ${}^{R}/_{\alpha} < {}^{T}/_{\beta(1-h_2)}$, there is a unique B_1^*, satisfying $B_1^* < B_2^*$. Obviously, the analysis result is consistent with the condition $B_1 < B_2$.

(2) $B_1 > B_2$: This means player 1 with lower mining power would choose a bigger block size than player 2. Now, each player's expected payoff can be expressed as

$$\begin{aligned} P_1 &= (R + \alpha B_1) h_1 e^{-(1-h_1)\beta(B_1 - B_2)/_{T}} \\ P_2 &= (R + \alpha B_2) \left[1 - (1 - h_2) e^{-h_2 \beta (B_1 - B_2)/_{T}}\right]. \end{aligned} \tag{3.3-13}$$

Then by calculating $\partial P_1 / \partial B_1 = 0$, we obtain the optimal block size B_1^* for player 1, which is listed below.

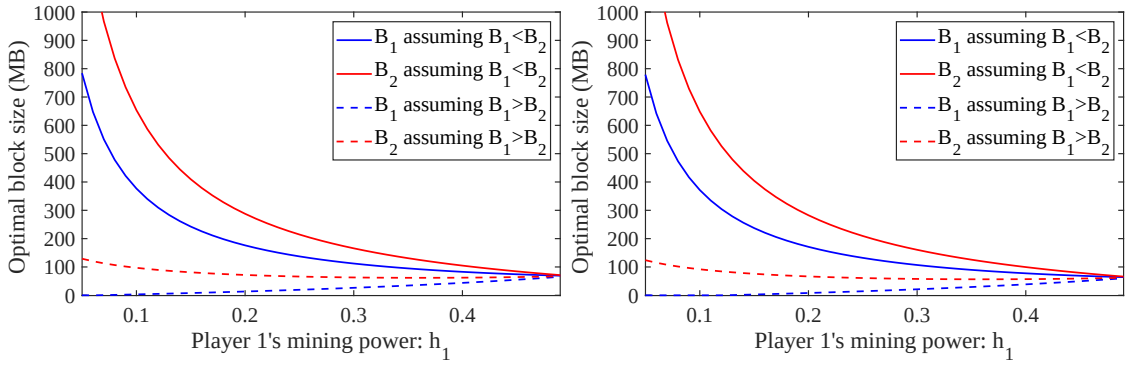
$$B_1^* = \begin{cases} 0 & \text{if } \frac{R}{\alpha} \geq \frac{T}{\beta(1-h_1)} \quad \text{case (a)} \\ \dfrac{T}{\beta(1-h_1)} - \dfrac{R}{\alpha} & \text{otherwise} \qquad\quad \text{case (b)} \end{cases}$$

We verify if $B_2^* < B_1^*$ holds. We begin with case (a), where ${}^{R}/_{\alpha} \geq {}^{T}/_{\beta(1-h_1)}$ and $B_1^* = 0$. Thus, ${}^{\partial P_2}/_{\partial B_2} = \alpha \left(1 - h_1 e^{{}^{h_2 \beta B_2}/_{T}}\right) - ({}^{h_1 h_2}/_{T}) \beta (R + \alpha B_2) e^{{}^{h_2 \beta B_2}/_{T}}$. When $B_2 = 0$, we obtain ${}^{\partial P_2}/_{\partial B_2}|_{B_2=0} = (\alpha - {}^{h_1}/_{T} \beta R) h_2$.

Based on ${}^{R}/_{\alpha} \geq {}^{T}/_{\beta(1-h_1)}$, we can see $B_2^* = 0$ if $\alpha \leq {}^{R\beta h_1}/_{T}$ (since ${}^{\partial P_2}/_{\partial B_2}|_{B_2=0} \leq 0$), and $B_2^* > 0$ if ${}^{R\beta h_1}/_{T} \leq \alpha \leq {}^{R\beta h_2}/_{T}$ (since ${}^{\partial P_2}/_{\partial B_2}|_{B_2=0} > 0$). Thus, $B_2^* \geq B_1^*$ holds in case (a). We proceed with case (b), where ${}^{R}/_{\alpha} < {}^{T}/_{\beta(1-h_1)}$ and $B_1^* = {}^{T}/_{\beta(1-h_1)} - {}^{R}/_{\alpha}$. When $B_2 = B_1^*$, we obtain $\frac{\partial P_2}{\partial B_2}|_{B_2=B_1^*} = (1 - \frac{h_1}{h_2}) \alpha h_2$, which is bigger than 0 given $h_1 < h_2$ and P_2 is a concave function in B_2 as ${}^{\partial^2 P_2}/_{(\partial B_2)^2} < 0$ holds, then $B_2^* > B_1^*$. Thus, the result violates the condition $B_1 > B_2$.

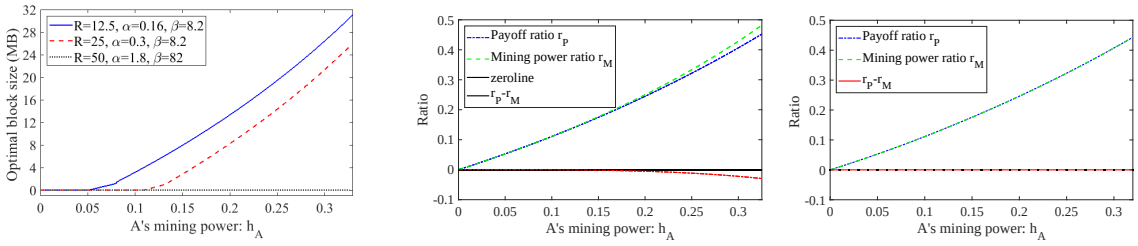
Based on the previous discussion, it is obvious to see that, given $h_1 < h_2$, $B_1^* < B_2^*$ always holds. □

We also use real-world data from the Bitcoin network to validate our conclusions.



(a) $R = 12.5, \alpha = 0.16, \beta = 8.2$ (b) $R = 25, \alpha = 0.3, \beta = 8.2$

Figure 3.1: Player with lower mining power will have smaller optimal block size.



(a) Optimal block size in different settings of parameter triple (R, α, β).

(b) Payoff ratio Vs Mining power ratio between A and M when $R = 12.5$.

(c) Payoff ratio Vs Mining power ratio between A and M when $R = 25$.

Figure 3.2: Numerical analysis based on real-time information from [10].

The numeric results can be seen in Fig. 3.1. We plot the optimal block sizes for both players under different conditions. In Fig. 3.1(a), the solid lines represent B_1^* and B_2^* under the condition $B_1 < B_2$. The red solid line is above the blue solid line, which is consistent with the condition $B_1 < B_2$. As h_1 increases, *i.e.*, h_1 is close to h_2, these two solids approach and finally intersect when $h_1 = h_2$. However, the result reflected by the dashed lines violates the condition $B_1 > B_2$ (the blue dashed line is below the red one). In the Fig. 3.1(b), we modify the Bitcoin network settings by changing the block subsidy, transaction fee density and the network delay, but we still get the same trend. Thus, a player with low mining power should choose a small block size while a player with high mining power should choose a big one.

3.4 System Equilibrium Analysis and Search

The payoff presented in Eq.(3.3-9) is derived given all players' strategies. If a player changes his strategy, then the payoffs of all the other players will also be affected. We are interested in finding equilibria, *i.e.*, strategies of all players such that no player can improve its payoff by changing its strategy. It is infeasible to express a player's payoff in a symbolic manner, since it is a function of all players' strategies as well as the difficulty parameter, which is expressed as an implicit function. Therefore, we use numerical analysis to find equilibria in the system.

We implement an equilibrium-search-tool, a tool we use to numerically search for an equilibrium, and that works in the following manner. The equilibrium-search-tool receives as an input for the system income and expenses parameters, as well as a list of tuples representing all players' strategies. Each tuple of that list is in the form of $\{i, h_i, B_i\}$, where i is a player's index, h_i is the mining power controlled by player i, and B_i is the block size selected by player i.

Iteratively, the equilibrium-search-tool randomly chooses an input tuple $\{i, h_i, B_i\}$, and searches what value of a block size B_i will result in maximal payoff for player i. This process is repeated until no player increases its payoff by changing any of its rigs, meaning an equilibrium is reached. Note that all equilibria found by such process are only ϵ-Nash equilibria, as they are limited by the numerical precision of the calculation. To counter that predicament, we repeat the search process with different random start times and different optimizing order. In all conducted experiments, the randomness introduced had no effect on the output equilibrium. That strengthens our analysis of an equilibrium.

3.5 One Misbehaving Miner

We begin our analysis with an assumption that there is exactly one miner with misbehavior. For simplicity, we assume that miners are divided into two groups, a corrupted pool A controlled by the misbehaving miner, and the rest of the miners M behaving heuristically. It is irrelevant whether M operates as a single pool, as a collection of pools, or individually. Each miner in M always honestly mines with the default block size $\overline{B}$, while A manipulates his block size B_A to optimize his expected payoff.

3.5.1 Attacker's Expected Payoff

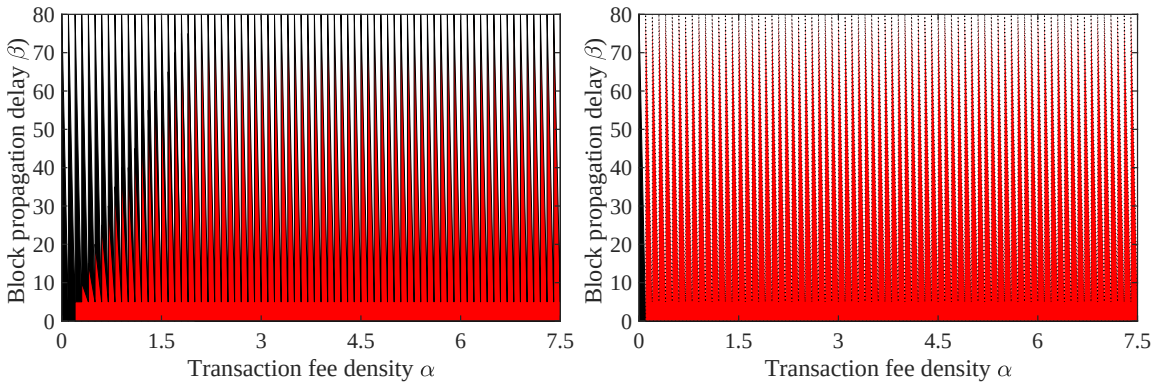
Let A's mining power equal to h_A, then M controls $h_M = 1 - h_A$ of the total mining power. Based on Eq.(3.3-9), we calculate A's expected payoff in Eq.(3.5-14).

$$P_A = \begin{cases} (R + \alpha B_A)\left[1 - (1 - h_A)e^{-h_A\beta(\overline{B} - B_A)/T}\right] & \text{if } B_A \leq \overline{B}. \\ (R + \alpha B_A)h_A e^{-(1-h_A)\beta(B_A - \overline{B})/T} & \text{otherwise.} \end{cases} \tag{3.5-14}$$

3.5.2 Numerical Analysis on One-Sided Misbehavior

The expected payoff presented in Eq.(3.5-14) is derived given A's mining power and block size. In fact, A's optimal block size B_A^* can be decided according to $\partial P_A / \partial B_A = 0$, and is an implicit function related to h_A. Now, we focus on how A's mining power h_A would influence his decision on B_A. Thus, we allow A to put as many (or few) transactions as he wants in the block. Since it is infeasible to express B_A^* in a symbolic manner, we use numerical analysis to find B_A^* under different values of h_A, in hopes of finding out a unified and reasonable explanation to these numerical results.

Fig. 3.2(a) shows how A's optimal block size B_A^* is related to his mining power h_A, given $\overline{B} = 1$ and $T = 600$. We fix the parameters R, α, and β and vary the parameter h_A. Values of each set (R, α, β) are based on the real-time information

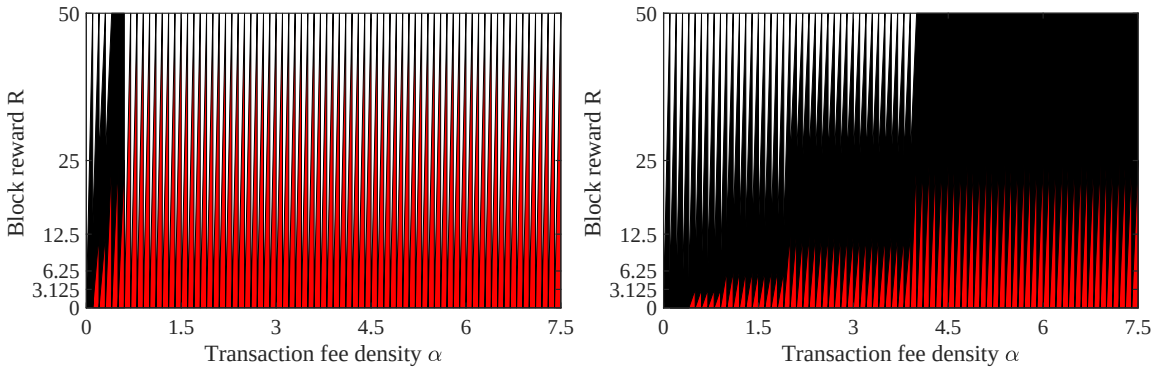


(a) $R = 12.5$ (b) $R = 0$

Figure 3.3: Existence of the peaceful equilibrium when R is fixed.

from [39]. From the black dashed line we can see A always puts few transactions in his block. $B_A^* = 0$ is reasonable due to the huge network delay, *i.e.*, $\beta = 82$. When we set the network delay to a normal level ($\beta = 8.2$), we find A's optimal block size becomes larger as his mining power increases. We can also see that decreasing the block subsidy motivates A to increase his block size to optimize payoff, even if the transaction fee density decreases. Thus, we could predict that, once the transaction fee dominates the Bitcoin reward mechanism, optimal block size increases for each player no matter what his mining power is. Results in Fig. 3.2(b)-(c) show that, the payoff ratio is equivalent to the mining ratio between A and M when A adopts his optimal block size while M follows default block size. In fact, if only A seeks to gain more by manipulating his block size, we could see the payoff distribution between A and M still follows the fairness requirement: the payoff should be distributed proportionally to the mining power in the long run.

Since any block over 1 MB will be denied in the real Bitcoin network, we are interested in finding peaceful equilibria which refrain A with any h_A from manipulating his block size. That is, we want to find on what conditions, A can optimize his payoff by always filling up his 1-MB block. Obviously, $B_A^* \geq 1$ must hold on those conditions, such that A is forced to fill up his block, in order to maximize his expected payoff



(a) $\beta = 8.2$ (b) $\beta = 82$

Figure 3.4: Existence of the peaceful equilibrium when β is fixed.

(although it is still not his optimal payoff due to the $1MB$ limitation).

In Fig. 3.3(a), we firstly fix the parameter R as 12.5, and we vary the parameters α and β. The red shaded areas represent the existence of peaceful equilibria given different values of α and β (if you are reading this dissertation in a black-and-white version, those grey parts represent equilibria). The figure shows that, a peaceful equilibrium tends to appear when α is high and β is low. That is, if the transaction fee density is high enough and the network delay is within a low range, A's optimal block size is always over 1 MB, which forces him to maximize his obtainable payoff by choosing his block size as the default 1 MB. In Fig. 3.3(b), we assume the block reward runs dry, *i.e.*, $R = 0$, and we cannot observe a peaceful equilibrium unless either $\alpha \to 0$ or $\beta \geq 600$ (which is impossible in reality unless a delay attack exists). This is reasonable since transaction fees are the only incentive.

In Fig. 3.4, we fix the parameters β (81.92 in (a) and 8.192 in (b)), and we vary the parameters R and α. These figures show that if R is low and α is high, then a peaceful equilibrium is possible; however, if either of these parameters is deviant, then there can be no peace. This further confirms the result in Fig. 3.3(b), as the block subsidy goes low (even dry), the transaction fee dominates, A with any mining power has the motivation to choose a larger block size. Besides, Fig. 3.4 also implies β is especially important. As we can compare, the red shaded area obviously shrinks

as β goes up. That is, dramatic increase on the network delay will lead A to choose a block size smaller than 1 MB. In practice, if someone issues a delay attack (which can delay a message for at most 20 minutes), then players may have no incentive to collect transactions in their blocks. This would be a disaster for the liveness of Bitcoin.

3.6 Two Pools

We proceed to analyze the case with two misbehaving miners where miner L has a small pool and miner H has a big pool. By size comparison, we simply mean that L has less mining power than H. Obviously, $B_L^* \leq B_H^*$. A third entity M represents the rest of the Bitcoin mining market and behaves heuristically, using the default block size $\overline{B}$.

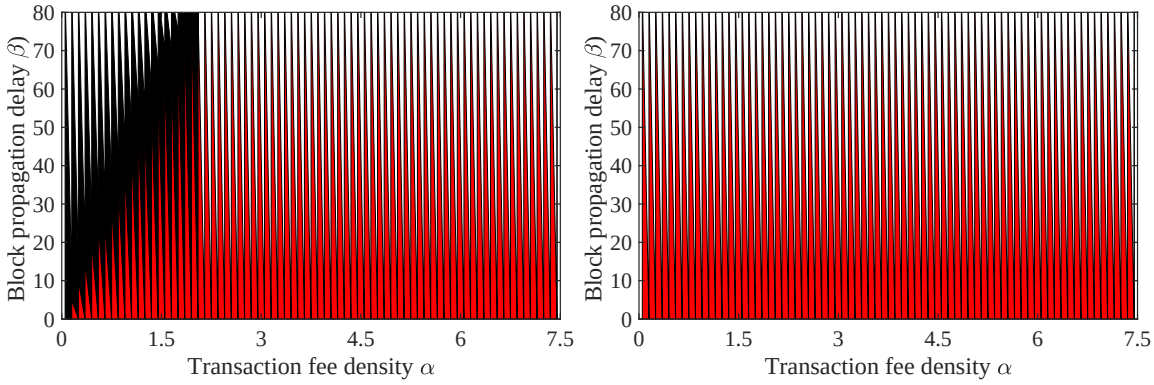
3.6.1 Peaceful Equilibria

First, we are interested in finding peaceful equilibria which refrain both L and H from deviating from $\overline{B}$. On these conditions, both B_L^* and B_H^* must be no less than $\overline{B}$ so that if L and H want to maximize their expected payoffs, they have to fill up their blocks. The payoff functions of L and H under the condition of $\overline{B} \leq B_L \leq B_H$ are listed in Eq.(3.6-15):

$$\begin{aligned} P_L &= (R + \alpha B_L)\frac{h_L}{h_M + h_L} \times \\ &\left[e^{-h_M \beta (B_L - \overline{B})/T} - h_H e^{-h_M \beta (B_H - \overline{B})/T - h_L \beta (B_H - B_L)/T}\right] \\ P_H &= (R + \alpha B_H) h_H e^{-h_M \beta (B_H - \overline{B})/T - h_L \beta (B_H - B_L)/T}. \end{aligned} \tag{3.6-15}$$

We can calculate B_L^* and B_H^* by solving the equations $\partial P_L/\partial B_L = 0$ and $\partial P_H/\partial B_H = 0$ if R, α, and β are all given. Again, we use numerical analysis here and the corresponding results are present in Fig. 3.5 and Fig. 3.6.

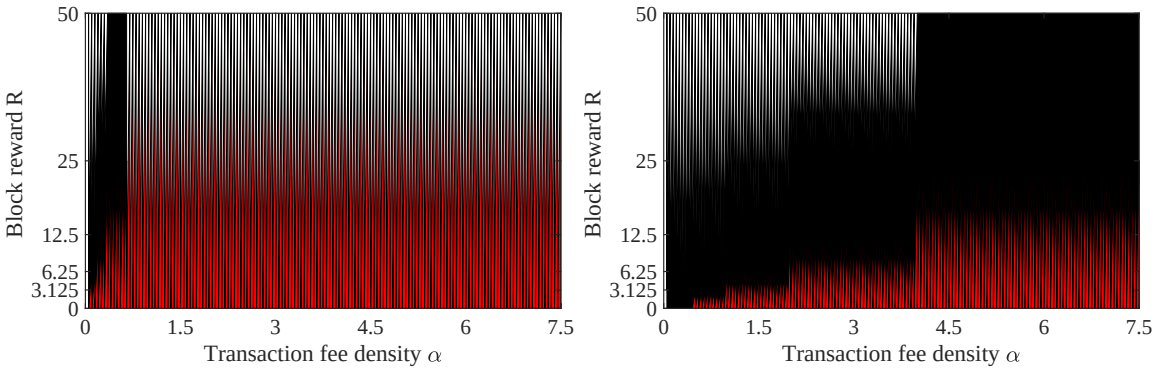
In Fig. 3.5, we fix R while vary α and β and the red shaded areas are peaceful



(a) $R = 12.5$ (b) $R = 0$

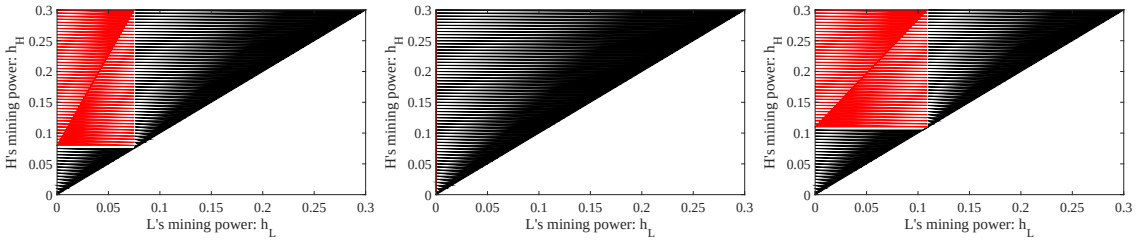
Figure 3.5: Existence of the peaceful equilibrium when R is fixed.

equilibria given different values of α and β. Comparing Fig. 3.5 with Fig. 3.3, we find the peaceful equilibria are reduced. This is because more misbehaved miners leads to a more unstable and unpredictable mining environment. When $R = 12.5$, Fig. 3.5(a) shows that a peaceful equilibrium appears if α is high and β is low. That is, if transaction fee density is high enough and network delay is within a low range, both B_L^* and B_H^* are over 1 MB, thereby they have to choose the default block size to obtain a maximized payoff in expectation. In Fig. 3.5(b), we assume the block reward runs dry, *i.e.*, $R = 0$, we find that a peaceful equilibrium comes as long as the network delay β falls into a reasonable range since transaction fees are the only income source. In Fig. 3.6, we fix β while vary R and α. These figures show that, if R is low and α is high, then a peaceful equilibrium is possible; however, if either of these parameters is deviant, then there can be no peace. Thus, when transaction fees dominate, L and H with any mining power have the motivation to choose larger block sizes. Fig. 3.6 also shows the red shaded area obviously shrinks as the number of misbehaved players increases. Thus, the more misbehaved players, the less stable Bitcoin mining will be.



(a) $\beta = 8.2$ (b) $\beta = 82$

Figure 3.6: Existence of the peaceful equilibrium when β is fixed.



(a) $\beta = 8.2$ (b) $\beta = 82$ (c) $\beta = 82$

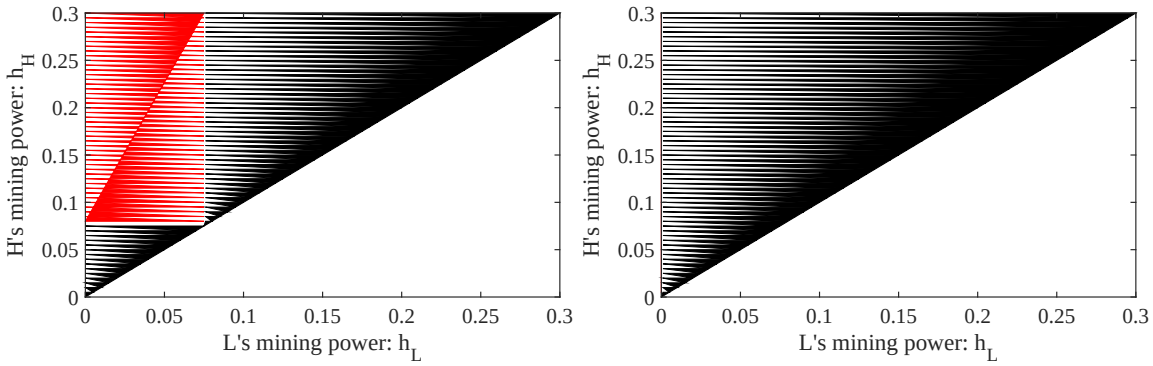
Figure 3.7: Existence of the peaceful equilibrium when β is fixed. Red shaded areas represent parameter combinations where the peaceful equilibrium exists.

3.6.2 One-Sided Misbehaviors

We also want to know when only L is deviant from the default block size, which means $B_L^* < \overline{B}$ and $B_H^* \geq \overline{B}$. Below are the payoff functions under the assumption $B_L < \overline{B}$ and $B_H = \overline{B}$. The equation given below is the corresponding payoff functions when only L misbehaves.

$$\begin{aligned} P_L &= (R + \alpha B_L) \times \\ &\left[1 - \frac{h_M}{h_L + h_M} e^{-h_L \beta (\overline{B} - B_L)/T} - \frac{h_L h_H}{h_M + h_L} e^{-h_L \beta (\overline{B} - B_L)/T}\right] \\ &= (R + \alpha B_L)\left[1 - (1 - H_L) e^{-h_L \beta (\overline{B} - B_L)/T}\right] \\ P_H &= (R + \alpha \overline{B}) h_H e^{-h_L \beta (\overline{B} - B_L)/T}. \end{aligned} \tag{3.6-16}$$

Again, we conduct numerical analysis to find Nash equilibrium and see how parameters affect the achieved equilibria. The red shade in Fig. 10 shows all possible



(a) $\beta = 8.2$ (b) $\beta = 82$

Figure 3.8: Existence of the peaceful equilibrium when β is fixed.

equilibria when L's mining power varies given some fixed parameter(s). We can see with a low transaction rate and a high network delay, a miner with lower mining power cannot achieve more payoff even by choosing a smaller block size than the default block size. Thus, we can conclude that, the manipulation on the block size doesn't work here. In practice, however, the transaction rate is high and the network delay is usually controlled at a reasonable level. In those cases, we find that there exists an upper bound for L's mining power. See the example in Fig. 10(a) given the parameters: $R = 12.5, \alpha = 6, \beta = 8.2$, the upper bound is around 8%. Once exceeding this bound, there is no existence of equilibria, which means, if L holds more than 8% mining power, then his optimal block size is definitely smaller than $\overline{B}$. As a misbehaved player, L could choose a B_L^* instead of $\overline{B}$. Thus, the current default block size 1 MB can never ensure any player would behave well, since a misbehaved player with over 8% mining power can manipulate his block size by setting it smaller than the default to gain more than his fair reward share.

3.6.3 Two-Sided Misbehaviors

We now analyze when both sides want to misbehave, which means $B_L^* < \overline{B}$ and $B_H^* < \overline{B}$. Given these assumptions, we start our analysis on the two-sided-misbehavior scenario by expressing the utilities of both parties.

According to our previous analysis, we know that the player with higher mining power always performs better if its block size is bigger than that of a miner with lower mining power, *i.e.*, $B_H \geq B_L$ given $h_H \geq h_L$. Below are the payoff functions under the assumption $B_L \leq B_H < \overline{B}$:

$$\begin{aligned} P_L =& (R+\alpha B_L)\left[1-\frac{h_H}{h_L+h_H}e^{-h_L\beta(B_H-B_L)/T}\right. \\ & \left.-\frac{h_L h_M}{h_L+h_H}e^{-h_L\beta(\overline{B}-B_L/T)-h_H\beta(\overline{B}-B_H)/T}\right] \\ P_H =& (R+\alpha B_H)\left[\frac{h_H}{h_L+h_H}e^{-h_L\beta(B_H-B_L)/T}\right. \\ & \left.-\frac{h_H h_M}{h_L+h_H}e^{-h_L\beta(\overline{B}-B_L)/T-h_H\beta(\overline{B}-B_H)/T}\right]. \end{aligned} \tag{3.6-17}$$

According to the numerical analysis, we found the anticipated equilibrium is hardly to be found in the current Bitcoin network. This means, even in such a simplified scenario, the default block size 1 MB cannot refrain players from manipulating block sizes to gain more. Thus, we need to reconsider how the Bitcoin network designs the default block size $\overline{B}$. An important design protocol is that, $\overline{B}$ should be smaller the optimal block size of every player with any mining power. In the next part, we will use the current information of the Bitcoin network to recommend a suitable default block size according to the previously-mentioned design protocol.

3.6.4 Extension on Real Bitcion Mining Network

We now make an educated estimation on the real Bitcion network. In Bitcoin today, there are 7 mining pools [11] controlling about 85% of the mining power, while the rest is divided among many smaller mining pools. Although they vary in size, we approximate that situation by assuming 8 equal-size miners.

Currently, the rewards from minting and fees are 12.5 BTC and about 1 BTC, respectively. We extend from previous analysis and try to find peaceful equilibrium among eight mining pools. With the help of Matlab, we find the default block size, which is suggested to be 4 MB.

3.7 Related Work

A vast majority of previous work examines possible types of misbehaviors against the Bitcoin protocol and suggest adaptations of the protocol to encourage honest mining, and thereby ensuring its security. We very briefly mention some of these works here. Usually, misbehaviors at the miners' side tend to be referred to as *Mining attacks*. Eyal and Sirer [47] develop the selfish mining attack, a deviant mining strategy that enables miners to get more than their fair share of rewards. Other works, notably Sapirshtein *et al.* [94] have analyzed selfish mining in more detail using Markov Decision Processes (MDP). Various other attacks have been studied. For example, members of a mining pool can launch a block withholding attack against the pool itself [37], and this harms the victim pool and its other members, but actually increases the revenue of the rest of the network. In [45], the author considers attacks performed between different pools where users are sent to infiltrate a competitive pool giving rise to a pool game. [23] deals with information propagation and Sybil attacks. Most of them are consider a model where the subsidy is the dominant incentive for mining. In this work, we analyze how a miner's behavior differs according to his block size in a reward mechanism with block subsidies and transaction fees. Möser and Böhme [79] review and analyze the history of transaction fees in Bitcoin. They conclude that historically miners prefer to follow the protocol rules rather than optimize their gains. They predict such a state is sustainable only when fees are a negligible part of the incentive.

There also exists real attacks at the network layer. Each Bitcoin node is connected over TCP to many peers, with a default maximum of 125. The peer-to-peer connections between these nodes can be inferred through various techniques [78, 27]. Heilman *et al.* [54] demonstrated a network-level eclipse attack where a single node monopolizes all possible connections to a victim and eclipses it from the network. Thereby, the eclipsing node can filter the eclipsed node's view of the blockchain.

Although a few proposed counter-measures have been implemented that reduce the feasibility of carrying out an eclipse attack by a single node, multiple nodes can collude and still succeed in eclipsing. Besides, Coinscope [78] proposed non-trivial techniques to map out the Bitcoin network topology as well as the mining power of various nodes. This network knowledge can further help a network-level attacker.

In addition, game theory has been widely applied to analyze Bitcoin attacks. Several recent works have examined the game theoretic consequences of attacks. Kiayias *et al.* [49] performed a theoretical analysis of various selfish mining strategies in the fixed-reward model, and proved that when miners are small enough, the default mining behavior is an equilibrium. See also [70] for a (cooperative) game theoretic analysis regarding pool mining.

3.8 Conclusion

We define and analyze the block size game exploring how block sizes form as a function of block subsidy and transaction fees. We show that once fees become significant, then manipulation on the block size appears. However, it does not happen uniformly as previously believed, while it has a significant effect on blockchain security. This means that base rewards are critical for system security, and should be achieved either by subsidies, fee backlogs, or alternative fee schemes [68, 90]. We show that the default block size 4 MB is sufficient to avoid deviant mining behavior of the block sizes in presented scenarios; we expect Bitcoin to drop below this threshold within a decade.

CHAPTER 4

EXTENSION: STORAGE MANIPULATION

4.1 Motivation

There exist several types of consensus mechanisms and most of them requires node to show their qualification by contributing resources, such as computation power and storage space. Meanwhile, since it is append-only, a blockchain grows forever, incurring a high requirement on node's storage. These high resource requirements pose a challenge on mobile devices, and thus hindering the development of mobile blockchain services. To facilitate blockchain-powered applications in future mobile IoT systems, offloading appears to be a viable solution.

We will focus on how computation/storage offloading can help nodes in the mobile blockchain network. To solve the nodes' resource management problem, we exploit game theory to analyze the complex interactions among service providers and mobile blockchain nodes. Then, we turn the original resource allocation problem into a Nash equilibrium problem, in which each miner's equilibrium strategy is his optimal strategy if NE exists. In this chapter, we will focus on blockchain storage offloading and next chapter will be mining-power-oriented.

4.2 Introduction

The requirement of storing the entire blockchain is a major challenge for blockchain mining in mobile environments, and thus has hindered the development of blockchain-powered mobile applications. Storage outsourcing to a cloud service provider (CSP)

is a viable solution. An individual miner can store his blockchain in the cloud and then validate transactions by querying the CSP. However, validation outsourcing to a remote CSP incurs delay and damages a miner's winning probability in the mining competitions. To shorten such an unwanted delay, miners can also cache the unspent transaction output (UTXO) set in a nearby edge service provider (ESP) for fast transaction validations, which definitely brings extra costs. In this chapter, we consider a two-layer outsourcing paradigm to solve storage shortage for mobile miners. Due to the delay-cost tradeoff when selecting service providers, we can model interactions among miners as a non-cooperative game and formulate a Nash equilibrium problem to investigate the effects of outsourcing on miners' utilities. We also study the access probability of UTXOs with different generation times. This will guide miners on how to select unspent transaction outputs if they decide only to cache the partial UTXO set in the edge. We further extend our game by modeling multiple mining rounds as a one-shot game to see how the cache update frequency affects miners' strategies. Numerical evaluation is conducted to show the feasibility of storage outsourcing and to validate the proposed models and theoretical results.

4.3 System Model and Problem Formulation

This chapter focuses on a mobile blockchain mining network. The basic setting on blockchain technique is based on Bitcoin. That is, we assume all blockchain users follow the Proof-of-Work (PoW) consensus protocol and apply the UTXO account model. Corresponding notations are listed in Table 4.1. Our model includes two service providers and a set of n miners using mobile devices. The SP side consists of a remote CSP and a nearby ESP, offering storage and query services to miners. Usually, large datasets are outsourced to the CSP given that its resources are rich and cheap. If users want to get query answers quickly, then the ESP is a better choice due to its close physical location. The delay-cost tradeoff makes the coexistence of the

Table 4.1: Summary of Notations.

Symbol	Description
p_e / p_c	price of edge VM / cloud query
d_e / d_c	single transaction validation delay from edge / cloud
b / ρ	block base reward / transaction fee density
S	number of network-wide unspent transaction outputs
a_k	probability of accessing unspent transaction output k
n	number of miners
m_i / B_i / U_i	miner i's mining power / budget / utility
R_i / P_i / C_i	miner i's expected reward / winning probability / cost
x_i / y_i	miner i's block size / edge storage request
z_{ik}	miner i's cache decision on unspent transaction output k
X / Y	total cloud-mining / self-mining units
X_{-i}	total cloud-mining units except m_i's, *i.e.*, $X_{-i}=X-x_i$
Y_{-i}	total self-mining units except m_i's , *i.e.*, $Y_{-i}=Y-y_i$
r_i	m_i's request vector, in the form of (x_i, y_i)
$\boldsymbol{r_{-i}}$ / $\boldsymbol{r}$	all miners except m_i's / all miners' request profile
β	discount factor caused by a unit-time delay

Table 4.1: Edge cache storage unit is tailored as an unspent transaction output size.

CSP and the ESP in order to satisfy users of different service quality requirements and different budgets.

The user side is a network with n miners using different mobile devices. Miners compete against each other in hopes of generating new blocks and getting rewards. The process of adding a block to the blockchain is viewed as a mining round. In a mining round, each miner has to create his own candidate block. The process of creating a candidate block consists of 3 steps. First, a miner validates unconfirmed transactions, and then bundles them to form a Merkle tree structure, which produces a Merkle root. Finally, the miner uses his computation, *i.e.*, mining power, to solve a PoW puzzle based on the previously produced Merkle root. Due to the storage limitation of their devices, all miners outsource their blockchain in the cloud. A miner i will issue queries to the CSP when he needs to validate transactions. Each query is charged at the price of p_c and the corresponding answer returns at a delay of

d_c. Thus, the cost and the time for miner i who wants to validate x_i transactions are $x_i p_c$ and $x_i d_c$, respectively. Once a miner finds a PoW solution, he will broadcast his block in the mining network for consensus. A miner whose block reaches consensus first will be the winner and get rewarded in that round. Mining rewards come from two sides: one is a fixed base reward for a block creation, the other is transaction fees accumulated in this block. Thus, if miner i successfully mines a block containing x_i transactions, his expected reward is $R_i = b + \rho x_i$, given the base reward b and the transaction fee density ρ.

Obviously, miners can have different mining rewards, depending on how they decide their block sizes. Selecting a large value of x_i (under the constraint that x_i is more than the number of unconfirmed transactions in the network) definitely brings a higher expected reward R_i for miner i. However, it also takes a longer delay to validate those transactions. As we mentioned before, miner i wins unless he is the first to solve his PoW puzzle and propagate his block to reach consensus. A large delay $x_i d_c$ damages miner i's winning probability in the mining competition. The tradeoff between the expected reward and the winning probability poses a challenge to miner i on deciding the value of x_i. Except shrinking his block for a shorter delay, miner i can turn to the ESP for a low-latency query service. In this case, miner i can cache the UTXO set in the edge for fast transaction validations. A transaction is valid if each of its inputs is available in the UTXO set. If miner i maintains a cache of the UTXO set in the edge, he can validate transactions without accessing the blockchain in the cloud. The ESP processes each query from miner i by iterating his cache space. Let d_e denote a single transaction validation delay from the ESP, and then the total transaction validation delay will be $x_i d_e$ if he wants to validate x_i transactions.

In fact, $x_i d_e$ is the optimal result for miner i with a block size x_i, given the fact that he maintains the entire UTXO set in the edge. In reality, given budget constraints,

high prices and limited capacity of edge resources, miner i may prefer to cache part rather than all of UTXOs in the edge. Thus, the ESP will charge miner i a cost of $y_i p_e$ if he maintains a cache with y_i UTXOs (p_e is a combo charge of storage and query services). We assume that, the CSP offers a lower price, *i.e.*, $p_c < p_e$ while the ESP guarantees a shorter delay, *i.e.*, $d_e < d_c$. These assumptions always hold in the real world and also guarantee the problem discussed in this chapter is meaningful.

Miners participate in mining processes by requesting resources and services from the SPs. Each miner i's request is in the form of $r_i = (x_i, y_i)$, where x_i represents the number of transactions miner i decides to validate and put in his current block, and y_i represents the number of unspent transaction outputs he decides to cache in the edge. Let $\boldsymbol{r} = \{r_1, \cdots, r_n\}$ and $\boldsymbol{r}_{-i}$ represent the request profile of all miners and all other miners except i, respectively. As miners all want to make as much profit as possible, a competition among miners is formed, in which each miner optimizes his utility by deciding his request r_i under the current resource prices (p_e, p_c), while considering his own budget B_i and other miners' strategies $\boldsymbol{r}_{-i}$. Since requests are generated for individual utility maximization, a non-cooperative game is also formed. Miner i's optimization problem is defined as follows.

Problem 1 (OP_{MINER}).

$$\text{maximize} \qquad U_i = R_i \cdot P_i - C_i, \tag{4.3-1a}$$

$$\text{subject to} \qquad C_i \leq B_i, \quad x_i \geq 0, \quad y_i \geq 0, \tag{4.3-1b}$$

P_i and C_i represent miner i's winning probability and his cost charged by the CSP and/or the ESP, respectively. Given their complexity, accurate definitions and detailed explanations of P_i and C_i will be given in the following part. Each miner i aims to maximize his utility and constraint (4.3-1b) ensures that i is within his budget B_i.

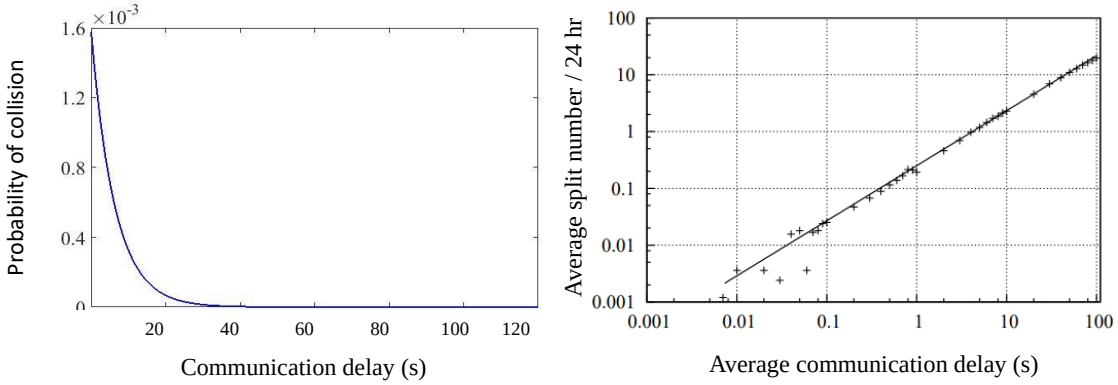
4.4 Individual Winning Probability and Cost

4.4.1 Cache Hit/Miss and Cost

As we mentioned before, miner i can validate any unconfirmed transaction without accessing his cloud storage only if he maintains the complete UTXO set in the edge. Currently, the UTXO set is close to 4 GB and it also grows as fast as the blockchain itself. It has been predicted that the size of the UTXO set will grow to close to 20 GB within the next few years, even when some effective actions are implemented for size reductions. Taking cost-efficiency into consideration, miner i may maintain partial instead of all UTXOs in the edge. We assume the complete UTXO set contains S unspent transaction outputs in total and the size of each is identical. If miner i caches a size of y_i UTXOs ($y_i < S$), then not all transactions can be validated in the edge, given the fact that some UTXOs are missing in the miner i's edge cache. In the case of cache missing, all those transactions failing to be validated in the edge will be redirected to the CSP for further confirmation, and miner i is responsible for paying the corresponding query cost to the CSP. Let h_i denote miner i's cache hit rate in the edge, then his total cost can be expressed as $C_i = p_c x_i(1 - h_i) + p_e y_i$. Definitely, h_i is a function over the variable y_i. Intuitively, the relationship between h_i and y_i can be characterized by a uniform distribution, *i.e.*, $h_i = y_i/S$.

4.4.2 Validation delay and Winning Probability

The mining power m_i characterizes the probability that miner i happens to solve a PoW puzzle. However, outsourced transaction validations incur delay and discount miners' winning probability. Miner i starts mining until his x_i transactions are validated. The waiting time damages miner i's winning probability, since other miners with shorter delay can find and publish their blocks during that time. Thus, mining is not just a race on miners' contributed computing power. Generally, miner i's winning



(a) Probability density function of a conflicting block being found while there exists another block being propagated in the network [39].

(b) Average number of blockchain forks per 24 hours as a function of communication delay, averaged over all the nodes in the network [52].

Figure 4.1: Communication delay can cause damage winning probability.

probability P_i is discounted by the delay. Their relation has been studied in Bitcoin [81], which is subject to an exponential distribution as shown in Fig. 4.1(a). Thereby, the discount rate is almost linearly proportional to the delay, as shown in Fig. 4.1(b).

In this chapter, we assume that the proposed network follows the same pattern in Bitcoin. In our setting, the transaction validation delays between the SPs and miners can be an important inducement that lowers miner i's winning probability. We define β as the unit-time discount factor. Given each miner j's delay d_j and his mining power m_j, the weighted average delay is $\sum_{j=1}^{n} m_j d_j$, which leads to a winning probability discount rate of $\beta \sum_{j=1}^{n} m_j d_j$ in the entire mining network. Thus, miner i's winning probability can be captured as $P_i = m_i \left(1 - \beta \sum_{j=1}^{n} m_j d_j\right)$. To focus on the influence of the transaction validation delay, we neglect the block broadcast delay. Miner i's transaction validation delay d_i depends on his request. If he decides to only use the service provided by the CSP, *i.e.*, $y_i = 0$, then d_i is linear to his block size x_i. However, if he caches UTXOs in the edge, all x_i transactions are filtered by the ESP first, and then, with a cache miss rate of $1 - h_i$, the remaining $x_i\,(1 - h_i)$ transactions are redirected to the CSP for further confirmation. Thus, d_i can be expressed in

Eq. (4.4-2).

$$d_i = \begin{cases} d_c x_i & y_i = 0 \\ d_e x_i + d_c x_i \left(1 - h_i\right) & y_i > 0 \end{cases}. \tag{4.4-2}$$

4.5 Game under Uniform Access Probability

In a single mining round, all miners focus on validating transactions and solving their own PoW puzzles. Once a block is found, all miners move on to find the next block. This process is repeated indefinitely. The repeated generation of blocks becomes a series of independent one-shot competitions. We consider each mining round as a one-shot game played by all miners. A miner's strategy is the choices of the block size and the cache size in the edge. The choices are made a-priori by all miners. The cached unspent transaction outputs are just randomly picked from the complete UTXO set based on a miner's request on the edge cache size, given the uniform distribution assumption in the cache hit rate.

4.5.1 Unlimited Resource Capacity of ESP

We start with a scenario where the ESP has unlimited resource capacity, which means all miners' requests to the ESP will be completely fulfilled. In this scenario, a miner optimizes his utility by solving Problem 1.

Theorem 4. A Nash equilibrium exists in $\mathrm{OP}_{\mathrm{MINER}}$ if the ESP has unlimited resource capacity.

Proof. Any game has NEs if its equivalent variational inequality (VI) problem [51] has a nonempty solution set. Given a VI problem, $VI(K, G)$, if K is convex and compact, and F is monotone on K, then the solution set of $VI(K, G)$ is nonempty, closed, and convex.

We start with the definition on the equivalent VI problem $VI(K, G) \equiv OP(X, U)$,

where

$$G := (\nabla_i U_i)_{i=1}^n, \quad X := ((x_i, y_i))_{i=1}^n, \quad U := (U_i)_{i=1}^n,$$
$$K := \prod_{i=1}^n K_i, \quad K_i := \{(x_i, y_i) | C_i \leq b_i, x_i, y_i \geq 0\}.$$

It can be easily verified that K_i is convex and closed, $\forall i$. Thus, K is convex and compact. G is monotone if and only if $U_i(r_i, \boldsymbol{r}_{-i})$ is concave in r_i for given $\boldsymbol{r}_{-i}$, $\forall i$, which is true as shown below. Since the VI problem has a nonempty solution set, the existence of NE thus follows the sufficient conditions.

We start with the simple case where miner i decides to query the CSP for transaction validation without investing edge cache resources. In this case, $y_i = 0$ holds and miner i needs to solve a single-variable maximization problem if other miners' decisions are known to him. Obviously, $U_i = (b+rx_i)m_i \left(1 - \beta \sum_{j \neq i} m_j d_j - \beta m_i d_i\right) - p_c x_i$ is a concave quadratic function. We then move to the case where miner i decides to cache some UTXO sets in the edge in order to speed up transaction validation. In this case, he has to determine how much storage to request from the ESP as well as his block size. His transaction validation delay is $d_e x_i + d_c x_i (1 - h_i)$ and his cost charged by both the CSP and the ESP is equal to $p_e y_i + p_c x_i (1 - h_i)$. To find miner i's best response strategy, we investigate the concavity of his utility function.

Denote H for the Hessian matrix of U_i:

$$H := \begin{bmatrix} U_{xx}^i & U_{xy}^i \\ U_{yx}^i & U_{yy}^i \end{bmatrix}$$

where

$$U_{xx}^i = \frac{\partial^2 U_i}{\partial x_i^2}, U_{xy}^i = U_{yx}^i = \frac{\partial^2 U_i}{\partial x_i \partial y_i}, U_{yy}^i = \frac{\partial^2 U_i}{\partial y_i^2}.$$

Then the first-order derivative of miner i's utility function is:

$$\begin{aligned}\frac{\partial U_i}{\partial x_i} =& m_i r\left(1-\beta\sum\nolimits_{j=1}^{n} m_j d_j\right) - p_c(1-h_i) \\ & - \beta m_i^2(b+rx_i)\left[d_e + d_c(1-h_i)\right], \\ \frac{\partial U_i}{\partial y_i} =& \left[\beta d_c x_i(b+rx_i)m_i^2 + p_c x_i\right]/S - p_e.\end{aligned}$$

The expressions of the Hessian elements are as below:

$$\begin{aligned}U_{xx}^i =& -2\beta r m_i^2\left[d_e + d_c\left(1-h(y_i)\right)\right], \\ U_{xy}^i =& U_{yx}^i = \left[\beta d_c m_i^2(b+2rx_i) + p_c\right]/S \\ U_{yy}^i =& 0.\end{aligned}$$

Next, we show H is negative definite by proving its leading principal minors, *i.e.*, U_{xx}^i and $det(H)$, are smaller than 0.

$$\begin{aligned}det(H) &= U_{xx}^i U_{yy}^i - U_{xy}^i U_{yx}^i \\ &= -\left[(p_c + \beta b d_c m_i^2 + 2\beta r d_c m_i^2 x_i)^2\right]/S^2,\end{aligned} \tag{4.5-3}$$

the sign of which is always negative for a non-empty block. Obviously, miner i has a concave utility function that definitely will yield a maximal utility point. Therefore, we have proved that U_i is strictly concave with respect to (x_i, y_i). Accordingly, the Nash equilibrium exists in this game. The proof is now completed. □

Here, we provide a distributed algorithm (Algorithm 3) which computes the NE solution to the OP_{MINER}. When updating his request vector, miner i always applies a standard Lagrange multipliers optimization solution based on his own OP_{MINER}.

4.5.2 Resource Limitation

Edge Computing is praised for its short delay while also being criticized for its limited resource capacity. In reality, it is possible that the ESP cannot fulfill all

requests from miners.

Generalized Nash Equilibrium

In the perspective of game theory, we can model this game as a generalized Nash equilibrium problem (GNEP). GNEPs differ from classical Nash equilibrium problems (NEP) in that, while in an NEP only the players' objective functions depend on the other players' strategies, in a GNEP both the objective functions and the strategy sets depend on the other players' strategies. In our case, the ESP only has a total of Y_{max} resource units, where Y_{max} is a common knowledge in this game. It has to reject some requests when overloaded. Thus, the aggregate requests from all miners should be no more than Y_{max} in order to avoid being rejected. Thus, given other miners' requests $\boldsymbol{r}_{-i}$, miner i should ensure that y_i can be satisfied by the ESP. Mathematically, this can be written as $\sum_{j=1}^{n} y_j \leq Y_{max}$.

Now, we reformulate the OP_{MINER} problem in the following.

Problem 1b ($\text{GNEP}_{\text{MINER}}$).

$$\begin{aligned} &\text{maximize} && U_i = R_i \cdot P_i - C_i, && \text{(4.5-4a)} \\ &\text{subject to} && \sum\nolimits_{j=1}^{n} y_j \leq Y_{max}, && \text{(4.5-4b)} \\ & && C_i \leq B_i, \quad x_i \geq 0, \quad y_i \geq 0. && \text{(4.5-4c)} \end{aligned}$$

Constraint (5.4-24b) ensures that miner i's request to the ESP can be fully satisfied. Since all miners' requests are mutually dependent, the $\text{GNEP}_{\text{MINER}}$ problem is a Generalized Nash Equilibrium Problem (GNEP). In $\text{GNEP}_{\text{MINER}}$, the dependence of each miner's strategy set on the other miners' strategies is represented by the (linear) constraint (5.4-24b), which includes each miner's request y_i to the ESP. More specifically, since the miners all share a jointly convex shared constraint, this game is known as a jointly convex game.

Algorithm 3 Best-Response Algorithm

Output: $\boldsymbol{r} = \{r_1, \cdots, r_n\}$ where $r_i = (x_i, y_i), \quad i \in \{1, n\}$
Input: Choose any feasible starting point $\boldsymbol{r}^{(0)}$: each miner chooses the decision using the local computing

1: **for** round k **do**
2: **for** miner i **do**
3: Decide $r_i^{(k)} = r_i^{(k-1)} + \Delta \frac{\partial U_i\left(r_i, r_{-i}^{(k-1)}\right)}{\partial r_i}$
4: Send the request $r_i^{(k)}$ to SPs
5: SPs collect the request profile $\boldsymbol{r}^{(k)}$
6: **if** $\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)}$ **then** Stop

Theorem 5. Given a price set (p_e, p_c) from the SP side, there exist at least one Nash equilibrium for the non-cooperative game at miner side given that the ESP's resource capacity Y_{max} is a common knowledge to all miners.

Similar with the proof for OP_{MINER} NE in Theorem 1, the existence of NE in $\text{GNEP}_{\text{MINER}}$ is easily followed by capitalizing on the variational inequality theory. In general, a GNEP could have infinite solutions. Namely, there are multiple NEs among miners, and thus there is no efficient algorithm to obtain the global optimal strategy in the proposed game. Algorithm 3 is still feasible here if each miner i updates his request vector using a standard Lagrange multipliers optimization solution based on his own $\text{GNEP}_{\text{MINER}}$. Note that, Algorithm 3 promises to compute a solution while there is no guarantee that the produced NE is a global optima.

Auction-based Partial Fulfillment

The application of GNEP has two deficiencies in reality. First, it usually has infinite solutions and no guarantee on global optimization, leading to an unpredictable equilibrium. Second, its convergence speed is a big concern as well. Another real-world problem is that Y_{max} may not be revealed to all miners and usually it may vary in each mining round, given the fact that the ESP also serves users outside the mining network.

Since all the resources/services are requested before a mining game starts, we can consider an alternative solution: a miner-side auction. Auctions help allocate and price scarce resources in settings of uncertainty. In this situation, each miner i simultaneously reports his bid on the amount y_i and the unit price p_e^i to the ESP. Then, the ESP applies a VCG mechanism to allocate resources to miners with certain charges, based on the value of Y_{max} and miners' bids at that time. This will result in a case that miner i's request on y_i is partially satisfied, which fits well with the reality.

4.6 Joint Optimization of Cache Size and Content

Previously, we simply characterized the access probability of each UTXO in a given mining round to be identical. However, lots of recent works [67, 41] on the Bitcoin UTXO set reflects a fact that the lifespan of a UTXO, the period from the time when it becomes spendable to the time when it is confirmed to be spent by its owner, varies. This observation indicates that, UTXOs with different birth times should have different access probabilities at a given time. Thus, our basic model, which assumes all UTXOs have an identical access probability no matter when they become spendable, seems a little bit rough and should be refined to be more in line with reality. Section 4.7 will discuss how the access probability of an unspent transaction output changes as time goes. In this section, we assume the relationship is given.

Given the fact that UTXOs may have different probabilities of being accessed in a specific mining round, miner i who plans to invest on edge resources cannot randomly pick from the UTXO set for caching any more. He should not only consider the cache size, but also the cache content, *i.e.*, which UTXOs should be selected in the requested cache space. Thereby, miner i faces a joint optimization problem where the cache size and the cache content have to be decided simultaneously. In the below, we focus on

this more realistic setting where contents to be selected and cached have different accessing probabilities.

4.6.1 Single Mining Rounds as a One-shot Game

Intuitively, miner i always tends to cache contents with high access probabilities in the hope of improving his cache hit rate and hence shortening his delay and avoiding extra costs to the CSP. Thus, in a given round, all UTXOs, S in total, are sorted based on their access probability a_k in the descending order. Define z_{ik} as a decision variable, indicating whether miner i decides to cache the k-th UTXO in the edge. That is, z_{ik} equals to either 1 if the k-th UXTO is selected by miner i, or 0 otherwise. Obviously, his requested cache size can be expressed as $y_i = \sum_{k=1}^{S} z_{ik}$, and the corresponding cache hit rate also can be rewritten as $h_i = \sum_{k=1}^{S} z_{ik}a_k / \sum_{k=1}^{S} a_k$. Now, miner i's strategy space is extended into three dimensions: (1) block size x_i, (2) cache size y_i, and (3) cache content z_{ik}, $\forall k \in [1, S]$. And his utility becomes a function over variables x_i and z_{ik}, $\forall k \in [1, S]$. We reformulate the optimization problem as follows.

Problem 1c (OP_{MINER}).

$$\text{maximize} \qquad U_i = R_i \cdot P_i - C_i, \tag{4.6-5a}$$

$$\text{subject to} \qquad x_i \geq 0, \quad z_{ik} \in \{0, 1\}, \forall k \in [1, S]. \tag{4.6-5b}$$

where $y_i = \sum_{k=1}^{S} z_{ik}$, and $h_i = \sum_{k=1}^{S} z_{ik}a_k / \sum_{k=1}^{S} a_k$.

Corollary 1. Nash equilibrium still exists even if each unspent transaction output has non-uniform access probability in a given mining round.

The uniform-access-probability setting is a special case where all a_ks are identical. Similar to the proof for NE in Theorem 1, the existence of NE for miners in a non-uniform-access-probability setting is followed by capitalizing on the variational inequality theory. Based on the previous analysis, we need to show that U_i in Prob-

lem 1c is a concave function over variables x_i and z_{ik}, $\forall k \in [1, S]$. According to the proof in Theorem 1, we can obtain the fact that U_i is a concave function over variables x_i and y_i. For a given mining round, all a_ks are constants, so y_i is an affine function over z_{ik}, $\forall k \in [1, S]$. Therefore, the composite function U_i is still concave over x_i and z_{ik}, $\forall k \in [1, S]$.

4.6.2 Multiple Mining Rounds as a One-shot Game

Now, miner i figures out how to dedicate his cache storage in order to maximize his cache hit rate. His cache replacement policy is still to delete transaction outputs spent in the previous mining round, and refill with new unspent transaction outputs with high access probabilities in the next mining round. In fact, the update of cached contents may not be very frequent (*e.g.* on the order of hours) so as to reduce overload cost and complexity. Another issue is that, usually users take advantage of edge resources in a pre-ordered way instead of a preemptive way, as a preemption process incurs too much uncertainty. A non-preemptible usage of edge resources requires a user to report what and how many resources he wants, as well as how long he will occupy the requested resources. These two facts lead us to think about a more realistic scenario, *i.e.*, miners dedicate their cache storage (both sizes and contents) in a relatively longer-term view, *i.e.*, the minimum time period requested by the ESP for providing non-preemptible services.

Assuming that the minimum time period contains T mining rounds, then each miner's strategy is made to maximize his utility in the following T mining rounds (our previous analysis can be viewed as a special case of $T = 1$). When these T mining rounds end, miner i resubmits his requests to the SPs and updates his cache in the edge. Now, we move to a new scenario where every T mining rounds are viewed as a one-shot game. In such a one-shot game with T mining rounds, we assume all miners start mining at the same time point, *i.e.*, mining round 1. Before round 1 begins,

the cached contents should be reasonably updated so that they can be repeatedly accessed in a long timescale, *i.e.*, from round 1 to round T. Based on our previous analysis, the probability of accessing unspent transaction output k varies as time goes. We now denote $a_k(t)$ to represent the probability of accessing unspent transaction output k at mining round t in this game. Thereby, the expected cache hit rate for miner i at mining round t can be formulated as $h_i(t) = \sum_{k=1}^{S} z_{ik} a_k(t) / \sum_{k=1}^{S} a_k(t)$. The expected delay and cost at mining round t for miner i are also updated as $d_i(t) = d_e x_i + d_c x_i (1 - h_i(t))$ and $C_i(t) = p_e y_i + p_c x_i (1 - h_i(t))$, respectively. Now, the OP_{MINER} problem in this scenario can be reformulated as in Eq. (4.6-6), of which the utility function U_i is a sum over T mining rounds.

Problem 1d ($\text{OP}_{\text{MULTIROUND}}$).

$$\text{maximize} \qquad U_i = \sum\nolimits_{t=1}^{T} R_i \cdot P_i(t) - \sum\nolimits_{t=1}^{T} C_i(t) \tag{4.6-6a}$$

$$\text{subject to} \qquad P_i(t) = m_i \left(1 - \beta \sum\nolimits_{j=1}^{n} m_j d_j(t)\right) \tag{4.6-6b}$$

$$x_i \geq 0, \quad z_{ik} \in \{0, 1\}, \forall k \in [1, S]. \tag{4.6-6c}$$

Corollary 2. $\text{OP}_{\text{MULTIROUND}}$ can converge to some point(s) where each miner will keep a certain strategy given the fact that all miners simultaneously update their cache contents every T rounds.

The objective function presented in Eq. (4.6-6) is a concave function, since it is a sum of T concave functions, as we have proved in Corollary 2. This concavity guarantees the existence of the Nash equilibrium of its corresponding game.

4.7 Modeling Access Probability Dynamics

This section seeks to determine how likely a certain unspent transaction output will be accessed in a given mining round. We resort to an educated approximation.

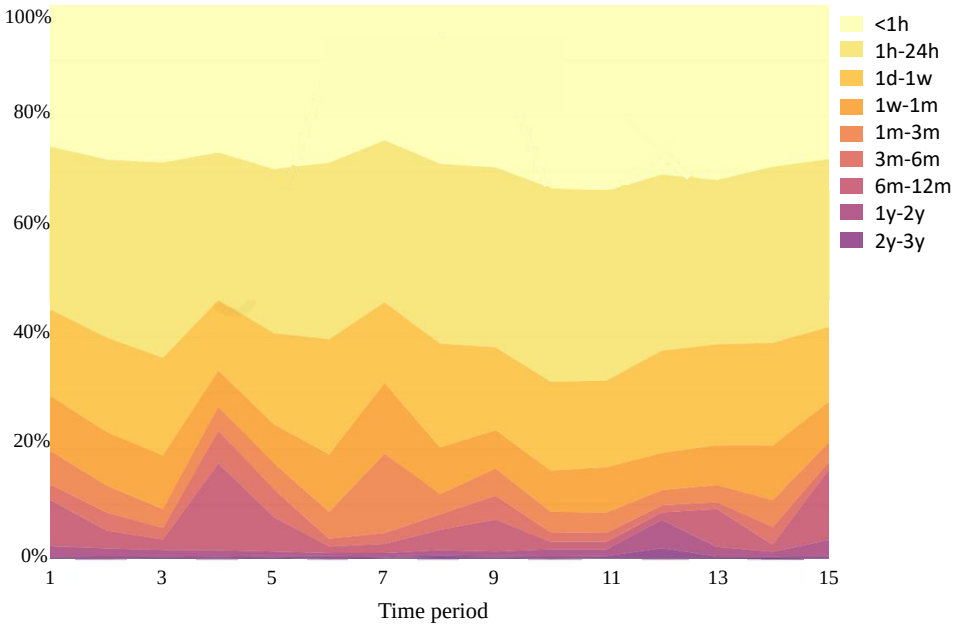


Figure 4.2: Daily spent transaction output lifespan within 15 days.

Our objective is to find an access probability function $a_k(t, t_b)$ to model how the chance of spending an unspent transaction output k evolves as time t goes, by taking its unique birth time t_b into account. (Note that, both t and t_b represent a certain mining round rather than an exact time point.) Accurately predicting over time the possibility that an unspent transaction output is spent by its owner is out of the scope of this work, since more factors are involved, *e.g.* the amount of this output, its owner's activeness, and even the cybercurrency's market price. We consider proposing a general model to capture the most cases. Thus, we also ignore some special outputs generated by transactions (*i.e.*, coinbase transactions in Bitcoin) that reward block creators, as those outputs by default have a longer waiting time before they become spendable.

4.7.1 Data Collection and Analysis

Our data is collected from Glassnode Studio [16] and Blockchain.info [6]. First, we conduct the following measurement starting from April 1st, 2020. We obtain all

Time period	Median lifespan (day)	Average lifespan (day)
April 01 - April 07	1.60	41.60
April 08 - April 14	1.44	37.33
April 15 - April 21	1.90	50.23
April 22 - April 28	1.36	34.60
April 29 - May 05	1.29	31.72

Table 4.2: Comparison of median lifespan and average lifespan every 7 days.

spent transaction outputs and their corresponding lifespans, *i.e.*, the duration from the time an output becomes spendable to the time it is confirmed to be spent by its owner, and then present the daily spent transaction output lifespan bands from April 1st to April 15th in Fig. 4.2. It is obvious that most transaction outputs are spent within 24 hours, but there exist some transactions that stayed in the system for years before they were spent. We further analyze both the median lifespan and the average lifespan of those collected spent transaction outputs. We take 7 days as a period and list the median lifespans and the average lifespans of each period from April 1st to May 5th in Table 4.2. (Note that, outputs with a lifespan of more than 3 months are discarded.) The result indicates that the median lifespan is much shorter than the average lifespan. This observation guides us to fit $a_k(t, t_b)$ with either an exponential distribution or a lognormal distribution.

4.7.2 Parameter Fitting and Model Validation

To decide the shape (exponential or lognormal) and corresponding parameter values of $a_k(t, t_b)$, we further collect the birth time and the redeeming time of 5000 spent transaction outputs, as our training dataset. These transaction outputs are divided into 4 traces, each containing 1250 outputs, based on their creation years from 2016 to 2019. We finally decide to use lognormal distribution for the access probability model. (In fact, we also tried exponential distribution as well as Gaussian distribution, and we found lognormal distribution fits best.) We apply a generalized

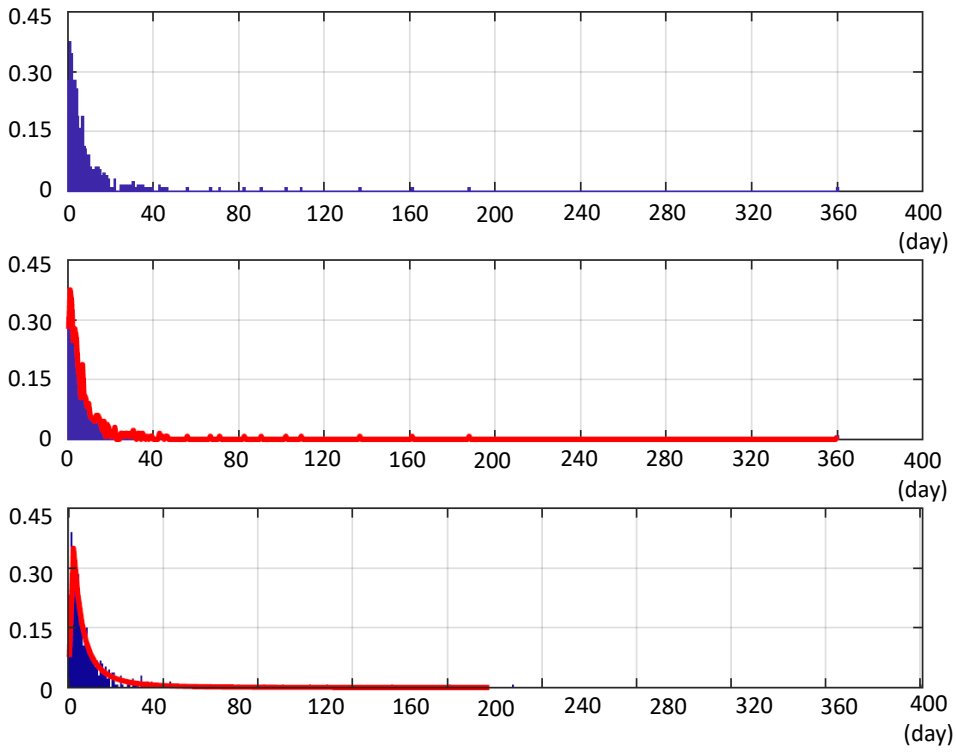


Figure 4.3: Access probability function fitting.

linear model with a log transformation on the access probability. In Fig. 4.3, we show the potential access probability as a function of time. The first and second pictures are the histogram of the original data and its corresponding frequency polygon, and the last picture is the fitting result. We calculate the squared correlation value and get an average of $R^2 = 0.91$. Now, we can conclude that a lognormal approximation is reasonable and the access probability of different unspent transaction outputs can be different in a given mining round. Therefore, it is necessary for miners to tactically select cache contents for utility optimization.

4.8 Evaluation

Our evaluation includes two main parts. First, we examine how miners decide their optimal strategies using our proposed algorithm, if we model each single mining round as a one-shot game (Subsection VII.A). We conduct our experiments based on

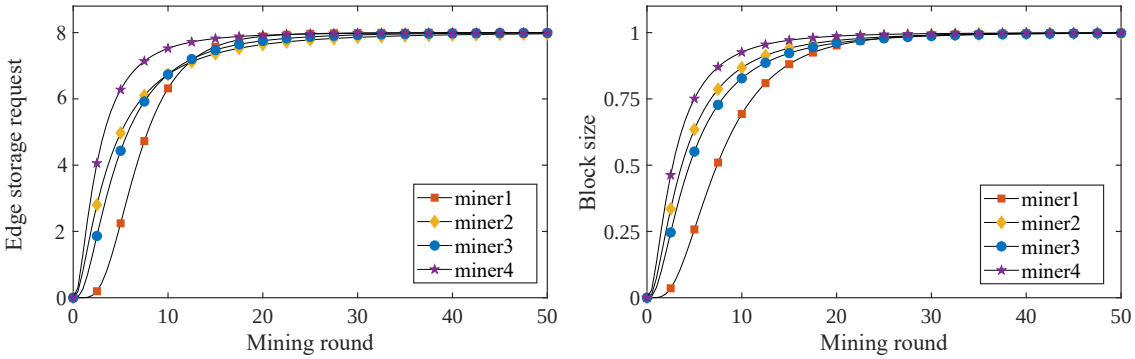
different sets of parameters to show how miners' decisions will be affected by external factors. Second, we take the network settings into consideration and analyze how the number of mining rounds in a one-shot game can influence the achieved equilibrium in our proposed game (Subsection VII.B).

4.8.1 Equilibrium in Games of a Single Mining Round

Our experiments evaluate the influence of important parameters on each miner's strategies. We start with a small mobile mining network with 4 homogeneous miners with unlimited budgets. We first consider the simplest case, where we just assume all unspent transaction outputs have the uniform access probability. Then we differentiate miners' mining power in order to make our simulation more realistic. Finally, we focus on how the non-uniform access probability will affect a miner's strategy.

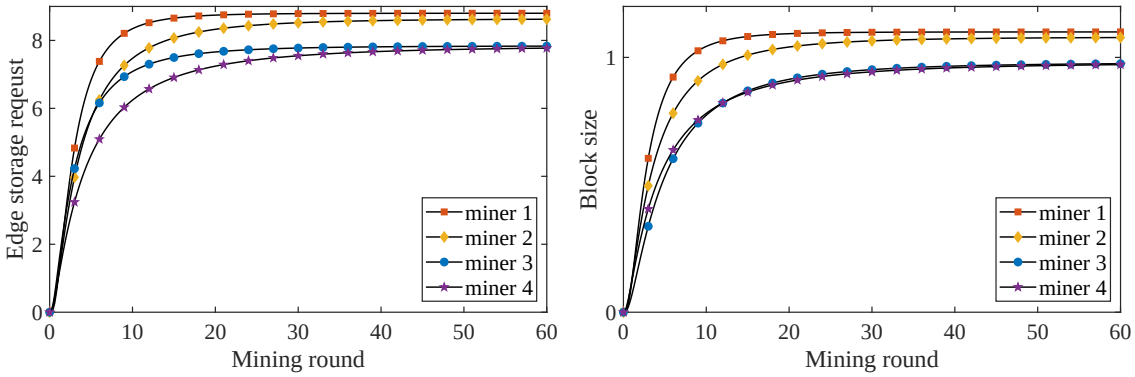
Uniform Access Probability

We analyze the results of 4-identical-miner game. Fig. 4.4 shows how miners' strategies evolve using our proposed algorithm 1. Since miners are identical, their optimal strategies converge to the same point. As we can observe in Fig. 4.4, the equilibrium is reached after 25 rounds, which is efficient. Next, we move to a heterogeneous miner setting and modify the mining power of each miner as $(m_1, m_2, m_3, m_4) = (0.18, 0.22, 0.3, 0.3)$. Based on the results shown in Fig. 4.5, we could see miner 3 and miner 4 share the same utility since they are identical in this experiment. Meanwhile, we could see miner 1 and miner 2 have a very close utility while the utility of miner 2 is always higher than that of miner 1. This is reasonable as we are discussing a budget-unlimited situation, where the mining power matters. This result is in line with the Bitcoin mining design principle.



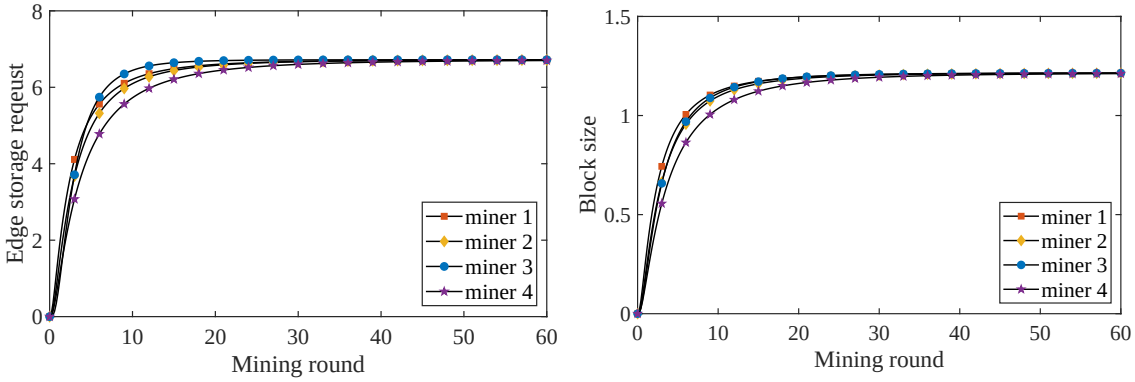
(a) Miners' cache requests (b) Miners' block sizes

Figure 4.4: 4 identical miners' single-round repeated mining game.



(a) Miners' cache requests (b) Miners' block sizes

Figure 4.5: 4 heterogeneous miners' single-round repeated mining game.

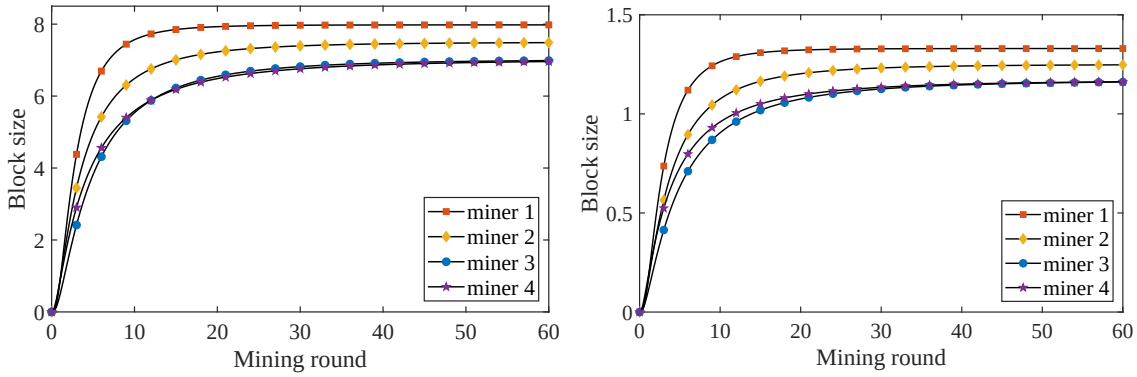


(a) Miners' cache requests (b) Miners' block sizes

Figure 4.6: 4 identical miners' single-round repeated mining game.

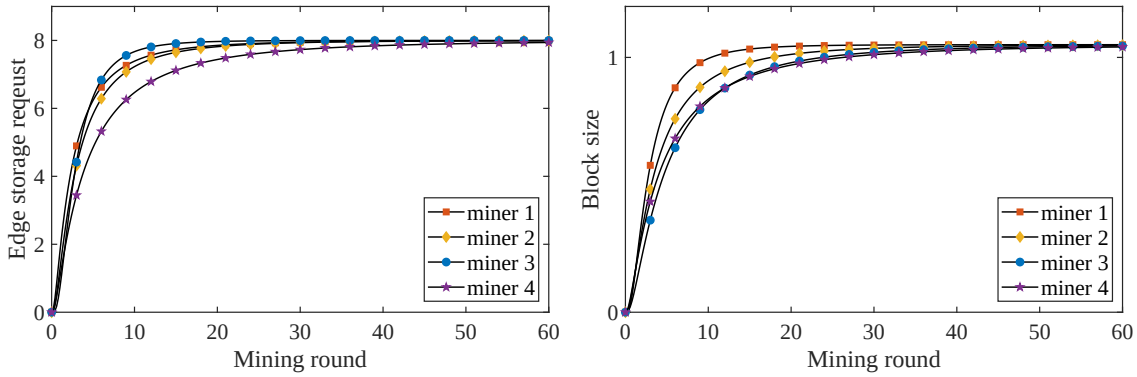
Non-uniform Access Probability

We now investigate how the non-uniform access probability will affect miners' strategies. We modify the 4-identical-miner setting by assigning different values to



(a) Miners' cache requests (b) Miners' block sizes

Figure 4.7: 4 heterogeneous miners' single-round repeated mining game.



(a) Miners' cache requests (b) Miners' block sizes

Figure 4.8: 4 identical miners' 3-round repeated mining game.

each unspent transaction output and the corresponding result is given in Fig. 4.6. When comparing with Fig. 4.4, we can see miners' strategies have changed. Each miner enlarged his block size while shrinking his cache size. It is reasonable as miners have more information on each output's access probability so that they can filter some outputs since caching them only brings a negative marginal utility. Fig. 4.7 shows the result of the 4-heterogeneous-miner setting with a similar observation.

4.8.2 Equilibrium in Games with Multiple Mining Rounds

We perform our experiment in the 4-identical-miner setting. We assume that T is fixed as 3. We compare each miner's strategy in Fig. 4.8 with that in Fig. 4.4. Since miners are identical, their strategies finally converge to the same point. Obviously,

the converge speed becomes slow compared with that in Fig. 4.4, as lowering update frequency decreases the chance for miners to adjust their own strategies. However, we find that in this setting, miners become more aggressive on purchasing resources from the ESP. This is reasonable as since some cached contents will become stale in the processing of the game, miners definitely will get benefit from storing more contents in advance if they don't have budget limitations. For the decision of block sizes, all miners' decisions are nearly the same as the decisions in a single-round game.

4.9 Related Work

Scalability Problem of Blockchain Size

Blockchain is an append-only ledger and should be fully replicated by all users in an untrustworthy environment. The exponentially increased blockchain size poses a challenge for its application in the IoT field. Many works focus on the blockchain size reduction. Pruning is a solution that has been implemented in BitcoinCore [15]. A node in its pruned mode only stores the UTXO set and several most recent blocks. [88] uses summary blocks to replace the actual blocks with storage compression. An improved memorization mechanism for the Bitcoin blockchain is proposed in [80]. Another idea is to split the entire blockchain into pieces so that each node only needs to store some of them [38, 108, 99]. We apply a traditional database outsourcing solution by considering both cloud and edge resources.

Game Theory in Offloading Mechanism

Game theory is a widely-used model in the field of offloading mechanisms. A large body of existing literature [105, 76, 100, 95, 111, 55, 96, 112, 72, 43, 104] focuses on minimizing offloading users' computation overhead in terms of energy and latency. To this end, researchers have developed distributed decision making methodologies.

In the field of mobile blockchain mining offloading [73, 106, 60], there are few works and most of them are in the PoW-mining scenario where mobile miners only offload their computation to a service provider. Our outsourcing scheme can be viewed as a combination of computation offloading and storage offloading.

Blockchain Balance Model

There are two popular models for recording each user's balance in today's blockchain networks. One is the unspent transaction output model, and the other is the account model. The UTXO model is applied by Bitcoin [82]. It can be abstracted as a directed graph of assets moving between users. The account model is used in Ethereum [102]. It is a database reflecting the asset distribution state in the current network. In our work, we focus on the UTXO model and characterize the unspent transaction output's access probability. Some researches have started to analyze the properties of the Bitcoin UTXO set [67, 41].

4.10 Conclusion

We consider storage outsourcing as a solution to deal with the storage shortage problem for miners using mobile devices, and then propose a non-cooperative game among miners to obtain optimal storage outsourcing strategies given the existence of both the CSP and the ESP. We analyze how each unspent transaction output's access probability evolves over time and model a single mining round and multiple mining rounds as a one-shot game, respectively. We prove the existence of Nash equilibrium and design a distributed algorithm to achieve NE point(s) for the proposed game. Both numerical evaluation and testbed experiment on Bitcoin are conducted to show the correctness of the proposed access probability pattern and to validate the proposed models and theoretical results. Through our evaluation, we see how different game settings and parameters affect miners' strategies and utilities.

CHAPTER 5

EXTENSION: MINING POWER MANIPULATION

Computation, *i.e.*, Mining Power, offloading has been considered as a viable solution to PoW blockchain mining in mobile environments. In this chapter, we first present a two-layer computation offloading paradigm that includes an edge computing service provider (ESP) and a cloud computing service provider (CSP). We formulate a multi-leader multi-follower Stackelberg game to address the computing resource management problem in such a network, by jointly maximizing the profits of each service provider (SP) and the payoffs of individual miners. Two practical scenarios are investigated: a fixed-miner-number scenario for permissioned blockchains and a dynamic-miner-number scenario for permissionless blockchains. For the fixed-miner-number scenario, we discuss two different edge operation modes, *i.e.*, the ESP is *connected* (to the CSP) or *standalone*, which form different miner subgames based on whether each miner's strategy set is mutually dependent. The existence and uniqueness of Stackelberg equilibrium (SE) in both modes are analyzed, according to which algorithms are proposed to achieve the corresponding SE(s). For the dynamic-miner-number scenario, we focus on the impact of population uncertainty and find that the uncertainty inflates the aggressiveness in the ESP resource purchasing.

Besides PoW consensus, in this chapter, we also take PoC consensus into consideration. We modify our previously-proposed model by updating miners' payoff function based on the specific feature of PoC mining. In this setting, mining power is redefined as capacity for energy saving. To focus on miner side, we only consider

a one-layer game by removing the participation of multiple service providers. That is, we model interactions among miners as a non-cooperative game and formulate a Nash equilibrium problem to investigate the effects of offloading on miners' utilities. We analyze the existence and uniqueness of equilibrium and propose a distributed algorithm to achieve the equilibrium in a uniform-delay setting. Further, we extend our results to non-uniform delays since miners may choose different network settings, *e.g.* 5G, 4G, or 3G. Both numerical evaluation and testbed experiments on Burstcoin are conducted to show the feasibility of storage offloading and to validate the proposed models and theoretical results.

5.1 Mining Power Offloading in PoW Mining Network

Most public blockchain networks today use processes referred to as Proof of Work (PoW) to provide consensus, however, the energy consumption and the computing power required to perform PoW computation are prohibitively high for mobile devices, thus hindering the practical usage of blockchain in mobile environments. Offloading PoW computation to the external machines has been proven effective in overcoming the aforementioned limitations and promoting mobile blockchain applications. Specifically, both an *edge computing service provider* (ESP) and a *cloud computing service provider* (CSP) can provide computing resources for mobile devices. While a CSP can guarantee a good isolation among multiple computation offloading requests (*i.e.*, there is no competition for cloud computing resources) with a relatively cheap price, significant network delays hamper the performance of cloud computing. Due to the delay-sensitive nature of mining, an ESP is considered as an efficient proximity alternative with the capability of providing low-latency service. However, mobile miners may have to compete against each other for the limited and expensive edge computing resources.

In this work, we present a hierarchical computation offloading paradigm consisting

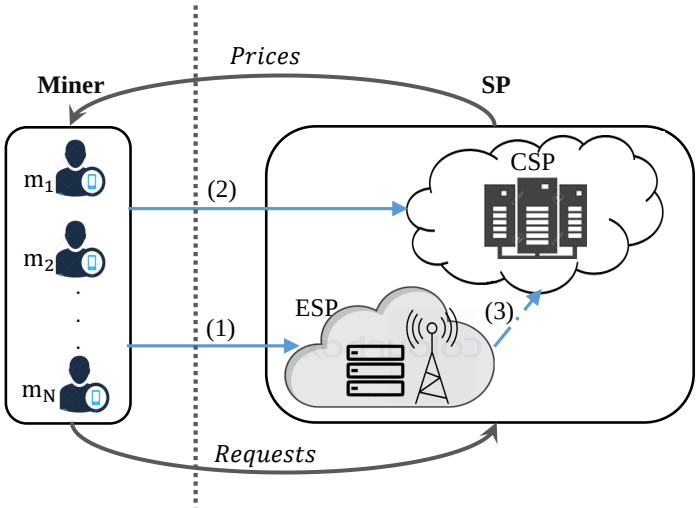


Figure 5.1: Mobile blockchain mining network: (1) offloading to the ESP; (2) offloading to the CSP; (3) automatic transfer from the ESP to the CSP.

of two service providers (SPs), *i.e.*, a nearby ESP and a remote CSP, and a set of miners in a mobile blockchain mining network. As depicted in Fig. 5.1, each miner is willing to offload its PoW computation to either of these two SPs or both of them. Once the ESP is overloaded with requests, it responds differently according to its operation mode. Specifically, two edge computing operation modes, *i.e.*, the ESP *connected* to the CSP and *standalone*, have been implemented in practice. Consequently, for an edge computing request which fails to be satisfied by the ESP, it will be sent to the backup CSP in the connected mode (characterized by the line(3) in Fig. 5.1), or will be rejected in the standalone mode. In the standalone mode, miners can resend those requests rejected by the ESP to the CSP. However, the communication delay will be considerably longer than that in the connected mode where the ESP executes automatic transfers. In the standalone mode, miners' requests are mutually affected and should be dedicated to avoid overloading the ESP.

We exploit game theory to analyze the complex interactions among SPs and mobile miners. To solve the price-based resource management problem, we leverage a multi-leader multi-follower Stackelberg game, which includes two subgames for the SPs (as leaders) and the miners (as followers), respectively. In the SP subgame, each SP has

a privilege to set unit prices on its computing resources by anticipating the miners' responses. In the miner subgame, the miners decide their requests according to the observed unit prices. Moreover, we investigate how edge operation modes will affect the miner subgame. In the connected mode, the miner subgame is formulated as a classical Nash equilibrium problem (NEP). However, the miner subgame becomes a generalized Nash equilibrium problem (GNEP) in the standalone mode. GNEPs differ from NEPs in that, while in an NEP only the players' objective functions depend on the other players' strategies, in a GNEP both the objective functions and the strategy sets depend on the other players' strategies. In the standalone mode, due to the limited computing units at the ESP side, whether a miner's edge computing request can be satisfied is affected by other miners' requests.

All previous studies assume that the miner number is fixed as a common knowledge in the proposed games. In practice, for permissionless blockchains where miners can randomly join or leave, the miner number may change. Thus, we also discuss the impact of population uncertainty on the miners' strategies by modeling the miner number as a random variable.

5.2 System Model and Game Formulation

5.2.1 A Mobile Blockchain Mining Network

This work focuses on a mobile blockchain mining network. Corresponding notations are listed in Table 5.1. We consider N end users, which we also call miners, and two service providers. Fig. 1 depicts an overview of this network. The SP side consists of a nearby ESP and a remote CSP that make profits by contributing their computing power, sold by unit. One unit from the ESP is functionally equivalent to one from the CSP. In the proposed network, message transmission time is viewed as communication delay. For simplicity, we assume communication delay between the

ESP and miners is negligible as 0, while communication delay between the CSP and the ESP/miners is the same as D_{avg}. Besides, the ESP is assumed to have limited computing capability, while the CSP owns unlimited computing power.

The end-user side is a network with N miners using different mobile devices. We differentiate them in terms of available budget which gives an upper bound on the amount of computing units they can afford. Thus, different types of miners have different requests on computing power. We employ a utility function to describe each miner's expected payoff, *i.e.*, the difference between its expected income and expected cost. The SPs and the miners have bidirectional communications for exchanging price and request information. Miners receive prices from the SPs and transmit their requests to them.

We consider two practical edge operation modes, *i.e.*, connected to the CSP or standalone, differing in whether or not the ESP would share loads with the CSP if it is computationally overloaded. Based on these two modes, we characterize the limited computing capability of the ESP in two different ways. In connected mode, the ESP's computing limitation is captured by an expected transfer rate, *i.e.*, $(1-h)$. That is, A request to the ESP may automatically be transferred to the CSP with a probability of $(1-h)$ in expectation. As an empirical value, $(1-h)$ is common knowledge in the game. Instead, if operating in standalone mode without load sharing, the ESP is limited with E_{max} computing units and hence rejects requests once overloaded.

5.2.2 SP-Miner Interaction: A Stackelberg Game

We focus on interactions between the SPs and the miners. Each miner's income depends all miners' strategies and its cost varies according to the prices set by each SP. In fact, each SP decides its unit price by considering miners' requests as well as the rival SP's price. Game theory provides a natural paradigm to model the interactions between the SPs and the miners in this network. Each SP sets the unit price and

Table 5.1: Summary of Notations.

Symbol	Description
P_e/P_c	unit price set by the ESP/CSP
C_e/C_c	unit cost of the ESP/CSP
$1-h$	the ESP's expected transfer rate in connected mode
E_{max}	total computing capacity of the standalone ESP
D_{avg}	average transmission delay with the CSP
N	total number of miners
m_i	the i-th miner
e_i/c_i	number of ESP/CSP units requested by m_i
r_i	m_i's request vector to the SPs, in the form of $[e_i, c_i]^\intercal$
$\boldsymbol{r}$	stacked request vectors of all miners
$\boldsymbol{r}_{-i}$	stacked request vectors of all miners excluding m_i's
R	blockchain mining reward
β	blockchain fork rate

announces it to the miners. The miners respond to the price by requesting an optimal amount of computing units to the SPs. Since the SPs act first and then the miners make their decision based on the prices, the two events are sequential. Thus, we model the interactions between the SPs and the miners using a Stackelberg game. In our proposed game, the SPs are the leaders and the consumers are the followers. It is a multi-leader multi-follower Stackelberg game, two stages of which can be described as follows.

In the first stage, the competition between the ESP and the CSP forms a non-cooperative leader subgame, where each SP optimizes its unit price (P_e/P_c) by predicting the miners' reactions as well as considering the other SP's price strategy. In the second stage, each miner, *i.e.*, m_i, responds to the current prices, by sending request(s) to the target SP(s), considering its budget B_i and requests of other miners'. Since requests are generated for individual utility maximization, a non-cooperative follower subgame is also formed.

Miner Side Utility

Let e_i and c_i be m_i's requests on the ESP and the CSP, respectively. Given the constant R as the mining reward, we define m_i's optimization problem below.

Problem 2 (MINER SUBGAME: OP_{MINER}).

$$\text{maximize} \qquad Ui = R \cdot W_i - (P_e \cdot e_i + P_c \cdot c_i), \tag{5.2-1a}$$

$$\text{subject to} \qquad P_e \cdot e_i + P_c \cdot c_i \leq B_i, \quad e_i \geq 0, \quad c_i \geq 0. \tag{5.2-1b}$$

where W_i represents m_i's expected winning probability, an accurate definition and detailed explanations of which will be given in Section 5.8. Each miner m_i aims to maximize its utility and constraint (5.2-1b) ensures that m_i is within its budget.

SP Side Utility

The objective of each SP is to optimize its profit by determining the corresponding unit price. The optimization problem (including OP_{ESP} and OP_{CSP}) at SP side is thus defined as in Eq.(5.2-2a) and Eq.(5.2-2b) for the ESP and the CSP, respectively.

Problem 3 (SP SUBGAME: OP_{SP}).

$$\text{maximize} \qquad V_e = (P_e - C_e) \cdot E \quad \text{where } E = \sum_{i=1}^{N} e_i \tag{5.2-2a}$$

$$\text{maximize} \qquad V_c = (P_c - C_c) \cdot C \quad \text{where } C = \sum_{i=1}^{N} c_i \tag{5.2-2b}$$

Stackelberg Game

OP_{SP} and OP_{MINER} together form the proposed Stackelberg game. To achieve equilibrium in this game, where neither the leaders (SPs) nor the followers (miners) have incentives to deviate, we need to find its subgame perfect Nash equilibria (NE) in both the leader stage and the follower stage, by applying backward induction. Formally, the SE point(s) is defined as follows.

Definition 1. Let $[E^*, C^*]$ and $[P_e^*, P_c^*]$ denote the optimal resource request vector of all miners and the optimal computing unit price vector of SPs, respectively. Let $[e_i^*, c_i^*]_{i=1}^N = [E^*, C^*]$, then the point (E^*, C^*, P_e^*, P_c^*) is the Stackelberg equilibrium if the following conditions hold:

$$V_e(P_e^*, E^*) \geq V_e(P_e, E^*), \forall P_e, \tag{5.2-3a}$$

$$V_c(P_c^*, C^*) \geq V_c(P_c, C^*), \forall P_c, \tag{5.2-3b}$$

$$U_i(e_i^*, c_i^*, P_e^*, P_c^*) \geq U_i(e_i, c_i, P_e^*, P_c^*), \forall i. \tag{5.2-3c}$$

5.2.3 Main Results

We summarize the main results of our analysis based on whether the miner number is a constant or a random variable. Scenario 1: the miner number N is a constant.
(1) The ESP operates in the connected mode:

- In the heterogeneous-miner case, we prove the existence and uniqueness of SE (Theorem 7) and provide a best response algorithm (Algorithm 4) to find the unique SE point.
- In the homogeneous-miner case, we derive explicit-form expressions of the optimal pricing for the SPs (Theorem 9) and resource management strategies (Theorem 8) for all the miners, given miners' identical budget.

(2) The ESP operates in the standalone mode:

- In the heterogeneous-miner case, the existence of the SE is validated by capitalizing on the variational inequality theory (Theorem 10). An effective distributed price bargaining algorithm (Algorithm 5) with guaranteed convergence is proposed to find one SE point.
- In the sufficient-budget case, we symbolically express the optimal prices and

resource management strategies (Table 5.2) given a sufficiently large budget for the miners.

Scenario 2: the miner number N is a random variable.

- We reformulate the proposed game given that the miner number is subject to a Gaussian distribution.
- Numerical analysis shows that, population uncertainty renders miners more aggressive to buy resources from the ESP, causing the profit to increase at the ESP side.
- A reinforcement learning framework is applied to further confirm the numerical results.

5.3 A Miner's Winning Probability

5.3.1 Parameter Analysis

As the core part of each miner m_i's utility, W_i is determined by multiple parameters. To win mining rewards, m_i has to be the first to solve its PoW puzzle and propagate its block to reach consensus. The chance for m_i to find a PoW solution is positively correlated to its relative computing power, which is the ratio of m_i's computing power out of all computing power in the network. A mined block will be broadcast among all miners and its propagation time may vary due to the underlying factors like network topology and block size. During its propagation time, blockchain forks and this mined block is to be discarded if another conflicting block is found and reaches consensus faster. Generally, W_i diminishes if the corresponding block takes more propagation time. The relation between the blockchain fork rate and the propagation time has been studied in Bitcoin [81], a classic PoW-based blockchain application. Fig. 4.1(a) provides its block collision probability density function (PDF)

with respect to communication delay, which is subject to an exponential distribution. Thereby, the split rate, *i.e.*, the block collision cumulative distribution function (CDF), is almost linearly proportional to communication delay, as shown in Fig. 4.1(b).

In this work, we assume that the proposed network follows the same pattern of collision PDF and CDF as in Bitcoin. We denote the blockchain fork rate by β, a function of the block propagation time. In the conventional rig mining, propagation time only includes the block broadcast time within the network. However, a mobile blockchain mining network also views communication delay between the SPs (the CSP in this work) and miners as part of the block propagation time. For simplicity, we consider that the time taken to broadcast a mined block among all miners is identical to 0. Hence, communication delay with the CSP D_{avg} is the only possible source of the propagation time in the proposed network, which can lead to a blockchain fork rate equal to β_{avg}. We use β to represent β_{avg} for simplified writing. Thus, W_i mainly depends on m_i's *relative computing power* and the blockchain fork rate caused by *transmission delay*.

5.3.2 Expression of Individual Winning Probability

In this part, we will derive an expression of W_i under the assumption that each miner m_i's request, denoted by a vector $r_i = [e_i, c_i]^\intercal$, is fully satisfied at the SP side. Let $\boldsymbol{r} \triangleq \{r_1, r_2, \cdots, r_N\}$ and $\boldsymbol{r}_{-i}$ represent the request profile of all miners and all other miners except m_i, respectively. We denote E in Eq. (5.2-2a) and C in Eq. (5.2-2b) as the total number of computing units requested on the ESP and the CSP, respectively. $S = E + C$ therefore represents the total requested computing units in the network. The winning probability, in the form of $W_i = W_i^e + W_i^c$, consists of two parts, W_i^e and W_i^c, jointly contributed by the ESP and the CSP, where W_i^e and

W_i^c are functions of r_i and $\boldsymbol{r}_{-i}$ given below:

$$W_i^e(r_i, \boldsymbol{r}_{-i}) = e_i/S + e_i \sum\nolimits_{j \neq i} \beta c_j/ES, \tag{5.3-4}$$

$$W_i^c(r_i, \boldsymbol{r}_{-i}) = c_i/S - c_i \sum\nolimits_{j \neq i} \beta e_j/ES. \tag{5.3-5}$$

For a better understanding, we begin with the analysis on W_i^c.

• W_i^c: c_i/S represents the expected chance that m_i mines a cloud-solved block B. Now we discuss the probability that B is discarded before it reaches consensus. With a chance of β, a conflicting block B' would be found during the propagation time D_{avg}. A cloud-solved B' has the same propagation time D_{avg} and thus cannot beat B. However, B will be discarded if B' is found in the edge and hence reaches consensus immediately. W_i^c in Eq. (5.3-5) characterizes the fact that, the probability of a successful cloud mining is discounted by the chance that the mined block is discarded due to any conflicting edge-solved block. Here, e_j/E approximates the rate that B' is mined in the edge by another miner m_k. We don't consider the situation, where B' is an edge-solved block for m_i itself, as a discount factor, since m_i still wins.

• W_i^e: m_i's winning probability of edge mining is the addition of (i) the chance that m_i is the first to successfully mine a block using its edge computing power, expressed as e_i/S and (ii) a summed chance that m_i's edge-solved block surpasses a cloud-solved block mined by any other miner m_k. The corresponding expression is shown in Eq. (5.3-4).

We verify the validity of W_i as a probability mass function.

Theorem 6. $W_i = W_i^e + W_i^c$ is a valid probability mass function to express the winning probability of individual miners in a mobile blockchain mining network.

Proof. We present the full verification process by checking that $\sum_{i=1}^{N} W_i = 1$ holds.

$$\sum\nolimits_{i=1}^{N} W_i = \sum\nolimits_{i=1}^{N} (W_i^e + W_i^c)$$

$$
\begin{aligned}
&= \sum_{i=1}^{N} [(e_i+c_i)/S+\beta\cdot(e_iC+c_iE)/ES] \\
&= \sum_{i=1}^{N} e_i \left[1+\beta(C-c_i)/E\right]/S+c_i \left[1-\beta(E-e_i)/E\right]/S \\
&= \sum_{i=1}^{N} [e_i/S+c_i/S] \\
&\quad + \beta \sum_{i=1}^{N} [e_i(C-c_i)/ES + c_i(E-e_i)/ES] \\
&= 1+\beta \sum_{i=1}^{N} (e_iC-c_iE)/ES = 1. \qquad \square
\end{aligned}
$$

Thus, we are now ready to conclude that, the winning probability we use is valid, hence our model as well. Note that m_i's winning probability and hence its utility depends not only on its own request but also on the other miners'.

5.3.3 User Requests and SP Responses

All above analysis is based on the assumption that m_i's request r_i is responded to by the ESP and the CSP as what it expects, *i.e.*, if r_i is fully satisfied by the ESP and the CSP as its original form $[e_i, c_i]^\intercal$, the individual winning probability on this condition is denoted by W_i^h and shown in Eq. (5.3-6)

$$
W_i^h = (e_i + c_i)/S + \beta(e_iC - c_iE)/ES. \tag{5.3-6}
$$

However, this assumption cannot always hold when we take the ESP's computing capability into consideration. It is possible that overall requests from the miner side are beyond the ESP's computing capability. Thus, we further refine the individual winning probability based on whether e_i can be satisfied by the ESP or not. Now we discuss how r_i will be responded to if e_i is beyond the ESP's capability, in connected mode and in standalone mode, respectively. We denote the corresponding winning probability by W_i^{1-h}.

Failure in connected mode

In this case, e_i would be transferred from the ESP to the CSP, and therefore, r_i is degraded as $[0, e_i + c_i]^\intercal$. The total computing power in the network stays unchanged as S, while $E - e_i$ and $C + e_i$ represent the actual resource allocation by the ESP and the CSP, respectively. Eq. (5.3-7) gives the winning probability.

$$W_i^{1-h} = (1 - \beta)(e_i + c_i)/S. \tag{5.3-7}$$

Failure in standalone mode

Since e_i would be rejected by the ESP, r_i is degraded as $[0, c_i]^\intercal$. Thus, the total computing power of edge computing and that in the network are reduced to $E - e_i$ and $S - e_i$, respectively. Eq. (5.3-8) gives the corresponding winning probability.

$$W_i^{1-h} = (1 - \beta)c_i/(S - e_i). \tag{5.3-8}$$

5.4 Fixed Miner Number Scenario

In the fixed miner number scenario, we assume that the network contains a fixed set of miners. That is, the number of miners is modeled as a constant, *i.e.*, $N \triangleq n$. We consider two edge computing operation modes: connected and standalone. We apply backward induction to analyze the subgame perfect NE in each stage for both modes. In the connected mode, the uniqueness of the SE is validated by identifying the best response strategies of the miners. In the standalone mode, the existence of the SE is proved by capitalizing on the variational inequality theory. Then, we get the closed-form price and resource allocation solutions to a special homogeneous-miner case for both modes. Besides, we compare the profits at the SP side and the miner side in these two modes.

5.4.1 Connected Mode

In this mode, the ESP's limited computing capability is characterized by the ESP's expected transfer rate $(1-h)$.

Miner Subgame Equilibrium

Consequently, m_i should consider two possible results: (i) with a probability of h, its request on the ESP is satisfied; (ii) with a probability of $(1-h)$, its request on the ESP is automatically transferred to the CSP with a degraded service. Thus, W_i can reflect these two results by applying the law of total expectation as shown in Eq. (5.4-9)

$$\begin{aligned} W_i &= h \cdot W_i^h + (1-h) \cdot W_i^{1-h} \\ &= h \cdot [(e_i + c_i)/S + \beta \cdot (e_i C - c_i E)/ES] \\ &\quad + (1-h) \cdot (1-\beta)(e_i + c_i)/S \\ &= (1-\beta)(e_i + c_i)/S + \beta h e_i / E, \end{aligned} \tag{5.4-9}$$

then the OP_{MINER} problem can be concreted as below.

Problem 1a (MINER SUBGAME: $\text{NEP}_{\text{MINER}}$).

$$\text{maximize} \qquad U_i = R \cdot W_i - (P_e \cdot e_i + P_c \cdot c_i)\,, \tag{5.4-10a}$$

$$\text{subject to} \qquad P_e \cdot e_i + P_c \cdot c_i \leq B_i, \quad e_i \geq 0, \quad c_i \geq 0, \tag{5.4-10b}$$

where $W_i = (1-\beta)(e_i + c_i)/S + \beta h e_i / E$.

Thus, the existence and uniqueness of an NE of this miner subgame is given by the following theorem.

Theorem 7. A unique Nash equilibrium exists in $\text{NEP}_{\text{MINER}}$.

Proof. Denote the equivalent variational inequality (VI) problem $\mathcal{VI}(\mathcal{K}, \text{F}) \equiv \mathcal{NEP}(\mathcal{X}, U)$,

where

$$\begin{aligned}
&\mathrm{F} := (\nabla_i U_i)_{i=1}^n, \quad \mathcal{X} = ([e_i, c_i]^\intercal)_{i=1}^n, \quad \mathcal{U} = (U_i)_{i=1}^n, \\
&\mathcal{K} := \mathcal{K}_1 \times \mathcal{K}_2 \times \cdots \times \mathcal{K}_n, \\
&\mathcal{K}_i := \{(e_i, c_i) | P_e \cdot e_i + P_c \cdot c_i \leq B_i, e_i \geq 0, c_i \geq 0\}.
\end{aligned} \tag{5.4-11}$$

Obviously, (i) $\mathcal{K}_i$ is closed and convex, $\forall i$ and (ii) U_i is continuously differentiable and convex w.r.t. $[e_i, c_i]^\intercal$, $\forall i$, then the VI problem has a non-empty solution set. The existence of NE thus follows the sufficient conditions. Further details and the proof of its uniqueness can be found in Appendix. □

As a rational player, each miner optimizes its utility by solving the $\mathrm{NEP}_{\mathrm{MINER}}$ problem as follows. Using Lagrange's multipliers λ_1, λ_2, and λ_3 for the constraints in Eq. (1e), the optimization problem is thus converted to the form

$$\begin{aligned}
L_i = R \cdot [(1-\beta)(e_i + c_i)/S + \beta h e_i / E] - (P_e \cdot e_i + P_c \cdot c_i) \\
- \lambda_1 (P_e \cdot e_i + P_c \cdot c_i - B_i) + \lambda_2 e_i + \lambda_3 c_i,
\end{aligned} \tag{5.4-12}$$

and the complementary slackness conditions are

$$\begin{aligned}
&\lambda_1 (P_e \cdot e_i + P_c \cdot c_i - B_i) = 0, \\
&\lambda_2 e_i = 0, \quad \lambda_3 c_i = 0, \quad \lambda_1 > 0, \lambda_2, \lambda_3, e_i, c_i \geq 0.
\end{aligned} \tag{5.4-13}$$

By the first-order optimality condition $\nabla L_i = 0$, it immediately follows that $\lambda_2 = \lambda_3 = 0$. Thus

$$\begin{aligned}
e_i &= \sqrt{\frac{h\beta E_{-i} R}{(1+\lambda_1)(P_e - P_c)}} - E_{-i}, \\
c_i &= \sqrt{\frac{R(1-\beta)(E_{-i}+C_{-i})}{(1+\lambda_1)P_c}} - \sqrt{\frac{h\beta E_{-i} R}{(1+\lambda_1)(P_e - P_c)}} - C_{-i}, \\
B_i &= P_e e_i + P_c c_i, \text{where } E_{-i} := \sum\nolimits_{j \neq i} e_j, C_{-i} := \sum\nolimits_{j \neq i} c_j.
\end{aligned} \tag{5.4-14}$$

Solving Eq. (5.4-14) yields that

$$1+\lambda_1 = \left[\frac{(P_e - P_c)\sigma_1\sqrt{E_{-i}} + P_c\sigma_2\sqrt{E_{-i}+C_{-i}}}{B_i + P_c C_{-i} + P_e E_{-i}}\right]^2, \tag{5.4-15}$$

where: $\sigma_1^2 = h\beta R/(P_e - P_c)$ and $\sigma_2^2 = (1-\beta)R/P_c$. Hence substituting Eq. (5.4-15) back into Eq. (5.4-14) gives the explicit form of the solution to the $\text{NEP}_{\text{MINER}}$ problem, *i.e.*, each miner's best response strategy. This naturally gives a distributed iterative algorithm, allowing each miner to iteratively update its strategy, given the strategies of other miners. When this unique subgame NE is ensured, the algorithm's convergence is also guaranteed. The uniqueness of NE in $\text{NEP}_{\text{MINER}}$ is guaranteed given that F defined in Eq. (5.4-11) is strictly monotone.

SP Subgame Equilibrium

With the knowledge of the miners' strategies, each SP makes its decision by solving the NEP_{SP} defined below.

Problem 2a (SP SUBGAME: NEP_{SP}).

$$\text{maximize} \qquad V_e = (P_e - C_e)\cdot E \text{ where } E = \sum\nolimits_{i=1}^{n} e_i, \tag{5.4-16a}$$

$$\text{maximize} \qquad V_c = (P_c - C_c)\cdot C \text{ where } C = \sum\nolimits_{i=1}^{n} c_i. \tag{5.4-16b}$$

Stackelberg Equilibrium in Connected Mode

We take advantage of a distributed algorithm called Asynchronous Best-response, as is shown in Algorithm 4, to find the unique NE point in OP_{SP} defined in Problem 3. The solution's uniqueness further guarantees the global convergence and SE is achieved, given that NE is found in the leader stage.

Algorithm 4 Asynchronous Best-Response Algorithm

Output: $k, k \in \{e, c\}$
Input: Choose any feasible starting point P_k
1: **for** iteration i **do**
2: Receive the optimal requests of miners
3: Predict the strategy of the other SP
4: Decide $P_k^{(i)}$
5: **if** $P_k^{(i)} == P_k^{(i-1)}$ **then** Stop
6: **else** send P_i to miners

5.4.2 Homogeneous Miners with Identical Budgets

The solutions to the $\text{NEP}_{\text{MINER}}$ are infeasible to express in a symbolic manner. Fortunately, we are able to get the closed-form computation offloading solutions for the $\text{NEP}_{\text{MINER}}$ in a special case. We consider a homogeneous-miner case where each miner is homogeneous with an identical budget B. We are interested in finding an NE where miners decide on a mixed request, buying computing units from both the ESP and the CSP. Thus, constraint (5.4-10b) is modified as $e_i > 0$, $c_i > 0$. The corresponding miner side optimization problem can be rewritten as the $\text{NEP}_{\text{HOMOMINER}}$ problem in the following.

Problem 1b (MINER SUBGAME: $\text{NEP}_{\text{HOMOMINER}}$).

$$\text{maximize} \qquad U_i = R \cdot W_i - (P_e \cdot e_i + P_c \cdot c_i)\,, \tag{5.4-17a}$$

$$\text{subject to} \qquad P_e \cdot e_i + P_c \cdot c_i \leq B, \quad e_i > 0, \quad c_i > 0, \tag{5.4-17b}$$

where $W_i = (e_i + c_i)/S + \beta \cdot (e_i C - c_i E)/(ES)$.

We will provide the explicit-form expression or the pricing strategy for the homogeneous-miner case defined above in Problem (1d).

Theorem 8. The unique Nash equilibrium for miner m_i in the $\text{NEP}_{\text{HOMOMINER}}$ problem

is given below

$$\begin{cases} e_i^* = \dfrac{B\beta h}{(1-\beta+h\beta)(P_e-P_c)}, \\ c_i^* = \dfrac{B\left[(1-\beta)(P_e-P_c)-P_c\beta h\right]}{P_c(1-\beta+h\beta)(P_e-P_c)}, \end{cases} \tag{5.4-18}$$

provided that the prices set by the ESP and the CSP satisfy $P_c < \dfrac{1-\beta}{1-\beta+h\beta}P_e$.

Proof. According to (13), we have $E^2 = \sigma_1^2 \sum_{j\neq i} e_j/(1+\lambda)$ and $S^2 = \sigma_2^2 \sum_{j\neq i}(e_j+c_j)/(1+\lambda)$ for each miner m_i, which will yield $E^2/S^2 = \sigma_1^2(E-e_i)/[\sigma_2^2(S-e_i-c_i)]$. Then, we calculate the summation of this expression for all the miners: $E/S = \sigma_1^2/\sigma_2^2 = [h\beta/(1-\beta)]\cdot P_c/(P_e-P_c)$. In order to get a mixed strategy, $E/S > 1$ must hold, *i.e.*, Eq.(5.4-27) holds. Since all miners are homogeneous, their best response strategies are identical as well, *i.e.*, $E = ne_i$ and $S = n(e_i+c_i)$. By substituting these two equations into Eq. (5.4-15), we obtain the NE for miner m_i in Eq.(5.4-18). □

Corollary 3. If the budget B is sufficiently large, the explicit solution to the $\text{NEP}_{\text{HOMOMINER}}$ problem is shown in Eq.(5.4-30)

$$\begin{cases} e_i^* = \dfrac{\beta h R(n-1)}{n^2(P_e-P_c)}, \\ c_i^* = \dfrac{R(n-1)\left[(1-\beta)P_e-P_c\right]}{n^2P_c(P_e-P_c)}. \end{cases} \tag{5.4-19}$$

Now, we start to analyze the SP optimization problem, which can be rewritten as follows.

Problem 2b (SP SUBGAME: $\text{NEP}_{\text{SPHOMOMINER}}$).

$$\text{maximize} \qquad V_e = (P_e - C_e)\cdot e_i^*, \;\; V_c = (P_c - C_c)\cdot c_i^*, \tag{5.4-20a}$$

$$\text{subject to} \qquad P_c < \frac{1-\beta}{1-(1-h)\beta}P_e, \tag{5.4-20b}$$

where $e_i^* = \dfrac{B\beta h}{(1-\beta+h\beta)(P_e-P_c)}$, $c_i^* = \dfrac{B\left[(1-\beta)(P_e-P_c)-P_c\beta h\right]}{P_c(1-\beta+h\beta)(P_e-P_c)}$.

Theorem 9. The unique Nash equilibrium for the SPs in the $\text{NEP}_{\text{SPHOMOMINER}}$ problem is given below:

$$\begin{cases} P_e^* = \bar{p}, \\ P_c^* = \dfrac{C_c\bar{p}(1-\beta) - \bar{p}\sqrt{C_c h\beta(\bar{p}-C_c)(1-\beta)}}{[1-\beta(1-h)]C_c - \beta h P_e}. \end{cases} \tag{5.4-21}$$

Proof. We start with the optimal P_c^* by analyzing the convexity of V_c. We calculate the first derivative of V_c and find that it is a concave function given the constraint (34). Thus, its optimal P_c^* can be achieved by solving $\partial V_c/\partial P_2 = 0$ where $P_c < P_e 1-\beta/[1-(1-h)\beta]$ and P_c^* is shown in Eq. (9), as is a function dependent on P_e set by the ESP. Given the response strategy of the CSP, the ESP can optimize his payoff by maximizing the re-written V_e.

$$V_e = \frac{nB\beta h}{(1-\beta+h\beta)(P_e - P_c^*)} \cdot (P_e - C_e) \tag{5.4-22}$$

We calculate the second derivative of V_e and find that $\partial^2 V_e/\partial P_e^2 \leq 0$ holds for any valid P_e value.

Thus, the ESP has his dominant strategy P_e^* (which exists, but is hard to be expressed symbolically) and the CSP has his best response strategy P_c^* (which is shown in Eq. (9)). In this situation, NE is achieved in the leader stage. We analyze P_e^* and P_c^* and find that they only depend on their own operating costs C_e, C_c, and the network delay penalty factor β. □

5.4.3 Standalone Mode

In standalone mode, the ESP only has a total of E_{max} computing units, where E_{max} is a common knowledge in this game. It has to reject some requests when

Algorithm 5 Price Bargaining

Input: Choose any feasible starting point P_e, P_c

for each miner i **do**
    Receive P_e, P_c
    Predict the optimal requests of other miners
    Decide its computing request $[e_i, c_i]^T$
    Send e_i to the ESP and send c_i to the CSP
for the ESP **do**
    Receive the optimal edge requests of miners
    Predict the strategy of the CSP
    Decide P_e and send P_e to miners
for the CSP **do**
    Receive the optimal cloud requests of miners
    Predict the strategy of the ESP
    Decide P_c and send P_c to miners

overloaded. Thus, the aggregate requests from all miners should be no more than E_{max} in order to avoid being rejected.

Subgame Equilibrium

In standalone mode, given other miners' requests $\boldsymbol{r}_{-i}$, m_i should ensure that e_i can be satisfied by the ESP. Mathematically, this can be written as $E = \sum_{k=1}^{n} e_k \leq E_{max}$. Under this constraint, its winning probability is expressed in Eq. (5.4-23).

$$W_i = (e_i + c_i)/S + \beta(e_i C - c_i E)/ES. \tag{5.4-23}$$

Now, we reformulate the OP_{MINER} problem in the following.

Problem 1c (MINER SUBGAME: $\text{GNEP}_{\text{MINER}}$).

$$\begin{aligned}
&\text{maximize} && U_i = R \cdot W_i - (P_e \cdot e_i + P_c \cdot c_i)\,, && \text{(5.4-24a)}\\
&\text{subject to} && E \leq E_{max}, && \text{(5.4-24b)}\\
& && P_e \cdot e_i + P_c \cdot c_i \leq B_i, \;\; e_i, c_i \geq 0, && \text{(5.4-24c)}
\end{aligned}$$

where $W_i = (e_i + c_i)/S + \beta \cdot (e_i C - c_i E)/ES$. Constraint (5.4-24b) ensures that m_i's request to the ESP can be satisfied.

Since all miners' requests are mutually dependent, the $\text{GNEP}_{\text{MINER}}$ problem is a Generalized Nash Equilibrium Problem (GNEP). In $\text{GNEP}_{\text{MINER}}$, the dependence of each miner's strategy set on the other miners' strategies is represented by the (linear) constraint (5.4-24b), which includes each miners' request e_i to the ESP. More specifically, since the miners all share a jointly convex shared constraint (JCSC), this subgame is known as a jointly convex game. $\text{GNEP}_{\text{MINER}}$ can be considered as a special case of $\text{NEP}_{\text{MINER}}$, where $h = 1$ and $(1 - h) = 0$ due to the given constraint (5.4-24b).

Existence of Stackelberg equilibria

Similar with the proof for $\text{NEP}_{\text{MINER}}$ NE in Theorem 7, the existence of $\text{GNEP}_{\text{MINER}}$ NE is easily followed by capitalizing on the variational inequality theory.

Theorem 10. Given a price set $\{P_e, P_c\}$ from the SP side, there exists at least one Nash equilibrium for the non-cooperative subgame at miner side in standalone mode.

In general, a GNEP could have infinite solutions. Namely, there are multiple NEs in the follower stage, and thus there is no efficient algorithm to obtain the global optimal pricing and computation offloading strategy for the proposed Stackelberg game. Here, we provide a distributed algorithm which first computes a unique variational solution to the $\text{GNEP}_{\text{MINER}}$ problem and then finds the corresponding solution to the SP SUBGAME: GNEP_{SP} problem (defined later) based on the computed miner Nash equilibrium. Note, there is no guarantee that the SE produced by Algorithm 5 is a global optima.

Problem 2c (SP SUBGAME: GNEP_{SP}).

$$\text{maximize} \qquad V_e = (P_e - C_e) \cdot E, \quad V_c = (P_c - C_c) \cdot C, \tag{5.4-25a}$$

$$\text{subject to} \qquad E = E_{max}. \tag{5.4-25b}$$

Homogeneous Miners with Sufficiently Large Budgets

In standalone mode, solutions to the $\text{GNEP}_{\text{MINER}}$ are infeasible to express in a symbolic manner. Fortunately, we are able to get the closed-form computation offloading solutions for the $\text{GNEP}_{\text{MINER}}$ in a special case. We consider a homogeneous-miner case where each miner is homogeneous with an identical budget B. We assume B is quite large so that each miner's cost under optimal request is affordable. Under this assumption, the constraint (5.4-24c) on budget $\text{GNEP}_{\text{MINER}}$ can be removed. We are interested in finding a Nash equilibrium where miners decide a mixed request, buying computing units from the ESP and the CSP. Thus, constraint (5.4-24c) is modified as $x_i > 0, \quad y_i > 0$. The corresponding miner side optimization problem can be rewritten as the $\text{GNEP}_{\text{HOMOMINER}}$ problem in the following.

Problem 1d (MINER SUB-GAME: $\text{GNEP}_{\text{HOMOMINER}}$).

$$\begin{aligned}
&\text{maximize} && U_i = R \cdot W_i - (P_e \cdot x_i + P_c \cdot y_i) && \text{(5.4-26a)}\\
&\text{where} && W_i = \frac{x_i + y_i}{S} + \beta \cdot \frac{x_i C - y_i E}{SE} &\\
&\text{subject to} && E \leq E_{max} && \text{(5.4-26b)}\\
& && x_i > 0, \quad y_i > 0 && \text{(5.4-26c)}
\end{aligned}$$

To achieve such an equilibrium in the follower level, the prices set by the ESP and the CSP matters. Then, the Eq. (5.4-27) gives the relation between P_e and P_c under which all miners will yield an equilibrium with mixed requests.

$$\begin{cases} P_c < (1-\beta)P_e \\ P_c < P_e - \dfrac{\beta R(n-1)}{nE_{max}} \end{cases} \tag{5.4-27}$$

Given P_e and P_c satisfying the Eq. (5.4-27), we compute the explicit expression of a miner's request in Nash equilibrium, as is shown in Eq. (5.4-28).

$$\begin{cases} x = \dfrac{\beta R(n-1)}{n^2(P_e - P_c)} \\ y = \dfrac{R(n-1)\left[(1-\beta)P_e - P_c\right]}{n^2 P_c(P_e - P_c)} \end{cases} \quad \text{where} \;\; n \cdot x \leq E_{max} \tag{5.4-28}$$

There is a special case where all computing units of the ESP are sold out, *i.e.*, $n \cdot x = E_{max}$, by setting dedicate P_e if the following holds:

$$\begin{cases} P_e - \dfrac{\beta R(n-1)}{E_{max} n} < P_c < \dfrac{R(n-1)(1-\beta)}{n E_{max}} \\ P_e < \dfrac{R(n-1)}{n E_{max}}. \end{cases} \tag{5.4-29}$$

Then the corresponding equilibrium request of each homogeneous miner is captured by Eq.(5.4-30)

$$\begin{cases} x = \dfrac{E_{max}}{n} \\ y = \dfrac{R(n-1)(1-\beta)}{n^2 P_c} - \dfrac{E_{max}}{n} \end{cases} \tag{5.4-30}$$

Given the NE point at miner side, utilities of the ESP and the CSP can be rewritten as follows:

$$\begin{aligned} V_e &= (P_e - C_e) \cdot E \\ &= n \cdot (P_e - C_e) \cdot x \\ &= \frac{R(n-1)\beta}{n} \cdot \frac{C_e - P_c}{P_e - P_c} \end{aligned} \tag{5.4-31}$$

$$\begin{aligned} V_c &= (P_c - C_c) \cdot C \\ &= n \cdot (P_c - C_c) \cdot y \\ &= \frac{R(n-1)\beta}{n} \cdot \frac{P_c - C_c}{P_c} \cdot \frac{P_e(1-\beta) - P_c}{(P_e - P_c)} \end{aligned} \tag{5.4-32}$$

Thus, the optimization problems for the ESP and the CSP are in the below.

Problem 2d (SP SUBGAME: $\text{GNEP}_{\text{SP}_{\text{HOMOMINER}}}$).

$$\text{maximize} \qquad V_e = (P_e - C_e) \cdot E, \quad V_c = (P_c - C_c) \cdot C, \tag{5.4-33a}$$

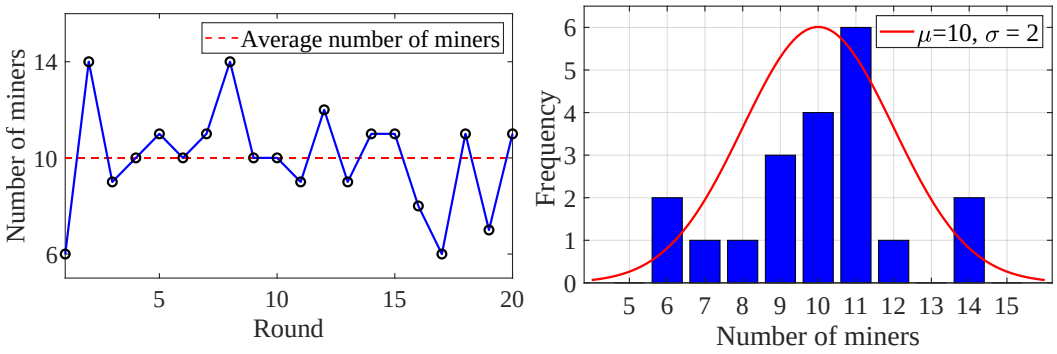
$$\text{subject to} \qquad E = E_{max}. \tag{5.4-33b}$$

The Nash equilibrium in the leader level can be captured by the following equation.

$$\begin{cases} P_e = \dfrac{\beta R(n-1)h}{n^2\left[(P_e - P_c)h + P'_e(1-h)\right]} \\ P_c = \dfrac{C_c P_e(1-\beta) - P_e\sqrt{C_c\beta(P_e - C_c)(1-\beta)}}{C_c - \beta P_e} \end{cases} \tag{5.4-34}$$

5.4.4 Comparison of Two Modes

We sum up the main results by comparing these two modes in a homogeneous-miner case. The explicit expressions of all miners' requests in equilibrium are summarized in Table 5.2, where $\gamma \triangleq \frac{(n-1)R}{n}$. As can be explicitly seen in Table 5.2, the amount of all miners' requests is identical in these two modes, given the same pricing of the CSP. Thus, the total requested computing units is only related to P_c. That is, pricing of the CSP decides the upper bound of the P2P network computing power. Since $h \leq 1$, the ESP would sell more units in standalone mode than in connected mode. Thus, connected mode maximizes the profits of the CSP and also lowers the cost at miner side, while standalone mode maximizes the ESP. The numerical results provided in Section 5.6 also show that the ESP's equilibrium prices in the standalone mode is higher compared to those in the connected mode. Thus, we conclude that the ESP in the standalone mode gains more profits.



(a) Statistics on the miner number among 20 mining rounds.

(b) Corresponding histogram and underlying distribution $N(\mu, \sigma^2)$.

Figure 5.2: A toy example for population dynamics of moblie miners.

Table 5.2: Optimal requests of homogeneous miners with sufficiently large budgets where $\gamma = (n-1)R/n$.

Mode	E^*	C^*	S^*
Connected	$\frac{\gamma\beta}{P_e - P_c}h$	$\gamma\left[\frac{(1-\beta)P_e - P_c}{P_c} + \beta(1-h)\right]$	$\frac{\gamma(1-\beta)}{P_c}$
Standalone	$\frac{\gamma\beta}{P_e - P_c}$	$\gamma\frac{(1-\beta)P_e - P_c}{P_c}$	$\frac{\gamma(1-\beta)}{P_c}$

5.5 Dynamic Miner Number Scenario

Obviously, in the above analysis, we assume the miner number N is common knowledge in the proposed games. In practice, this scenario is applicable to permissioned blockchains, where miners are pre-selected by a central authority or consortium. However, most blockchains are permissionless, in which anyone can participate in or retreat from the mining process, so the previous scenario may not be suitable. For such situations, we consider a more general scenario by introducing population uncertainty. Games with population uncertainty relax the assumption that the exact number of players is common knowledge. Thus, we model the miner number, N, as a random variable. In particular, N follows a Gaussian distribution with mean μ and variance σ^2. We have $N \sim \mathcal{N}(\mu, \sigma^2)$ where $N = k$ with probability $P(k) = \Phi(k) - \Phi(k-1)$. Fig. 5.2 gives a toy example where the number of miner

can be fit to a Gaussian distribution with $\mu = 10$ and $\sigma^2 = 4$.

In this scenario, we only consider standalone mode and derive the miner utility function U_i as below.

$$U_i(\mu, \sigma^2) = 0.5 \cdot U_i^h + 0.5 \cdot U_i^{1-h} \tag{5.5-35}$$

$$\begin{aligned}
&U_i^h = P_e \cdot e_i + P_c \cdot c_i - R \cdot W_i^h \\
&U_i^{1-h} = P_e \cdot e_i + P_c \cdot c_i - R \cdot W_i^{1-h} \\
&W_i^h = \sum_{k=l}^{u} P(k)\left[(e_i+c_i)/S_k + \beta(e_i C_k - c_i E_k)/(S_k E_k)\right] \\
&W_i^{1-h} = (1-\beta)(e_i+c_i)/S_\mu \\
&S_k = E_k + C_k, E_k = \sum_{j=1}^{k} e_j, \\
&C_n = \sum_{j=1}^{k} c_j, \forall k \in [l,\, u]
\end{aligned}$$

Thus, the OP_{MINER} problem in this scenario can be reformulated as in Eq. (5.5-36).

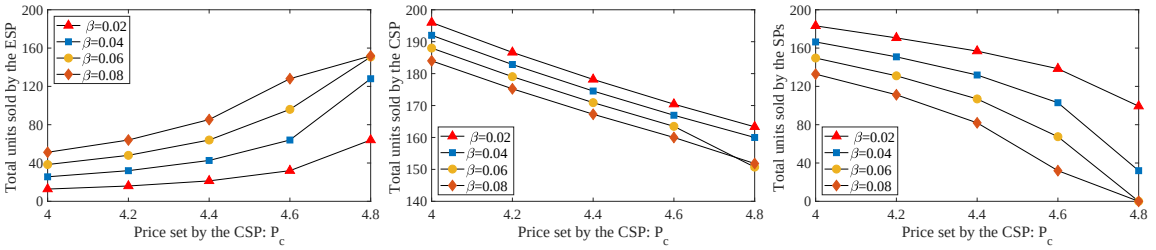
Problem 1e (MINER SUBGAME: $\text{OP}_{\text{DYNAMICMINER}}$).

$$\begin{aligned}
&\text{maximize} && U_i(\mu, \sigma^2) && && \text{(5.5-36a)} \\
&\text{subject to} && P_e \cdot e_i + P_c \cdot c_i \le B_i, \quad e_i \ge 0, \quad c_i \ge 0 && && \text{(5.5-36b)}
\end{aligned}$$

Problem 2e (SP SUBGAME: OP_{SP}).

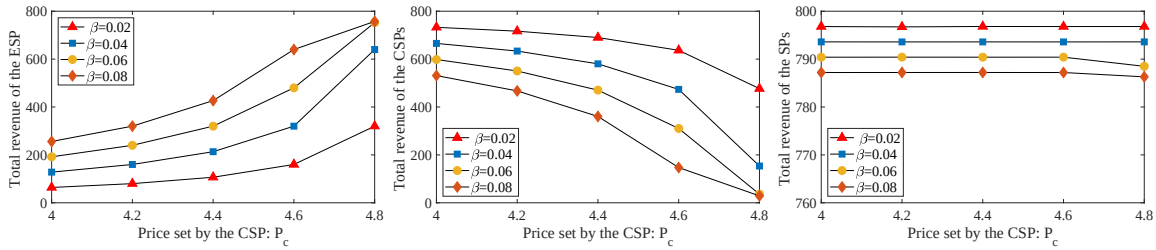
$$\text{maximize} \quad V_e = (P_e - C_e) \cdot E, \;\; V_c = (P_c - C_c) \cdot C \tag{5.5-37}$$

The objective function presented in Eq. (5.5-36) is so complex that it is infeasible to express its equilibrium expression in a symbolic manner. Therefore, we use numerical analysis to find equilibria in the network. As numerical results will later show in Section 5.6, we find that with an identical P_e, the uncertainty incurred by the dynamic population renders miners more aggressive to buy computing units from the ESP, even beyond its capability E_{max}.



(a) Miners' requests to ESP. (b) Miners' requests to CSP. (c) Miners' requests to SPs

Figure 5.3: Homogeneous miners with identical budgets and $P_e = 5$.



(a) The ESP's revenue. (b) The CSP's revenue. (c) Total revenue at SP side.

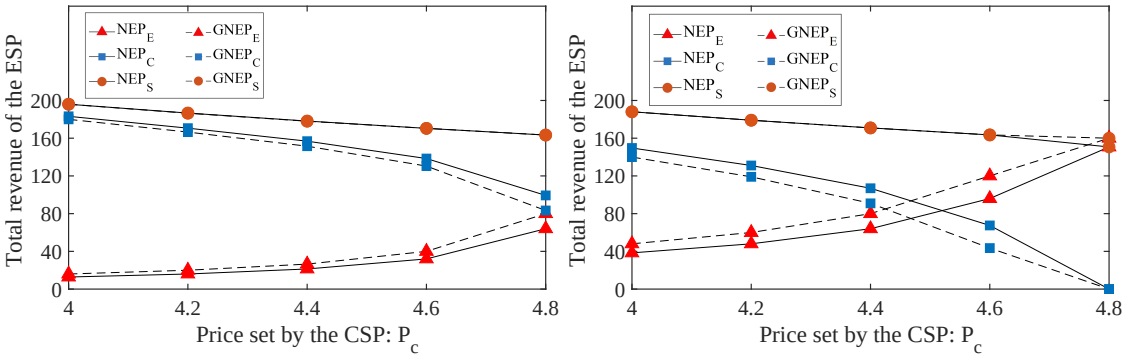
Figure 5.4: Homogeneous miners with identical budgets and $P_e = 5$.

5.6 Simulation

In this section, numerical examples are provided to examine how miners figure out their optimal requests based on the prices of the SPs and how the SPs optimize their unit prices based on their available power and the miners' budgets. We start with a small mobile blockchain mining network with only 5 miners with budgets $B_i,\ \forall i \in [1, 5]$.

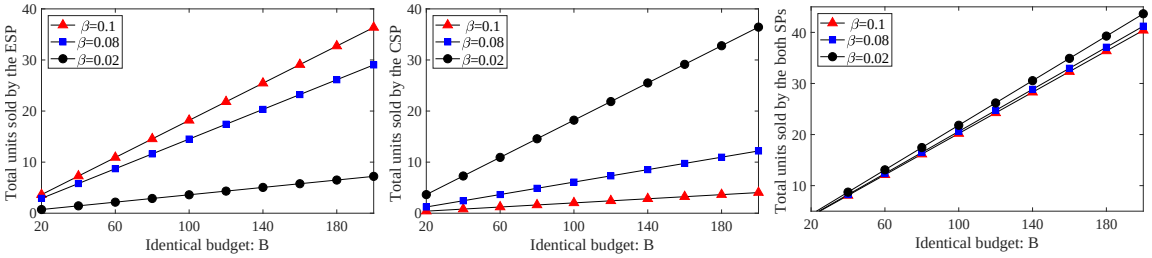
5.6.1 Miner Subgame Equilibrium

Our experiments evaluate how the corresponding miner subgame Nash equilibrium is influenced if the parameter values change.



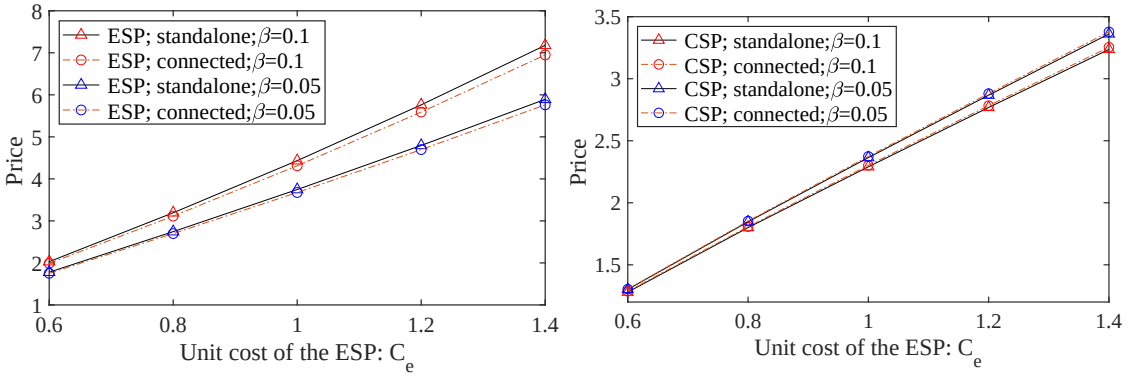
(a) $\beta = 0.02$. (b) $\beta = 0.06$.

Figure 5.5: Connected Vs Standalone.



(a) A miner's request to ESP. (b) A miner's request to CSP. (c) A miner's request to SPs.

Figure 5.6: m_i's budget B_1 varies from 20 to 200, with 5 miners in total.



(a) $\beta = 0.02$. (b) $\beta = 0.06$.

Figure 5.7: The CSP's unit cost is 0.5, while the ESP's unit cost changes.

Influences from SP side

We first consider the different prices at SP side. Assuming a homogeneous-miner case in the connected mode, where $B_i = 200$, $\forall i \in [1, 5]$ holds, Fig. 5.3 obviously reflects that, if the CSP's price P_c unilaterally increases, miners tend to buy more units

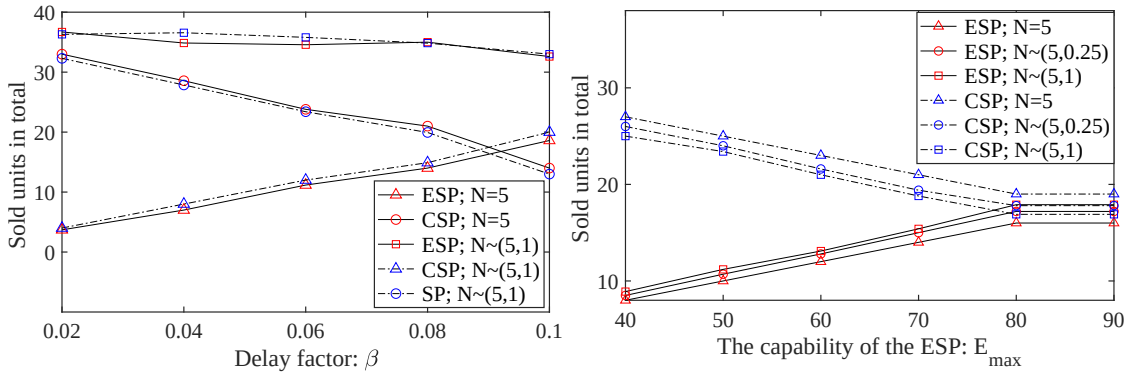
from the ESP, leading to more revenue at the ESP side. Similarly, from Fig. 5.3, we can also conclude that the blockchain fork rate β caused by the CSP's communication delay also has a negative effect on the number of total units sold by the CSP as well as his total revenue. However, from Fig. 5.4(c), we find the total revenue at the SP side remains almost unchanged no matter how prices and communication delay change.In the same miner configuration, we analyze the impact of edge operation modes. If the ESP operates in the standalone mode, we see its computing capability is positively related to miners' requests, which can be easily followed in Fig. 5.5. From this figure, we can conclude that, miners are discouraged from buying units from an ESP working in the connected mode. We see a cross in the Fig. 5.5. This explains the CSP's optimal prices under different communication delays. The longer the communication delay, the lower the optimal price.

Influences at miner side

Miners also mutually affect each other in this mining network. Fig. 5.6 shows the changes on all the miners' utilities when their budget of B_i varies from 20 to 200. m_i's requests to the ESP and the CSP keep increasing and its utility follows a similar trend. However, we can see that m_1's total requests to both SPs are similar even with different communication delays at the CSP side.

5.6.2 SP Subgame

We also study how communication delay and edge operation modes as well as the SP's operating costs affect their equilibrium prices. Fig. 5.7 depicts the equilibrium prices of the SPs. The ESP's prices increase linearly as its unit operating cost increases. In both modes, the ESP charges a higher price, because it has less power available and its communication delay is 0 in the proposed network. However, its advantage will be shaded if the communication delay at the CSP side decreases.



(a) $\beta = 0.02$. (b) $\beta = 0.06$.

Figure 5.8: Miner number: fixed vs dynamic.

Besides, the ESP's computation limitation also poses an upper bound on its profits. We also discover that the standalone mode allows the ESP a higher price while it decreases the CSP's price and its profits.

According to numerous experiments, we find that the total amounts of sold-out computing units are always approximately equal, if we allow a sufficiently large budget and a fixed number of miners. Besides, we can see that the SP-side welfare is bounded by the total miner budgets in the beginning. However, as the budgets increase to a certain degree, the total welfare of these two SPs are positively related to the blockchain mining reward.

5.6.3 Population Uncertainty

In Section 5.5, we consider the miner number as a variable subject to a specific Gaussian distribution. To capture the dynamics of the miner number, we use a reinforcement learning (RL) framework to allow miners to learn the population uncertainty and hence improve their strategies. We conduct our simulation within a small mining network of 5 homogeneous miners. We define a time period T as adding 50 blocks. During T, prices from these two SPs are fixed and the miner number changes subject to $N(\mu, \sigma^2)$. The reason why we choose $T = 50$ in our all experiments is that miners' strategies converge after at most 50 blocks added even in such an unstable-

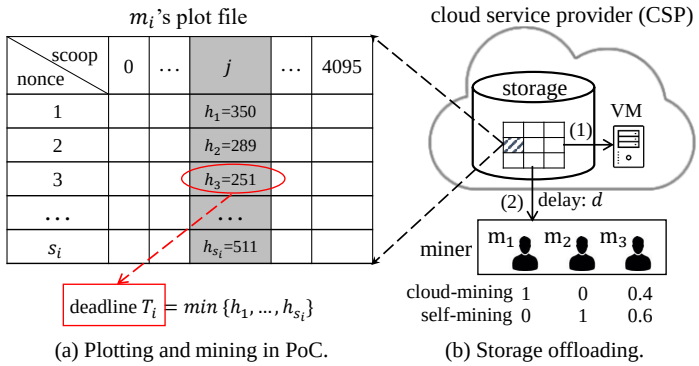


Figure 5.9: Miners offload plot files to a cloud storage and mine blocks (1) via cloud-mining using VMs and/or (2) via self-mining using mobile devices.

population mining network. Once the miners' behavior converges, both the ESP and the CSP update their pricing strategies adaptively. These two steps repeat until a fixed point for both sides is reached. We also apply such a process to a fixed number scenario where $N = \mu$.

In Fig. 5.8, all unfilled points are the results produced by the RL framework, while all lines are computed using our proposed model. Obviously, the results of our model are anastomotic with the learned strategies. In Fig. 5.8(a), we conclude that the uncertainty caused by the miner number renders each miner to buy more units from the ESP, making the total requests sometimes can exceed the ESP's capability. Besides, we also find the variance also affects a miner's request to the ESP, *i.e.*, a larger variance leads to a more ESP-prone miner, according to Fig. 5.8(b). Both Fig. 5.8(a) and (b) are generated based on the ESP's connected mode. We also conduct a check experiment in the standalone mode. The results in the dynamic scenario show that the miners are discouraged from buying units at the ESP side in the connected mode. These results are similar to those in the fixed scenario.

5.7 Mining Power Offloading in PoC Mining Network

Recently, a new mechanism called *Proof of Capacity* (PoC) [8] has emerged as a promising solution to the previously-mentioned problems of energy efficiency and fairness. As its name implies, PoC mining relies on idle storage capacity, *i.e.*, miners invest disk space, as opposed to computation in PoW mining, and the amount of space dedicated to mining determines the chances of winning a block. The mining process consists of two steps: one-time plotting and repeated mining in rounds. In the plotting step, miners configure their available storage with mining-related plot files. Fig. 5.9 (a) shows miner m_i's plot file, which is arranged into s_i units. Each unit is called a *nonce* and each nonce is divided into 4096 *scoop*s. In each mining round, m_i executes hashing over a scoop column (which is randomly selected by the system to avoid miner-side cheating) individually to get the smallest value among all cells of the scoop column, which is called m_i's deadline. A miner cannot publish a block until his deadline comes. The one that finds the smallest deadline among all miners wins. Since more storage space gives a higher chance of finding the smallest deadline, a miner's probability of winning a block is related to the ratio between his own space to the total space invested by all miners.

The PoC mechanism has been applied to several blockchain applications, *e.g.* Burstcoin [8], Chia [9], *etc*. However, its storage requirement poses a challenge on mobile devices, thus hindering its applications to mobile services. To facilitate PoC-based blockchain application in future mobile IoT systems, storage offloading appears to be a viable solution. Miners with mobile devices can overcome capacity limitations by offloading all plot files to an external cloud storage. Given that a small amount of computation is required in each PoC mining round, a miner can execute hashing in his device by downloading corresponding scoops (*self-mining*) and/or in the cloud by employing virtual machines (VMs) (*cloud-mining*) provided by the cloud service provider (CSP). Fig. 5.9 (b) gives an example where miners apply different mining

strategies. Self-mining requires no extra cost, but miners cannot start mining until a selected scoop column (one in 4096 rather than all of his plot file) is downloaded. The incurred download delay will reduce a miner's winning probability. Cloud-mining can avoid such a disadvantage, however, it also adds miners' cost on VM employment. Thus, a miner has to determine a suitable mining strategy, *i.e.*, the optimal ratio between self-mining and cloud-mining to maximize his utility, based on his storage offloading decision. In Fig. 5.9 (b), user m_3 deploys 40% for cloud-mining and the rest for self-mining.

From here, we study the interactions among multiple PoC mobile miners, who aim to maximize their own utilities. Tho whole model is quite similar to the follower-game proposed above, except that the winning probability should be modified based on PoC mining features. Note that, each miner maximizes his utility by deciding his strategies of **storage offloading and mining**, where in PoW mining, miners only decide on mining offloading. The proof of the existence and uniqueness of NE is skipped as it is similar to the previous model. The distributed algorithm we proposed in Algorithm 4 is also applicable to achieve the equilibrium here.

5.8 Miner's Winning Probability in PoC Mining

In this section, we start with a model for traditional PoC mining (subsection 5.8.1), where miners contributes their disk resources and mine in their own devices like desktop computers or laptops. This basic model allows us to quantify the relation between the block finding time and miners' storage size (subsection 5.8.2). We then analyze how the winning probability is influenced by the individual storage size and total storage size (subsection 5.8.3). We extend the basic model to include mobile mining and formulate how the download delay affects the winning probability (subsection 5.8.4). Finally, we derive the expression for miner's winning probability with download delays after combining all the related parameters (subsection 5.8.5).

5.8.1 Overview of PoC Mining

Generally, PoC mining consists of plotting and mining. The plotting process pre-generates and stores mining-related files on miners' storage. Fig. 5.9 (a) shows m_i's plot file, which is arranged into s_i fixed-length units (a row in Fig. 5.9 (a)). Each unit is called a nonce and is evenly divided into 4096 scoops (a cell in Fig. 5.9 (a)). In each mining round, m_i retrieves the jth scoop from each of his units (the grey column in Fig. 5.9 (a)), where j is selected by the system. m_i executes hashing over every retrieved scoop and gets a hash value as a storage proof. All hash values are within the range of $[0, D]$ and the smallest one is measured as m_i's best proof, named as a deadline (the circle in Fig. 5.9 (a)), which represents the waiting time before m_i is allowed to publish his block. Thus, the smallest deadline in the network will be measured as the network-wide best proof and its owner will win the block. In fact, D is a parameter controlling the mining difficulty for the miners. (Note, described above is a basic model, where a miner stores his plot file and self-mines using the same device, not involving storage offloading and cloud mining.)

5.8.2 Block Finding Time and Individual Storage Size

To find the winning probability of each player, we start by analyzing the block finding time probability distribution. We model the block finding time, *i.e.*, the network-wide deadline, as a random variable denoted T. This is a function related to all miners' selection of storage units.

Single miner's distribution functions

In a mining round, a miner can get a proof (*i.e.*, a hash value), denoted h, for each of his contributed storage units. In fact, h is a random variable, of which the value is subject to a uniform distribution within the range of $[0, D]$ in each round. Given that miner m_i commits s_i units of his storage in total, we model his deadline

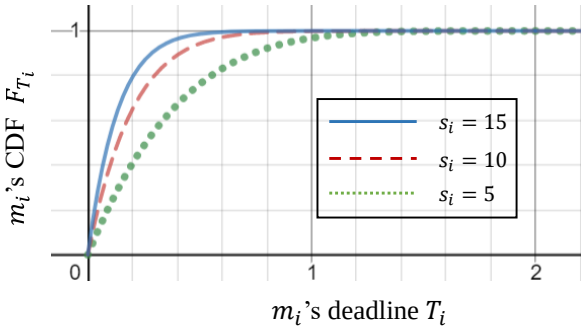


Figure 5.10: CDFs under different storage sizes given $D = 2$ min.

as a random variable denoted T_i. Obviously, $T_i = \min\{h_1, \cdots, h_{s_i}\}$. T_i's cumulative distribution function (CDF), denoted $F(t, s_i, D)$, can be obtained in the below.

$$F(t, s_i, D) = \begin{cases} 0 & t \leq 0, \\ 1 - (1 - t/D)^{s_i} & 0 < t < D, \\ 1 & t \geq D. \end{cases} \tag{5.8-38}$$

Thereupon, the corresponding probability density function (PDF), denoted $f(t, s_i, D)$, follows through performing derivative over $F(t, s_i, D)$, as is shown in Eq. (5.8-39).

$$f(t, s_i, D) = \begin{cases} \frac{s_i}{D}(1 - t/D)^{s_i - 1} & 0 < t < D, \\ 0 & otherwise. \end{cases} \tag{5.8-39}$$

Whole mining network's distribution functions

Given a mining network with n miners, the block finding time can be expressed as $T = \min\{T_1, \cdots, T_n\}$. We have already calculated the PDF and the CDF over T_i for $\forall i \in [1, n]$, thus the corresponding distribution functions $F_T(t, S, D)$ and $f_T(t, S, D)$ can be expressed using Eqs. (5.8-38-5.8-39), by replacing s_i with $S = \sum_j s_j$.

5.8.3 Influences of Total Storage Size

Now, we analyze how storage sizes affect P_i. Intuitively, a miner's winning probability should be positively related to his own storage size, since more storage units lead to a higher chance of smaller deadlines. Fig. (5.10) further confirms our guess, where T_i's CDF (as Eq. (5.8-38) describes) hits 1 faster under a larger s_i. Meanwhile, P_i should also be affected by the total mining storage space. To capture the exact mathematical relation between the individual winning probability and storage sizes (both individual and total), we start with a competition between only two miners m_i and m_j, owning the storage sizes of s_i and s_j, respectively. Obviously, m_i wins when he finds a smaller deadline. The probability that m_i wins is calculated as follows:

$$\begin{aligned} P_i = Pr[T_i < T_j] &= \iint f\left(t_i, s_i, D\right) f\left(t_j; s_j, D\right) \mathrm{d}t_i \mathrm{d}t_j \\ &= 1 - s_j/(s_i + s_j) = s_i/(s_i + s_j). \end{aligned}$$

Thus, the probability that m_i wins is proportional to his fraction of the total storage size. That is, given a total storage of size S, m_i's individual winning probability is $P_i = s_i/S$. Obviously, dedicating more storage space yields a proportionally higher expectation of successfully mining a block. Therefore, a PoC-based incentive mechanism can reward smaller miners fairly according to their contribution to the network, thus incurring more distributed participation.

5.8.4 Influences of Delay

Download delay in self-mining

The expression s_i/S characterizes the probability that m_i happens to hold the smallest deadline among all miners. However, it is possible that the owner of the smallest deadline is not the block winner. Suppose, in a certain mining round, a miner m_i's deadline is 100 seconds, the smallest one among all miners' deadlines, and

another miner m_j's deadline is 105 seconds, only next to m_i's deadline. If m_i finds his deadline within 100 seconds, then he propagates his block until that time comes and becomes the winner. However, if m_i's mining is delayed for some reason and hence he fails to finds his deadline within 105 seconds, at which time m_j succeeds in broadcasting his block, then m_j becomes the winner, although he is not the owner of the smallest deadline. Thus, with delay, a miner's winning probability is definitely discounted. In the traditional PoC mining, since plot file storing and mining happen in the same device, usually a desktop computer or a laptop, all deadlines can be calculated before the smallest one comes. Thus, it is just a race on miners' contributed storage. However, when applying storage offloading, a delay, denoted d, can be incurred in self-mining due to the scoop download from the cloud to a miner's device. Miners cannot start self-mining until the required scoops are downloaded. During the waiting time, if there is a deadline no greater than d calculated using VMs in the cloud, then the corresponding block can be successfully published and rewarded, although there may exist smaller deadlines not yet computed by self-mining. In reality, block propagation delay also damages a miner's winning probability. To focus on the influence of the download delay, we assume propagation delay among miners is negligible.

Download delay and winning probability

We now extend the basic model with the download delay d. We show how d discounts miner's winning probability. During the download delay d, if the speed of cloud mining is v, then there should be roughly vdn proofs computed in total. If the best one among them is less than d, then the corresponding block definitely wins whether or not it is a network-wide optimal deadline. The probability, denoted β, that a cloud-mined block wins before the self-mining starts can be expressed as $\beta\left(d, v\right) = 1 - \left(1 - d/D\right)^{vdn}$. We simplify our model by assuming cloud-mining can

perform deadline calculations fast, *i.e.*, all deadlines over total cloud-mining units X, *i.e.*, $\sum_{i=1}^{n} x_i$, can be calculated within d, then the corresponding probability β can be refined as: $\beta\left(d, X\right)|_{v\to+\infty} = 1 - \left(1 - \frac{d}{D}\right)^X$.

5.8.5 Expression of Winning Probability

We are now ready to express P_i in the model of storage offloading, *i.e.*, a miner stores his plot file in the cloud instead of his own device. P_i consists of two parts, P_i^c and P_i^s, jointly contributed by cloud-mining and self-mining, where P_i^c and P_i^s are functions of r_i and $\boldsymbol{r}_{-i}$ given below:

$$\begin{aligned} P_i^c(r_i, \boldsymbol{r}_{-i}) &= \frac{x_i}{S} + \frac{x_i}{X}\frac{Y}{S}\beta, \\ P_i^s(r_i, \boldsymbol{r}_{-i}) &= \frac{y_i}{S} - \frac{y_i}{Y}\frac{Y}{S}\beta = y_i\frac{1-\beta}{S}, \end{aligned} \tag{5.8-40}$$

where $Y = \sum_{i=1}^{n} y_i$ and $\beta = \beta(d, X)$ is for simplicity. Next, we verify the validity of P_i as a probability mass function.

Theorem 11. *$P_i = P_i^c + P_i^s$ is a valid probability mass function to express the winning probability of individual miners in a mobile blockchain mining network.*

Proof. We show the verification process by checking that $\sum_{i=1}^{n} P_i = 1$ holds, i.e.,

$$\begin{aligned} \sum\nolimits_{i=1}^{n} P_i &= \sum\nolimits_{i=1}^{n} (P_i^c + P_i^s) \\ &= \sum\nolimits_{i=1}^{n} (x_i + y_i)/S + Y\beta \cdot (x_i/X - y_i/Y)/S. \\ &= 1 + \frac{Y\beta_d}{S} \cdot (\frac{X}{X} - \frac{Y}{Y}) = 1. \end{aligned}$$

□

We can conclude that, the winning probability we use is valid, hence our model is as well. Note that m_i's winning probability and hence its utility depends not only on its request but also on those of the other miners.

5.9 Related Work

Mobile Blockchain Applications

There exist two different categories of research in the field of blockchain applications in wireless networks. The first category focuses on blockchain protocols [65, 92, 113] to eliminate overhead while maintaining most of blockchain's security and privacy. These research works are beneficial for secure and decentralized data communication in wireless networks. Instead of designing and implementing light-weight blockchain-based protocols, the second category [73, 106, 77, 61, 109] investigates pricing and resource management schemes for supporting blockchain applications in a mobile environment. The focus here is on the mining under the PoW consensus [81], which results in the competition among miners to receive a mining reward. Due to limited computing resources of their mobile terminals, miners offload the PoW computations to local edge servers [73, 106]. We also study the problem of pricing and computation offloading in mobile blockchain mining under the PoW consensus. However, we consider a more complicated assumption in which miners can perform a two-layer computation offloading to either/both of the ESP and the CSP.

Stackelberg Game in Offloading Mechanism

Stackelberg Game is a widely-used model in the field of offloading mechanisms. A large body of existing literature [105, 76, 100, 95, 111, 55, 96, 112, 22] focuses on minimizing offloading users' computation overhead in terms of energy and latency. To this end, researchers have developed distributed decision making methodologies. In the field of mobile blockchain mining offloading [73, 106, 107], there are few works and most of them are in the single-leader scenario where mobile miners only offload their computation to an SP, *e.g.* fog. In our work, we consider a multi-leader multi-follower Stackelberg game to jointly maximize the profit of the SPs and the individual

utilities of mobile miners. We assume a resource-limited edge layer working in either stand-alone or connected operation mode with the cloud layer.

Reinforcement Learning in Incomplete Information Game

Although analysis in game theory always assumes the observable strategies of other players [93, 91], it is more realistic that the miner's action is the private information which is unobservable by others. In addition to applying game-theoretical analysis on the proposed game, we also develop a reinforcement learning framework [85, 44, 29, 56, 97, 63] in our evaluation, allowing all players to select their best response strategies and update their beliefs about unobservable actions of others through repeated interactions with each other in a stochastic environment. This framework confirms our proposed model.

5.10 Conclusion

In this work, we have proposed a Stackelberg game between the SPs for optimal price strategies and among the mobile miners for optimal computation offloading requests. Two practical edge computing operation modes are investigated, *i.e.*, the ESP is connected to the CSP or standalone. First, we characterize the miner number as a constant in both modes. We discuss the existence and the uniqueness of Stackelberg equilibrium in the proposed games and provide algorithms to achieve SE point(s). Our analysis indicates that the connected mode discourages miners from buying computing resources from the ESP. Then, we study the impact of a dynamic miner number. Interestingly, we find that uncertainty incurred by the dynamic population renders miners more aggressive to buy computing resources from the ESP. Numerical experiments based on a reinforcement learning framework have been conducted to further confirm our analysis. We also extend our model by considering another consensus mechanism, *i.e.*, PoC, where the theoretical analysis is quite similar.

CHAPTER 6

APPLICATION: DATA MARKET

Lots of privacy and security issues in the current cloud-based data markets will be eliminated by taking advantage of blockchain-based decentralized storage services, which can provide a new paradigm for safe data outsourcing and correct remote search. However, existing data markets are also questioned on their inflexible and opaque pricing, where the value of data ownership and the cost of query search are mixed. Thus, a better pricing model is necessarily needed in an emerging decentralized data market. In this chapter, we envision an Ethereum-based data market, in which the pricing model for each query includes two parties: owner (paid for his data ownership) and miner (rewarded by query search). We study a new cooperative search scheme through a proxy to reduce cost on the client (user) side. Suppose each user query is charged based on the number of keywords in the query. The cost reduction is based on combining multiple queries into a group subject to the constraint that the resulting combined query is not significantly larger than any of its original queries in terms of the number of keywords. The total price is based on the total number of keywords in all groups. As the optimal grouping depends on the pricing of both owners and miners, we build a small testbed to analyze how price setting will affect grouping results. Since it is a cooperative model with shared resources, we also study various incentive properties on the client side, thereby yielding a cost sharing mechanism to split joint cost in a truth-revealing and fair manner. We further extend our market with a set of substitute data owners and propose a double auction mechanism to match buyers and owners based on their requirements. Experiments have

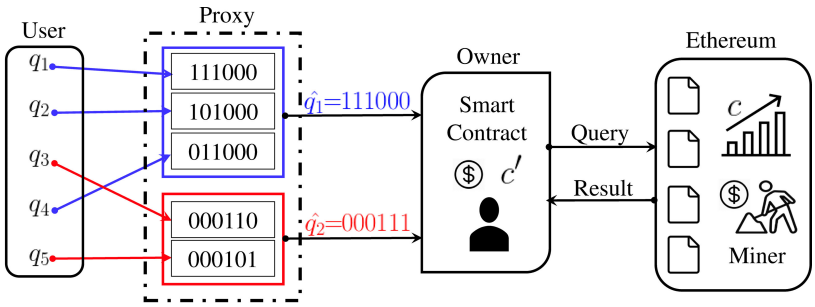


Figure 6.1: Given a database with six keywords, five queries q_1, q_2, q_3, q_4, q_5, each being a six-bit binary string with 1 for search and 0 for not search, are issued from users to retrieve information.

been conducted on real query trace to demonstrate the effectiveness of our proposed scheme.

6.1 Introduction

Data has become a tradeable good in our society nowadays. Most online data markets, *e.g.* Amazon Athena and Xignite, are cloud-based. Cloud offers a convenient single platform for trading data, and provides value-added services that help derive data products. However, centralized clouds also give rise to privacy and security issues in data storage and search. Thus, works on decentralized storage services have been proposed to alleviate these concerns. Blockchain techniques [31] are widely used to guarantee data integrity and provide scalability for handling big data. In terms of remote search, smart contract [57] is leveraged to protect the correctness and immutability of search results. Unfortunately, lots of existing works focus on the single-user setting, where a database is queried only through its owner. As in real data markets, a data owner makes profit by sharing his data with legitimate clients. Thus, a more complex multi-user setting should be discussed in the decentralized storage.

In this chapter, we envision an Ethereum-based data market that provides services related to data storage, search and trades. Since Ethereum [103] is a blockchain-based

decentralized computing platform [57] merged with smart contract, it guarantees reliable data storage and correct remote search. As is shown in Fig. 1, this new type of data market has three basic entities. As a data provider, an *owner* profitably shares his database by allowing third-party called *users* to query his database. Unlike any cloud-based data market, where private data is uploaded to the cloud and in the central control of the cloud provider, an Ethereum-based data market allows databases to be managed in a decentralized fashion. That is, an owner can either send his small-size information to the Ethereum blockchain [57] for the convenience of searching, or distribute his large data in an off-blockchain storage [115], *e.g.* IPFS [18], with a pointer to the data on the distributed ledger of blockchain. We assume all data is encrypted under a searchable symmetric encryption (SSE) scheme before being outsourced to keep it confidential while still allowing query-based searches. Users are data consumers and are willing to retrieve information from a designated database with some payment. We assume that users directly send queries to a corresponding owner, and the owner issues search transactions as if he is querying. As data searchers, *miners* in Ethereum make money by executing search functions in smart contracts.

Thus, query pricing in such a decentralized data market is unilaterally decided by both owners and miners. Each user query is charged for two parts: one for the data owner at a pre-set price based on data value, and the other for miners based on their workload, which is a pay-as-you-go mode. Thanks to Ethereum's gas system, a miner's computation consumption is traceable and transparent in Ethereum. Besides, a query's computation consumption and the corresponding search delay, as well as its cost, tend to be proportional to its keyword number.

Since all users want to query at a cheap price, *cooperative search* can be a good approach to save total cost, hence decreasing individual payments. To illustrate, let us consider an example in Fig. 6.1. Five users want to search over the same database to retrieve information with six keywords. Given per-query cost c' from the owner and

per-keyword cost c from miners, each query will be separately charged by the owner and miners. For example, the cost of query $q_1 = 111000$ (a six-bit binary string for six keywords, with 1 for select and 0 for not select) is c' for the owner plus $3c$ due to 3-keyword search in Ethereum. Without cooperation, the overall cost of these queries is $5c' + 11c$ (5 users and 11 accumulative keywords). To save cost, their queries can be grouped. Among all the grouping strategies, we briefly mention two here: (1) a *cost-saving-based* strategy without considering delay constraints. Five users are grouped together, and their combined query is 111111 at a total cost of $c' + 6c$. Search latency largely increases for each client. (2) a *delay-tolerant-oriented* strategy (as is shown in Fig. 6.1) that groups q_1, q_2, q_4 in G_1, and q_3, q_5 in G_2. Thus, the total cost for all users is $2c' + 6c$ and any user at most waits additional 1-keyword search time. This example reflects a tradeoff between delay constraint and cost efficiency.

We consider users with limited delay-tolerance. We design a client-biased cooperative search scheme, which facilitates group formation among users driven by cost savings subject to a uniform delay constraint all users agree upon. Users submit their individual queries and delay constraints to a front *proxy*. The proxy gathers users with similar delay constraints into a set, and matches users from the same set as search groups in order to minimize the overall cost subject to the delay constraint (*i.e.*, the number of 1s in each group query should not exceed a given number). After grouping, the data owner receives combined queries from the proxy, and issues search requests to Ethereum. This cooperative search scheme improves a data owner's processing capacity by reducing the query number.

In a cooperative model, cost sharing must be regulated in a truth-revealing and fair manner. Truth-revealing helps avoid free-riding users who want to get some information without payment, and fairness can promote a stable and long-term cooperation among users. For example, we consider a common cost sharing mechanism, which equally distributes group cost among its members. After applying it to the grouping

result in Fig. 1, q_3 and q_5 will be equally charged with $(c' + 3c)/2$. It seems unfair because q_5 has fewer keywords in its original form compared with q_3. In this chapter, we design a keyword-based cost sharing mechanism according to original queries, which yields some desirable properties like group-strategyproofness and sharing incentive. In Fig. 6.1, the cost $2c'$ paid to the owner will be equally distributed among 5 users, each of whom is responsible for $2c'/5$. The total cost of searching for the first keyword is c, equally shared by q_1 and q_2. The last keyword also costs c, which is only borne by q_5, since there is no other user querying this keyword.

Specifically, given a set of n queries over a database, our objective is to classify n queries into k groups with each group satisfying a uniform delay constraint so that the total cost over all groups is minimized.

6.2 Preliminaries

Scheme Overview

Our multi-user cooperative search scheme consists of two stages, involving four entities, as is shown in Fig. 6.1. The first stage consists of three entities: users, a proxy and a data owner. Although we assume a data owner and a proxy, it can be easily extended to multiple owners and proxies. All users send their queries along with delay constraints to the proxy. The proxy gathers users with similar delay constraints into the same sets. We assume each set is sufficiently large, and thus treat them separately. In each set, the minimal value of all users' delay constraints is set as the set delay upper bound. Given an n-query set, the main function of the proxy is to run our grouping strategies which classify n queries into k groups (k is a variable) and send those combined queries to the data owner. In the second stage, a data owner issues search transactions to Ethereum, based on queries received from his proxy. Then miners execute related functions to obtain search results. The total

cost is shared among n users using our cost sharing mechanism.

Design Goals

To implement a client-biased cooperative search scheme in the Ethereum-based data market, our main goals are divided into three parts. First, we need to design effective grouping strategies by simultaneously achieving *social optimum*: the total cost among n users is minimal, and *latency limitation*: each user can be guaranteed to retrieve desired information within their uniformly-agreed delay tolerance. Second, we want to design a cost sharing mechanism among n users, which offers *group-strategyproofness*: each user should truthfully reveal individual query request even if collusion is permitted, since lying provides no benefit to his interest, and *sharing incentive*: each user should achieve individual cost reduction if their total cost gets reduced. Third, we would like to implement an Ethereum testbed to verify the practicality of our proposed search scheme, and the actual relationship between the keyword number and the search delay.

6.3 Grouping strategy

6.3.1 Notation and Cost Model

Assuming we have a set of n queries, $Q = \{q_1, q_2, \cdots, q_n\}$, that are issued from different users but over the same database. Corresponding notations are listed in Table 6.1.

The cost $C(q)$ of a query q consists of two parts: c' is charged by a corresponding owner due to his contribution on data and $c \cdot |q|$ is paid to a miner in search of information.

Table 6.1: Summary of Notations.

Symbol	Description
d	number of keywords in the dictionary
w_i	the i-th keyword in the dictionary
n	number of queries
Q	n-query set
G_i	group i, which is a subset of Q
$\|G_i\|$	number of queries in G_i
q_i	query from user i, in the form of a binary string
$\|q_i\|$	number of keywords in q_i
$\hat{q}_i$	combined query of G_i
$\|\hat{q}_i\|$	number of keywords in $\hat{q}_i$
k	number of groups
P_i	a partition over Q with i non-overlapping groups
c	cost of miner searching for a keyword
c'	cost of a data owner

Cost without Grouping

The n queries in the set Q are individually executed and their total cost is the accumulation of their individual costs.

$$\sum_{i=1}^{n} C(q_i) = c' \cdot n + c \cdot \sum_{i=1}^{n} |q_i| \tag{6.3-1}$$

Cost with Grouping

Another way to execute Q is to create a single group G that contains all n queries and execute it as a single query $\hat{q} = |q_1 \vee \cdots \vee q_n|$.

$$C(\hat{q}) = c' + c \cdot |\hat{q}| \tag{6.3-2}$$

We can determine if grouping is beneficial by comparing Eq. (6.3-1) with Eq. (6.3-2).

$$\sum_{i=1}^{n} C(q_i) - C(\hat{q}) = c'(n-1) + c(\sum_{i=1}^{n} |q_i| - |\hat{q}|) \tag{6.3-3}$$

Even in the worst case, where no overlapping keywords are among n queries, *i.e.*, $\sum_{i=1}^{n} |q_i| = |\hat{q}|$, Eq. (6.3-3) ≥ 0 still holds. It is obvious that grouping is always beneficial. Thus, greedily grouping all queries into a combined query is always a socially optimal choice, if no user has a constraint on search delay.

6.3.2 Problem Formulation

Given $Q = \{q_1, q_2, ..., q_n\}$, let α be the pre-agreed maximal search delay, measured in the numbers of keywords among n users. That is, each user is willing to wait for at most α-keyword search time, whatever their original keyword numbers before grouping. An optimal grouping problem is defined as:

Problem 4 (Optimal Query Grouping, OQG). Given a set of queries $Q = \{q_1, q_2, ..., q_n\}$, group the n queries into k non-overlapping groups $G_1, G_2, ..., G_k$ (k is a variable), such that the number of keywords in each combined query is no more than α and the overall cost of all groups is minimized.

6.3.3 Problem Hardness

Theorem 12. The OQG problem is NP-hard.

Proof. The Set Partitioning (SP) problem , known as NP-hard, can be reduced to the OQG problem. The SP problem is expressed as: given a set of n positive integers $\{a_1, a_2, \cdots, a_n\}$ and an integer A, where $\forall i \in [1, n],\ a_i \leq A$, such that $\sum_{i=1}^{n} a_i = 2A$, decide if this set can be partitioned into two subsets with the same sum A. We can construct every instance of the SP problem as a valid instance of the OQG problem as follows. Let α, the upper limit of the keyword number in each combined query, be equal to A, and d, the number of keywords in the dictionary, be $2A$. Each a_i is mapped to a query q_i. We define the number of keywords for each query $|q_i|$ as a_i. We also assume that there is no overlapping among all queries. This is a valid instance of the OQG problem.

An optimal solution to the OQG problem with two groups exists, if and only if there exists a partition in the original SP problem. Obviously, merging n queries into a single group G is infeasible, because the combined query $\hat{q}$ contains $2A$ keywords, exceeding the upper limit. Hence, at least one query should be removed from G. According to our previous analysis, for any group G_i in the optimal solution, its cost is $C(\hat{q}_i) = c' + c \cdot |\hat{q}_i|$. We minimize $C(\hat{q}_i)$ over all k groups, but $c\sum_{i=1}^{k} |\hat{q}_i|$ is constant among all grouping strategies, hence our problem is equivalent to minimizing the number of groups.

If the OQG problem has an optimal solution with two groups G_1 and G_2, then there exists a partition of n queries into two sets, such that $\sum_{q_i \in G_1} |q_i| = \sum_{q_j \in G_2} |q_j| = \alpha$. This partition is definitely an optimal solution to SP problem. On the other hand, if the original SP problem has an equal-sum partition, the OQG problem also has an optimal strategy with two groups, since three groups yield more cost. Now, we can conclude that the OQG problem is NP-hard. □

6.3.4 Grouping Strategy

Since the OQG problem is NP-hard, we solve it using linear programming relaxation and greedy algorithms.

Mathematic Relaxation

First, we give the matrix representation of the above problem. Since each user has a query string with length d, we define an $n \times d$ matrix $\mathbf{Q}$ as representing all queries from n users. Let $\mathbf{Y} \in \{0,1\}^{n \times n}$ denote the grouping result, then each (i,j)-th element of $\mathbf{Y}$ takes either 0 or 1. $Y_{ij} = 1$ means query q_i is classified into group G_j. The matrix representation is shown below.

$$\arg\min_{\mathbf{Y}} \qquad c' \cdot tr(\delta(\mathbf{Y}^T\mathbf{E})) + c \cdot tr(\mathbf{E}^T\delta(\mathbf{Y}^T\mathbf{Q})) \tag{6.3-4a}$$

$$\mathbf{Y} \in \{0,1\}^{n \times n},\ \delta(\mathbf{Y}^T\mathbf{Q})\mathbf{e} \leq \alpha\mathbf{e},\ \mathbf{Y}\mathbf{e} = \mathbf{e} \tag{6.3-4b}$$

where tr is the trace operator which sums up diagonal elements of a given matrix. $\mathbf{E}$ and $\mathbf{e}$ denote an all-1's matrix and an all-1's vector, respectively, and both of their size can be adjusted to fit the context. And $\delta(\cdot)$ is an indicator function such that $\delta(c) = 1$ if c is non-zero and $\delta(c) = 0$, otherwise. The value of (i, j)-th element in matrix $\mathbf{Y}^T\mathbf{Q}$ represents how many times the keyword w_j is queried among all members in the group G_i. Since each overlapping keyword in a group only needs querying once, $\delta(\mathbf{Y}^T\mathbf{Q})$ indicates combined queries for all groups. Thus, $tr(\mathbf{E}^T\delta(\mathbf{Y}^T\mathbf{Q}))$ calculates the total keyword number of all combined queries. Similarly, $tr(\delta(\mathbf{Y}^T\mathbf{E}))$ indicates the true group number.

We use the example in Fig. 6.1 to better explain each term in Eq. (6.3-4a). Five queries q_1, q_2, q_3, q_4, q_5 are represented as a matrix $\mathbf{Q}$ in Fig. 6.2(a), where the i-th row represents q_i. The grouping result $\mathbf{Y}$ is provided in Fig. 6.2(b), indicating there are two groups, q_1, q_2 and q_4 in G_1, and q_3, q_5 in G_2. Thus, the number of groups can be calculated using the expression $tr(\delta(\mathbf{Y}^T\mathbf{E})) = 2$ based on Fig. 6.2(c). Besides, the combined queries of G_1 and G_2 are $\hat{q}_1 = 111000$ and $\hat{q}_2 = 000111$, which are consistent with its matrix representation $\delta(\mathbf{Y}^T\mathbf{Q})$ shown in Fig. 6.2(d). Hence, the total keyword number over all combined queries $tr(\mathbf{E}^T\delta(\mathbf{Y}^T\mathbf{Q})) = 6$ is equal to $|\hat{q}_1| + |\hat{q}_2|$.

Our objective is to find a grouping strategy $\mathbf{Y}$, which will result in a minimal overall cost for n queries, as Eq. (6.2) shows, while each combined query has no more than α keywords ($\delta(\mathbf{Y}^T\mathbf{Q})\mathbf{e} \leq \alpha\mathbf{e}$) and any original query only belongs to a group ($\mathbf{Y}\mathbf{e} = \mathbf{e}$). Since it is an NP-hard problem, we consider a relaxation. According to [75], given a large constant number β, *e.g.* $\beta = 10,\ 20,\ 30$, the indicator function over the matrix $\mathbf{Y}^T\mathbf{Q}$, *i.e.*, $\delta(\mathbf{Y}^T\mathbf{Q})$, can be approximated with a smooth function, $\mathbf{E} - e^{(-\beta\mathbf{Y}^T\mathbf{Q})}$, and the indicator function over the matrix $\mathbf{Y}^T\mathbf{E}$, *i.e.*, $\delta(\mathbf{Y}^T\mathbf{E})$, can be approximated with a smooth function, $\mathbf{E} - e^{(-\beta\mathbf{Y}^T\mathbf{E})}$. With the above relaxations, we

$$\mathbf{Q}_{5\times 6}=\begin{bmatrix}1&1&1&0&0&0\\1&0&1&0&0&0\\0&1&1&0&0&0\\0&0&0&1&1&0\\0&0&0&1&0&1\end{bmatrix} \qquad \mathbf{Y}_{5\times 5}=\begin{bmatrix}1&0&0&0&0\\1&0&0&0&0\\0&1&0&0&0\\1&0&0&0&0\\0&1&0&0&0\end{bmatrix}$$

(a) (b)

$$\delta(\mathbf{Y}^T\mathbf{E})=\delta\left(\begin{bmatrix}1&1&0&1&0\\0&0&1&0&1\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}\times\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}\right)=\begin{bmatrix}1&&&&\\&1&&&\\&&0&&\\&&&0&\\&&&&0\end{bmatrix}$$

(c)

$$E^T\delta(\mathbf{Y}^T\mathbf{Q})=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}\times\begin{bmatrix}1&1&1&0&0&0\\0&0&0&1&1&1\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}=\begin{bmatrix}1&&&&&\\&1&&&&\\&&1&&&\\&&&1&&\\&&&&1&\\&&&&&1\end{bmatrix}$$

(d)

Figure 6.2: Matrix representation of queries and grouping result (We don't fill all elements in some matrices for saving space).

transfer the integer matrix $\mathbf{Y}$ into the continuous domain. Then, we get a relaxed version of the OQG problem below:

$$\begin{aligned}\arg\min_{\mathbf{Y}} \quad & c'\cdot tr(\mathbf{E}-e^{-\beta\mathbf{Y}^T\mathbf{E}})+c\cdot tr(\mathbf{E}^T[\mathbf{E}-e^{-\beta\mathbf{Y}^T\mathbf{Q}}])\\ & \mathbf{Y}\in[0,1]^{n\times n},\ \mathbf{Y}\mathbf{e}=\mathbf{e},\ e^{-\beta\mathbf{Y}^T\mathbf{Q}}\mathbf{e}\geq(d-\alpha)\mathbf{e}\end{aligned}$$

which equals to its dual problem as below:

$$\begin{aligned}\arg\max_{\mathbf{Y}} \quad & c'\cdot tr(e^{-\beta\mathbf{Y}^T\mathbf{E}})+c\cdot tr(\mathbf{E}^Te^{-\beta\mathbf{Y}^T\mathbf{Q}})\\ & \mathbf{Y}\in[0,1]^{n\times n},\ \mathbf{Y}\mathbf{e}=\mathbf{e},\ e^{-\beta\mathbf{Y}^T\mathbf{Q}}\mathbf{e}\geq(d-\alpha)\mathbf{e}\end{aligned} \tag{6.3-5}$$

The objective of Eq. (6.3-5) is to maximize a convex function over multiple variables

Table 6.2: Examples of 7 User Queries.

Queries	Content
q_1	11010000
q_2	00001101
q_3	11000000
q_4	00000111
q_5	00001100
q_6	00000011
q_7	10000000

(a) Example One.

Queries	Content
q_1	11010000
q_2	00001101
q_3	11000000
q_4	00000111
q_5	00001100
q_6	10000001
q_7	00110000

(b) Example Two.

with nonlinear constraints. We apply the interior-point approach, transforming the original inequality-constrained problem into a sequence of equality constrained problems. A logarithmic barrier function with a dynamic coefficient μ is constructed and added to the original objective function to remove all inequality constraints. This algorithm has a two-level iteration. The outer level iterates over the coefficient μ, while the inner level optimizes the augmented objective function using Newton method under a fixed μ. Since Newton method may lead to a local optima, we can run the algorithm with different initial values and select the best one.

In addition, we also design a rounding algorithm to obtain a feasible $0-1$ solution based on the continuous optima achieved above. In each iteration, the rounding algorithm greedily sets Y_{ij} as 1 to maximize the objective function in Eq. (6.3-5) while still satisfying all constraints. To make our integer solution closer to the optimum one, the rounding order of queries matters. We always start with queries with the most keywords first, because chances are higher for them to overlap with other queries. Once their grouping result is determined, other queries can be assigned to thecorresponding groups.

We consider a dictionary that consists of (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8) and two sample queries are as shown in Table 6.2(a) and 6.2(b). We also assume $c = c'$. To show how it affects grouping results, the constraint of search delay will be set differently. Table 6.3 shows grouping results under different delay constraints using

our Mathematic Relaxation.

Projected Gradient Descent Method

The Newton method is quite simple and gives a relatively fast rate of convergence. However, this method is very expensive in each iteration - it needs the function evaluation and then the derivative evaluation. If the volume of queries is large, then this might not be a good choice. Thus, to improve the efficiency when facing large amounts of queries, we consider a simple modification of gradient descent for constrained optimization: a projected gradient descent method. In general, projected gradient algorithms minimize an objective $f(x)$ subject to the constraint that $x \in \chi$ for some convex set χ. They do this by iteratively updating $x := \prod_{\chi}(x + \eta \nabla f(x))$, where η represents a step length of learning rate, and $\prod_{\chi} = \arg\max_x \{\|z - x\| \, | x \in \chi\}$ is the Euclidean projection onto set χ. First order projected gradient algorithms are effective when second order methods are infeasible because of the dimension of the problem.

Greedy Algorithm

Since Mathematic Relaxation uses a first order local optimization method to solve a non-convex optimization problem, of which computational complexity isn't proven, this strategy has no guarantee on time. Thus, we consider a greedy algorithm with a guaranteed bound.

According to section 6.3.1, combination always brings about cost saving. Here, we reconsider the OQG problem in terms of cost saving. Given a set of queries $Q = \{q_1, q_2, ..., q_n\}$ from n different users over the same database, group these n queries into k non-overlapping groups $G_1, G_2, ..., G_k$ where the number of keywords in each group is no more than α such that the overall sum of savings $\sum_{i=1}^{k} S(G_i)$, is maximized.

Table 6.3: Grouping Results using Mathematic Relaxation.

Constraint	Group	Combined Query
4	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_4, q_5, q_6\}$	$\hat{q}_2 = 00001111$
3	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_5\}$	$\hat{q}_2 = 00001101$
	$G_3 = \{q_4, q_6\}$	$\hat{q}_3 = 00000111$

(a) Example One.

Constraint	Group	Combined Query
5	$G_1 = \{q_1, q_3, q_6, q_7\}$	$\hat{q}_1 = 11010001$
	$G_2 = \{q_2, q_4, q_5\}$	$\hat{q}_2 = 00001111$
4	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_4, q_5\}$	$\hat{q}_2 = 00001111$
	$G_3 = \{q_6\}$	$\hat{q}_3 = 10000001$

(b) Example Two.

Considering two queries, q and q', we define the saving of merging q with q' as $c' + c(|q| + |q'| - |q \vee q'|)$. In other words, combining two queries, q and q' will save one charge from the owner and overlapping-keyword search cost. Since one token generation cost can be saved by combining any two queries q and q', the more overlapping keywords q and q' have, the more search cost it will save through their combination. If q and q' have a containment relationship, we can prove there always exists some optimal grouping result that contains a group with both q and q', as is shown in Theorem 2.

Theorem 13. Given a set of queries $Q = \{q_1, q_2, ..., q_n\}$, for any two queries q and q', if the keywords of q are entirely contained in the keywords of q', then there is some optimal grouping strategy that contains a group with both q and q'.

Proof. Assume the optimal partition P over Q has two groups G_1 and G_2, such that $q \in G_1$ and $q' \in G_2$. Thus, individual group saving of G_1 is $S(G_1) = c'(|G_1| - 1) + c(\sum_{i=1}^{|G_1|} |q_k| - |\hat{q}_1|)$, and the same for G_2. We can obtain the overall savings of G_1 and

G_2 :

$$S(G_1) + S(G_2) = \tag{6.3-6}$$
$$c'(|G_1| + |G_2| - 2) + c(\sum_{i=1}^{|G_1|+|G_2|} |q_k| - |\hat{q}_1| - |\hat{q}_2|)$$

Moving q to G_2 leads to new groups $G_{1'} = G_1 \setminus \{q\}$ and $G_{2'} = G_2 \cup \{q\}$, with $|G_{1'}| = |G_1| - 1$ and $|G_{2'}| = |G_2| + 1$. q is entirely contained by q', hence $|\hat{q}_{2'}| = |\hat{q}_2|$. Thus, $S(G_{2'}) = c'(|G_2|) + c(\sum_{i=1}^{|G_2|} |q_i| + |q| - |\hat{q}_2|)$. Although it is difficult to directly know the exact value of $|\hat{q}_{1'}|$, we can bound it as $|\hat{q}_1| - |q| \leq |\hat{q}_{1'}| \leq |\hat{q}_1|$, such that $S(G_{1'}) \geq c'(|G_1| - 2) + c(\sum_{i=1}^{|G_1|} |q_i| - |q| - |\hat{q}_1|)$. Listed below is a lower bound of the overall group savings for new groups $G_{1'}$ and $G_{2'}$:

$$S(G_{1'}) + S(G_{2'}) \geq \tag{6.3-7}$$
$$c'(|G_1| + |G_2| - 2) + c(\sum_{i=1}^{|G_1|+|G_2|} |q_i| - |\hat{q}_1| - |\hat{q}_2|)$$

Comparing Eq. (6.3-6) and Eq. (6.3-7), we conclude that, moving q to G_2 yields a new partition, which is at least as good as P and thus still optimal. □

Hence, we can greedily group queries q and q', and treat them cost-wise as a single query q', which can be further merged with other queries. The containment can easily be determined by OR operation. This is a Naive Greedy solution, of which the time complexity is $O(n^2)$. For the rest of the chapter, we assume all such containments have been identified.

It is obvious that, without any constraints on the search delay, the optimal solution is to combine n users as a group, and issue a single combined query. However, when constraints are added, it becomes an NP-hard problem, thereby we consider a Greedy Partition solution to efficiently approximate its optimal result with an upper bound. Greedy Partition starts with a single group, iterating to split a group of which the splitting cost is minimal among all existing groups, until all query groups are subject

to the search delay constraints. As it is a typical set partition problem, we will consider Problem 4 in terms of set theory as follows.

First, given a query set Q, we define a set function $C : 2^Q \to \mathbb{R}$ where

$$\forall\, G \subseteq Q,\ C(G) = \begin{cases} 0 & G = \emptyset \\ c' + c \cdot |\hat{q}| & \text{otherwise} \end{cases} \tag{6.3-8}$$

Then, we can reformulate the OQG problem as below. Given a system (Q, C, k), where Q is a set of queries, $C : 2^Q \to \mathbb{R}$ is a set function, and k is a variable with $1 \leq k \leq n$.

$$\text{minimize} \qquad C(G_1) + C(G_2) + \cdots + C(G_k) \tag{6.3-9a}$$

$$\text{subject to} \qquad G_1 \cup G_2 \cup \cdots \cup G_k = Q \tag{6.3-9b}$$

$$G_i \cap G_j = \emptyset \qquad 1 \leq i < j \leq k \tag{6.3-9c}$$

$$|\hat{q}_i| \leq \alpha \qquad 1 \leq i \leq k \tag{6.3-9d}$$

As is proven in the paper [114], given a nondecreasing submodular system (V, f, k), where $f(V) + f(\emptyset) \geq f(S)$ holds for any nonempty subset S of V, the set partition problem can be approximated within a factor of $(2 - 2/k)$ in polynomial time. They provide a greedy algorithm to guard this result. Since our system is also submodular (proven below), we present Greedy Partition which satisfies delay constraints.

As is shown in Algorithm 1, Greedy-Partition has two functions. The main function Greedy-Partition returns the final partition P over a given set Q. Starting with the 1-partition $P_1 = Q$, in its ith iteration, we obtain an $i+1$-partition P_{i+1} by partitioning some members of the previous i-partition P_i. We halt when a partition P satisfies the delay constraints. For any member W in i-partition P_i, we call function Optimal-Subset [114] to find its minimal-partitioning-cost subset. Since a minimal-cost solution is desired, we choose the least-cost partition among all members. The time complexity of this algorithm is $O(kn^3)$, where k is the number of groups, and n is the query number. Grouping results of the previous examples are listed in Table 6.4.

Algorithm 6 Greedy Partition

Input: a system, (Q, C)
Output: a partition over Q, P

```
1: function GREEDY-PARTITION(Q, C)
2:     P_1 ← {Q}
3:     for each i ∈ [1, n] do
4:         for all W ∈ P_i do
5:             (S, W) ← OPTIMAL-SUBSET(W)
6:         (S_i, W_i) ← argmin (C(S) + C(W/S) − C(W))
7:         P_{i+1} ← (P_i − {W_i}) ∪ {S_i, W_i/S_i}
8:         if each W ∈ P_i satisfies search delay then
9:             return P_i
```

Table 6.4: Grouping Results using Greedy Partition.

Constraint	Group	Combined Query
4	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_4, q_5, q_6\}$	$\hat{q}_2 = 00001111$
3	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_5\}$	$\hat{q}_2 = 00001101$
	$G_3 = \{q_4, q_6\}$	$\hat{q}_3 = 00001111$

(a) Example One.

Constraint	Group	Combined Query
5	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_4, q_5, q_6\}$	$\hat{q}_2 = 10001111$
4	$G_1 = \{q_1, q_3, q_7\}$	$\hat{q}_1 = 11010000$
	$G_2 = \{q_2, q_4, q_5\}$	$\hat{q}_2 = 00001111$
	$G_3 = \{q_6\}$	$\hat{q}_3 = 10000001$

(b) Example Two.

In the rest of this section, we will demonstrate that Greedy Partition can solve the OQG problem within a factor of $(2-2/k)$ in polynomial time. According to [114], our proposed algorithm on a given system (Q, C) can achieve the above properties if the following two conditions hold: (1) C is submodular, and (2) C is non-decreasing.

Before showing Greedy Partition satisfies the above two conditions, we introduce their definitions. Given a finite set V and a set function $f{:}2^V{\to}\mathbb{R}$, f is (1) submodular if $\forall S{\subseteq}V$ and s_1, $s_2{\in}V{\setminus}S$, $f(S \cup \{s_1\}) + f(S \cup \{s_2\}) \geq f(S \cup \{s_1, s_2\}) + f(S)$ always holds; (2) non-decreasing if $f(V) + f(\emptyset) \geq f(S)$ holds for any nonempty subset S of V .

Lemma 1. The set function $C : 2^Q \to \mathbb{R}$ is submodular.

Proof. Now, we will show for every $G \subseteq Q$ and $q_1, q_2 \in Q \setminus G$, Eq. (6.3-10) always holds. Without loss of generality we can assume that $G \neq \emptyset$, otherwise the answer is immediate.

$$C(G \cup \{q_1\}) + C(G \cup \{q_2\}) \geq C(G \cup \{q_1, q_2\}) + C(G) \tag{6.3-10}$$

Since $C(G \cup \{q_1\}) + C(G \cup \{q_2\}) = 2c' + c(|\hat{q} \vee q_1| + |\hat{q} \vee q_2|)$ and $C(G \cup \{q_1, q_2\}) + C(G) = 2c' + c(|\hat{q} \vee q_1 \vee q_2| + |\hat{q}|)$, we need to prove Eq. (6.3-11) ≥ 0 always holds.

$$\begin{aligned} & C(G \cup \{q_1\}) + C(G \cup \{q_2\}) - C(G \cup \{q_1, q_2\}) - C(G) \\ &= c(|\hat{q} \vee q_1| + |\hat{q} \vee q_2|) - c(|\hat{q} \vee q_1 \vee q_2| + |\hat{q}|) \\ &= c(|\hat{q} \vee q_1| + |\hat{q} \vee q_2| - |\hat{q} \vee q_1 \vee q_2| - |\hat{q}|) \end{aligned} \tag{6.3-11}$$

Assume that, nonnegative integers x, y, z represent the number of overlapping keywords between $\hat{q}$ and q_1, $\hat{q}$ and q_2, q_1 and q_2, respectively. Let m be the overlapping keyword number among $\hat{q}$, q_1, and q_2. It is obvious that $z \geq m \geq 0$.

$$\begin{aligned} & |\hat{q} \vee q_1| + |\hat{q} \vee q_2| = 2\,|\hat{q}| + |q_1| + |q_2| - x - y \\ & |\hat{q} \vee q_1 \vee q_2| + |\hat{q}| = 2\,|\hat{q}| + |q_1| + |q_2| - x - y - z + m \\ & |\hat{q} \vee q_1| + |\hat{q} \vee q_2| - |\hat{q} \vee q_1 \vee q_2| - |\hat{q}| = z - m \geq 0 \end{aligned} \tag{6.3-12}$$

Based on Eq. (6.3-12), we conclude, for every $G \subseteq Q$ and $q_1, q_2 \in Q \backslash G$, Eq. (6.3-10) always holds. Thus, $C : 2^Q \to \mathbb{R}$ is a submodular set function. □

Lemma 2. The set function $C : 2^Q \to \mathbb{R}$ is non-decreasing.

Proof. Based on Definition 2, we should prove $C : 2^Q \to \mathbb{R}$, $C(Q) + C(\emptyset) \geq C(G)$ holds for any nonempty subset G of Q. According to Eq. (6.3-8), $C(\emptyset) = 0$. Since $\forall G \subseteq Q$, it is obvious that set Q's combined query contains more keywords than its subset G's combined query. Thus, we can obtain $C(Q) - C(G) \geq 0$, hence, $C(Q) + C(\emptyset) \geq C(G)$. □

Theorem 14. The QOG problem can be approximated within a factor of $(2 - 2/k)$ by Greedy Partition.

This theorem easily follows from Lemmas 1 and 2. This grouping strategy is suitable for those users who have requirements on cost reductions.

6.4 Fair Cost Sharing

Our grouping strategies will yield a total cost for n users. Thus, one must find a way to distribute the cost among all users. A major purpose of our proposed grouping strategies is to seek high efficiency of the whole network, in the fields of both finance and computation. As self-interested and autonomous entities, clients may behave strategically by misreporting their willingness to query to maximize their profit, thereby harming the efficiency. Thus, we want our cost sharing mechanism to be incentive compatible, i.e., it is in the interest of clients to be truth telling [19]. Also, it should provide an incentive for clients in their assigned groups to participate in the coalition without coercion, *i.e.*, it is fair and maintains the stability of a given grouping result.

6.4.1 Cost Sharing Mechanism

To address this challenge, we design a cost sharing mechanism with two desirable properties: (1) *group-strategyproofness* and (2) *sharing incentive*. In the following, we first present our mechanism, then prove it can satisfy the above two properties. In our cost sharing mechanism, the total cost of n clients is composed of two parts: one part goes to the data owner's account, and the other is for Ethereum miners; so does it for individual cost. Each user is equally responsible for the total payment to the data owner. Given a grouping result of k combined queries, the data owner will make a revenue of kc', each client paying kc'/n to him.

Table 6.5: An Example of User Cost Sharing.

Keyword	Cost	Shared by
w_1	$1 \cdot c$	q_1, q_3, q_7
w_2	$1 \cdot c$	q_1, q_3
w_3	$0 \cdot c$	
w_4	$1 \cdot c$	q_1
w_5	$2 \cdot c$	q_2, q_5
w_6	$2 \cdot c$	q_2, q_4, q_5
w_7	$1 \cdot c$	q_4, q_6
w_8	$2 \cdot c$	q_2, q_4, q_6

(a) Cost of each keyword

Clients	Cost
q_1	$\frac{3}{7}c' + (\frac{1}{3} + \frac{1}{2} + 1) \cdot c$
q_2	$\frac{3}{7}c' + (\frac{2}{2} + \frac{2}{3} + \frac{2}{3}) \cdot c$
q_3	$\frac{3}{7}c' + (\frac{1}{3} + \frac{1}{2}) \cdot c$
q_4	$\frac{3}{7}c' + (\frac{2}{3} + \frac{1}{2} + \frac{2}{3}) \cdot c$
q_5	$\frac{3}{7}c' + (\frac{2}{2} + \frac{2}{3}) \cdot c$
q_6	$\frac{3}{7}c' + (\frac{1}{2} + \frac{2}{3}) \cdot c$
q_7	$\frac{3}{7}c' + \frac{1}{3} \cdot c$

(b) User individual cost

Any keyword in a combined query may be redundant for some of its group members, and it is unfair for a user to pay for a keyword he never requests. Thus, the total cost of searching a certain keyword is only borne by those users who request it. Thus, the cost sharing is at the granularity of n clients instead of each group. For each unique keyword, we calculate its total cost in all combined queries, and then evenly distribute the cost among all users querying this keyword. That is, if a keyword is queried by m of n users and appears in t of k combined queries, its total search cost is tc, and each one from m users is equally responsible for a cost share of tc/m.

We show how to share the total cost using the grouping result shown in Table 6.4(a) under the constraint of 3 keywords. For the rest of this paragraph, each client i is identified by his query q_i. The grouping result is $G_1 = \{q_1, q_3, q_7\}$, $G_2 = \{q_2, q_5\}$, and $G_3 = \{q_4, q_6\}$. Thus, the cost paid to the corresponding data owner is $3c^{'}$, equally distributed among 7 clients. Table 6.5(a) presents total cost for each keyword and who should be fairly responsible for the corresponding cost. Table 6.4(b) gives the final split cost for each client. For example, client 1's total cost is $3c'/7 + (1/3 + 1/2 + 1)c$, where (1) $3c'/7$ is paid to the data owner, shared with all other 6 clients; (2) $c/3$ comes from querying keyword w_1, shared with users q_3 and q_7; (3) $c/2$ comes from querying keyword w_2, shared with q_3; (4) c comes from querying keyword w_4 by himself.

6.4.2 Theoretical Analysis

We present theoretical analysis to demonstrate that our cost sharing mechanism achieves some desirable properties. For group-strategyproofness, we should demonstrate each client will honestly disclose his real query request even if they are permitted to collude. For each keyword, if a client's dominant strategy is to truthfully tell whether he wants to query it or not, then truth-revealing is his dominant strategy. Thus, we can divide the whole proof into d steps, and the j-th step shows that each client would prefer revealing his real request on the keyword w_j in our cost-sharing mechanism. Thereby, we divide our cost sharing mechanism on keyword search part into d cost sharing methods, one for each keyword, then we prove each method satisfies group-strategyproofness.

The cost sharing method of keyword w_j is a function, ξ_j, which distributes the total cost of searching for the j-th keyword, denoted as C_j, to its requesters. More formally, ξ_j takes two arguments, a subset of users G and a user q_i, and returns a nonnegative real number satisfying the following: (1) if $q_i \notin G$ then $\xi_j(G, q_i) = 0$, and (2) $\sum_{q_i \in G} \xi(G, q_i) = C_j$. As is proven in [42], if ξ_j is a cross-monotone, then the mechanism specified above is group-strategyproof- for keyword w_j. Thus, we need to prove ξ_j is cross-monotone. A cost sharing method can be said as cross-monotone if for $G \subseteq R$, $\xi_j(G, q_i) \geq \xi_j(R, q_i)$ for every $q_i \in G$.

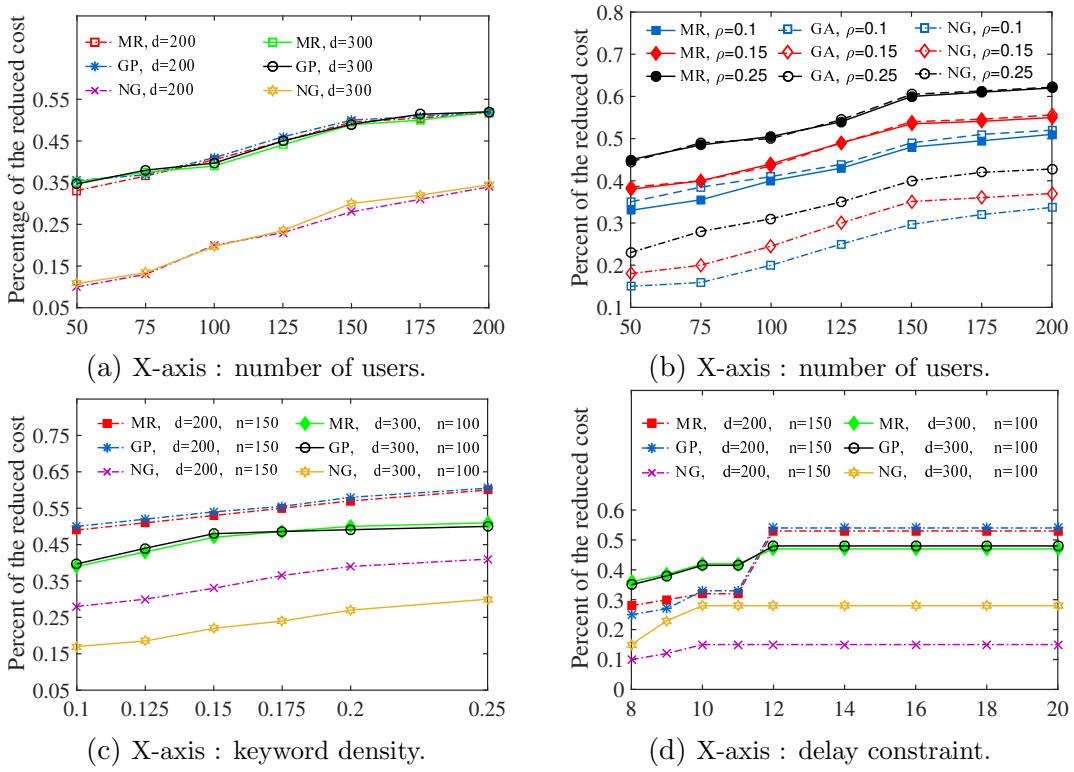
Lemma 3. For every $j \in [1, d]$, ξ_j is cross-monotone.

Proof. Any $q_i \in R \setminus G$ refers to a client not requesting the j-th keyword, thereby they are charged zero cost share. Thus, $G \subseteq R$, $\xi_j(G, q_i) = \xi_j(R, q_i)$ for every $q_i \in G$. Thus, ξ_j is a special cross-monotone cost sharing mechanism. □

Theorem 15. Our cost sharing mechanism satisfies group-strategyproofness and sharing incentive for all clients.

Proof. The property of group-strategyproofness can be proven using Lemma 3. To

show sharing incentive, we should reveal, for any client, leaving his current assigned group would not bring him more benefits. Sending an individual query definitely brings more cost paid to the data owner, which is cost-inefficient. As is shown in Eq. (6.3-3), grouping is always beneficial for each client to save cost paid to data owners. Thus, no one has incentive to leave. □



(a) X-axis : number of users.

(b) X-axis : number of users.

(c) X-axis : keyword density.

(d) X-axis : delay constraint.

Figure 6.3: Evaluations of grouping strategies on real query traces. MR: Mathematic Relaxation, GP: Greedy Partition, and NA: Naive Greedy.

6.5 Performance Evaluation

Our evaluation consists of two parts. First, we focus on evaluating our proposed cooperative search scheme on real query traces AOL [3]. sECOND, we implement an Ethereum testbed to demonstrate the practicality of our scheme, and also analyze the actual relationship between the keyword number and the search delay.

6.5.1 Cooperative Search Scheme

Our experiments evaluate two grouping strategies in terms of total cost reductions and the cost sharing mechanism in terms of individual cost saving. Mathematic relaxation was implemented with MATLAB-R2017b and greedy algorithms were implemented with Eclipse 4.6 in Java. All experiments are conducted on AOL. Ad AOL is a huge query collection, we randomly choose 200 users with 31 804 queried keywords in total, among which 17 786 are unique. Thus, a 400×17786 binary matrix is constructed to reveal each query's request on each keyword. Since it is still a large array, we semi-randomly select part of the matrix in each experiment to satisfy pre-set constraints on dictionary size, query number, and keyword density. For simplicity, we define two parameters: *keyword density* and *charge ratio*. Given a d-size dictionary and an n-query set Q, *keyword density* ρ of Q is defined as $\rho = \sum_{i=1}^{n} |q_i| / (n \times d)$. Given c' from a data owner and c from miners, *charge ratio* r is defined as $r = c'/c$.

Grouping strategies: We analyze the percentage of reduced total cost using our proposed grouping strategies: Mathematic Relaxation (using PGD here since the query volume is large), Naive Greedy and Greedy Partition. Fig. 6.3 shows, in all parameter settings, all strategies achieve cost reduction by at least 24.8%. Greedy Partition works slightly better than Mathematic Relaxation, and Naive Greedy achieves the least total cost reduction, which is around 50% of the other two strategies. Since the complexity of Naive Greedy and Greedy Partition is $O(n^2)$ and $O(n^3)$, respectively, we could see an inevitable tradeoff between efficiency and performance. Now, we analyze how each parameter influences the total cost reduction. In Fig. 6.3(a), as n increases, the total cost reduction also increases. Given a fixed ρ, changing d has little effect on the cost reduction. Fig. 6.3(b) reflects, as ρ increases from 0.1 to 0.25, the total cost is reduced by about 10% for each unique n. In Fig. 6.3(c), we have two set comparable parameters: $(d = 200, n = 150)$ and $(d = 300, n = 100)$. Given a fixed ρ, each set has the same number of 1s. From this experiment, we could see the

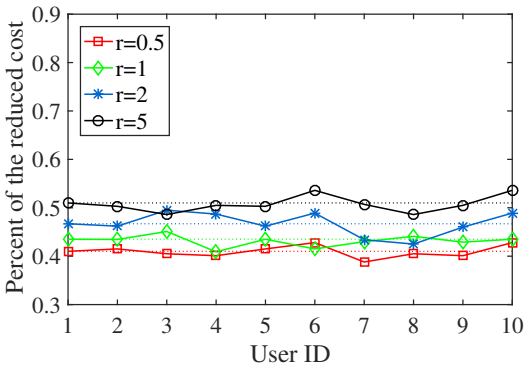


Figure 6.4: Individual vs average saving.

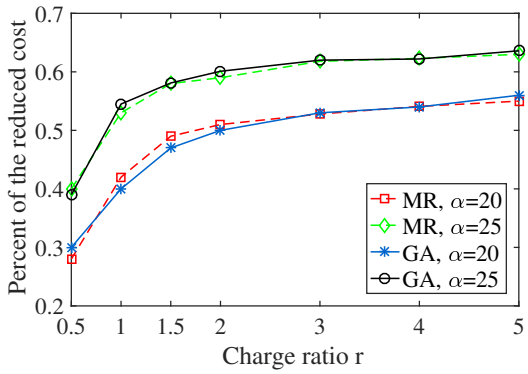


Figure 6.5: Effect caused by charge ratio.

first set has more cost savings, since smaller size of d yields higher chances of keyword overlapping. Fig. 6.3(d) reflects that the effect of delay constraint α on the total cost reduction decreases as its value increases.

Cost sharing mechanism: In the second part, we study individual cost saving under our cost sharing mechanism by picking up 10 users with $d = 100$. Each time, we change r and select a better one from grouping results from Mathematic Relaxation and Greedy Partition. We compare the cost reduction between individuals and the average. As is shown in Fig. 6.4, individuals can benefit from our grouping strategies. Besides, no user largely deviates from the average level, which shows our cost sharing mechanism can achieve fairness.

Summary: Both grouping strategies are local optimal. Mathematic Relaxation uses random restarts to produce multiple rounds to mitigate this problem. The larger the number of random restarts, the better the performance, but the more the execution time. Therefore, Greedy Partition is more appropriate for large scale query systems, and Mathematic Relaxation can be used as the baseline to measure the grouping quality. In terms of the cost sharing mechanism, each user can achieve cost savings near around the average saving.

6.5.2 Ethereum Testbed

To demonstrate the practicality of our scheme, we implemented a testbed in a simulated Ethereum network called TestRPC [20]. TestRPC is a fast and customizable blockchain emulator. It sets mining time as 0 while truly revealing execution time and gas consumption of a transaction. This design allows us to focus on the search delay itself without being affected by mining or waiting delays. Our Ethereum testbed can be helpful in revealing the real relationship between the number of keywords and search delay by the miners. This provides a better estimate for search delay in the real system, and thereby, a better estimation of the grouping constraint.

Keyword number and search delay: We conduct two experiments to verify the actual relation between the keyword number and search delay. We stored a 5KB database with 20 keywords and 30 files, each tagged with one or more keywords. In the first experiment, we randomly select 5 keywords and incrementally add 20 more each time to see how the execution time changes. The result shows, as the number of keywords increases, the delay time also increases while there exists a slowdown in its growth rate. In the second experiment, we dedicatedly design 5 query sets, each including 4 queries. The total keyword number in each set is fixed at 10, while the unique keyword number changes. For each query set, we execute them in two ways: (1) executing all queries individually, and (2) executing a single query composed of 4 queries. We compare accumulative execution times and combined execution times, and analyze how execution time reduces as the number of overlapping keywords increases. The result of this experiment shows that the relationship between the number of overlapping keywords and execution time is nearly proportional. Based on the above results, the search delay is at least sublinear to keyword numbers.

Charge ratio: In our real implementation, we store a 1.4MB database with 300 unique keywords and 2000 files. Each file is tagged with some different keywords. We issue 75 transactions in order to store the entire database in blockchain. When

previously evaluating our cost sharing mechanism, we find that charge ratio r can affect individual cost savings as well as total cost reductions. Thus, when performing experiments on our testbed, we first analyze how charge ratio r can affect grouping results, hence changing cost reductions. As is shown in Fig. 6.5, there is a positive sublinear relationship between the total cost reduction and charge ratio r. In our previous sections, assuming $r = 1$ to yield a maximal reduction on total cost is acceptable, since it can be adjusted by a factor.

Four-user cooperative search: We also envision a small four-user setting with different queries, and conduct several optimal cooperative searches and their individual searches. Fig. 6.6 reflects the cost reduction in the form of transaction number and gas consumption amount, both of which are important cost measures in Ethereum. These two parameters follow a very similar changing pattern if given the same inputs. The reason is each transaction invokes execution of the same search function. As we can see, the cost reduction is positively related to the ratio of overlapping matched file number and the unique matched file number, which is a reflection of overlapping keyword number in original queries.

Summary: Using our testbed, we analyze the actual relation between the keyword number and the search delay, which is sublinear. Experiments are conducted to see how charge ratio affects cost reduction. The pricing for search part has more effects on the total cost reduction compared with the owner's pricing. The last experiment on four-user cooperative search has demonstrated the practicality of our proposed scheme.

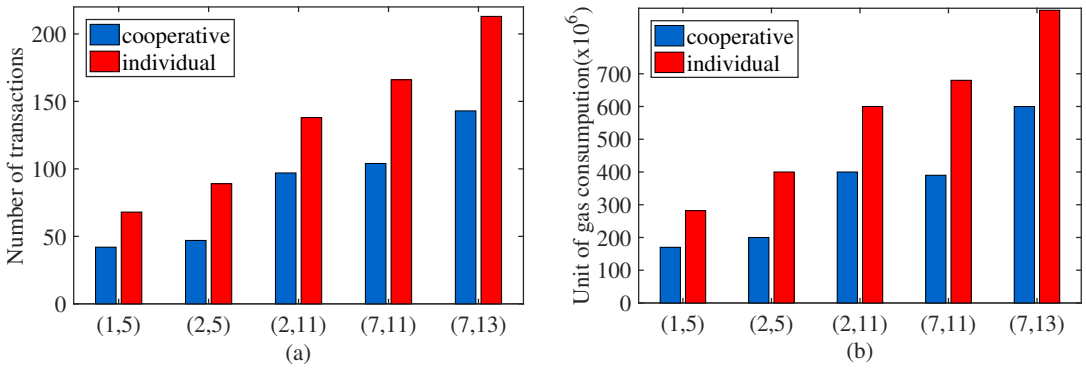


Figure 6.6: Cost reduction using testbed (X-axis : number of (overlapping files, unique files) tuples in non-grouping search result).

6.6 Related Work

Blockchain-based storage platforms

Decentralized storage platforms [14, 11, 19, 115, 71] are designed to allow users to store outsourced files on the peers who make profits by leasing their unused storage. Data can be stored either on-blockchain [14, 11, 19] or off-blockchain [115]. Filecoin [14] implements blockchain structured file storage and Datacoin [11] uses the blockchain as a data store for file blocks. Storj [19] employs end-to-end encryption and stores files on the blockchain, while [115] stores files in an off-blockchain storage, retaining only a pointer to the data on the public ledger.

Online data markets

On traditional online data markets, independent data owners sell their data through some digital platforms [1, 10]. Usually, buyers (users) have to buy the entire database from a seller, then download and query it offline, which is inefficient for buyers. On cloud-based data markets [4, 17], data owners upload their data in the cloud and make money by sharing it with users. Users pay to obtain what they want through a query interface instead of buying the entire database. Cloud providers are rewarded by providing services, *e.g.* storage and search. Our Ethereum-based data market is

similar to cloud-based data markets, where users pay for querying, while owners and miners earn by providing data and services, respectively.

Cost models

Traditional online markets take a one-time payment set by data owners. On cloud-based data markets, either data owners [4] or cloud providers [17] can be sellers, with term-based offers such as monthly subscriptions. Some cloud providers use a pay-as-you-go mode. Each query is charged based on bytes of scanned data [2]. On the Ethereum-based data market, owners and miners are all sellers, jointly charging users. Previously-mentioned online data markets all adopt a unilateral cost model [24], while the cost model can also be bilateral, such as auction, in a more complex situation.

Cooperative search

The cooperative keyword-based search scheme has been proposed in [75] by grouping all received queries together within a fixed timeout to achieve privacy. In [74], authors equally divide queries into a fixed number of groups to achieve k-anonymity and load balancing. Our work also deals with query grouping problems, while focusing on user-side cost saving and search delay guarantee.

Cost sharing schemes

A good cost sharing mechanism will distribute shared cost among users in a truth-revealing and fair manner [58]. The current cost sharing mechanism is group-based [74]. Our proposed mechanism is keyword-based, which guarantees group-strategyproofness and sharing incentive.

6.7 Conclusion

In this chapter, we present a cooperative search scheme on an Ethereum-based data market. We take advantage of smart contract and gas system in Ethereum to separate a query cost into two parts: one for data owners and the other for miners. We also make use of grouping strategies to provide efficiency and cost savings at the user side. We provide three methods, suitable for different scenarios, to compute an efficient grouping result. Besides, we propose a fair cost sharing mechanism to split total cost among users given a grouping result. This mechanism guarantees some desirable properties such as group-strategyproofness and sharing incentive to avoid free-riders. The experiment results show that our scheme is efficient in terms of cost reduction for both the group as a whole and individuals.

CHAPTER 7

CONCLUSION

7.1 Summary of Contributions

In this dissertation, we summarize our study on the blockchain-based systems from the aspects of design, extension, and application.

1. A major weakness of the current blockchain systems is the low transaction throughput, which is resulted from the small block size, the long block generation time, and the delayed block propagation. To shorten the block propagation delay, we design a neighbor selection algorithm, improving the communication speed in the P2P network of the blockchain. This design is an attempt on topology optimization and reduces the frequency of blockchain forking. Forking is a situation where the blockchain temporarily diverges into two or more branches, which wastes mining power and causes security issues.

2. Although a consensus mechanism aims to regulate mining nodes' behavior, it also can be manipulate by strategic miners within legal limits due to their profit-driven nature. In the dissertation, we analyze strategies that miners apply for utility maximization under different consensus mechanisms from the perspective of game theory. And then we propose distributed algorithms to optimize resource allocation among miners.

3. We also propose a blockchain-based market where we believe the application of blockchain will enhance the corresponding applications. It is a blockchain-

powered data market that allows multi-user cooperative search. We also integrate smart contract into these applications.

7.2 Future Works

Blockchain has shown its potential in industry and academia. We discuss possible future directions with respect to two areas: stop the tendency to centralization and balance efficiency and privacy in payment channel networks.

7.2.1 Stop the Tendency to Centralization

Blockchain is designed as a decentralized system. However, there is a trend that miners are centralized in the mining pool. Up to now, the top 5 mining pools together owns larger than 51% of the total hash power in the Bitcoin network. Apart from that, selfish mining strategy showed that pools with over 25% of total computing power could get more revenue than fair share. Rational miners would be attracted into the selfish pool and finally the pool could easily exceed 51% of the total power. As the blockchain is not intended to serve a few organizations, mining power should be more decentralized by design.

For PoW mechanism, one possible approach to democratizing mining power is to design ASIC proof hashing algorithms. For example, Litecoin and Dogecoin use scrypt, a hash algorithm that is designed to be ASIC proof by its high memory consumption. Another possible solution is to design a new reward policy which cannot differentiate pool mining with solo mining for indididuals. In this case, pool mining becomes less effective as it charges service fee.

7.2.2 Payment Channel Network Efficiency-Privacy Tradeoff

Payment channel networks (PCNs) are viewed as one of the most promising scalability solutions for cryptocurrencies today. As a layer-2 solution, a PCN is a network

lying on top of a blockchain, where each node represents a user and each directed, weighted edge represents funds escrowed on the blockchain; these funds can be transacted only between the endpoints of the edge. Users efficiently transmit funds from node A to B by relaying them over a path connecting A to B, as long as each edge in the path contains enough balance (escrowed funds) to support the transaction. Whenever a transaction succeeds, the edge weights are updated accordingly. In deployed PCNs, channel balances (*i.e.*, edge weights) are not revealed to users for privacy reasons; users know only the initial weights at time 0. Hence, when routing transactions, users typically first guess a path, then check if it supports the transaction. This guess-and-check process dramatically reduces the success rate of transactions. At the other extreme, knowing full channel balances can give substantial improvements in transaction success rate at the expense of privacy. Routing algorithms that can balance efficiency and privacy in PCNs should be proposed. We consider using a probabilistic model to partially reveal the balance information to protect node privacy as well as improving routing efficiency.

PUBLICATIONS

Journal Publications

1. **Suhan Jiang** and Jie Wu, "A Blockchain-Powered Data Market for Multi-User Cooperative Search", accepted to appear in IEEE Transactions on Network and Service Management.

2. **Suhan Jiang** and Jie Wu, "A Blockchain-based NFV Market in the Multi-node Edge Computing Network", accepted to appear in International Journal of Security and Networks (IJSN).

3. **Suhan Jiang** and Jie Wu, "A reward response game in the blockchain-powered federated learning system", Vol. 37, No. 1, 2022, 68-90. International Journal of Parallel, Emergent and Distributed Systems (IJPEDS).

4. **Suhan Jiang**, Xinyi Li, and Jie Wu, "Multi-Leader Multi-follower Stackelberg Game in Mobile Blockchain Mining", accepted to appear in IEEE Transactions on Mobile Computing.

5. **Suhan Jiang** and Jie Wu, "Multi-resource Allocation in Cloud Data Centers: A Trade-off on Fairness and Efficiency", Concurrency and Computation: Practice and Experience, Vol. 33, No. 6, 2021, e6061.

Conference Publications

6. **Suhan Jiang** and Jie Wu, "Taming Propagation Delay and Fork Rate in Bitcoin Mining Network", Proc. of the 4rd IEEE International Conference on Blockchain (Blockchain 2021).

7. **Suhan Jiang** and Jie Wu, "On Game-theoretic Computation Power Diversification in the Bitcoin Mining Network", Proc. of the IEEE Conference on Communications and Network Security (CNS 2021).

8. **Suhan Jiang** and Jie Wu, "A Reward Response Game in the Federated Learning System", Proc. of the 18th IEEE International Conference on Mobile Ad Hoc and Smart Systems (MASS 2021).

9. Jie Wu, and **Suhan Jiang**, "Local Pooling of Connected Supernodes in Lightning Networks for Blockchains", Proc. of the 3rd IEEE International Conference on Blockchain (Blockchain 2020).

10. **Suhan Jiang** and Jie Wu, "Game Theoretic Storage Outsourcing in the Mobile Blockchain Mining Network", Proc. of the 17th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS 2020).

11. **Suhan Jiang** and Jie Wu, "A Game-theoretic Approach to Storage Offloading in PoC-based Mobile Blockchain Mining", Proc. of the 21th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc 2020).

12. **Suhan Jiang**, Yubin Duan, and Jie Wu, "A Client-Biased Cooperative Search Scheme in Blockchain-Based Data Markets", Proc. of the 28th International Conference on Computer Communications and Networks (ICCCN 2019).

13. **Suhan Jiang**, and Jie Wu, "Bitcoin Mining with Transaction Fees: A Game on the Block Size", Proc. of the 2nd IEEE International Conference on Blockchain (Blockchain 2019).

14. **Suhan Jiang**, Xinyi Li, and Jie Wu, "Hierarchical Edge-Cloud Computing for Mobile Blockchain Mining Game", Proc. of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019).

15. **Suhan Jiang** and Jie Wu, "2-Dominant Resource Fairness: Fairness-Efficiency Tradeoffs in Multi-resource Allocation", Proc. of the 16th IEEE International Performance Computing and Communications Conference (IPCCC 2018).

Under Review

16. **Suhan Jiang** and Jie Wu, "A Game-theoretic Approach to Storage Offloading in PoC-based Mobile Blockchain Mining", submitted to Proc. of the 23rd International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc 2022).

17. Jie Wu, and **Suhan Jiang**, "On Increasing Scalability and Liquidation of Lightning Networks for Blockchains", submitted to IEEE Transactions on Network Science and Engineer.

18. **Suhan Jiang** and Jie Wu, "Approaching an Optimal Bitcoin Mining Overlay", submitted to IEEE/ACM Transactions on Networking.

BIBLIOGRAPHY

[1] Aggdata. http://www.aggdata.com/.

[2] Amazon athena. https://aws.amazon.com/athena/.

[3] Aol. https://archive.org/details/AOL_search_data_leak_2006.

[4] Azure data market. https://datamarket.azure.com/.

[5] Bitnodes. https://bitnodes.earn.com/.

[6] Blockchain.info. https://blockchain.info/q.

[7] Blockchain.pool. https://www.blockchain.com/pools.

[8] Burstcoin. https://www.burst-coin.org/.

[9] Chia. https://www.chia.net/.

[10] Customlists.net. http://www.customlists.net/home.

[11] Datacoin. http://datacoin.info/.

[12] Falcon - a fast bitcoin backbone. https://www.falcon-net.org/.

[13] Fibre: Fast internet bitcoin relay engine. https://www.bitcoinfibre.org/.

[14] Filecoin. https://filecoin.io/.

[15] A full node in pruned mode. https://bitcoin.org/en/full-node.

[16] Glassnode. https://studio.glassnode.com.

[17] Infochimps. http://www.infochimps.com/.

[18] Ipfs. https://ipfs.io/.

[19] Storj. https://storj.io/storj.pdf.

[20] testrpc. https://www.npmjs.com/package/ethereumjs-testrpc.

[21] Yusuke Aoki and Kazuyuki Shudo. Proximity neighbor selection in blockchain networks. *arXiv preprint arXiv:1906.00719*, 2019.

[22] Gagangeet Singh Aujla, Rajat Chaudhary, Neeraj Kumar, Joel JPC Rodrigues, and Alexey Vinel. Data offloading in 5g-enabled software-defined vehicular networks: A stackelberg-game-based approach. *IEEE Communications Magazine*, 55(8):100–108, 2017.

[23] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73, 2012.

[24] Magdalena Balazinska, Bill Howe, and Dan Suciu. Data markets in the cloud: An opportunity for the database community. *Proc. of the VLDB Endowment*, 2011.

[25] Massimo Bartoletti and Livio Pompianu. An empirical analysis of smart contracts: platforms, applications, and design patterns. In *International conference on financial cryptography and data security*, pages 494–509. Springer, 2017.

[26] Wei Bi, Huawei Yang, and Maolin Zheng. An accelerated method for message propagation in blockchain networks. *arXiv preprint arXiv:1809.00455*, 2018.

[27] Alex Biryukov and Ivan Pustogarov. Bitcoin over tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134. IEEE, 2015.

[28] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[29] Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, 2000.

[30] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[31] Chengjun Cai, Xingliang Yuan, and Cong Wang. Towards trustworthy and private keyword search in encrypted decentralized storage. In *ICC'17*.

[32] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

[33] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[34] Duan-Bing Chen, Hui Gao, Linyuan Lü, and Tao Zhou. Identifying influential nodes in large-scale directed networks: the role of clustering. *PloS one*, 8(10):e77455, 2013.

[35] M Corallo. High-speed bitcoin relay network. *November*, 2013.

[36] Matt Corallo. Compact block relay. bip 152, 2017.

[37] Nicolas T Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

[38] Mingjun Dai, Shengli Zhang, Hui Wang, and Shi Jin. A low storage room requirement framework for distributed ledger in blockchain. *IEEE Access*, 6:22970–22975, 2018.

[39] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P Proceedings*.

[40] Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. Txprobe: Discovering bitcoin's network topology using orphan transactions. In *International Conference on Financial Cryptography and Data Security*. Springer, 2019.

[41] Sergi Delgado-Segura, Cristina Pérez-Sola, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. Analysis of the bitcoin utxo set. In *International Conference on Financial Cryptography and Data Security*, pages 78–91, 2018.

[42] Nikhil R Devanur, Milena Mihail, and Vijay V Vazirani. Strategyproof cost-sharing mechanisms for set cover and facility location games. *Decision Support Systems*, 2005.

[43] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2018.

[44] Ido Erev and Alvin E Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American economic review*, pages 848–881, 1998.

[45] Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE, 2015.

[46] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 45–59, 2016.

[47] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[48] Muntadher Fadhil, Gareth Owenson, and Mo Adda. Locality based approach to improve propagation delay on the bitcoin peer-to-peer network. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 556–559. IEEE, 2017.

[49] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the*

theory and applications of cryptographic techniques, pages 281–310. Springer, 2015.

[50] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[51] Franco Giannessi and Antonino Maugeri. *Variational inequalities and network equilibrium problems*. Springer, 1995.

[52] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[53] Michael T Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing*, pages 792–799. IEEE, 2011.

[54] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on {Bitcoin's}{Peer-to-Peer} network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, 2015.

[55] Tai Manh Ho, Nguyen H Tran, Cuong T Do, SM Ahsan Kazmi, Tuan LeAnh, and Choong Seon Hong. Data offloading in heterogeneous cellular networks: Stackelberg game based approach. In *2015 Asia-Pacific Network Operations and Management Symposium*.

[56] Junling Hu and Michael P Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250, 1998.

[57] Shengshan Hu, Chengjun Cai, Qian Wang, Cong Wang, Xiangyang Luo, and Kui Ren. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. In *INFOCOM'18*.

[58] Nicole Immorlica, Mohammad Mahdian, and Vahab S Mirrokni. Limitations of cross-monotonic cost-sharing schemes. *ACM Trans. on Algorithms*, 2008.

[59] Muhammad Anas Imtiaz, David Starobinski, Ari Trachtenberg, and Nabeel Younis. Churn in the bitcoin network: Characterization and impact. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 431–439. IEEE, 2019.

[60] Suhan Jiang, Xinyi Li, and Jie Wu. Hierarchical edge-cloud computing for mobile blockchain mining game. In *Proc. of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)*, volume 15, 2019.

[61] Yutao Jiao, Ping Wang, Dusit Niyato, and Zehui Xiong. Social welfare maximization auction in edge computing resource allocation for mobile blockchain. In *2018 IEEE international conference on communications*, pages 1–6, 2018.

[62] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[63] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[64] Jia Kan, Lingyi Zou, Bella Liu, and Xin Huang. Boost blockchain broadcast propagation with tree routing. In *International Conference on Smart Blockchain*, pages 77–85. Springer, 2018.

[65] Minhaj Ahmad Khan and Khaled Salah. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411, 2018.

[66] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), 2012.

[67] Robert Konrad and Stephen Pinto. Bitcoin utxo lifespan prediction. *CS229.stanford.edu*, 2015.

[68] Ron Lavi, Or Sattath, and Aviv Zohar. Redesigning bitcoin's fee market. In *The world wide web conference*, pages 2950–2956, 2019.

[69] Charles Lee. Litecoin (2011), 2011.

[70] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 919–927. Citeseer, 2015.

[71] Ruinian Li, Tianyi Song, Bo Mei, Hong Li, Xiuzhen Cheng, and Limin Sun. Blockchain for large-scale internet of things data storage and protection. *IEEE Trans. on Services Computing*, 2018.

[72] Liqing Liu, Zheng Chang, Xijuan Guo, Shiwen Mao, and Tapani Ristaniemi. Multiobjective optimization for computation offloading in fog computing. *IEEE Internet of Things Journal*, 5(1):283–294, 2017.

[73] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Joint computation offloading and content caching for wireless blockchain networks. In *2018 IEEE Conference on Computer Communications Workshops*.

[74] Qin Liu, Yuhong Guo, Jie Wu, and Guojun Wang. Effective query grouping strategy in clouds. *J. Computer Science and Technology*, 2017.

[75] Qin Liu, Chiu Chiang Tan, Jie Wu, and Guojun Wang. Cooperative private searching in clouds. *J. Parallel Distrib. Comput.*, 2012.

[76] Yang Liu, Changqiao Xu, Yufeng Zhan, Zhixin Liu, Jianfeng Guan, and Hongke Zhang. Incentive mechanism for computation offloading using edge computing: a stackelberg game approach. *Computer Networks*.

[77] Nguyen Cong Luong, Zehui Xiong, Ping Wang, and Dusit Niyato. Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach. In *2018 IEEE International Conference on Communications*, pages 1–6, 2018.

[78] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. *et al*, 2015.

[79] Malte Möser and Rainer Böhme. Trends, tips, tolls: A longitudinal study of bitcoin transaction fees. In *International conference on financial cryptography and data security*, pages 19–33. Springer, 2015.

[80] Ulfah Nadiya, Kusprasapta Mutijarsa, and Cahyo Y Rizqi. Block summarization and compression in bitcoin blockchain. In *International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–4, 2018.

[81] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[82] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[83] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

[84] Lam Pak Nian and David Lee Kuo Chuen. Introduction to bitcoin. In *Handbook of digital currency*, pages 5–30. Elsevier, 2015.

[85] Aissam Outchakoucht, ES-Samaali Hamza, and Jean Philippe Leory. Dynamic access control policy based on blockchain and machine learning for the internet of things. *International Journal of Advanced Computer Science and Applications*, 8(7):417–424, 2017.

[86] Gareth Owenson, Mo Adda, et al. Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2411–2416. IEEE, 2017.

[87] A Pinar Ozisik, Gavin Andresen, Brian N Levine, Darren Tapp, George Bissias, and Sunny Katkuri. Graphene: efficient interactive set reconciliation applied to blockchain propagation. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 303–317. ACM, 2019.

[88] Asutosh Palai, Meet Vora, and Aashaka Shah. Empowering light nodes in blockchains with block summarization. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2018.

[89] Sehyun Park, Seongwon Im, Youhwan Seol, and Jeongyeup Paek. Nodes in the bitcoin network: Comparative measurement study and survey. *IEEE Access*, 7:57009–57022, 2019.

[90] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM symposium on principles of distributed computing*, pages 315–324, 2017.

[91] Eric Rasmusen and Basil Blackwell. Games and information. *Cambridge, MA*, 15, 1994.

[92] Danda B Rawat and Amani Alshaikhi. Leveraging distributed blockchain-based scheme for wireless network virtualization with security and qos constraints. In *2018 International Conference on Computing, Networking and Communications*, pages 332–336, 2018.

[93] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–10, 2010.

[94] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[95] Lingyang Song, Dusit Niyato, Zhu Han, and Ekram Hossain. Game-theoretic resource allocation methods for device-to-device communication. *IEEE Wireless Communications*.

[96] Youming Sun, Hongxiang Shao, Xin Liu, Jian Zhang, Junfei Qiu, and Yuhua Xu. Traffic offloading in two-tier multi-mode small cell networks over unlicensed bands: A hierarchical learning framework. *TIIS*.

[97] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. 1998.

[98] Peter Tschipper. Buip010: Xtreme thinblocks. In *Bitcoin Forum (1 January 2016). https://bitco. in/forum/threads/buip010-passed-xtreme-thinblocks*, volume 774, 2016.

[99] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[100] Xiumin Wang, Xiaoming Chen, Weiwei Wu, Ning An, and Lusheng Wang. Cooperative application execution in mobile cloud computing: A stackelberg game approach. *IEEE Communications Letters*.

[101] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[102] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[103] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 2014.

[104] Qiufen Xia, Weifa Liang, Zichuan Xu, and Bingbing Zhou. Online algorithms for location-aware task offloading in two-tiered mobile cloud environments. In *Proceedings of the 2014 IEEE/ACM 7th international conference on utility and cloud computing*, pages 109–116, 2014.

[105] Liang Xiao, Caixia Xie, Tianhua Chen, Huaiyu Dai, and H Vincent Poor. A mobile offloading game against smart attacks. *IEEE Access*.

[106] Zehui Xiong, Shaohan Feng, Dusit Niyato, Ping Wang, and Zhu Han. Optimal pricing-based edge computing resource management in mobile blockchain. In *2018 IEEE International Conference on Communications*.

[107] Zehui Xiong, Yang Zhang, Dusit Niyato, Ping Wang, and Zhu Han. When mobile blockchain meets edge computing. *IEEE Communications Magazine*, 56(8):33–39, 2018.

[108] Yibin Xu. Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. In *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 115–125, 2018.

[109] Minjae Yoo and Yoojae Won. A study on the transparent price tracing system in supply chain management based on blockchain. *Sustainability*, 10(11):4037, 2018.

[110] Ian Young. Dogecoin: A brief overview & survey. *Available at SSRN 3306060*, 2018.

[111] Huaqing Zhang, Yong Xiao, Shengrong Bu, Dusit Niyato, Richard Yu, and Zhu Han. Fog computing in multi-tier data center networks: a hierarchical game approach. In *2016 IEEE international conference on communications*.

[112] Xiaonan Zhang, Linke Guo, Ming Li, and Yuguang Fang. Social-enabled data offloading via mobile participation-a game-theoretical approach. In *2016 IEEE Global Communications Conference*.

[113] Yuanyu Zhang, Shoji Kasahara, Yulong Shen, Xiaohong Jiang, and Jianxiong Wan. Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, 6(2):1594–1605, 2018.

[114] Liang Zhao, Hiroshi Nagamochi, and Toshihide Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Math Program*, 2005.

[115] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *SPW'15*.