Fairness Matters: A Tit-For-Tat Strategy Against Selfish Mining

Weijie Sun
The Hong Kong University of Science and Technology
Hong Kong SAR, China
wsunan@cse.ust.hk

Zihuan Xu
The Hong Kong University of Science and Technology
Hong Kong SAR, China
zxuav@cse.ust.hk

Lei Chen
The Hong Kong University of Science and Technology
Hong Kong SAR, China
leichen@cse.ust.hk

ABSTRACT

The proof-of-work (PoW) based blockchains are more secure nowadays since profit-oriented miners contribute more computing powers in exchange for fair revenues. This virtuous circle only works under an incentive-compatible consensus, which is found to be fragile under selfish mining attacks. Specifically, selfish miners can conceal and reveal blocks strategically to earn unfairly higher revenue compared to honest behaviors. Previous countermeasures either require incompatible modifications or fail to consider the asynchronous network and multiple honest nodes setting in reality.

In this paper, we introduce the unfairness measurement based on the KL-divergence from the computing power distribution to the revenue distribution of miners. To improve fairness with the existence of selfish miners, we propose a novel block promotion strategy namely Tit-for-Tat (TFT), for honest miners. In particular, based on a miner's local observation of forks, we design the suspicious probability measurement of other nodes. Rather than promoting a fresh block instantly, miners withhold it for different time periods according to others' suspicious probability before delivery. Meanwhile, to minimize the attacker's unfair revenue, we formulate the delay vector (DV) problem for honest miners to determine the optimal withholding time. We prove that DV problem is nonconvex, and thus propose two approximation algorithms that yield ϵ-suboptimal solutions. In addition, we extend TFT strategy to support dynamic networks. Extensive experiments validate the efficiency and effectiveness of our strategy and algorithms to reduce unfairness by 54.62% within bounded withholding time.

PVLDB Reference Format:
Weijie Sun, Zihuan Xu, and Lei Chen. Fairness Matters: A Tit-For-Tat Strategy Against Selfish Mining. PVLDB, 15(13): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/SwJay/tft.

1 INTRODUCTION

Blockchain has evolved from the original cryptocurrency ledger into an almighty platform that copes with troublesome scenarios lacking trust in fields like finance [39, 51], crowdsourcing [24], e-voting [40], etc.. To ensure system security and reliability, the consensus protocol plays an essential role. Nakamoto consensus, the de facto consensus mechanism for most permissionless chains (*e.g.*, Bitcoin [35], Ethereum [53]), adopts the *proof-of-work* (PoW) to nominate a node to append a new block every round and the *longest-chain rule* [35] to ensure the eventual consistency in an honest majority environment. In addition, *incentive mechanisms* motivate miners to join mining and increase computing power for more profits, thus contributing to the growing system hash rate that raises the attack cost and strengthens the overall security.

However, such enhanced security of Nakamoto consensus is based on a vulnerable property which is taken for granted – **fairness**. Nowadays, a growing interest in the miner's fairness of permissionless blockchain has emerged in the blockchain community and database field [12, 26, 27, 42]. Fairness assumes that on expectation, each miner has the same relative ratio of the revenue and computing power [7]. However, fairness may not hold due to the asynchronous network in reality [41], where different propagation delays among nodes will give some miners an unfair lead over others to solve PoW puzzles [12]. What's worse, such delay can be manipulated by malicious nodes to enforce a fork to partition the network, and the total computing power is dispersed and undermined. Due to the uneven distribution of nodes' wasted computing power leading to mismatched rewards, unfairness emerges.

Selfish mining is such a threat that gives attackers unfairly higher revenue yet with a budget attack cost much lower than the typical 51% attack in Nakamoto Consensus [7, 18, 42]. In particular, for a freshly mined block, honest miners promote it immediately, while the selfish attacker conceals it and continues mining on its local branch privately. The attacker will reveal withheld blocks to initiate a fork only when receiving a competing block from the honest community. First, let's consider an example:

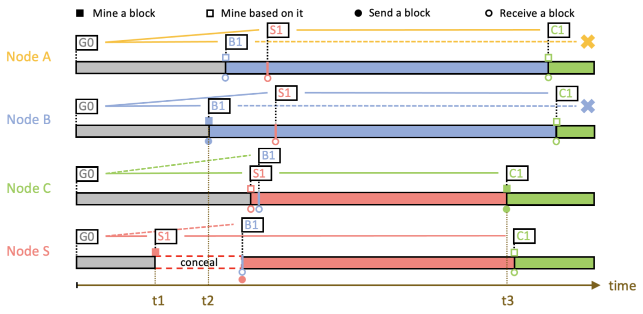


Figure 1: Example workflow for selfish mining.

Example 1.1. Fig. 1 shows three honest miners (A, B, C) and one selfish miner (S) that are fully connected. When S first mines a block S_1 at time t_1, S conceals it and continues mining on this private branch, while A, B, and C remain unaware of S_1 and still endeavor

Proceedings of the VLDB Endowment, Vol. 15, No. 13 ISSN 2150-8097.
doi:XX.XX/XXX.XX

to mine on G_0. Later at time t_2, B mines a block B_1 and promotes it immediately. Once S receives B_1, it reveals S_1 to others, resulting in a split where A and B mine on B_1, while C and S mine on S_1. Eventually, this fork tie is broken by node C who mines a higher block C_1 at time $t3$. The branch G_0-S_1-C_1 becomes the longest chain and is accepted as the main branch by the whole network. In this case, S still gets rewards from S_1 which is mined by C, while wasting A and B's computing power on B_1 in this malicious fork.

In Example 1.1, block mining becomes an unfair game since the selfish attacker doesn't disclose the information of an existing block and gets profits in two aspects. First, it can reveal private blocks strategically to cause forks, dispersing the computing power of the honest community, *s.t.* blocks from the honest community may be discarded. Second, even honest miners mining based on blocks proposed by attackers, are less likely to compete with the attacker, since the attacker has an unfair lead to mine on concealed blocks.

To design solutions for the unfairness problem originated from selfish mining, there are three challenges to overcome:

(1) **Compatibility**. Compatibility is essential to blockchain clients which operate in a decentralized manner, thus hard forks from outdated nodes are not desired.
(2) **Asynchronous network**. The network connectivity plays a vital role in selfish mining, thus the propagation delay of the underlying network cannot be ignored.
(3) **Multiple honest nodes**. Multiple honest nodes rather than a single honest party is more practical. It gives deeper insight on the impact of selfish mining, and enables partial participation in the countermeasure.

Most existing solutions to selfish mining require heavy modifications on blockchain structures which are mainly categorized into three aspects, time witness mechanisms [25, 42, 47], reward incentives [4], and fork selection rules [18, 57]. Pass [42] Solat [47] and Heilman [25] introduce time witness by adding fruits, dummy blocks, or unforgeable timestamps respectively, and Bahack [4] modifies block reward rules which all require incompatible modifications causing hard forks thus fail to address challenge 1. Eyal [18] and Zhang [57] propose novel fork resolving policies for synchronous networks which neglect challenge 2. In addition, existing works on selfish mining simply split the network into an honest majority and a selfish pool with no propagation delay [21, 36, 45]. Though these works make contributions to the theoretic game modeling, considerations for challenges 2 and 3 are still missing.

In this paper, we propose a novel solution namely *Tit-for-Tat* (TFT), with compatibility to existing blockchains to improve fairness against selfish mining in an asynchronous network with multiple honest players. Particularly, **For challenge 1**, TFT strategy targets block promotion and can be easily adapted by miners to decide promotion schedules without system incompatible modifications. **For challenge 2**, TFT strategy utilizes selfish miner's late-mover drawback and amplifies the targeted propagation delay to the suspicious attacker in an asynchronous network. **For challenge 3**, we introduce the unfairness measurement by using the KL-divergence from miners' computing power to their revenue. Besides, we propose the suspicious probability of other nodes under the n-player model to assist with the decision of promotion delay. Meanwhile, to determine the optimal delay schedules in TFT strategy, we model an optimization problem named the delay vector (DV) problem to minimize the attacker's unfair revenue. We prove that the DV problem is non-convex which is hard to find a global optimum. Thus, we propose approximation algorithms yielding ϵ-suboptimal solutions with bounded promotion delay. In addition, we extend TFT strategy for practicality in a dynamic network. Extensive experiments validate the efficiency and effectiveness of the TFT strategy to improve fairness against selfish mining.

We summarize our main contributions below:

- To the best of our knowledge, we are the first to tackle selfish mining with compatibility under an asynchronous network with multiple honest players. Specifically, we target the block promotion and propose a novel tit-for-tat strategy to undermine selfish mining and reduce unfairness.
- We formulate the delay vector (DV) problem to minimize the attacker's profit and prove its non-convexity. Thus, we propose approximation algorithms yielding ϵ-optimal solution with bounded promotion delay.
- We implement a simulator to evaluate our methods. Extensive results show that for the selfish miner with computing power $< \frac{1}{3}$ (deduced by theoretical proof), we reduce the selfish miner's revenue to 60.26% of an honest miner on average. When the attacker's computing power $\geq \frac{1}{3}$, we reduce the system unfairness by 54.62% on average.

The rest of paper is organized as follows. Sec. 2 introduces the background. Sec. 3 defines the unfairness and analyzes the effect of selfish mining. Sec. 4 proposes TFT strategy and the DV problem, followed by solutions in Sec. 5. Sec. 6 extends TFT for dynamic networks. Sec. 7 shows experiment results. We discuss related works in Sec. 8 and conclude in Sec. 9.

2 BACKGROUND

In this section, we introduce our background including the Nakamoto consensus protocol and the selfish mining attack.

2.1 Nakamoto Consensus

Nakamoto consensus is introduced to reach consensus among nodes without a trusted party [35], which consists of three components:
Leader selection policy. Whoever solves the PoW puzzle is nominated as the leader to append a block, where the difficulty is periodically updated to retain a stable block interval [35]. Thus, we can treat the overall mining as a Poisson process [13]. Generally, a miner's hash rate (normalized computing power over the network) is proportional to the probability to mine a block [26].
The longest-chain rule. The rule is to solve fork conflicts caused by the asynchronous network, requiring miners to select the longest chain as the main branch which represents the honest majority [35]. However, such security is only guaranteed when competing blocks in a fork have an equal chance to be selected. In reality, some influential nodes (*e.g.*, mining pools) have network advantages to deliver blocks [34], thus more likely to gain support from other nodes in a fork. What worse, deliberate forks make miners with strong network connectivity have higher rewards [54].
Incentive mechanism. The incentive of the Nakamoto consensus includes the reward for mining a block and transaction fees within the block. However, it lacks the incentive to other honest behaviors such as transaction propagation and block validation [3, 33]. In

consequence, although winners are supposed to publish the block expeditiously, it turns out that miners who maliciously conceal the block strategically can gain revenue higher than its fair share [18]. Such malicious behavior is known to be selfish mining.

2.2 Selfish Mining Attack

Intuitively, miners are motivated to promote a block immediately once they mine it *s.t.* it's accepted by others as soon as possible. However, selfish mining [18] shows that an *immediate promotion* (IP) is not incentive compatible. Instead, by withholding a freshly mined block and strategically revealing blocks to enforce forks, the selfish miner can gain higher rewards than its fair share. Such an attacker can control a majority of confirmed blocks with minor computing power, which severely damages the system fairness and security. Worse yet, its profitable threshold is much lower than the typical 51% attack: 25% for attackers who win half forks, and 33.3% for attackers who even never win forks. According to the data monitored on Mining Pool Stats[48], such thresholds can be easily achieved in popular blockchains like BCH, ETH, not to mention other projects. Below we illustrate the details of selfish mining.

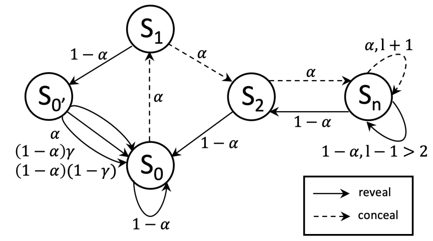


Figure 2: State machine of the selfish mining process.

Attack workflow. As studied in [36], selfish mining can be modeled as a Markov process illustrated by the state machine in Fig. 2 for an attacker with the ***computing power*** α and the ***network connectivity*** γ. In particular, α is normalized, denoting the percentage of computing power that an attacker possesses ($1-\alpha$ for other honest nodes). The network connectivity γ denotes the ratio of honest miners who mine based on the selfish miner's block during a fork. For the honest community h, the selfish miner sm with l concealed blocks ahead of h, the Markov chain works as follows:

State S_0: sm and h mines at the same height with $l = 0$. If h mines a block first, sm accepts it and stays at S_0. If sm mines a block first, sm conceals it and moves to S_1 with $l = 1$.

State S_1: sm mines on his private branch with $l = 1$ block lead. If h mines a block first, sm reveals his private block to enforce a fork immediately when informed and goes to $S_{0'}$ with $l = 0$. If sm mines a block first, sm conceals it again and goes to S_2 with $l = 2$.

State $S_{0'}$: sm and the h are mining on competing blocks in a fork. The network is split into two groups: **1)** sm and a proportion γ of honest miners h are working on sm's block. **2)** The rest $(1-\gamma)$ of honest miners h are working on the block from h. Whoever mines the next block gets reward, and sm always moves to S_0.

State S_2: sm mines on his private branch with $l = 2$ blocks lead. If h finds a block first, sm reveals two private blocks at once to beat h and goes to S_0 with $l = 0$. If sm mines a block first, sm conceals it again and goes to S_n with $l = 3$.

State S_n: sm mines on his private branch with $l > 2$ blocks lead over h. If h finds a block first, sm reveals the earliest private block to cause a fork with $l = l - 1$. If $l = 2$, sm moves to S_2, otherwise stays. If sm mines a block first, it conceals this block with $l = l + 1$.

Attack revenues. Given the states above, the probability of state transition depends on the winner selection at the current state. It follows Bernoulli distribution where sm wins with $p = \alpha$ and h wins with $p = 1-\alpha$. Denote by p_{S_i} the state probability for a certain state S_i, and the revenue for the honest community (R_h) and the selfish attacker (R_{sm}) are summarized by Eyal as follows [18]:

$$R_h = p_{S_{0'}}[\gamma(1-\alpha) + 2(1-\gamma)(1-\alpha)] + p_{S_0}(1-\alpha) \quad (1)$$

$$R_{sm} = p_{S_{0'}}[2\alpha + \gamma(1-\alpha)] + (2p_{S_2} + p_{S_n})(1-\alpha) \quad (2)$$

The stationary distribution of this chain can represent the attacker's relative revenue $r_{sm} = \frac{R_{sm}}{R_{sm}+R_h}$ determined by α, γ, and the profitable threshold $t(\gamma)$ is computed as follows [18]:

$$r_{sm} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)} \quad (3)$$

$$t(\gamma) = \inf\{\alpha | r_{sm}(\alpha, \gamma) > \alpha\} = \frac{1-\gamma}{3-2\gamma} \quad (4)$$

Two types of attackers. $t(\gamma)$ in Eq. (4) monotonically decreases from the upper bound $\frac{1}{3}$. We interpret $t(\gamma)$ for two different attackers: ***convincing attacker*** with small $\alpha < \frac{1}{3}$ but high γ, and ***powerful attacker*** with large $\alpha \geq \frac{1}{3}$. When a convincing attacker's γ is close to 1, the profitable threshold drops to 0, *i.e.*,it can invest minor computing power to profit unfairly, so long as it dominates the network propagation. A powerful attacker always gains unfairly since the α surpasses the upper bound of $t(\gamma) = \frac{1}{3}$.

With the prerequisites above, we give insights on countering both convincing and powerful attackers in Sec. 3.2 and demonstrate our detailed defense strategy in Sec. 4.

3 UNFAIRNESS

In this section, we give a formal definition of unfairness for a n-player blockchain network under Nakamoto consensus and analyze the overall unfairness caused by selfish mining.

3.1 Formal Definition of Unfairness

To quantify unfairness in blockchain, most works [16, 26, 42] follow the theory of inequity with the essence to compare perceived revenue and investment between oneself and another entity [1]. However, such a comparison implies a 2-player model, failing to capture the unfairness within the integrated reference.

Therefore, we extend the 2-player model to n-player with both *computing power distribution* $\boldsymbol{a}$ and *mining revenue distribution* $\boldsymbol{r}$ normalized among all miners. As discussed in Sec. 2.1, the single block mining process is modeled with the same probability distribution of $\boldsymbol{a}$ [26]. Therefore, we define unfairness as follows:

*Definition 3.1 (**Unfairness**).* For a blockchain network with n players, let $\boldsymbol{a} = (a_1, a_2, ..., a_n)$ be the computing power distribution and $\boldsymbol{r} = (r_1, r_2, ..., r_n)$ be the revenue distribution. Unfairness U is defined as the KL-divergence from $\boldsymbol{a}$ to $\boldsymbol{r}$:

$$U = D_{KL}(\boldsymbol{r} || \boldsymbol{a}) \quad (5)$$

In this definition, unfairness is isolated from the incentive mechanism, and purely measures the difference between the designed and measured distributions of leader selection. For a fair blockchain, we would expect the actual leader selection allocation to comply with miners' computing power distribution. Thus, the deviation of selection can be well captured by the entropy loss in KL-divergence.

3.2 Unfairness in the Selfish Mining Scenario

We then analyze how selfish mining aggravates unfairness with higher relative revenue. Based on the honest community h's revenue in Eq. (1) in Sec. 2.2, we further elaborate the revenue of every single honest miner $i \in h$ since the unfairness metric is proposed for multiple players. Specifically, there are three cases for i to profit given the state machine depicted in Fig. 2. We denote $B_{i,s}$ as the block mined by i at state S_s, and γ as the network connectivity defined in Sec. 2.2, which is the ratio of honest miners who mine on a selfish block during a fork:

(1) Node i mines a block $B_{i,0}$ at state S_0 with the probability of a_i.
(2) Node i mines a block $B_{i,0'}$ at state $S_{0'}$ with the probability of a_i. Note that, whether i mines based on an honest block $B_{h,1}$ or a selfish block $B_{sm,1}$, does not affect the reward of $B_{i,0'}$.
(3) $(1-\gamma)$ of miners in h successfully mines a block at state $S_{0'}$ where the last honest block $B_{i,1}$ mined at state S_1 from i get rewarded. Though the reward for honest block $B_{h,0'}$ in case (2) is irrelevant to the former blocks proposed at S_1, the last honest block $B_{i,1}$ from i only receives reward when the $(1-\gamma)$ of miners in h wins the fork competition. Conditioned on $S_{0'}$, the probability for i to propose the former block $B_{i,1}$ is $\frac{a_i}{1-\alpha}$.

We sum the three cases up and calculate the revenue for a single honest miner i as follows:

$$R_i = p_{S_0} a_i + p_{S_{0'}} a_i + \frac{a_i}{1-\alpha} p_{S_{0'}} (1-\gamma)(1-\alpha) = \frac{a_i}{1-\alpha} R_h$$

We can observe that the proportion of i's revenue R_i among the overall honest revenue (*i.e.*, R_h in Eq. (1)), equals to i's computing power proportion within the overall honest miners $\frac{a_i}{\sum_{j\neq sm}^{n} a_j}$. Thus, i's relative revenue r_i can be computed as $r_i = \frac{a_i}{1-\alpha} \cdot (1 - r_{sm})$ and we can compute the unfairness U from Eq. (5) as follows:

$$\begin{aligned} U = D_{KL}(\boldsymbol{r}||\boldsymbol{a}) &= r_{sm} \log \frac{r_{sm}}{\alpha} + \sum_{i=1}^{n} r_i \log \frac{r_i}{a_i} \\ &= r_{sm} \log \frac{r_{sm}}{\alpha} + (1 - r_{sm}) \log \frac{1 - r_{sm}}{1-\alpha} \qquad (6) \end{aligned}$$

Given the unfairness evaluation under selfish mining attack in Eq. (6), we further elaborate it with an example.

	CP distribution	Propagation time			
		Node A	Node B	Node C	Node S
Node A	20%	0	2	5	3
Node B	25%	2	0	6	4
Node C	25%	5	6	0	1
Node S	30%	3	4	1	0

Table 1: Computing power (CP) distribution and propagation time of the example network.

Example 3.2. Consider a fully connected network with three honest miners (A, B, C) and one selfish miner (S). Table 1 shows their computing power distribution and propagation time. We evaluate the expectation of γ as $\sum_{i\in\{V\setminus\{S\}\}} \frac{a_i}{1-\alpha} \sum_{j\in\{V\setminus\{i,S\}\}} \frac{a_j}{1-\alpha} \mathbb{1}_{\{t_{i,j} > t_{i,S,j}\}} \approx 0.4592$, where $\mathbb{1}_{\{t_{i,j} > t_{i,S,j}\}}$ indicates whether it takes shorter time to traverse from node i to j when passing node S. From Eq. (3) and Eq. (6), we compute attacker's relative revenue $r_{sm} \approx 0.3225 > \alpha$ and overall unfairness $U = 0.12\% > 0$.

Intuition to solve unfairness. We compute the first-order and second-order derivatives for unfairness U and list below:

$$\frac{\mathrm{d}U}{\mathrm{d}r_{sm}} = \log\left(\frac{1-\alpha}{\alpha} \frac{r_{sm}}{1-r_{sm}}\right), \ \frac{\mathrm{d}^2U}{\mathrm{d}r_{sm}^2} = \frac{1}{r_{sm}(1-r_{sm})} > 0$$

Therefore, U gets optimal when $\log(\frac{1-\alpha}{\alpha} \frac{r_{sm}}{1-r_{sm}}) = 0$, i.e. $r_{sm} = \alpha$. We assume the selfish miner is rational that always tries to maximize the revenue which means the attacker only performs selfish mining when it's more profitable, otherwise behaves honestly. Thus, a fair system should minimize the relative revenue r_{sm} of selfish miners.

To achieve so, the intuition is to minimize the attacker's connectivity γ. We analyze why lower γ helps to reduce r_{sm} for both *convincing attackers* and *powerful attackers* introduced in Sec. 2.2.

For convincing attackers. From Eq. (4), the profitable threshold $t(\gamma)$ is monotonically decreasing over γ. By minimizing γ, we raise the attack cost for convincing attackers higher than its profit. Thus a rational convincing attacker won't launch selfish mining anymore.

For powerful attackers. The relative revenue r_{sm} from Eq. (3) is also monotonically decreasing over γ. Hence, for a powerful attacker with computing power larger than the upper bound of $t(\gamma)$, the best we can do is directly reduce his relative revenue via lowering his network responsiveness.

To tackle unfairness caused by selfish attackers above, we formally define the problem in Sec. 4.3 and solve it in Sec. 5.

3.3 Network and Threat Model

To better capture γ under the n-player setting, a more detailed model is required to reveal the communication among nodes.

Network model. We study the honest community at a finer granularity of mining pools since they dominate computing power and provide a natural abstract of the underlying P2P network. By December 2018, Romiti analyzed that more than 80% mining shares in Bitcoin were occupied by known mining pools whose percentage was more than 4%, where top 3 or 4 pools held more than 50% of the overall computing power [43]. The largest pool in other popular blockchains like BCH, ETH, etc., can sometimes occupy more than $\frac{1}{3}$ computing power according to Mining Pool Stats. Studies on the underlying P2P network also suggest the impact from well-connected influential nodes and community structures formed in network clustering [14, 34]. Moreover, influential nodes and pools are considered to connect directly [36] thus forming a complete graph that is robust against network attacks. With the granularity fixed, we model the spatial relationship and communication delays with Göbel's assumptions [22]. We assume all miners follow the spatial Poisson point process. We assume the communication delay between any two miners follows the normal distribution with mean $k \cdot d$ proportional to their distance d, with a constant variance σ^2. Therefore we can treat the overall network as a complete graph $K_{n+1} = (V, E)$, where V is composed of n honest players $N = \{1, 2, ..., n\}$ and 1 selfish miner with $|V| = n + 1$, and the weight for each edge (u, v) in E represents the Euclidean distances between two nodes $d(u, v)$. From the perspective of our agency, a single honest miner, the structure it is aware of is a star S_n since it has no information of edges between any other two nodes.

Threat model. The strongest selfish mining comes from a single attacker who is more profitable than multiple attackers and damages the fairness most severely [6]. Therefore, without loss of generality, we adapt a solo selfish miner as our threat model. Moreover, as we've analyzed in Sec. 3.2, a lower connectivity γ suppresses the unfair profits for both convincing attacker and powerful attacker. Hence for our threat model, we consider the strongest attacker that completes block delivery in a flash *s.t.*it has the largest connectivity γ. That means once it receives a new block from the honest community, the rest uninformed honest nodes will immediately receive and mine on the selfish miner's block. It is under such a strong threat model that we can tell the effectiveness of our strategy.

4 TFT STRATEGY

In this section, we illustrate the workflow of our tit-for-tat (TFT) strategy in static networks, improving fairness against selfish mining. TFT strategy enables honest miners to delay promotions to suspicious attackers with optimized scheduling, *i.e.*, delay vector.

4.1 Tit-for-tat Strategy

As discussed in Sec. 3.2, to tackle unfairness from selfish mining, we need to paralyze the attacker's network connectivity γ, *i.e.*, the number of honest miners mining on selfish blocks in forks. To reduce γ, the insight of TFT strategy is to utilize the late-mover drawback of the selfish miner and amplify such gap by delaying promotions in a tit-for-tat manner. Selfish miner only reveals private blocks until receiving a competing honest block, thus it's always one step behind the honest community to promote blocks. Such late-mover promotion can be aggravated if its receipt of the honest block is delayed by honest miners. Specifically, TFT strategy helps the honest miner develop a well-scheduled promotion plan, namely *delay vector*, to delay block promotions to suspicious attackers. Before introducing the calculation of delay vectors, let's first check its effectiveness on countering selfish mining through an example.

	CP	**Propagation time with delay vector**			
	distribution	Node A	Node B	Node C	Node S
Node A	20%	0	$2+0$	$5+1$	$3+3$
Node B	25%	$2+0$	0	$6+1$	$4+3$
Node C	25%	$5+1$	$6+1$	0	$1+3$
Node S	30%	3	4	1	0

Table 2: Computing power (CP) distribution and propagation time with delay vector of the example network.

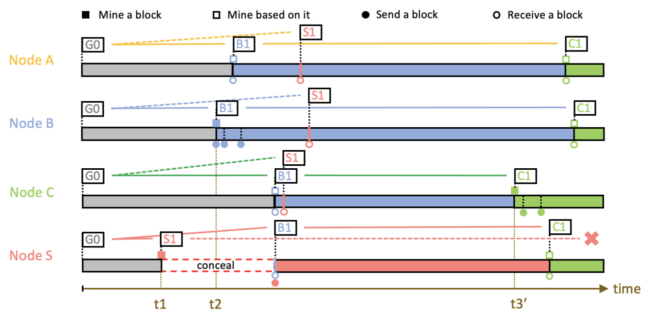


Figure 3: Example TFT strategy against selfish mining.

Example 4.1. Consider the same network in Example 3.2, but now we allow honest miners to delay their promotion following delay vectors in Table 2. For example, when B generates the block B_1 at t_2, B doesn't propagate it instantly to the whole network, but promote to A at t_2, C at t_2+1 and S at t_2+3 in Fig. 3. This strategy wins B the support from node C since it receives block B_1 earlier than block S_1. Now only S is mining on S_1 in the fork tie which is later broken by C and the longest chain $G_0-B_1-C_1$ leaves S no reward. Same as Example 3.2, we compute $\gamma=0$ and the relative revenue $r_{sm}\approx 0.2731<\alpha$, hence it's not profitable for the attacker.

Now we dive into the TFT strategy with three steps: **1)** Detect the selfish mining behavior (in Sec. 4.2). **2)** Compute the selfish mining suspicious probability of neighbor nodes (in Sec. 4.2). **3)** Promote the honest block according to optimized delay vector (in Sec. 4.3).

For step 1, the essence of selfish mining is to disperse the honest community's computing power through deliberate forks, thus fork events serve as a significant indicator [11, 17, 52].

For step 2, based on the fork events detected in step 1, we learn the latent variables behind their fork patterns to compute the suspicious probability of other nodes in Sec. 4.2.

For step 3, according to the suspicious probability, the honest miner can measure the attacker's network connectivity by the expectation of γ it perceives.To minimize γ, it's modeled as the delay vector problem in Sec. 4.3 to calculate the optimized delay scheduling. We prove it to be non-convex and propose two approximation algorithms to calculate the ϵ-suboptimal delay vector in Sec. 5.

4.2 Selfish Detection and Suspicious Probability

Selfish mining detection. Since the attacker profits by withholding its private blocks to initiate forks, the network's fork events are more frequent and severe [11]. Therefore, honest miners can easily observe abnormal fork rate, which serves as a significant indicator of selfish mining [17, 52]. Now we introduce a sliding window of width L_w to measure the fork rate in the latest L_w blocks. Every honest miner maintains a fork count vector $\boldsymbol{X}=(X1,...,X_n)\in\mathbb{Z}^n_{\geq 0}$ to record the number of competing blocks from other nodes. Whenever an honest miner receives two competing blocks from nodes i and j, it records on X_i and X_j, thus the honest miner can measure the fork rate as $pr_f=\frac{\sum_i^n X_i}{L_w}$. Once the observed fork rate is higher than *normal fork rate threshold* which is a dynamically estimated system parameter, the honest miner can determine the existence of selfish mining in progress. An example normal fork rate in Bitcoin from block height 180,000 to 190,000 is 1.69%, measured by [13].
Suspicious probability measurement. Among all those fork events recorded in $\boldsymbol{X}$, we model the number of forks caused by node i as a Binomial distributed random variable with probability θ_i, where the attacker should have the highest probability $\theta_s=\max_{i\in N}\{\theta_i\}$ to launch a fork. Given node i's observed fork count X_i, all n Binomial distributions can generate this result only at different probability, thus we represent it with latent probability $Pr[z_i|X,\boldsymbol{\theta}]$ where latent variable $z_i\in\{1,2,\ldots,n\}$ as the z_i-th distribution and $\boldsymbol{\theta}$ as the unknown parameters to be predicted. We are interested in the latent probability for each node i to be the attacker conditioned on data in the current window $Pr[z_i=s|\boldsymbol{X},\boldsymbol{\theta}]$, where $s=\arg\max_i^n\{\theta_i\}$. To compute the distribution of latent variable z and unknown parameter $\boldsymbol{\theta}$, an honest miner can apply the EM algorithm [15] with hash rate as the initial parameter estimation. Based on the converged parameter $\boldsymbol{\theta}$, we can obtain the suspicious probability $p_i=\frac{Pr[z_i=s|X,\theta]}{\sum_j^n Pr[z_j=s|X,\theta]}$, $i=1,2,...,n$.

4.3 Delay Vector Problem

Now we formulate delay vector problem for the optimal promotion delay scheduling. Here, we first define some basic notations.

Given a blockchain network K_{n+1}, what an honest miner $m \in h$ can perceive is the star subgraph $S_n(m)$ with itself as the internal node. For a perceived network $S_n(m)$, each leaf i has computing power a_i, and each edge (m, i) has a weight d_i indicating the Euclidean distance between nodes m and i. It complies with our threat model since the attacker's propagation distribution is unknown and we assume it's finished immediately for a strong attacker.

*Definition 4.2 (**Delay Vector**).* Given a perceived network $S_n(m)$, let $\boldsymbol{t} = (t_1, t_2, ..., t_n) \in \mathbb{R}^n_{\geq 0}$ be the vector of proactive delay time determined by m. After m mines a block, m will withhold this block for t_i before promoting it to node i.

With the proper delay vector $\boldsymbol{t}$, an honest miner can win extra support from the community before the attacker is informed and reacts. To determine the delay vector in an asynchronous network, we follow Göbel's work [22] to model the communication as follows.

*Definition 4.3 ((**$\boldsymbol{k}, \boldsymbol{\sigma}$**)**-Communication Condition**).* For a given constant pair (k, σ) where $k \geq 0$ and $\sigma \geq 0$, a perceived network $S_n(m)$ satisfies the (k, σ)-communication condition if for any edge (m, i), the message arrival time through this edge T_i follows normal distribution $N(k \cdot d_i + t_i, \sigma^2)$.

For simplicity, we write $k \cdot d_i + t_i$ as μ_i. Given a perceived network $S_n(m)$ satisfying (k, σ)-communication condition, since both k and d_i for every edge (m, i) are fixed, deciding the value of delay vector $\boldsymbol{t}$ is equivalent to determining $\boldsymbol{\mu}$. As for constraints, we require $\boldsymbol{\mu} \geq k \cdot \boldsymbol{d}$ for $i = 1, 2, ..., n$.

*Definition 4.4 (**Precedence Matrix**).* For a perceived network $S_n(m)$ satisfying (k, σ)-communication condition, let $D_{i,j}$ denote the difference between T_i and T_j where $i \neq j$, hence $D_{i,j} \sim N(\mu_i - \mu_j, 2\sigma^2)$. When $i = j$, $D_{i,j} = 0$. Let $\boldsymbol{P}$ be the precedence matrix where $P_{i,j}$ indicates the probability that node i receives the block from m earlier than node j,

$$P_{i,j} = P(D_{i,j} \leq 0) = \begin{cases} 0 & i = j \\ \Phi(\frac{-\mu_i + \mu_j}{\sqrt{2}\sigma}) & \text{esle} \end{cases} \tag{7}$$

Φ denotes the cumulative distribution function of standard normal distribution, $\Phi(x) = Pr[X \leq x] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{\mu^2}{2}} \mathrm{d}u$. Thus for any $1 \leq i < j \leq n$, we have $P_{i,j} + P_{j,i} = 1$.

*Definition 4.5 (**Delay Vector (DV) Problem**).* Given a perceived network $S_n(m)$ satisfying (k, σ)-communication condition, a suspicious vector p, the honest miner m wants to determine the value of $\boldsymbol{\mu}$ under certain constraints *s.t.* his measured expectation of network connectivity γ over the suspicious distribution is minimized:

$$\min \quad \gamma = \sum_{i=1}^{n} p_i \sum_{j=1}^{n} P_{i,j} \cdot \frac{a_j}{1 - a_i}, \tag{8}$$

$$\text{s.t.} \quad \boldsymbol{\mu} \geq k \cdot \boldsymbol{d} \tag{9}$$

5 DELAY VECTOR OPTIMIZATION

In this section, we first prove the DV problem is non-convex. To solve it, we study its benign property and prove it shares the lower bound with a transformed convex problem. We propose two approximation algorithms yielding ϵ-suboptimal solutions with a bounded withholding time.

5.1 Hardness Analysis

THEOREM 5.1. *The DV problem is non-convex.*

PROOF. It's an inequity constrained problem with affine inequality constraint functions Eq. (9), therefore it is the convexity of the objective function Eq. (8) that matters. For simplicity, we define weighted suspicious probability $\boldsymbol{b} = \frac{\boldsymbol{p}}{1-\boldsymbol{a}}$ and transform the objective function as follows:

$$\gamma = \Gamma(\boldsymbol{\mu}) = \sum_{i=1}^{n} b_i \sum_{j=1}^{n} P_{i,j} \cdot a_j = \boldsymbol{b}^T \boldsymbol{P} \boldsymbol{a}. \tag{10}$$

The transformed objective γ has the Hessian Matrix $\boldsymbol{H}$ with the following (i, j)th entry $\boldsymbol{H}_{i,j}$:

$$\boldsymbol{H}_{i,j} = \begin{cases} \frac{(b_i\alpha_j - b_j\alpha_i)(\mu_j - \mu_i)}{4\sqrt{\pi}\sigma^3} e^{-\frac{(\mu_j - \mu_i)^2}{4\sigma^2}} & i \neq j \\ \frac{\sum_{k=1}^{n}(b_i\alpha_k - b_k\alpha_i)(\mu_i - \mu_k)}{4\sqrt{\pi}\sigma^3} e^{-\frac{(\mu_i - \mu_k)^2}{4\sigma^2}} & i = j \end{cases}$$

Assume for any $\boldsymbol{a}$ and $\boldsymbol{b}$, the objective function is convex on domain $\mathcal{D} = \mathbb{R}^n_{\geq 0}$. Under this assumption, its Hessian matrix $\boldsymbol{H}$ is positive semidefinite and should guarantee that all of its eigenvalues are non-negative. For any given pair $(\boldsymbol{a}, \boldsymbol{b})$, consider the region $\mathcal{F} = \{\boldsymbol{\mu} \in \mathcal{D} | (\mu_i - \mu_j)(\frac{b_i}{a_i} - \frac{b_j}{a_j}) < 0\}$ where we have every $H_{i,j} > 0, i \neq j$. Compute the sum of all eigenvalues of $\boldsymbol{H}$ as follows:

$$\sum_{i=1}^{n} \lambda_i = \text{tr}(\boldsymbol{H}) = -\sum_{i=1}^{n} \sum_{j \neq i}^{n} H_{i,j} < 0.$$

The fact that the sum of eigenvalues is negative on $\mathcal{F}$ violates the assumption and completes the proof. □

5.2 Interior-point Method in Smith's Region

Basic Idea. To tackle the non-convex DV problem, we first show its optimal value $\gamma^\star$ is finite but not attained (Sec. 5.2.1). Then, we solve the ϵ-suboptimal set (Sec. 5.2.2) and prove it's contained in a convex region. Therefore, we construct a corresponding convex problem namely RDV and we show how to apply the *interior-point method* (IPM) [8, 37] with barrier functions [20] (Sec. 5.2.3). Due to space limitations, please refer to App. A for detailed proof of lemma and theorem in this section.

5.2.1 *Optimal Value.*

First, we consider the lower bound $\gamma^\star$ of the DV problem. Inspired by the weighted shortest processing time (WSPT) rule proposed by Smith to optimize a scheduling problem [46], we apply its definition of Smith's permutation to the delay vector problem.

Definition 5.2. (Smith's permutation π) Given the computing power distribution $\boldsymbol{a}$ and the weighted suspicious distribution $\boldsymbol{b}$, the Smith's permutation π satisfies $\frac{b_{\pi(i)}}{a_{\pi(i)}} < \frac{b_{\pi(i+1)}}{a_{\pi(i+1)}}$.

With Smith's permutation π, we construct the matrix $\boldsymbol{P}^\pi$ where $P^\pi_{i,j} = P_{\pi(i),\pi(j)}$. Denote $\boldsymbol{a}^\pi = \{a^\pi_1, a^\pi_2, ..., a^\pi_n\}$ and $\boldsymbol{b}^\pi = \{b^\pi_1, b^\pi_2, ..., b^\pi_n\}$ where $a^\pi_i = a_{\pi(i)}$ and $b^\pi_i = b_{\pi(i)}$ for any $i = 1, 2, ..., n$. From Def. 5.2 we have $\frac{b^\pi_i}{a^\pi_i} < \frac{b^\pi_j}{a^\pi_j}$ for any $i < j$. We can write the objective function Eq. (10) under π as $\gamma = \Gamma^\pi(\boldsymbol{P}^\pi) = \boldsymbol{b}^{\pi T} \boldsymbol{P}^\pi \boldsymbol{a}^\pi$.

Now, we consider the limit value of $\boldsymbol{p}^\pi$. First, we define a domain $\mathcal{D}_M = \{A_{i,j} \in [0, 1]; 1 \leq i, j \leq n | A_{i,j} + A_{j,i} = 1, A_{i,i} = 0, \forall i, j = 1, 2, ..., n\}$ which is a superset of the domain of $\boldsymbol{P}^\pi$. Because every P^π meets all conditions in $\mathcal{D}_M$. Thus P^π is also an element in $\mathcal{D}_M$. Then we prove in domain $\mathcal{D}_M$, when $\delta = \sum_{i=1}^{n-1} e^{\mu_{\pi(i)} - \mu_{\pi(i+1)}}$ approaches 0, we can obtain the limit value of $\boldsymbol{p}^\pi$ to be $\boldsymbol{I} \in \mathcal{D}_M$

which is a strictly upper triangular matrix with all $I_{i,j} = 1$ for $i < j$ in the following lemma:

Lemma 5.3. $\lim_{\delta \to 0} \boldsymbol{P}^{\pi} = \boldsymbol{I}$.

The semantic meaning of Lemma 5.3 is that, for any two nodes $\pi(i), \pi(j)$ where $i < j$ under Smith's permutation π, as the gap of their expected arrival times $\mu_{\pi(i)}$ and $\mu_{\pi(j)}$ grows large enough (*i.e.*, $\delta \to 0$), $P^{\pi}_{i,j}$ approaches 1 and $P^{\pi}_{j,i}$ approaches 0. In consequence, the limit value of $\boldsymbol{p}^{\pi}$ is $\boldsymbol{I}$. Then we give the theorem that the lower bound $\gamma^{\star}$ of DV problem is $\Gamma^{\pi}(\boldsymbol{I})$.

Theorem 5.4. *Given the power distribution* $\boldsymbol{a}$*, the weighted suspicious distribution* $\boldsymbol{b}$*, and their Smith's permutation* π*, the lower bound* $\gamma^{\star}$ *of DV problem is* $\Gamma^{\pi}(\boldsymbol{I}) = \sum_{i=2}^{n} a_{\pi(i)} \sum_{j=1}^{i-1} b_{\pi(j)}$.

5.2.2 ϵ-suboptimal Solution. Since the lower bound $\gamma^{\star}$ is unattainable, we discuss the region where suboptimal solutions lie in. Since the Smith's permutation π is a discrete scheduling order, we need to extend it to the set of positive real numbers $\mathbb{R}^{n}_{\geq 0}$, namely Smith's region, with the definition below:

Definition 5.5. (Smith's Region) For the given computing power distribution $\boldsymbol{a}$ and weighted suspicious distribution $\boldsymbol{b}$, Smith's region $\mathcal{S} = \{\boldsymbol{\mu} \in \mathcal{D} | (\mu_i - \mu_j)(\frac{b_i}{a_i} - \frac{b_j}{a_j}) \geq 0\}$ is a region where the order of $\{\mu_i\}$ follows the increasing order of $\{\frac{b_i}{a_i}\}$.

Def. 5.5 is to ensure $\mu_{\pi(i)} \leq \mu_{\pi(i+1)}$ for any $\boldsymbol{\mu} \in \mathcal{S}$, such that we can find the suboptimal solutions by enlarging the gap between $\mu_{\pi(i)}$ and $\mu_{\pi(j)}$ for all $1 \leq i < j \leq n$ to let $\boldsymbol{P}^{\pi}$ approach $\boldsymbol{I}$, which provides a guidance for our solution algorithm as well. Next, we give the theorem that $\mathcal{S}$ contains ϵ-suboptimal solution $\boldsymbol{\mu}^{\star}$.

Theorem 5.6. $\forall \epsilon > 0$*, the* ϵ*-suboptimal solution* $\boldsymbol{\mu}^{\star} \in \mathcal{S}$.

Algorithm 1: Interior-point Method in Smith's Region.

```
Input: computing power distribution a, weighted
       suspicious distribution b, lower bound l
1  $\pi \leftarrow$ the ascending order of $\frac{b}{a}$;
2  $A \leftarrow$ zero matrix;
3  for $i \leftarrow 1$ to $|\pi| - 1$ do
4      $A[i][\pi_i] \leftarrow -1$;
5      $A[i][\pi_{i+1}] \leftarrow 1$;
6  $S \leftarrow constrain(A, 0, \infty)$;// $\forall \mu \in S, 0 < A\mu < \infty$.
7  $B \leftarrow bound(l, \infty)$;// $\forall \mu \in B, l < \mu < \infty$.
8  $O \leftarrow objective(\Gamma|_{\mathcal{S}})$;
9  $\mu \leftarrow IPM(O, B)$;
10 return $\mu$;
```

5.2.3 Interior-point Method in Smith's Region. By theorem Theorem 5.6 that Smith's region $\mathcal{S}$ contains ϵ-suboptimal solutions, now we transform DV problem into a convex problem, namely restricted delay vector (RDV) problem, sharing the same lower bound. Based on the interior-point method (IPM), we propose IPM-S in Smith's region to find ϵ-suboptimal solutions. Specifically, RDV has a new objective function $\Gamma|_{\mathcal{S}}$, the restriction of Γ to $\mathcal{S}$: $\mathcal{S} \to \mathbb{R}, \boldsymbol{\mu} \to \Gamma(\boldsymbol{\mu})$. We give the definition of RDV and the theorem of its convexity.

Definition 5.7. (Restricted delay vector problem) Given a perceived network $S_n(h)$ under (k, σ)-communication condition, the weighted suspicious vector $\boldsymbol{b}$, an honest miner h wants to determine the value of $\boldsymbol{\mu} \in \mathcal{S}$ under certain constraints to minimize $\Gamma|_{\mathcal{S}}(\boldsymbol{\mu})$.

$$\min \quad \Gamma|_{\mathcal{S}}(\boldsymbol{\mu}), \qquad \text{s.t.} \quad \boldsymbol{\mu} \geq k \cdot \boldsymbol{d} \tag{11}$$

Theorem 5.8. *RDV problem is convex.*

To solve the convex RDV problem, we propose IPM-S as in Algo. 1. We first compute Smith's permutation π for given $\boldsymbol{a}, \boldsymbol{b}$ (line 1). With π, we construct a matrix $\boldsymbol{A}$ to *constrain* $\boldsymbol{\mu}$ satisfying $\boldsymbol{0} < \boldsymbol{A\mu} < \boldsymbol{\infty}$ that describes Smith's region S (line2-6). We define the *bound* B which $\boldsymbol{\mu}$ subjects to (line7) and set the Smith's region constrained *objective* function $\Gamma|_{\mathcal{S}}$ (line 8). Finally, we call IPM to tackle this transformed convex problem (line 9). We further illustrate IPM-S with the following example.

Example 5.9. Consider the same network in Example 3.2 but now satisfies $(1, 1)$-communication condition. Inputs for IPM-S are listed in Table 3(a) from node B's view where the weighted suspicious probability $\boldsymbol{b}$ is computed as $\boldsymbol{b} = \frac{\boldsymbol{p}}{1-\boldsymbol{a}}$. We first compute $\pi = (1, 2, 3)$ by sorting $\frac{\boldsymbol{b}}{\boldsymbol{a}}$ in ascending order (line 1). Then we compute the matrix $\boldsymbol{A}$ (line 2-5). When $i = 1$, for example, we have $A_{1,1} = -1$ and $A_{1,2} = 1$ which constrains $0 < -\mu_1 + \mu_2 < \infty$. The complete matrix $\boldsymbol{A}$ is presented in Table 3(b). Now we can construct the Smith's region S that constrains $\mu_1 < \mu_2 < \mu_3$ (line 6), bound B that complies with the topology distance (line 7), and objective function $\Gamma|_{\mathcal{S}}$ under S (line 8). Eventually we call IPM solver with error tolerance at 0.1 to get solution $\boldsymbol{\mu} \approx (3.75, 7.29, 10.14)$ (line 9).

Table 3: Tables for Example 5.9.

(a) Inputs for IPM-S.

	$\boldsymbol{a}$	$\boldsymbol{p}$	$\boldsymbol{b}$	$\boldsymbol{l}$
Node A	20%	0%	0	2
Node C	25%	10%	0.13	6
Node S	30%	90%	1.29	4

(b) Matrix $\boldsymbol{A}$.

−1	1	0
0	−1	1
0	0	0

Performance analysis of Algo. 1. The RDV problem is convex and holds Slater's condition, *s.t.* its optimum and dual optimum satisfy the Karush-Kuhn-Tucker (KKT) condition. Current interior-point methods with barrier functions to approximate the indicator of inequality constraints, implied in a rewritten objective can tackle the restricted problem. $\forall \epsilon > 0$, since there are $2n - 1$ inequality constraints including $\mathcal{S}$ restriction in the RDV problem, set the approximation accuracy as $\frac{2n-1}{\epsilon}$ and the barrier method is guaranteed to converge to an ϵ-suboptimal solution [19]. The resulting $\boldsymbol{\mu}$ is also the ϵ-suboptimal solution for the original DV problem.

5.3 Step Algorithm

The interior-point method can converge efficiently within polynomial time, however, the solution is unbounded since for any $\epsilon > 0$, the ϵ-suboptimal set for the RDV problem is unbounded. Thus, we propose an approximation algorithm, *step algorithm* (STA), that bounds the maximum delay time μ^{π}_{n} without sacrificing the tolerance ϵ. Specifically, we define $\beta = \max_{1 \leq i < j \leq n} b^{\pi}_{j} a^{\pi}_{i} - b^{\pi}_{i} a^{\pi}_{j}$. Meanwhile, for a given $\epsilon > 0$, we define the step function as

$$\tau(\epsilon) = \sqrt{2}\sigma \cdot \Phi^{-1}(1 - \frac{1}{\beta} \cdot \frac{\epsilon}{(n-1)!}). \tag{12}$$

With the step function $\tau(\epsilon)$, the Step Algorithm is described in Algo. 2. We first compute Smith's permutation π for given $\boldsymbol{a}, \boldsymbol{b}$ (line 1). $\forall\epsilon > 0$, the step size is fixed and the value of μ_1^π is irrelevant with ϵ (line 2). We construct $\boldsymbol{\mu}$ by following the order of π (line 3). At each iteration, the value for the current element in $\boldsymbol{\mu}$ is the maximum between its lower bound and last element with one step length to satisfy both bound and step constraint (line 4-5).

To further illustrate, we refer to the same network setting as Example 5.9. The same permutation $\pi = (1, 2, 3)$ is generated (line 1). Given the error tolerance at 0.1, we compute the step size $st = \tau(0.1) \approx 1.31$ (line 2). The first item $\mu[\pi_1] = \mu[1] = l[1]$, equals to its lower bound (line3). When $i = 2$, for example, $\mu[\pi_2] = \mu[2]$ should be assgined as $l[2]$ since $l[2] > \mu[1] + st$. The eventual result $\boldsymbol{\mu} \approx (2, 6, 7.31)$ (line 4-5) indicates shorter delays than IPM-S.

Algorithm 2: Step Algorithm.

```
Input: power distribution $a$, weighted suspicious
       distribution $b$, lower bound $l$, tolerance $\epsilon$
1 $\pi \leftarrow$ the ascending order of $\frac{b}{a}$;
2 $st \leftarrow \tau(\epsilon)$;
3 $\mu[\pi_1] \leftarrow l[\pi_1]$;
4 for $i \leftarrow 2$ to $|\pi|$ do
5     $\mu[\pi_i] \leftarrow \max\{l[\pi_i], \mu[\pi_{i-1}] + st\}$;
6 return $\mu$;
```

Performance analysis of Algo. 2 It's clear that for any ϵ, the maximum delay time μ_n^π for Step Algorithm is bounded by $\max\{k \cdot d_i^\pi + (n - \pi(i))\tau(\epsilon)\}$. For given $\boldsymbol{\mu}, \pi$, we construct a set $X_d(\boldsymbol{\mu}) = \{x = \mu_j^\pi - \mu_i^\pi | 1 \le i < j \le n\}$. Define the set $S(\epsilon) = \{\boldsymbol{\mu} \in \mathcal{D} | \arg\min_{x \in X_d(\boldsymbol{\mu})} \Phi(\frac{x}{\sqrt{2}\sigma}) \ge t(\epsilon)\}$. We give the theorem that for a given $\epsilon > 0$, the lower bound $t(\epsilon)$ for $x \in X_d(\mu)$ can guarantee that μ is an ϵ-suboptimal solution.

Theorem 5.10. *$\forall\epsilon > 0$, $S(\epsilon)$ is an ϵ-suboptimal set.*

Proof. $\forall\hat{\boldsymbol{\mu}} \in S(\epsilon)$, compute the difference between $\hat{\gamma}$ and $\gamma^\star$:

$$\begin{aligned}\hat{\gamma} - \gamma^\star &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (b_j^\pi a_i^\pi - b_i^\pi a_j^\pi)(I_{i,j} - \hat{P}_{i,j}) \\ &\le \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (b_j^\pi a_i^\pi - b_i^\pi a_j^\pi)(\frac{1}{\beta} \cdot \frac{\epsilon}{(n-1)!}) \le \epsilon.\end{aligned}$$

It's proved that $\forall\hat{\boldsymbol{\mu}} \in S(\epsilon)$ holds ϵ-suboptimality, thus $S(\epsilon)$ is an ϵ-suboptimal set. □

Thus the solution $\boldsymbol{\mu}$ of Step Algorithm is ϵ-suboptimal with bounded maximum delay time. Other than designating an error tolerance in the step function $\tau(\epsilon)$, STA also works with given bounds or step size. For example, one can follow the three-sigma rule of thumb to set its default value $st = 3\sigma$. That shall yield a suboptimal solution with relative error $\eta = \frac{\gamma(3\sigma)}{\gamma^\star} - 1 \le 1 - \Phi(\frac{3\sqrt{2}}{2}) \simeq 6.7\%$ which is acceptable in exchange for a fixed maximum delay.

6 PRACTICALITY IN DYNAMIC NETWORK

In this section, we handle dynamic issues, including the variation of network participants, their computing power and location.

6.1 Dynamic Issues

In reality, the blockchain network is dynamically varying accompanied by the following three cases. **1)** The node's computing power varies, influencing its mining ability. **2)** The node updates its location, resulting in different propagation delay. **3)** The node joins or leaves the network, *a.k.a. churn* [50]. An example of churn is that a node disguises its identity by modifying its IP or wallet address, which can be viewed as an old node leaves and a new node joins. These dynamic cases raise three new challenges which cannot be solved by the static version of TFT strategy.

(1) **Computing power estimation:** The node hash rate used by the static TFT strategy to compute the suspicious probability can vary from time to time.
(2) **Distance measurement:** The message propagation delay used in static TFT should be dynamically measured in real-time.
(3) **Practicality under churn:** Suspicious probability and delay vector in TFT require dynamic computation as nodes churn.

To tackle these three challenges in the dynamic scenario, we propose the dynamic TFT strategy below.

6.2 Dynamic TFT Strategy

To overcome challenge 1 and 2 we modify the sliding window in the static TFT and use round trip time to measure computing power and distance dynamically. To overcome challenge 3 we introduce three new concepts: node deprecation, cold start period and quick estimation. We next introduce these methods in detail.

Dynamic Power Estimation. To estimate the node computing power at any time τ, we modify the *sliding window* W_τ of length L_w in static TFT to record the miners of recent blocks from height $\tau - L_w$ to τ, and estimate the node computing power by the distribution of its mined blocks. For instance, suppose node i mines m_i blocks in previous L_w rounds, its computing power is estimated as $a_i = m_i / \sum_{j=1}^n m_j$. In reality, an example proper L_w is 100 blocks which is adopted by Mining Pool Stats to measure the network hash rate.

Dynamic distance measurement. To capture the latest location of pools, we measure the distance by the expected round trip time (RTT) of the message transmission. For example, messages on blocks and transactions are frequently exchanged in each round. Therefore, the honest node can estimate the latest distances with other pools by the RTTs of these messages. Such a method can cover cases of new connection establishment and location variance.

For node churn, we define the lifespan t_i of node i in the window W_τ, indicating i joins the network t_i blocks earlier. Meanwhile, if $t_i > L_w$, we set $t_i = L_w$ where i is regarded as a *static node.*

Node deprecation. When we lose the connection of a node, namely *deprecated node*, we do not compute its suspicious probability and delay vector any more. However, its fork count still contributes to the selfish mining detection, which will be detailed later.

Cold start. We introduce the *cold start* period with length $L_m < L_w$ for a newly joined node i with $t_i < L_m$, namely *immature node.* For immature nodes, we assign the delay vectors to make sure they are the last one to receive the block on expectation. Such a cold start period is for data collection of immature node for later estimation. Meanwhile, cold start can also restrict attackers, aiming to escape from the punishment by frequently disguising their identities.

Quick estimation. For a node i passing the cold start period ($t_i \in [L_m, L_w)$), namely *mature nodes*, its computing power and fork counts are roughly estimated by multiplying a ratio $\frac{L_w}{t_i}$. Specifically, consider a mature node i, its estimated hash rate is computed as $a_i = m_i \frac{L_w}{t_i} / \sum_j^n m_j \frac{L_w}{t_j}$, and estimated fork count is $X_i' = X_i \frac{L_w}{t_i}$ where X_i is node i's actual fork count within t_i blocks. While dynamic power estimation requires L_w rounds to measure a static node's hash rate and fork count, quick estimation can roughly estimate information of mature nodes after $L_m < L_w$ rounds.

We now introduce the workflow of the dynamic TFT strategy.

Dynamic TFT. The dynamic TFT works in rounds, as the sliding window W_τ moves one step forward. In each round, we first update the node distance $\boldsymbol{d}$ based on RTTs of the last round message exchange. Recall that the static TFT consists of three steps: selfish mining detection, suspicious probability measurement, and delay vector computation. Here we show how to modify these steps to address dynamic cases for *deprecated node* (who loses connection), *immature node* ($t_i > L_m$), *mature node* ($L_m \le t_i < L_w$) and *static node* ($t_i = L_w$) respectively.

- **Selfish mining detection (for all nodes)**: In each round τ, we monitor the fork rate $pr_f = \sum_i^{n'} X_i / L_w$ in W_τ, where n' is the number of nodes appeared in W_τ including deprecated nodes. Once pr_f is above the normal threshold (in Sec. 4.2), we enter the suspicious probability measurement step.
- **Suspicious probability measurement (for mature and static nodes only)**: For mature node i, we apply the quick estimation to roughly measure its estimated fork count X_i' and hash rate a_i. For static node i, we measure its fork count X_i within the sliding window W_τ and obtain its hash rate a_i through the dynamic power estimation. Given estimated hash rate and fork count, we use the EM algorithm in Sec. 4.2 to compute the suspicious probability $\boldsymbol{p}$ of mature and static nodes.
- **Delay vector computation (excluding deprecated nodes)**: Given hash rate $\boldsymbol{a}$, distance $\boldsymbol{d}$ and suspicious probability $\boldsymbol{p}$ of mature and static nodes, we use the STA algorithm in Sec. 5.3 to compute their delay vectors. Finally, immature nodes in the cold start period are arranged with the same expected block arrival time as the last one in mature and static nodes.

For other practical issues, due to space limitations, please refer to App. B for details.

7 EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate the effectiveness of the TFT strategy in countering selfish mining and improving fairness in both static and dynamic network settings.

7.1 Experiment Setting

Data set. To evaluate algorithms for the DV problem, we generate normalized inputs to simulate the distribution of node computing power and suspicious probability. In TFT strategy experiments, we generate different network graphs with random distance and power distributions to match given computing power α and attacker's connectivity γ, where the α is exact and the error for γ is bounded within 0.05. We compute γ by $\sum_{i \ne s} \frac{a_i}{1-\alpha} \sum_{j \ne i \ne s} \frac{a_j}{1-\alpha} \Phi(\frac{d_{is} - d_{ij}}{\sqrt{2}\sigma})$, where d_{ij} denotes the distance between nodes i, j. In addition, networks are generated in compliance with the actual median propagation time whose recent measurement is 8.7 seconds [12, 13].

Implementation. We implement the simulator to validate our strategy in Python with an event-driven fashion [2], *s.t.* it can capture any scale of concurrent forks and a complete selfish mining attack state machine. We list key details below.

- We consider the block generation from honest miners as events with total occurrences of 10000 for a single trial.
- We adopt the longest-chain rule to determine the main branch. Revenue distribution is computed by rolling back along the main branch *s.t.* there is no fork conflict.
- We decompose the selfish mining state machine *s.t.* it fits the honest event trigger of the simulator, while secret block mining is conducted within every single phase.

Experiment environment. We conduct experiments on an Azure server with Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz processors with 220GB memory. Data generation and simulation are implemented in a multi-process manner, while algorithm testing is single-threaded for execution time estimating. We repeat 50 times and count the mean results for nondeterministic experiments.

Compared approaches. For the DV problem, we compare our step (STA) algorithm (proposed in Sec. 5.3) with the following baselines:

- Interior-point method in Smith's region (IPM-S in Sec. 5.2).
- Differential evolution (DE), a heuristic optimization algorithm.

We adopt implementations for IPM with inequality constraints and DE from *scipy* package following classic designs [10, 49]. We compare them under different tolerance errors, controlling their performance, for a more comprehensive experiment.

For strategy validation, we compare our tit-for-tat strategy (TFT) proposed in Sec. 4 with the following baselines:

- *Instant promotion* (IP): miners promote a new block instantly.
- *Uniform tie breaking* (UTB) [18]: miners has 50% to select the block from honest community h.
- *Optimal tie breaking* (OTB) [45]: the theoretical lower bound where miners can always choose the honest block.

Parameters	Settings
Pool numbers n	8, **16**, 32, 64, 128
Maximum delay t_d	5, **10**, 15, 20, 25
Coefficient of variation CV	0.01, **0.02**, 0.04, 0.08, 0.16
Attacker's computing power α	5%, 10%, . . . , **30**%, 35%, 40%, 45%
Attacker's connectivity γ	0.05, 0.1, . . . , 0.55, 0.6, 0.65, **0.7**
Honest churn cycle c_h	10, **20**, 40, 80, 160
Attacker's disguise cycle c_{sm}	10, **20**, 40, 80, 160
Power adjustment cycle c_p	10, **20**, 40, 80, 160
Power adjustment range R_p	10%, **20**%, 30%, 40%, 50%

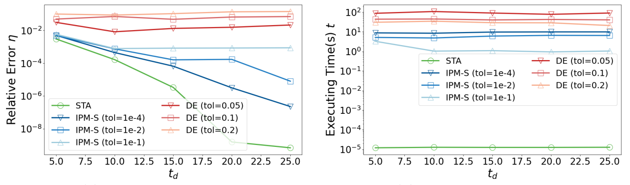
Table 4: Parameter Setting

Parameters and Metrics. Parameter settings are listed in Table 4 where default values (in bold) follow current empirical studies [12, 13, 22, 43]. Simulation experiments apply the default settings above like algorithm experiments where we adopt STA with step size determined by the three-sigma rule of thumb. In dynamic cases, we set the width of the sliding window as 100 and the mature period half its width at 50, which is consistent with the 100 blocks estimation in Mining Pool Stats. The resulting fork rate matches the empirical study in [13] while the measured attacker revenue and state probability for selfish mining also match its theoretical study [18] thus validating the implementation. Specifically, we vary parameters and measure metrics for experiments below.

- For DV algorithm, we test different network conditions by varying the pool numbers n, maximum delay t_d and coefficient of variation CV. We use executing time t and relative error η to measure the efficiency and effectiveness of DV algorithms.
- For static strategies, we vary attackers' computing power α and connectivity γ to measure it's relative ratio of revenue $\rho_{sm} = \frac{r_{sm}}{\alpha}$ for the overall performance, while the profitable threshold $t(\gamma)$ and unfairness U illustrate the effectiveness in detail. In addition, we measure the throughput and effective hash rate to evaluate the overhead cost of the TFT strategy.
- For dynamic strategies, we test different dynamic cases. *1) Peer dynamics:* a random node leaves while a new node joins the network every c_h rounds and attacker can disguise its identification every c_{sm} rounds. *2) Power dynamics:* each node changes its hash rate in the range of $[1 - R_p, 1 + R_p]$ every c_p rounds. We evaluate the attacker's relative ratio of revenue ρ_{sm} as the effectiveness of tested strategies.

7.2 Experimental Results of the DV Problem

First, we show the experimental result of the DV problem. Due to space limitations, we only show results on varying the maximum delay t_d. For the impact of pool number n and coefficient of variation CV, please refer to App. C for details.



(a) Relative error η (b) Executing time t

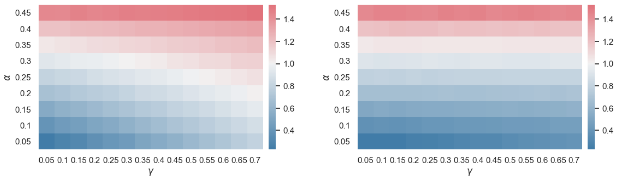
Figure 4: Impact of maximum delay t_d.

Impact of maximum delay t_d. As shown in Fig. 4(a), the relative error of STA and IPM-S drops down along with a wider search range while DE is not sensitive to t_d. For STA, a larger t_d grants it a larger step size *s.t.* the resulting ϵ-suboptimal solution can be bounded within a smaller ϵ. For IPM-S, relative error η drops as t_d increases where a larger tolerance yields smaller η except for a relaxed tolerance $tol = 0.1$. In additional, $t_d = 10$ serves as a watershed for $tol = 0.1$ where η doesn't decrease with a growing t_d. It indicates that $t_d = 10$ already provides enough search space for IPM-S to satisfy an easy termination criteria ($tol = 0.1$). For DE, a larger search space basically makes no huge difference in its performance. Fig. 4(b) denotes that the maximum delay won't affect executing time t significantly for most algorithms except for IPM-S with $tol = 0.1$. When $t_d = 5$, IPM-S ($tol = 0.1$) has t at a similar scale as IPM-S with stricter tolerances, and also has similar η as validated in Fig. 4(a). But once t_d is larger than 10 seconds, it suffices for IPM-S ($tol = 0.1$) to solve it efficiently under its error requirements hence both η and t becomes stable.

In conclusion, STA outperforms IPM-S and DE in both effectiveness and efficiency. Meanwhile, STA with a maximum delay time $t_d = 20$s can yield a solution pretty close to the optima efficiently. Thus, we adopt STA to solve DV problem in later experiments.

7.3 Experimental Results of Static Strategies

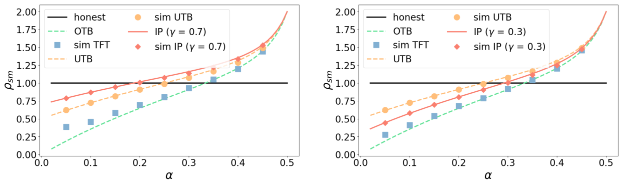
Overall performance. Fig. 5 shows the heat map of selfish miner's relative ratio of revenue ρ_{sm}. Fig. 5(a) indicates attacker's ρ_{sm} while honest miners promote blocks with default IP strategy. The results comply with the theoretical analysis Eq. (3) and validate our implementation and measurement of selfish mining. Generally, the selfish attacker with larger γ and α can gain higher ρ_{sm}. In addition, the profitable threshold $t(\gamma)$ from Eq. (4), represented by the white area in heat map Fig. 5(a), is monotonically decreasing over γ. Whereas, as Fig. 5(b) indicates, under TFT promotion strategy the higher revenue gained from large α is suppressed, and the unfair revenue from large γ is nearly evicted. Therefore, when honest miners apply TFT strategy, the attacker won't gain a large unfair revenue from his connectivity advantage. Attacker's profitable threshold is also kept at around $\frac{1}{3}$, which is elaborated in Fig. 7(a).



(a) Heat map of r_{sm} under (IP) (b) Heat map of r_{sm} under (TFT)

Figure 5: Heat map of relative ratio of revenue.

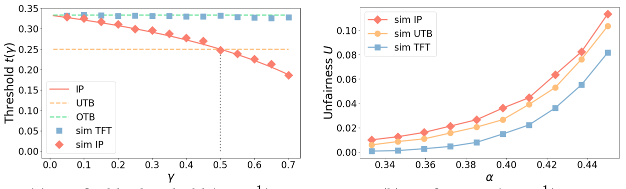
Effects on γ. Fig. 6 compares the relative ratio of revenue ρ_{sm} with different promotion and propagation strategies on both small and large γ. Fig. 6(a) illustrates ρ_{sm} for a well connected attacker ($\gamma = 0.7$). An honest miner keeps a constant relative ratio of revenue $\rho_h = 1$. The selfish miner's relative ratio ρ_{sm} is related to the honest community's promotion and propagation strategy. Results from IP strategy matches theoretical analysis Eq. (3), and the attacker gains unfair revenue when $\alpha > \frac{3}{16}$ ($\gamma = 0.7$) from Eq. (4). Theoretical propagation strategies, UTB and OTB, break the fork tie with the probability of either 50% or 100% to select the honest block. UTB alleviates the attacker's unfair revenue to some extent, and OTB gives the theoretical lower bound. Our strategy TFT can achieve a similar performance like OTB, especially for attackers with large computing power. Fig. 6(b) deals with poorly connected attacker ($\gamma = 0.3$). It clearly shows the flaw of UTB, *i.e.*, a poorly connected attacker even earns more since UTB strengthens his connectivity, while TFT strategy performs better for a less influential attacker ($\gamma = 0.3$), and is closer to the optimal lower bound.



(a) $\gamma = 0.7$ (b) $\gamma = 0.3$

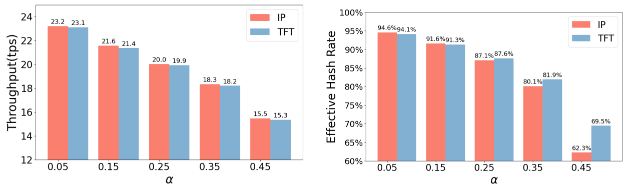
Figure 6: The relative ratio of revenue on γ.

Effects on α. Fig. 7 demonstrates the effects on α from the profitable threshold $t(\gamma)$ and overall unfairness U. On average, we decrease the revenue of convincing attacker ($\alpha < \frac{1}{3}$) to 60.26% of an honest miner, and reduce the system unfairness by 54.62% for a powerful attacker ($\alpha \geq \frac{1}{3}$) Fig. 7(a) evaluates the $t(\gamma)$ for different strategies. For IP strategy, our simulated $t(\gamma)$ matches the theoretical analysis in Eq. (4) that decreases monotonically. UTB strategy maintains the attacker's connectivity $\gamma = 50\%$ and $t(\gamma) = \frac{1}{4}$. The ideally optimal solution OTB gives the maximum $t(\gamma)$ at $\frac{1}{3}$ for all γ. Our simulated results for TFT strategy offer a similar $t(\gamma)$ as the optimal upper bound, though slightly smaller at 32.78% for a large initial $\gamma = 0.7$. Therefore, a selfish attacker with small computing power won't profit from his connectivity advantage. Rational miner assumption ensures selfish mining ($\alpha < \frac{1}{3}$) is straightly evicted. Though a powerful attacker ($\alpha \geq \frac{1}{3}$) can still profit from selfish mining, TFT decreases his revenue and improves the overall fairness. Fig. 7(b) shows the unfairness measurement for different strategies. The unfairness U rises for larger α, but both TFT and UTB decrease unfairness by a lower initial value and smaller gradients, where TFT outperforms UTB significantly.



(a) Profitable threshold ($\alpha < \frac{1}{3}$). (b) Unfairness ($\alpha \geq \frac{1}{3}$).

Figure 7: Effects on attackers with different α.



(a) Throughput (b) Effective hash rate

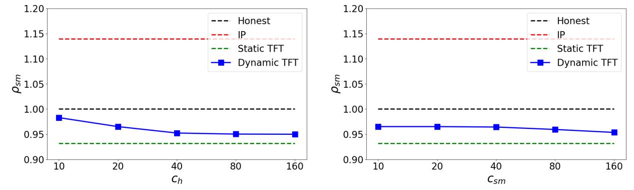
Figure 8: Overhead cost of the TFT strategy.

Overhead cost of TFT strategy. Fig. 8 displays the overhead cost by applying the TFT strategy. As shown in Fig. 8(a), with α growing the throughput under both IP and TFT strategies decrease by at most 33.41%. Because an attacker with higher hash rate is easier to perform selfish mining and withhold mined blocks, resulting in longer block delay and lower system throughput. On the other hand, TFT strategy brings extra delay in honest block promotion, leading to lower throughput than IP strategy. However, the decrease is minor (0.85% lower than IP on average) since the delay vector is controllable. As shown in Fig. 4(a), the 20s more delay time of TFT suffices to approach the minimized γ thus yielding a suboptimal r_{sm}. Such delay time is minor compared with the block interval delay (10 mins by default). Thus, TFT can effectively suppress selfish mining without sacrificing too much throughput.

Meanwhile, with higher α, the effective hash rate downgrades from 94.59% to 62.25% under IP strategy as in Fig. 8(b). Because an attacker with higher hash rate can withhold more private blocks, which aggravates the waste of honest computing power. Under TFT strategy, the effective hash rate decreases by 0.47% for small $\alpha = 5\%$ but improves by 7.2% for larger $\alpha = 45\%$ compared with IP. For larger α, there is more frequent withholding and intentional forks consuming honest power, which can be greatly saved by TFT and thus the effective hash rate is improved.

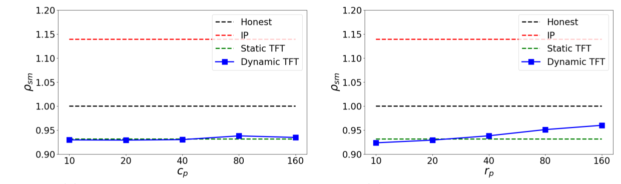
7.4 Experimental Results of Dynamic Strategies

Peer dynamics. Fig. 9 shows the attacker's relative ratio of revenue ρ_{sm} under different churn cycles. With smaller c_h, the attacker's ρ_{sm} under dynamic TFT is larger as in Fig. 9(a). Because a shorter c_h implies more honest but immature nodes are arranged at the last to receive honest blocks. Thus it increases attacker's connectivity γ and revenue r_{sm} compared to the static case. However, it still decreases attacker's ρ_{sm} by 15.66% for short $c_h = 10$ in the dynamic network. When attacker disguises its identification every c_{sm} round, its ρ_{sm} under dynamic TFT is relatively stable as in Fig. 9(b). For any c_{sm} less than the mature period L_m, such frequent disguise traps attacker in cold start period where it is arranged at the last to receive blocks. For $c_{sm} > L_m$, delay vectors to an attacker will be calculated after it survives the cold start, and the longer c_{sm} yields more accurate calculation. Therefore, whether peers join/leave frequently or rarely, dynamic TFT strategy can always reduce attackers' ρ_{sm}.



(a) Honest churn (b) Attacker disguise

Figure 9: Peer (honest/attacker) dynamics.



(a) Power adjustment cycle (b) Power adjustment range

Figure 10: Network power dynamics.

Power adjustment. When each node's hash rate varies dynamically, the sliding window based hash rate measurement is evaluated by ρ_{sm} in Fig. 10. With varying power adjustment cycles c_p, the attacker has relatively stable ρ_{sm} under dynamic TFT as in Fig. 10(a). Though power varies more frequently with shorter c_p, such modification is stable under the fixed adjustment range $R_p = 20\%$. As the adjustment range R_p grows, the attacker's ρ_{sm} under dynamic TFT increases as in Fig. 10(b). Given a fixed update cycle $c_p = 20$, the network's computing power varies dramatically with a larger R_p. Thus, the computing power and delay vector estimated from the sliding window is less accurate, resulting in attacker's better connectivity and higher ρ_{sm}. However, the TFT strategy is still effective for power dynamics, and decreases the attacker's ρ_{sm} by 17.92% with a large $R_p = 50\%$.

Summary. In static cases, the TFT promotion strategy outperforms both IP strategy [35] and UTB strategy [18]. It lifts the attacker's profitable threshold $t(\gamma)$ close to theoretical upper bound ($\alpha = \frac{1}{3}$) and decreases unfairness U by 54.62% for a powerful attacker. Compared to IP, the extra delay in TFT merely decreases throughput by 0.85%, and even increases 7.2% effective hash rate instead. For dynamic cases, TFT strategy is still effective and decreases the attacker's ρ_{sm} by 15.66% under frequent churn. In conclusion, TFT strategy resists the selfish mining attack and improves fairness effectively in both static and dynamic cases.

8 RELATED WORKS

Fairness in Blockchain. There is growing interest in miner fairness of permissionless blockchain [12, 26, 27, 42]. Existing works quantify fairness from three perspectives: *1) Malicious Attack:* Bitcoin-NG treats mining as a state machine and defines fairness as the ratio between transition rate and computing power of the network other than the largest miner [16]. Fruit chain defines fairness as any fraction coalition of honest community gains revenue at least the same as their computing power ratio [42]. *2) Propagation Delay:* Croman claimed that fairness is harmed by block propagation delay since top 10% miners may receive a block much earlier than the tail [12]. Kanda proposed the effective hash rate that incorporates both computing power and propagation distribution *s.t.* fairness is measured by the difference between effective hash rate and actual hash rate [27]. *3) Incentive Design:* Huang and Tang defined expectational fairness as one miner's expected revenue ratio equals to resource ratio, and proposed robust fairness to measure the confidence region of revenue [26].

However, the above works either merely aims at a single cause of unfairness or is limited within a 2-player scenario. In contrast, our measurement provides a quantified analysis that captures the network's overall fairness with multiple players.

Like [26], our definition of unfairness $U = D_{KL}(\boldsymbol{r}||\boldsymbol{a})$ in Sec. 3 can be extended to PoS chains by replacing hash rate $\boldsymbol{a}$ with stakes $\boldsymbol{s}$, which is the initial investment in PoS. For the PoS selfish mining, we can still detect attacker by frequent orphan blocks [38] and compute their suspicious probability. To undermine its profits from deliberate forks [9], similarly, we can punish it by delaying promotions.

Selfish Mining. Existing works detect selfish mining by supplementary indicators [25, 44] and intrinsic indicators [11, 52]. Heilman [25] added unforgeable timestamp to each block, then attacker's blocks can be detected due to withholding timeout. Saad [44] appended expected confirmation height to each transaction, thus a block's mean confirmation height is its time witness. While those supplementary indicators require structure modifications, other works utilize fork events for detection. Chicarino [11] studied different fork heights in selfish mining and showed high accuracy of the fork indicator. Wang [52] applies machine learning algorithms on fork samples to predicate selfish blocks in an offline manner. However, their works aim at improving accuracy of selfish block prediction and fail to incorporate it with the miner distribution for attacker suspect measurement, which we utilize for targeted delay.

Most selfish mining countermeasures focus on three directions: *1) Mining Incentive:* Bahack [4] changed the block rewards *s.t.* whoever attaches fork evidence get rewarded from forked blocks to disincentivize malicious forks by selfish miners. However, a new incentive can't be accepted by currently operating systems and has other potential flaws [57]. *2) Time Witness:* Heilman [25] adopted unforgeable timestamps to denote a block's actual age, yet it relied on a third party for a random beacon source. Without introducing external trust, Solat [47] witnessed time by generating dummy blocks after a timeout, while Pass [42] decoupled incentives from consensus safety by mining two different blocks embedded mutually in time. However, those implementations forced backward-incompatible upgrades on miners which would lead to hard forks. *3) Fork Selection:* Zhang [57] switched to a weight-based policy by incorporating uncle blocks. Though extra blocks are not involved to guarantee compatibility, it may still suffer from inconsistency from old clients which is especially vulnerable in an asynchronous network. Uniform tie breaking rule [18] suggested miners select competing blocks uniformly to maintain γ at 50%, which is incentive incompatible for rational miners to discard efforts already invested on the first block.

Those defenses didn't utilize the attacker's intrinsic weakness which is covered by the synchronous assumption, that it's a reactive attack always one step behind honest miners. We consider asynchronous networks and propose targeted promotion delays to paralyze the attacker's responsive speed, which performs closely to the theoretical optimal tie breaking rule without incompatible modifications. To our best knowledge, we are the first to target the block promotion process and counter selfish attacks by promotion delay. In contrary to existing works on selfish mining attack with multiple attackers [5, 29–31, 58, 59], we study the countermeasure in a network with multiple honest pools rather than a single party. By decomposing the honest party, we model attacker's connectivity and propagation delay more practically at a finer granularity even the honest part adopts a sharding organization [23, 28, 32, 55, 56]. In addition, solutions with single honest party model may fail if the individual honest miner doesn't follow the strategy. However, our solution with multiple honest miner model ensures that even some nodes don't utilize TFT, those nodes who use it can still benefit from winning forks caused by attacker. In conclusion, our solution enhances the system security by decreasing the risk of adversarial control of blockchain, and protects data reliability and integrity by preventing the monopoly due to unfairness [26].

9 CONCLUSION

In this paper, we target the unfairness issue caused by selfish mining attacks in PoW-based blockchains. To measure the global unfairness, we introduce KL-divergence from the computing power to the revenue of all miners. To reduce the attacker's unfair revenue, we propose the *TFT* promotion strategy utilizing suspicious probabilities and delay vectors. For the optimal performance of *TFT* strategy, we formulate the delay vector (DV) problem and prove it to be non-convex. To tackle that, we propose two approximation algorithms that generate ϵ-suboptimal solutions at given error tolerance. Extensive experiments have been conducted where both efficiency and effectiveness of our algorithm outperform the heuristic method. Our strategy also beats existing promotion and propagation strategies and performs close to the theoretically optimal effects.

ACKNOWLEDGMENTS

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16213620, CRF Project C6030-18G, C1031-18G, C5026-18G, CRF C2004-21GF, AOE Project AoE/E603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST).

REFERENCES

[1] J Stacy Adams. 1963. Towards an understanding of inequity. *The journal of abnormal and social psychology* 67, 5 (1963), 422.

[2] Yusuke Aoki, Kai Otsuki, Takeshi Kaneko, Ryohei Banno, and Kazuyuki Shudo. 2019. Simblock: A blockchain network simulator. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 325–329.

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

[4] Lear Bahack. 2013. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013* (2013).

[5] Qianlan Bai, Yuedong Xu, Nianyi Liu, and Xin Wang. 2021. Blockchain Mining with Multiple Selfish Miners. *arXiv preprint arXiv:2112.10454* (2021).

[6] Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, and Qingsheng Kong. 2019. A deep dive into blockchain selfish mining. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[7] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy*. IEEE, 104–121.

[8] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[9] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. 2019. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. 459–473.

[10] Richard H Byrd, Mary E Hribar, and Jorge Nocedal. 1999. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization* 9, 4 (1999), 877–900.

[11] Vanessa Chicarino, Célio Albuquerque, Emanuel Jesus, and Antonio Rocha. 2020. On the detection of selfish mining and stalker attacks in blockchain networks. *Annals of Telecommunications* (2020), 1–10.

[12] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün Sirer. 2016. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*. Springer, 106–125.

[13] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*. IEEE, 1–10.

[14] Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. 2019. Txprobe: Discovering bitcoin's network topology using orphan transactions. In *International Conference on Financial Cryptography and Data Security*. Springer, 550–566.

[15] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

[16] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*. 45–59.

[17] Ittay Eyal and Emin G Sirer. 2014. *How to Detect Selfish Miners*. Cornell University. Retrieved 2021-09-14 from https://hackingdistributed.com/2014/01/15/detecting-selfish-mining/

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

[19] Anthony V Fiacco and Garth P McCormick. 1964. The sequential unconstrained minimization technique for nonlinear programing, a primal-dual method. *Management Science* 10, 2 (1964), 360–366.

[20] Anthony V Fiacco and Garth P McCormick. 1990. *Nonlinear programming: sequential unconstrained minimization techniques*. SIAM.

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

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

[23] Siyuan Han, Zihuan Xu, and Lei Chen. 2018. Jupiter: a blockchain platform for mobile devices. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1649–1652.

[24] Siyuan Han, Zihuan Xu, Yuxiang Zeng, and Lei Chen. 2019. Fluid: A blockchain based framework for crowdsourcing. In *Proceedings of the 2019 international conference on management of data*. 1921–1924.

[25] Ethan Heilman. 2014. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*. Springer, 161–162.

[26] Yuming Huang, Jing Tang, Qianhao Cong, Andrew Lim, and Jianliang Xu. 2021. Do the Rich Get Richer? Fairness Analysis for Blockchain Incentives. In *Proceedings of the 2021 International Conference on Management of Data*. 790–803.

[27] Reiki Kanda and Kazuyuki Shudo. 2020. Block interval adjustment toward fair proof-of-work blockchains. In *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 1–6.

[28] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 583–598.

[29] Tin Leelavimolsilp, Viet Nguyen, Sebastian Stein, and Long Tran-Thanh. 2019. Selfish mining in proof-of-work blockchain with multiple miners: An empirical evaluation. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 219–234.

[30] Tin Leelavimolsilp, Long Tran-Thanh, and Sebastian Stein. 2018. On the preliminary investigation of selfish mining strategy with multiple selfish miners. *arXiv preprint arXiv:1802.02218* (2018).

[31] Quan-Lin Li, Yan-Xia Chang, and Chi Zhang. 2022. Tree Representation, Growth Rate of Blockchain and Reward Allocation in Ethereum with Multiple Mining Pools. *arXiv preprint arXiv:2201.10087* (2022).

[32] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 17–30.

[33] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. 2015. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 706–719.

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

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

[36] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 305–320.

[37] Yurii Nesterov and Arkadii Nemirovskii. 1994. *Interior-point polynomial algorithms in convex programming*. SIAM.

[38] Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. 2019. Selfish behavior in the tezos proof-of-stake protocol. *arXiv preprint arXiv:1912.02954* (2019).

[39] Wangze Ni, Peng Cheng, and Lei Chen. 2022. Mixing Transactions with Arbitrary Values on Blockchains. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2602–2614.

[40] Wangze Ni, Peng Cheng, Lei Chen, and Xuemin Lin. 2021. When the Recursive Diversity Anonymity Meets the Ring Signature. In *Proceedings of the 2021 International Conference on Management of Data*. 1359–1371.

[41] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.

[42] Rafael Pass and Elaine Shi. 2017. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. 315–324.

[43] Matteo Romiti, Aljosha Judmayer, Alexei Zamyatin, and Bernhard Haslhofer. 2019. A deep dive into bitcoin mining pools: An empirical analysis of mining shares. *arXiv preprint arXiv:1905.05999* (2019).

[44] Muhammad Saad, Laurent Njilla, Charles Kamhoua, and Aziz Mohaisen. 2019. Countering selfish mining in blockchains. In *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 360–364.

[45] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 515–532.

[46] Wayne E Smith. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3, 1-2 (1956), 59–66.

[47] Siamak Solat and Maria Potop-Butucaru. 2016. Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. *arXiv preprint arXiv:1605.02435* (2016).

[48] Mining Pool Stats. 2022. https://miningpoolstats.stream.

[49] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

[50] Daniel Stutzbach and Reza Rejaie. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. 189–202.

[51] Philip Treleaven, Richard Gendal Brown, and Danny Yang. 2017. Blockchain technology in finance. *Computer* 50, 9 (2017), 14–17.

[52] Zhaojie Wang, Qingzhe Lv, Zhaobo Lu, Yilei Wang, and Shengjie Yue. 2021. ForkDec: Accurate Detection for Selfish Mining Attacks. *Security and Communication Networks* 2021 (2021).

[53] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.
[54] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. 2020. Modeling the impact of network connectivity on consensus security of proof-of-work blockchain. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1648–1657.
[55] Zihuan Xu, Siyuan Han, and Lei Chen. 2018. CUB, a consensus unit-based storage scheme for blockchain system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 173–184.
[56] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 931–948.
[57] Ren Zhang and Bart Preneel. 2017. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*. Springer, 277–292.
[58] Shiquan Zhang, Kaiwen Zhang, and Bettina Kemme. 2020. A simulation-based analysis of multiplayer selfish mining. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–5.
[59] Shiquan Zhang, Kaiwen Zhang, and Bettina Kemme. 2020. Analysing the benefit of selfish mining with multiple players. In *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 36–44.

A PROOF

A.1 Proof of Lemma 5.3

Proof. Denote $x_{i,j} = \mu_{\pi(j)} - \mu_{\pi(i)}$ as the difference between $\mu_{\pi(j)}$ and $\mu_{\pi(i)}$, thus δ can be represented as $\sum_{i=1}^{n-1} e^{-x_{i,i+1}}$. Then, for any $i = 1, 2, ..., n-1$, we have the lower bound $x_{i,i+1} > \log \frac{1}{\delta}$. Not to lose generality, we consider for $1 \le i < j \le n$ the upper bound and lower bound of entry $P^{\pi}_{i,j}$:

$$1 > P^{\pi}_{i,j} = \Phi(\frac{\sum_{k=i}^{j-1} x_{k,k+1}}{\sqrt{2}\sigma}) > \Phi(\frac{\log \frac{1}{\delta}}{\sqrt{2}\sigma})$$

Since the upper bound is constant and the limitation for lower bound $\lim_{\delta\to 0} \Phi(\frac{\log \frac{1}{\delta}}{\sqrt{2}\sigma}) = 1$, by squeeze theorem we have $lim_{\delta\to 0} P_{i,j} = 1$ and $\lim_{\delta\to 0} P^{\pi}_{j,i} = 0$ for $1 \le i < j \le n$. In addition, we know that $P^{\pi}_{i,i} = 0$ for any $i = 1, 2, ..., n$. Therefore, under the given uniform convergence condition, it's sufficient to conclude that $\lim_{\delta\to 0} \boldsymbol{P}^{\pi} = \boldsymbol{I}$. □

A.2 Proof of Theorem 5.4

Proof. For any matrix $\boldsymbol{M} \in \mathcal{D}_M$, we can compute the difference between $\Gamma^{\pi}_M(\boldsymbol{I})$ and $\Gamma^{\pi}_M(\boldsymbol{M})$ as follows

$$\begin{aligned}\Gamma^{\pi}_M(\boldsymbol{I}) - \Gamma^{\pi}_M(\boldsymbol{M}) &= \sum_{i=1}^{n}\sum_{j=1}^{n} b^{\pi}_i a^{\pi}_j (I_{i,j} - M_{i,j}) \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (I_{i,j} - M_{i,j})(b^{\pi}_i a^{\pi}_j - b^{\pi}_j a^{\pi}_i) \le 0\end{aligned}$$

Thus, the minimum of $\Gamma^{\pi}_M(\boldsymbol{M})$ on $\mathcal{D}_M$ equals to $\Gamma^{\pi}_M(\boldsymbol{I})$. Since $\mathcal{D}^{\pi}_P \subset \mathcal{D}_M$, the minimum $\Gamma^{\pi}_M(\boldsymbol{I})$ on $\mathcal{D}_M$ gives the lower bound of the objective γ of DV problem. Besides, with Lemma 5.3, the optimal value $\gamma^{\star} = \lim_{\delta\to 0} \boldsymbol{b}^{\pi T} \boldsymbol{P}^{\pi} \boldsymbol{a}^{\pi}$ equals to the lower bound $\Gamma^{\pi}_M(\boldsymbol{I}) = \boldsymbol{b}^{\pi T} \boldsymbol{I} \boldsymbol{a}^{\pi} = \sum_{i=2}^{n} a_{\pi(i)} \sum_{j=1}^{i-1} b_{\pi(j)}$ which can't be attained on $\mathcal{D}^{\pi}_P$. □

A.3 Proof of Theorem 5.6

Proof. We construct a feasible solution in Smith's region $\hat{\boldsymbol{\mu}} \in \mathcal{S}$ with $\hat{\gamma} = \Gamma^{\pi}(\hat{\boldsymbol{\mu}}^{\pi})$ For any $\epsilon > 0$, we let $\hat{\mu}^{\pi}_j - \hat{\mu}'^{\pi}_i \ge \sqrt{2}\sigma \cdot \Phi^{-1}(1 - \frac{1}{b^{\pi}_j a^{\pi}_i - b^{\pi}_i a^{\pi}_j} \cdot \frac{\epsilon}{(n-1)!})$ for all $1 \le i < j \le n$. We prove that $\hat{\boldsymbol{\mu}}$ is an ϵ-suboptimal solution of the DV problem.

$$\begin{aligned}\hat{\gamma} - \gamma^{\star} &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (b^{\pi}_j a^{\pi}_i - b^{\pi}_i a^{\pi}_j)(I_{i,j} - \hat{P}_{i,j}) \\ &\le \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (b^{\pi}_j a^{\pi}_i - b^{\pi}_i a^{\pi}_j)(\frac{1}{b^{\pi}_j a^{\pi}_i - b^{\pi}_i a^{\pi}_j} \cdot \frac{\epsilon}{(n-1)!}) = \epsilon.\end{aligned}$$

Therefore, $\hat{\gamma} \le \gamma^{\star} + \epsilon$ for any $\epsilon > 0$, *s.t.* $\hat{\boldsymbol{\mu}}$ is an ϵ-suboptimal solution that lies in Smith's region $\mathcal{S}$. □

A.4 Proof of Theorem 5.8

Proof. Since the domain $\mathcal{S}$ is a cone and inequality constraints are affine, we focus on the convexity of the objective $\Gamma|_{\mathcal{S}}$.

First we prove the convexity of the restriction of $F(x) = \Phi(\frac{x}{\sqrt{2}\sigma})$ to $\mathbb{R}_{\le 0}$, *i.e.*, the function $F|_{\mathbb{R}_{\le 0}}(x)$. Denote $x_{i,j} = \mu_{\pi(j)} - \mu_{\pi(i)}$ as the difference between $\mu_{\pi(j)}$ and $\mu_{\pi(i)}$. $\forall \boldsymbol{\mu} \in \mathcal{S}$, we have $x_{j,i} \in \mathbb{R}_{\le 0}$ and $F|_{\mathbb{R}_{\le 0}}(x_{j,i}) = P_{j,i}$ for any $1 \le i < j \le n$. We prove its secondary derivative is non-negative:

$$\nabla^2 F|_{\mathbb{R}_{\le 0}}(x) = -\frac{x}{4\sigma^3\sqrt{\pi}} e^{-\frac{x^2}{4\sigma^2}} \ge 0.$$

Thus, $F|_{\mathbb{R}_{\le 0}}(x)$ is convex on its domain $\mathbb{R}_{\le 0}$. Moreover, we prove that $\Gamma|_{\mathcal{S}}$ can be transformed as the non-negative weighted sums of $F|_{\mathbb{R}_{\le 0}}$ plus a constant

$$\begin{aligned}\Gamma|_{\mathcal{S}}(\boldsymbol{\mu}) &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (b^{\pi}_i a^{\pi}_j P^{\pi}_{i,j} + b^{\pi}_j a^{\pi}_i P^{\pi}_{j,i}) \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (b^{\pi}_j a^{\pi}_i - b^{\pi}_i a^{\pi}_j) F|_{\mathbb{R}_{\le 0}}(x_{j,i}) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} b^{\pi}_i a^{\pi}_j\end{aligned}$$

Since the non-negative weighted sum operation preserves convexity, we conclude that the new objective function in Eq. (11) is convex, and the RDV problem is convex. □

B OTHER PRACTICAL ISSUES

Spy infiltration. Spy infiltration refers to a mining pool spreading some of its resources to other pools and expose their secret blocks. A selfish mining pool may send spy to honest pools who utilize TFT strategy, and get the leaked information of their blocks. However, the spy infiltration has already been properly considered in the selfish mining scenario [18], where honest pool may send spy to attacker. Thus, honest nodes can conduct a similar countermeasure when the selfish mining attacker performs spy infiltration.

As in [18], the infiltrated pool can selectively reveal blocks to subsets of its miners, then it can distinguish the spy through intersections and expel the node leaking information. Specifically, if there are n miners within the honest pool, its manager can divide them into m subsets, each containing $\frac{n}{m}$ members (assume dividable for simplicity). When a block is mined in the pool, the manager can selectively reveal it to m subsets with different informing time. Since the attacker can be quickly informed by the spy and the attacker can quickly promote its block, with enough observation the manager can identify the specific subset whose informing time is closely followed by the promotion of selfish block. To find the specific spy node among $\frac{n}{m}$ nodes in the targeted subset, the manager requires at least $\lceil \log_m n \rceil$ different division plans which should be applied in turn. Then, the pool manager can identify the common members in these targeted subsets through intersections. Finally, the spy node leaking information can be expelled.

For example, consider an honest pool with $n = 8$ miners numbered as $\{0, 1, 2, \ldots, 7\}$, the manager divides them into $m = 2$ subsets. To distinguish the spy node, the manager could prepare $\log_m n = 3$ different division plans according to the 1st, 2nd, 3rd bit in the binary form of their numbers correspondingly. Assume miner 5 with binary number 101 is the spy node, then the manager can distinguish its identity by intersecting three targeted subsets after observation: subset-1 in 1st division, subset-0 in 2nd division and subset-1 in 3rd division.

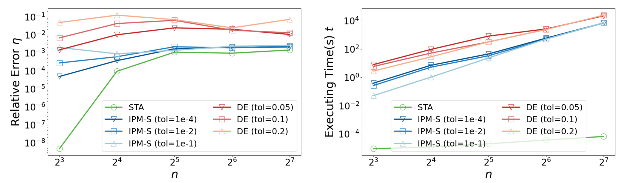
The parameter setting of m can be determined by the manager, where smaller m yields more division plans but larger accuracy to target the subset. With this solution, an honest pool can defend the attacker with both selfish mining and spy infiltration.

Semi-honest node. Semi-honest node refers to a mining pool who behaves honestly in communication with others but colludes with the attacker and leaks the latest honest blocks. The semi-honest node is beyond our multiple honest nodes model where honest nodes can either follow TFT or not and behave honestly without collusion. For rational honest miners, they will apply our strategy since TFT can improve its revenue decreased by the attack.

In addition, the requirement for semi-honest node is strict under TFT strategy: **1)** Semi-honest nodes with small computing power are ranked with low priority in DV problem and longer delay in consequence. **2)** Nodes that frequently join and leave will be arranged as the last in dynamic TFT. Therefore, the attacker can't be informed of newly mined blocks in time by including nodes with minor hash rate or frequently disguising their identities.

In consequences, the selfish attacker has to collude with qualified semi-honest node who has large power and is willing to take the risk to be identified without frequent identity disguise, which can be pretty costly. However, we can still counter this qualified semi-honest node with a similar solution as spy infiltration. The basic idea is that the honest pool can divide other nodes into different subsets and selectively reveal blocks. It can identify the semi-honest node through intersections and cut off the connection.

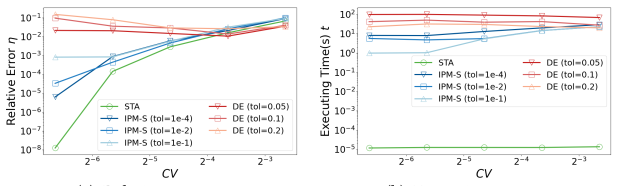
C EXPERIMENTAL RESULTS OF DV PROBLEM



(a) Relative error η. (b) Executing time t.

Figure 11: Impact of pool number n.

Impact of pool number n. Fig. 11(a) shows both IPM-S and STA converge to a worse η as n grows, while DE performs worst when n is around $16 - 32$. Under a fixed maximum delay t_d, the step size for STA decreases linearly along with a larger n, thus producing a worse result. However, while small-scale networks grant STA sufficient step size that guarantees much better performance, the shrinking step size in a large-scale network results in a worse η but with a gentle slope since the range of arrival time is fixed. For IPM-S, different error tolerance are distinguishable under a relatively small n, and they all converge to a higher η when n grows. The converged higher η comes from the bounded maximum search range under a relatively small t_d compared with the exponentially growing dimensions for DV problem. For DE, the outlier when $n = 128$ with relaxed error tolerance ($tol = 0.2$) indicates the poor performance of heuristic methods on large dimensions. Fig. 4(b) illustrates the executing time for those algorithms. It's clear that all algorithms are polynomial-time algorithms of n from the linear curve where both axes are in exponential scales, and STA is most efficient while DE performs slowest.



(a) Relative error η. (b) Executing time t.

Figure 12: Impact of coefficient of variance CV.

Impact of coefficient of variance CV. As shown in Fig. 12(a), three approaches all converge as CV grows no matter their error tolerance, where the relative error for STA and IPM-S increases and DE decreases. CV depicts the shape of the propagation distribution, where the block arrival time varies dramatically for a larger CV. Hence, even both STA and IPM-S guarantee solutions within Smith's region and the default maximum delay suffices to approximate adequate suboptimal γ as shown in Fig. 4(a), an intensively varying propagation network ($CV = 0.16$) produces unpredictable block arrival time thus algorithms on DV problem perform poorly. However, STA still serves as the lower bound of IPM-S even with a large CV. Fig. 12(b) shows that the efficiency of most algorithms doesn't depend on CV, except for IPM-S with a higher tolerance. Since a small CV (≤ 0.02) gives a steep gradient for the precedence matrix, IPM-S is efficient to converge within a relaxed error tolerance ($tol = 0.1$). However, when it becomes more variable ($CV > 0.02$), the resulting gentle slope requires more iterations to fall in the suboptimal region under the same error tolerance.