Towards Overcoming the Undercutting Problem

Tiantian Gong[1], Mohsen Minaei[2⋆], Wenhai Sun[1], and Aniket Kate[1]

[1] Purdue University, West Lafayette, USA,
{gong146, sun841, aniket}@purdue.edu
[2] Visa Research, Palo Alto, USA,
mominaei@visa.com

Abstract. Mining processes of Bitcoin and similar cryptocurrencies are currently incentivized with voluntary transaction fees and fixed block rewards which will halve gradually to zero. In the setting where optional and arbitrary transaction fee becomes the prominent/remaining incentive, Carlsten et al. [CCS 2016] find that an undercutting attack can become the equilibrium strategy for miners. In undercutting, the attacker deliberately forks an existing chain by leaving wealthy transactions unclaimed to attract petty complaint miners to its fork. We observe that two simplifying assumptions in [CCS 2016] of fees arriving at fixed rates and miners collecting *all* accumulated fees regardless of block size limit are often infeasible in practice and find that they are inaccurately inflating profitability of undercutting. Studying Bitcoin and Monero blockchain data, we find that the fees deliberately left out by an undercutter may not be attractive to other miners (hence to the attacker itself): the deliberately left out transactions may not fit into a new block without "squeezing out" some other to-be transactions, and thus claimable fees in the next round cannot be raised arbitrarily.
This work views undercutting and shifting among chains rationally as mining strategies of rational miners. We model profitability of undercutting strategy with block size limit present, which bounds the claimable fees in a round and gives rise to a pending (cushion) transaction set. In the proposed model, we first identify the conditions necessary to make undercutting profitable. We then present an easy-to-deploy defense against undercutting by selectively assembling transactions into the new block to invalidate the identified conditions. Indeed, under a typical setting with undercutters present, applying this avoidance technique is a Nash Equilibrium. Finally, we complement the above analytical results with an experimental analysis using both artificial data of normally distributed fee rates and actual transactions in Bitcoin and Monero.

1 Introduction

Bitcoin network [20] and several cryptocurrencies rely on nodes participating in transaction verification, ordering and execution, and mining new blocks for their security and performance. Specifically, with honest majority, Byzantine-fault

⋆ Part of this work was done while the author was at Purdue University.

tolerant consensus is possible with Proof of Work (PoW) assuming network synchrony. With honest majority, attacks like double spending [24] are also harder to implement in practice. Additionally, with more computing peers, liveness is provided with higher probability. A proper incentive design helps attract more honest miners to join. Bitcoin currently incentivizes nodes (or miners) with fixed block rewards and voluntary transaction fees. Historically, the block reward has been the dominating source of miners' revenues. However, for Bitcoin, it is a system parameter that halves approximately every four years.[3] Its domination is expected to vanish due to the deteriorating nature and transaction fees will then become the major mining revenue generator.

With a stable reward, a miner's expected revenues rely mostly on its probability of finding a block, which itself is contingent on the miner's hash power. However, in the fee-based incentive system, the revenues additionally depend on the amount of fees inside a block, which further relies on users' offerings and miners' transaction selections. The total fees inside blocks are market-dependent and time-variant because: (i) transaction arrival can be arbitrary; (ii) transaction fees are voluntary under the current mechanism, so they can be arbitrary (even 0) and the threshold fee rates for faster confirmation change with supply and demand in the block space market; (iii) miners have the freedom of sampling transactions to form new blocks. As a result, the fair sharing of revenue based on hashing power may not be maintained. For example, consider two miners A and B in the system with the same mining power. If A mines blocks each with total fees of 1 BTC and B always encounters wealthy transactions and mines blocks each with 2 BTC total fees, B's revenue is twice A's revenue.

In particular, the fee-based incentivization framework nurtures a possible new deviating mining strategy called undercutting [5]. In undercutting, the attacker intentionally forks an existing chain by leaving wealthier transactions out in its new block to attract other (petty compliant) miners to join the fork. Unlike honest miners, who follow the longest chain that appears first, petty compliant (PC) miners break ties by selecting the chain that leaves out the most fees. In [5], fees accumulate at a fixed rate and miners claim *all* accumulated fees when creating a new block. Thus, a miner undercuts another miner's block because it receives 0 of the fees in the target block but expects nonzero returns via forking. Similarly, PC miners join the fork because the undercutter leaves out more fees unclaimed (and they can claim *all* fees in the next block). Carlsten et al. find that undercutting can become the equilibrium strategy for miners, thus making the system unstable as miners undercut each other.

However, this result is based on a setting disregarding the block size limit. If the fees claimable in the next block is bounded and a pending transaction set exists due to the block size cap, PC miners may not join the fork and undercutting may not be more profitable than extending the current chain head. The intuition is that the extra claimable fees are bounded, and the fork does not win with absolute probability, while the main chain may provide slightly fewer fees but extends with probability 1 when there's no attack. We give an illustrative

[3] The next halving event to 3.125 BTC is scheduled for May 2024. [10]

example below where undercutting is not rational when we consider the limit. Let there be 33% honest, 17% undercutter, and 50% PC mining power, 100 total token fees with 20 claimable in each block. As we elaborate in Appendix F, the undercutter expects 3.4 token returns by extending the chain head. Suppose it instead undercuts and claims half of the tokens in the target block, 10 tokens, in its first forking block (as in [5]). If PC miners do not shift, they expect 10 tokens from the next main-chain block; if they follow the fork, they expect to gain 10 tokens. But, shifting is not rational for the owner of the undercutting target block and may not be rational for others as they have started mining the main chain for some time. Even if they shift, we find the undercutter's expected return to be $1.717 < 3.4$.

Towards modeling undercutting attacks more realistically and generally, we construct a new model to capture rational behaviors related to and performance of undercutting strategy. Miners in our model are either honest or rational. A rational miner may **undercut** or **arbitrarily shift among chains** as long as the action maximizes its returns. Fees in our model arrive with transactions. By sorting transactions in the unconfirmed transaction set and packing at most a block size limit of transactions, we obtain the maximum claimable fees at a certain timestamp. Miners can choose to claim no more than this maximum fee.

Essentially, when undercutting, the rational miner's goal is to earn more than what it can potentially gain not undercutting. The attacker needs to first **(i)** attract other rational miners to join its fork if necessary, and second **(ii)** avoid being undercut by others. If it leaves out too many fees, it may end up being worse off undercutting. If it claims more than necessary, other rational miners may undercut its fork, annihilating its efforts. Then how many fees should an undercutter take to achieve both goals simultaneously? And can others make it not possible to do so? We seek to first locate such a feasible area for an undercutter to secure its premiums and next, uncover defenses against this attack. Note that undercutting is not desired because it hurts the expected profits for honest miners. Successful undercutting also harms users who attach high fee rates to have their transactions processed faster.

1.1 Contributions

We define an analytical model that captures behaviors that are "rational" but not necessarily "honest" like undercutting and shifting rationally. This can be used to analyze other rational deviating strategies in fee-based incentive system. The key is to pinpoint reward distributions and probabilities of earning the rewards.

Specifically for undercutting and as a key contribution, we offer **closed-form conditions on the unconfirmed transaction set to make undercutting profitable**. The key quantity is the ratio (γ) between the maximum claimable fees in the next block (w.r.t. block size limit) and the fees in the current block. For clarity, let the mining power fraction of the undercutter be β_u and that of the honest miner be β_h, remaining rational miner be β_r. **(i)** In the best case for the undercutter in our model, the undercutter forgoes the fork after being one block behind instead of hanging on longer. **(ii)** When $\gamma < \frac{\beta_u}{1-\beta_u}$, the attacker

earns more through normal forking than extending the target chain. **(iii)** When $\frac{\beta_u}{1-\beta_u} < \gamma < \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$, the attacker can expect to earn a premium by proper undercutting. It should carefully craft the first block on its fork in such a way that rational miners can be attracted to join the fork but not tempted to undercut it again. We provide more details in Section 4. The conditions for the case where the undercutter holds on for one more block (Appendix A) are stricter, as noted in (i) and the overall returns are worse.

As a side-product and naturally, we provide an **alternative transaction selection rule** to counter undercutting, other than fitting all available transactions into a block. Once we have identified effective conditions for profitable undercutting, we work backward to proactively check the conditions before creating a new block. By making the conditions no longer satisfied, potential undercutters are no longer motivated to undercut. Applying the defense technique is Nash equilibrium in a typical setting. In the equilibrium, we additionally calculate the price of anarchy (PoA) to capture the inefficiency a strong undercutter brings or the advantage it has in a system. To make the system more stable, we can either strengthen the second potential undercutter or weaken the strongest undercutter through decentralization.

We **experiment with real-world data** from Bitcoin and Monero blockchains to evaluate the profitability of undercutting and the effectiveness of avoidance techniques. We decide on the two systems because Bitcoin is representative of swamped blockchains and Monero typically has a small unconfirmed transaction set. **(i)** In Bitcoin, for a 17.6% undercutter, the average return is 17.9%. For a hypothetical 49.9% attacker, the average revenue is 60.8%. In Monero, we observe a profit increase of around 8 percentage points from fair shares for a 35% attacker. **(ii)** After enabling defense, undercutting generates around fair share for Monero 35% undercutter where the two strongest rational miners possess the same mining powers. We test a strong undercutter's advantage in Bitcoin (49.9%, 20%), which gives the 49.9% attacker around 63.5% of the total returns.

1.2 Related Work

Carlsten et al. [5] introduce the undercutting mining strategy to show the instability of the future Bitcoin fee-based incentivization system because undercutting can become the equilibrium strategy. There, transaction fees accumulate at a constant rate and miners can include all fees when creating a new block. But fees essentially are *not* independent of transactions. If we dive into transaction level and account for block size limit, the fees one can claim is restricted and there can potentially be a large pending transaction set, which can cushion or even annihilating the effects of undercutting. Based on this intuition, we construct the new model focusing on transaction selection rules, which determine fees claimed and left out. Further, both undercutting and hopping among chains are modelled more generally as actions of rational miners instead of separately as two types of miners as in [5]. This helps quantify the profit margin and brings about opportunities for mitigation. Detailed examples for comparison and combining with other deviating mining strategies reside in Appendix F.

Sunk Cost. In traditional microeconomics [18], a rational agent makes decisions based on prospective costs and disregard sunk costs. In behavioral economics [3,28], decision-makers can have an irrational bias towards probability distribution of future events, loss aversion, and other illusions. When a miner decides whether to continue on a chain or shift to other chains, it can be influenced by sunk costs including time already spent. We capture this mindset by letting rational miners shift after its current chain is $D \geq 1$ block(s) behind.

Lemon Market. Another angle to look at the problem on a higher level is through the market for "lemons" [2], the brand-new car that becomes defective the minute one bought it. In the Bitcoin block space market, users are bidders, and miners are sellers. Users decide prices to pay based on their observation of the relationship between confirmation time and fee rates. They attach fee rates corresponding to the desired waiting time. If undercutting is prevailing, users who attach high fee rates but are ghosted are provided with "lemons" instead of "peaches" – fast confirmation. This can result in a decrease in the overall fee rates, diminishing the profitability of undercutting.

2 Background and Definitions

Mempool. Mempool [4] is an unconfirmed transaction set maintained by miners locally. When a transaction is announced to the network, it enters into miners' mempools. Miners select transactions from their mempools to form new blocks. Usually, a miner will choose the bandwidth set (Definition 1) with respect to the local mempool and global block size limit. When a new block is published, miners verify the block and then update their local mempools to exclude transactions included in the newly published block. A miner can also intentionally leave out wealthy transactions when forming blocks to attract rational miners. Wealthy transactions are those with high fee rates. Here we have two measures for fee rates. One is the fee per byte (F/B), which is calculated by dividing the total fee by the transaction size. After SegWit is introduced to the Bitcoin system, a new fee rate measure of fee per weight unit (F/WU) is adopted. For our analytical discussion, we refer to both terms simply as fee rates.

Definition 1. ***(Bandwidth Set.)*** *Given block size limit B and an unconfirmed transaction set A comprising N transactions, $S \in P(A)$ is one bandwidth set of A w.r.t B if $S.size \leq B$ and $\forall S_i \in P(A), S_i.size \leq B, S.fee \geq S_i.fee$, where $P(A) = \{S_j\}_{1 \leq j \leq 2^N}$ is the power set of A.*

Remark 1. A bandwidth set is the subset of transactions in a miner's mempool providing the largest amount of fees a miner can obtain in one block. If the unconfirmed transaction set is of size $\leq B$, then the bandwidth set is the memory pool itself. Note that bandwidth set is not necessarily unique.

Definition 2. ***(Safe margin D.)*** *A miner chooses the chain that is D block(s) ahead as main chain when there are multiple competing chains in the system.*

Remark 2. Honest miners apply the longest chain rule and always have $D = 1$.[4] For rational miners, $D \geq 1$. When the length discrepancy between competing chains is within D, they select the chain with the most expected returns.

3 Mining Game featuring Undercutting Strategy

In this section, we model the mining game involving undercutting strategy and played by honest and rational miners. Honest miners follow the default specifications and are only modeled as passive players. We address the rational miner considering undercutting as the undercutter under discussion. So the undercutter can refer to any rational miner.

Game definition. We define the mining game $G = \langle M, A, R \rangle$ as follows:

- Players $M = \{M_0, M_1, ..., M_{\eta-1}\}$: here η is the number of miners in the system. Without loss of generality, we label a subset of the miners that have a total of β_h mining power as honest; we label a miner with β_u mining power as the current undercutter under discussion; we label the remaining miners as (currently) non-undercutting rational miners and their total mining power is denoted as $\beta_r = 1 - \beta_h - \beta_u$. Honest miners are treated as one because they follow the same mining rules, and we assume they are informed the same way.
- Actions $A = \{undercut(\cdot), stay(\cdot), shiftToChain(\cdot)\}$: we index chains during a game according to their timestamps after the branching point, e.g. the original (main) chain with index $Chain_0$, abbreviated as C_0. For honest miners, they always honest mine and may choose to stay or shift depending on circumstances. Rational miners may choose to undercut an existing chain and start a new chain, stay on a working chain, or shift among existing chains.
- Utility functions $U = \{u_i\}_{M_i \in M}$: we let $u_i = R_i - c_i$, where R_i is the total transaction fees it receives and c_i is the cost. We treat the cost c_i as given and reduce the problem of maximizing utility to maximization of obtained fees.

Threat model. We allow no miner to own more than 50% mining power (i.e., $\beta_u \leq 0.5$). We let miners publish their discovered blocks immediately to attract other miners to join. We assume the best case for the undercutter and let the mempool be the same for miners on the same chain. Because undercutting is not practical or meaningful if miners have distinct mempools, since wealthy transactions an attacker left unclaimed may not exist in others' mempools in the first place. This assumption makes the attacker stronger, and we intend to uncover what the attacker can obtain in advantageous environment settings.

We let miners know of other miners' types (e.g. honest or rational) after sufficient observations. We assume miners can approximate the amount of mining power concentrated on a chain based on the block generation time on that chain.

Solution concept. We solve for Nash Equilibrium (NE) in the mining game with undercutting mining strategy. In a Nash Equilibrium, players do not earn extra utility by unilaterally deviating from the equilibrium strategy.

[4] When there is a tie, they choose the chain with the oldest timestamp. If timestamps should be the same, they select a chain at random.

3.1 Miner's Winning Probability

A miner's expected returns from mining equal the product of its winning probability of a block and the fees residing in that block. Firstly, miner M_i's winning probability of a block is simply its mining power when there is only one chain. In the case of competing chains, we need to additionally quantify a chain's winning probability when working in systems where only one chain survives.

A chain's winning probability. In undercutting, the attacker forks an existing chain by leaving out wealthy transactions. In the following discussions, we refer to the undercutting chain as C_1 and the current main chain as C_0. C_0 might not be on the main chain eventually if C_1 wins the race. The effective height of a chain is the number of blocks it has accumulated after the forking point. These competing blocks are called effective blocks in the game analysis.

Overall, the process proceeds as follows. The undercutter sees a new block is appended to C_0 by another miner. It starts to work on a forking block that excludes wealthy transactions appearing in the current chain head. With some probability, it can create the fork faster than the next block appearing on C_0. When the undercutter publishes its block, some rational miners consider shifting to C_1 because there are more high fee rate transactions that they can benefit from. To model this procedure, we screenshot the state of the system as a tuple that we denote as $\vec{S} = (m, n, \vec{F^0}, \vec{F^1}, O, \delta, \lambda_0, \lambda_1)$, where m and n are respectively the effective height of C_0 and C_1; $\vec{F^0}$ and $\vec{F^1}$ are the list of transaction fee total in effective blocks on C_0 and C_1; O is the mining power currently working on C_1, which updates upon new block appending events; $\delta \in (-1, 1)$ is the mining power shifting from the source chain to the destination chain, which is defined to be positive if miners are shifting to C_1 and negative if they are shifting to C_0; λ_0 and λ_1 are block generation rates for C_0 and C_1.

To obtain the winning probability measure for a chain from state $\vec{S}$, we view the block generation event as a Poisson process and use a random variable to represent the waiting time between block occurrence events. We denote waiting time for C_0 as X and C_1 as Y. They both follow exponential distribution but with different rates. The rate parameters depend on the mining power distribution. Given the state $\vec{S}$, we obtain the block occurrence rate as: $\lambda_0 = \frac{1-O}{I}$; and $\lambda_1 = \frac{O}{I}$, where I is block generation interval (e.g. 10 minutes for Bitcoin). This is derived from the thinning theorem of the Poisson point process. The main idea is that independent sub-processes of a Poisson process are still Poisson processes with individual rates. With this property, we can determine the time interval for the next block to appear on a chain. Then, the key is the mining power concentrated on a chain, and further is whether honest and rational miners shift.

For $D = 1$, there is only one state that the currently non-undercutting rational miners β_r need to make a decision, when the undercutter extends C_1 before the C_0 extends by one. The two competing chains are in a tie with relative height difference $\tilde{D} = 0$. The probability that C_1 wins is simply $p = \Pr[C_1\ Wins] = \Pr[Y < X] = O + \delta$.

For $D = 2$, there is an infinite number of states where flexible rational miners need to make decisions about shifting. We let $\tilde{D} = n - m < D$, denoting the

number of blocks by which C_1 leads C_0. For example, when $\tilde{D} = -1$, C_1 is one block behind C_0. Then C_1 wins if it creates 3 blocks before C_0 extends by 1, or discovers 4 blocks before C_0 extends by 2, and so on. Thus, we have $p = \sum_{i=0}^{\infty} \Pr[(D - \tilde{D} + i)Y < (i+1)X]$.

(i)When $\tilde{\mathbf{D}} = -\mathbf{1}$, C_1 is behind C_0. For C_1 to win, we need $p = \sum_{i=0}^{\infty} \Pr[(3+i)Y < (1+i)X] = \sum_{i=0}^{\infty} (\beta_u + \delta)^{3+i}(1 - \beta_u - \delta)^i$.

(ii)When $\tilde{\mathbf{D}} = \mathbf{0}$, there is a tie between C_1 and C_0. In this case, $p = \sum_{i=0}^{\infty} \Pr[(2+i)Y < (1+i)X] = \sum_{i=0}^{\infty} (\beta_u + \delta)^{2+i}(1 - \beta_u - \delta)^i$.

(iii)When $\tilde{\mathbf{D}} = \mathbf{1}$, C_1 is leading. We have $p = \sum_{i=0}^{\infty} \Pr[(1+i)Y < (1+i)X] = \sum_{i=0}^{\infty} (\beta_u + \delta)^{1+i}(1 - \beta_u - \delta)^i$.

A miner's probability of winning a block. Suppose a miner M_i with β_{M_i} mining power is mining on a chain C_j with β_{C_j} accumulated total mining power which has winning probability p_{C_j}. Then M_i's winning probability is $\frac{\beta_{M_i}}{\beta_{C_j}} p_{C_j}$.

4 Game Analysis

We analyze undercutting strategy with parameter $D = 1$ in this section and continue the discussion with $D = 2$ in Appendix A. The latter generates fewer profits. The high-level idea is that we differentiate between scenarios with "abundant" and "limited" amounts of fees. The extreme case where there are only negligible fees claimable for a long period ("drought") is described in Appendix B.

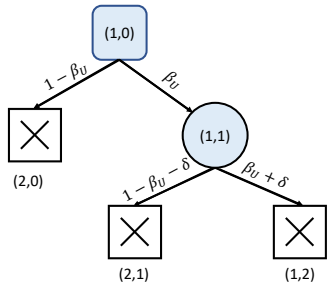


Fig. 1: State transition for $D = 1$. "X" Boxes are terminal states. For non-terminal states, circles indicate ties. Every left branch means C_0 extends by one and every right branch refers to C_1 creating a new block. The quantity on the arrow is the probability of state transition.

4.1 Giving Up If One Block Behind

Now we discuss $D = 1$. We use the abbreviated state $S^* = (m, n)$ in discussion. We denote the transaction fees inside the first two blocks of C_0 as F_1^0 and F_2^0, the transaction fees inside blocks of C_1 as F_1^1 and F_2^1, the expected returns for flexible rational miners β_r as R_r and the expected returns for the undercutter as R_u. When there is no undercutting, we denote their respective expected return as R'_r and R'_u.

For $D = 1$, rational miners only need to decide whether to shift at state $S^* = (1, 1)$ when undercutting becomes visible as shown in Figure 1. Suppose they shift x of its mining power β_r to C_1. They can decide x by solving $\max_{x\in[0,1]} E[R_r] =$

$$\max_{x\in[0,1]} \Big(\frac{\beta_r}{\beta_r + \beta_h}(1-p) \cdot F_1^0 + \frac{(1-x)\beta_r}{\beta_h + (1-x)\beta_r}(1-p) \cdot F_2^0 + \frac{x\beta_r}{x\beta_r + \beta_u} p \cdot F_2^1 \Big) \quad (1)$$

where p is the probability of C_1 winning and the term before each block fee total F is the miner's winning probability for this block. The shift can then be

calculated as $\delta = x\beta_r$. We can observe that the optimization problem involves fees inside succeeding blocks after the forking point. We seek to represent the fees in a relative way so that the analysis can be applied more generally. We let $F_1^0 = 1$ and have fee total in other blocks measured relative to it. Now we continue the discussion in two different mempool situations.

Mempools with limited bandwidth set By "limited" we mean the current bandwidth set on C_0 has a small enough transaction fee total ($< \frac{\beta_u}{1-\beta_u} F_1^0$). We provide more details concerning this quantity as we proceed. WLOG, we assume $F_1^0 = 1$, $F_2^0 = \gamma \geq 0$ (s.t. $\frac{F_2^0}{F_1^0} = \gamma$), $F_1^1 = a$ and $F_2^1 = 1 + \gamma - a$ where $a \in [0, 1]$. Here we are implicitly assuming the best case for the undercutter that it can compose the first block on C_1 in such a way that the second block can claim all unclaimed fees within one block. If a rational miner decides to undercut, with probability β_u, the undercutter can create a new chain and the game is started. With probability $p = \beta_u + \delta$, C_1 wins and with probability $\beta_h + \beta_r - \delta$, C_0 wins. The expected profit of the undercutter is $E[R_u] =$

$$\beta_u(\beta_u + \delta) \cdot (1 \cdot a + \frac{\beta_u}{\beta_u + \delta} \cdot (1 + \gamma - a))$$

The expected return for the rational miner if it does not undercut is $E[R'_u] = \beta_u \gamma$. The miner will undercut only if $E[R'_u] < E[R_u]$. Then

$$\gamma < \frac{\delta a + \beta_u}{1 - \beta_u} \tag{2}$$

With $\gamma < \frac{\beta_u}{1-\beta_u}$, $E[R'_u] < E[R_u]$ even when $\delta = 0$. That is, even no rational miner shifts to C_1, there are so few fees left in the mempool that the attacker is always better off by forking C_0 compared with extending it.

One extreme case is when there are no transactions left or the bandwidth set has negligible fees and $F_2^0 = 0$. The rational miner will fork because originally there is nothing left on C_0 and $E[R'_u] = 0$. One detail is that the attacker needs to craft the first block (determine a) it generates to avoid being undercut again. Suppose when $\gamma < T$ ($T = \frac{\beta_u}{1-\beta_u}$ in our current context), a potential undercutter initiates the attack. Then by choosing a in such a way that $\frac{1+\gamma-a}{a} \geq T$, the undercutter can avoid being undercut again (in the current context). Note that here when an undercutter decides a, it is picturing a potential undercutter other than itself so β_u is not the same as when we decide the condition for γ. We will revisit the choice of a after complete the discussion for $\gamma > \frac{\beta_u}{1-\beta_u}$ case.

In conclusion, for $D = 1$, when the attacker is stronger (β_u is larger), the requirements on the mempool bandwidth set fee total for undercutting to be profitable regardless of rational miners' actions is looser. When β_u approximates 0.5, the threshold ratio approaches 1, which occurs with high frequency. For $\beta_u = 0.2$, the upper bound is 0.25, where the current bandwidth set is 1/4 of the fees inside the chain head of C_0.

Mempools with sufficient bandwidth set By "sufficient" we mean the current bandwidth set in the mempool has more than "limited" transaction fee total

($\geq \frac{\beta_u}{1-\beta_u} F_1^0$). In this case, the undercutter needs to attract some rational miners at state (1,1) (make $\delta > 0$). We continue to let $F_1^0 = 1$, $F_2^0 = \gamma$, $F_1^1 = a$, $F_2^1 = 1 + \gamma - a$ ($a \in [0,1], \gamma \geq 0$). To decide whether to shift to the fork, the rational miner solves for x in $\max_{x\in[0,1]} E[R_r] =$

$$\max_{x\in[0,1]} \big(\frac{\beta_r}{\beta_r+\beta_h}(1-p) + \frac{(1-x)\beta_r}{\beta_h+(1-x)\beta_r}(1-p)\gamma + \frac{x\beta_r}{x\beta_r+\beta_u}p(1+\gamma-a)\big)$$

Here $p = O + \delta = \beta_u + x\beta_r$. One observation is that the rational miners either move to C_1 with all their mining power or none (function is linear in x). When $x = 1$, we have $E[R_{r|x=1}] =$

$$\frac{\beta_r}{\beta_r+\beta_h}\beta_h + \frac{\beta_r}{\beta_r+\beta_u}(\beta_u+\beta_r)(1+\gamma-a) = \frac{\beta_r\beta_h}{\beta_r+\beta_h} + \beta_r(1+\gamma-a)$$

Similarly setting $x = 0$, we obtain

$$E[R_{r|x=0}] = \frac{\beta_r}{\beta_r+\beta_h}(1-\beta_u) + \frac{\beta_r}{\beta_h+\beta_r}(1-\beta_u)\gamma = \beta_r(1+\gamma)$$

To encourage shifting of rational miners, we need $E[R_{r|x=1}] > E[R_{r|x=0}]$, thus

$$a < \frac{\beta_h}{\beta_r+\beta_h} \tag{3}$$

By transforming Equation 2 with our knowledge of a as indicated in Equation 3, we have

$$\gamma < \frac{\beta_r a + \beta_u}{1-\beta_u} < \frac{\beta_h\beta_r + \beta_u(1-\beta_u)}{(1-\beta_u)^2} \tag{4}$$

To avoid being undercut, the undercutter additionally needs to pick an a such that this condition is not satisfied for the first block on its C_1. This is to say the undercutter can profitably undercut C_0 in expectation, but others do not expect to attack its C_1 successfully. As previously touched on, we need

$$a \leq \frac{1+\gamma}{1+T} = \frac{1+\gamma}{1+\frac{\beta_{h_2}\beta_{r_2}+\beta_{u_2}(1-\beta_{u_2})}{(1-\beta_{u_2})^2}} < \frac{1+\frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}}{1+\frac{\beta_{h_2}\beta_{r_2}+\beta_{u_2}(1-\beta_{u_2})}{(1-\beta_{u_2})^2}} \tag{5}$$

where β_{u_2} is the mining power of the strongest potential undercutter for this attacker and β_{r_2}, β_{h_2} is the remaining flexible rational mining power and honest mining power in that case.

For $D = 1$, let the current bandwidth set containing fee total of γ relative to the fees inside the current C_0 chain head. We can conclude that with $\gamma < \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$, the undercutter with mining power β_u can expect a profitable undercutting attack. If $\gamma \geq \frac{\beta_u}{1-\beta_u}$, the undercutter needs to also set a, the fees inside C_1's chain head relative to current C_0 chain head, to be smaller than $\frac{\beta_h}{\beta_r+\beta_h}$ (Inequality 3). But since new transactions may arrive and change the bandwidth set, there is still some uncertainties when actually carrying out the attack. The undercutter in addition needs to make sure it does not get undercut again by ensuring a to be no greater than $\frac{1+\gamma}{1+T}$ (Inequality 5).

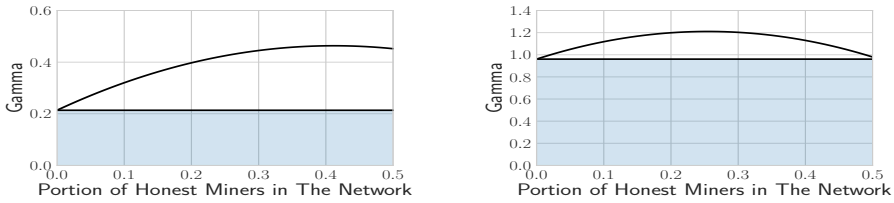
We present algorithms ($D = 1$) for undercutter to start an attack, for other rational miners to decide whether to join a chain and for miners to avoid being undercut below. Figure 2 demonstrates the undercutting conditions graphically.

Part 1. **undercutter** decides whether to undercut:

1. Check if $\gamma < \frac{\beta_u}{1-\beta_u}$. If Yes, start forking. Calculate $T = \frac{\beta_{h_2}\beta_{r_2}+\beta_{u_2}(1-\beta_{u_2})}{(1-\beta_{u_2})^2}$ and set the first block on C_1 to contain $a \leq \frac{1+\gamma}{1+T}$ of the fees in C_0 chain head; else, continue.
2. Check if $\gamma < \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$. If Yes, start undercutting. Calculate $T = \frac{\beta_{h_2}\beta_{r_2}+\beta_{u_2}(1-\beta_{u_2})}{(1-\beta_{u_2})^2}$ and set the first block on C_1 to contain $a < \min\{\frac{\beta_h}{\beta_r+\beta_h}, \frac{1+\gamma}{1+T}\}$ of the fees in C_0 chain head. Exit.

Part 2. **flexible rational miners** decide whether to join a chain: Solve for x (proportion of mining power to shift to the chain) in Equation 1.
Part 3. **miners** avoid being undercut: Calculate $T = \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$ [a]. Check if $\gamma < T$. If Yes, include in the current block $a < \frac{1+\gamma}{1+T}$ of the fees in bandwidth set; else, use the bandwidth set.

[a] Here β_u is the undercutter a miner is defending against unlike in undercutting reasoning, where β_u is the miner's own computation power.



(a) $\beta_u = 17.6\%$ **(b)** $\beta_u = 49.9\%$

Fig. 2: Undercutting conditions for $D = 1$ as summarized in Part 1 of the algorithm. The shaded area indicates the "limited" bandwidth set condition and the upper line depicts the undercutting conditions when faced with "sufficient" bandwidth set.

Treating rational miners as one miner. In the above analysis, rational miners make decisions from a collective perspective by maximizing $E[R_r]$ instead of the expected returns for a specific rational miner M_i ($i < n$). This can give rise to coordination problems. Fortunately, rational miners either move all their mining power or staying on their current chain. There is only one state $(1, 1)$ where they need to make a decision. There is one scenario in practice when a rational miner may not be flexible, which is when this miner owns the current chain head of C_0. In analysis, we only consider this scenario probabilistically

by applying a probability $\frac{\beta_r}{\beta_r+\beta_h}$ as the probability rational miners own the C_0 head. But in reality, a miner either owns the block or not. When a rational miner indeed owns the C_0 head, we treat it like honest miners when applying the above analysis. Since miners are aware of other miners' types across time, they will be able to adjust in their reasoning process.

When to apply undercutting avoidance. Suppose the current bandwidth set contains fees of 1 and the remaining next bandwidth set contains γ amount of fees. The mempool is always sorted so $\gamma \leq 1$ (except in edge cases). Suppose we have computed the corresponding threshold attacking condition T for a rational attacker and $\gamma < T$. Then this attacker undercuts if a miner simply assembles the current bandwidth set into a block. When a miner M_i does not apply avoidance, its expected return from this single block is $E[R_{M_i}] = 1-\beta_u-\delta$. When M_i applies avoidance, its expected return is $E[R_{M_i,avoid}] = \frac{1+\gamma}{1+T}$. Then if $\frac{1+\gamma}{1+T} > 1-\beta_u-\delta$, M_i applies avoidance routine. We state the following theorem.

Theorem 1. *In setting $D = 1$, each miner applying avoidance procedure when creating a new block is NE.*

Proof. Let $M_i \in M$ be a miner and M_i calculates $T = \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$. When $\gamma \geq T$, M_i proceeds as normal. Therefore, we only need to show that for M_i, when $\gamma < T$, M_i is better off by claiming $a < \frac{1+\gamma}{1+T}$ of the fees in bandwidth set.

(i) $\gamma < \frac{\beta_u}{1-\beta_u}$: this means $\beta_u > \frac{\gamma}{1+\gamma}$ and the undercutter does not attract others. $E[R_{M_i}] = (1-\beta_u)\cdot 1 \leq (1+\gamma)(1-\beta_u) = \frac{1+\gamma}{1+T} = E[R_{M_i,avoid}]$. This indicates that a miner expects higher returns by applying avoidance in this setting. **(ii)** $\frac{\beta_u}{1-\beta_u} \leq \gamma < \frac{\beta_h\beta_r+\beta_u(1-\beta_u)}{(1-\beta_u)^2}$: the undercutter attracts other miners and $E[R_{M_i}] = 1-\beta_u-\beta_r = \beta_h$. Since $E[R_{M_i,avoid}] = \frac{1+\gamma}{1+T} > \frac{1}{1+T} = \frac{(\beta_h+\beta_r)^2}{\beta_h\beta_r+\beta_h+\beta_r} > \beta_h$, a miner also expects higher profits when applying avoidance in this case.

By unilaterally deviating from avoidance when γ satisfies undercutting conditions for a potential undercutter, M_i receives smaller expected returns because the undercutter's best response is undercutting in that case. □

There are two special cases worth noting: (1) all miners are honest ($\beta_h = 1$) so that $T = 0$. We know that $\gamma \geq 0$. No effective avoidance is ever needed in this case; (2) M_i is the only rational miner ($\beta_r = 0$) so that $T = 0$ for itself. M_i also does not need to actually apply avoidance since $\gamma \geq 0$.

Quantifying Strong Undercutter's Advantage. Let the strongest undercutter have mining power β_u and the second strongest undercutter have mining power β_{u_2}. We know from the previous discussion that a miner should always apply avoidance techniques to avoid being undercut in our setting. For miners other than the strongest undercutter β_u, they need to defend against β_u while β_u itself only needs to defend against β_{u_2}. Let T, T' be the threshold ratio computed for β_u and β_{u_2} respectively. We can capture its advantage with the ratio $\frac{1+T}{1+T'}$. For example, if $\beta_u = 0.5, \beta_{u_2} = 0.2, \beta_h = 0$, $\frac{1+T}{1+T'} = 4$, which means that the strongest undercutter can claim 4 times than what the other miners are collecting each time. When the discrepancy between β_u, β_{u_2} approaches 0,

$\frac{1+T}{1+T'}$ approaches 1. More formally, we capture this inefficiency brought by selfish behavior with the price of anarchy (PoA) [12].

Corollary 1 (Price of Anarchy). *In setting $D = 1, \beta_h < 1, \beta_r > 0$, with the strongest and the second-strongest undercutters respectively having mining power β_u, β_{u_2}, the Price of Anarchy is $PoA = \frac{1+T}{(T-T')\beta_u+1+T'}$, where T, T' are as defined above.*

This follows from the above analysis. When all miners stay honest, the "undercutter" is expected to earn a fair share β_u. When miners apply avoidance, the strongest undercutter claims $\frac{1+\gamma}{1+T'}$ each time while others claim $\frac{1+\gamma}{1+T}$. We can obtain its share $\frac{\beta_u \frac{1+\gamma}{1+T'}}{\beta_u \frac{1+\gamma}{1+T'} + (1-\beta_u)\frac{1+\gamma}{1+T}}$. Then we can calculate the PoA as the ratio between the strongest undercutter's shares in its optimal situation (the worst-case NE for the system) and in its worst case (the optimal all honest outcome). We do not include other miners' returns in calculation because the total shares always sum up to 1 regardless of the outcome and our focus is capturing the advantage of the undercutter. One observation is that when β_u and $T-T'$ are large, PoA is large. To move it towards 1 (more stable system), we can either strengthen the second potential undercutter or downsize β_u through further decentralization. For example, let $\beta_u = 0.499, \beta_{u_2} = 0.176$ and $\beta_h \in \{0, 0.05, 0.10, \ldots, 0.30\}$, on average (over β_h) $T = 1.06, T' = 0.37$ and $PoA = 1.20$. This means that for β_u, the mean revenue proportion from undercutting is $0.499 \times 1.20 = 0.6$.

5 System Evaluation

In this section, we evaluate the profitability of undercutting using data obtained from Bitcoin and Monero. Bitcoin is a typical example of congested blockchains, and Monero is a more available one. The simulation codes and a sample data set have been made open source [19]. In previous analysis, we let the undercutter be aware of future transaction flows in and out of the mempool. In reality, there is more uncertainty involved. Another difference is that now mining powers are discrete, and we model each miner individually.

5.1 Data Collection and Experiment Setup

Transactions. We obtain the blocks from height 630,457 (May 15th, 2020 after the Bitcoin's block reward halving) to 634,928 (June 15th, 2020) from the Bitcoin blockchain using the API provided by blockchain.com [25], comprising 9,167,040 transactions. The Monero blockchain data are collected using a similar API from `xmrchain.net`. In total, we acquire 1,482,296 transactions from block height 2,100,000 (May 17th, 2020) to 2,191,000 (Sept 20th, 2020). For each of these transactions, we extract the size, fee, and timestamp attributes. Note that transactions that appeared during the sample period but not in any of the collected blocks are not included. Thus, the memory pools reconstructed are not

the exact mempools miners were faced with. We also create artificial transaction data sets with normally distributed fee rates.

Miners. To mimic the actual Bitcoin network, we follow the mining power distribution of miners published by blockchain.com [26] on July 30th, 2020. We make the strongest miner with 17.6% mining power the undercutting miner. We additionally consider a hypothetical undercutter with 49.9% mining power. This is to uncover the profitability of undercutting for a strong attacker and its advantage over other miners when avoidance techniques are adopted by all. For the Monero network, we follow the mining power distributions published by exodus [29] and moneropool.com [1]. The strongest pool with 35% mining power is made the undercutting miner.

Setup. We model the blockchain system as event-based, with new block creation being the event. Parameters and states of the system are updated upon a new block creation event that we denote as B_i for the remaining of this section. Miners have the same view of the network and the same latency in propagating the blocks and transactions. So miners working on the same chain see the same mempool. We initialize the time of the system to the earliest transaction timestamp. As shown in Algorithm 1, new block creation first happens (line 2-4). Then chains, miners, mempools are updated in line 5-7. We include more details for chain and miner updating routines in Algorithm 2. Detailed descriptions for each routine reside in Appendix C.1.

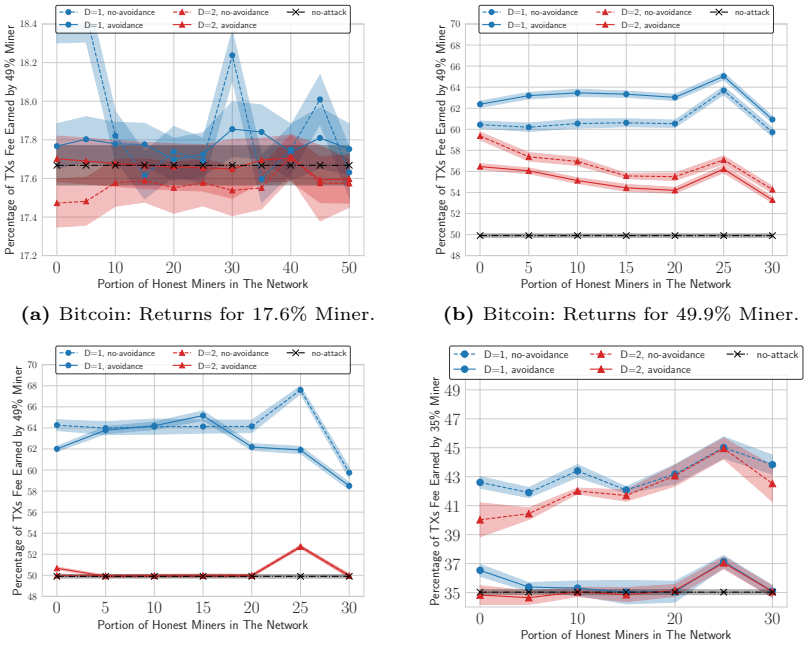
Algorithm 1: Simulation Overview

```
   input : txSet, minerSet, chainsTime
1: while txSet not empty do
2:     extChain ← nextChainToExtend(chainsTime);
3:     m ← selectNextBlockMiner(extChain);
4:     nextBlock ← publishBlock(m);
5:     updateChains(extChain, nextBlock);
6:     updateMiners(extChain);
7:     updateMempool(extChain);
```

Algorithm 2: Chain and Miner Updates

```
1: Function updateChains(extChain, nextBlock):
2:     extChain.append(nextBlock);
3:     foreach chain in chainsTime do
4:         remove from chainsTime if it is non-wining
5:     t ← NextBlockCreationTime(extChain);
6:     update chainsTime with tuple (extChain, t);

7: Function updateMiners(extChain):
8:     foreach miner in minerSet do
9:         if miner = undercutter then
10:            decide to fork or not and craft the new block as described in Part 1 of
                 the D = 1 algorithm in 4.1, the D = 2 algorithm in Appendix A;
11:        if miner = honest then
12:            if extChain longest chain then
13:                switch to extChain;
14:        if miner = rational then
15:            decide to switch to extChain or stay on current chain as described in
                 Part 2 of the D = 1 algorithm in 4.1, the D = 2 algorithm in
                 Appendix A;
```



(a) Bitcoin: Returns for 17.6% Miner.

(b) Bitcoin: Returns for 49.9% Miner.

(c) Artificial transactions with normal distributed fee rates: returns for 49% Miner.

(d) Monero: Returns for 35% attacker.

Fig. 3: Undercutting Returns: normal runs (dashed lines) and runs with avoidance feature enabled (solid lines). The shadowed band is statistics' 95% confidence interval.

Simulation run. In a normal run, we repeat the above steps until we exhaust all transactions. In an avoidance enabled simulation run, we repeat the procedure but with all miners actively defending against undercutting in line 4, according to the two summarized algorithms in Section 4.1 and Appendix A.

5.2 Experiment Results

Normal runs. Overall in a normal run, a strong undercutter can expect to earn more than fair shares by conditional undercutting as shown in figures 3b and 3d. **(i)** In Bitcoin runs, the 17.6% undercutter receives on average (for $D = 1$) 17.9% shares for 0-50% honest mining power (Figure 3a). The strong 49.9% undercutter receives a greater profit of 60.8% of the shares (Figure 3b). **(ii)** In runs with artificial transactions, the profits for $D = 1, 2$ bear a wider gap than with actual Bitcoin transactions (Figure 3c). **(iii)** In Monero runs, the 35% undercutter obtains 43.2% of the profit on average (for $D = 1, 2$) for different honest miner portions (Figure 3d). Undercutting is especially efficient in Monero because of its small mempools, which provide limited cushion effects.

With undercutting avoidance. As noted by PoA, the attacker has an advantage over others in equilibrium. The predicted average revenue proportion

(adjusted for rounds where the undercutter mines a block and attacking is unnecessary) for the 49.9% attacker is around 63.5%. **(i)** In Bitcoin actual and artificial data runs, the return proportion is close to this predicted average. Avoidance runs can result in better revenues for the undercutter if the attack *cannot* be carried out to its ideal extent. That is because a large mempool along with continual incoming transactions lower the profitability of undercutting. The implication is that if undercutting cannot be implemented ideally, avoidance can be relaxed from the exact extent. **(ii)** For Monero, we observe profit reduction for attackers in both margins after enabling avoidance, as shown in Figure 3d. **(iii)** Monero runs and Bitcoin runs for 17.6% undercutter provide more straightforward results, compared to Bitcoin runs with 49.9% attacker. Because the second undercutter in Monero has 35% mining power, which equals the strongest undercutter's mining power and in Bitcoin, the configuration is that the second-strongest mining power is 15.3% for 17.6% attacker and 20% for 49.9% attacker.

Minor changes to Bitcoin core codebase. We provide discussions concerning undercutting avoidance implementation in Appendix F.1. We note that only light code changes in Bitcoin core codebase are needed, which we demonstrate in this source [9] and also describe the code snippet in Appendix F.1.

6 Conclusion

We study the profitability of undercutting mining strategy with block size limit present. The intentional balancing of undercutting others and avoiding one's fork being undercut again demands specific conditions on the unconfirmed transaction set at the time of decision-making. Upon conditions being met, an attacker can expect positive premiums. However, because such conditions are not easy to satisfy, time-dependent (can be invalidated if new transactions arrive), and can be manipulated, it opens a door for mitigation. By applying an avoidance technique to invalidate the aforementioned conditions, miners can avoid being undercut. Avoidance encourages miners to claim fewer fees if the current bandwidth set is sufficiently wealthier than the next bandwidth set. As a result, the competition of undercutting can involuntarily promote the fair sharing of fees even in a time-variant fee system. Nevertheless, in a one-sided competition where the mining power discrepancy between the first and second strongest undercutters is large, the stronger undercutter has a natural advantage over others because it only has to defend against the weaker.

7 Acknowledgement

We would like to thank our shepherd Marko Vukolic and anonymous reviewers for valuable comments. We thank Dankrad Feist for his feedback in the early stage of this project. This work has been partially supported by the National Science Foundation under grant CNS-1846316.

References

1. Monero pools since 2016. http://moneropools.com/ (2020)
2. Akerlof, G.A.: The market for "lemons": Quality uncertainty and the market mechanism. In: Uncertainty in economics, pp. 235–251. Elsevier (1978)
3. Benartzi, S., Thaler, R.H.: Myopic loss aversion and the equity premium puzzle. The quarterly journal of Economics **110**(1), 73–92 (1995)
4. Bitcoin.org: Memory pool, https://developer.bitcoin.org/devguide/p2p_network.html#memory-pool
5. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167. ACM (2016)
6. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718 (2014)
7. Eyal, I.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy. pp. 89–103. IEEE (2015)
8. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International conference on financial cryptography and data security. pp. 436–454. Springer (2014)
9. Gong, T.: Changes to bitcoin core source code to account for undercutting avoidance, https://github.com/haas256/bitcoin
10. Half, B.B.: Bitcoin halving 2024. https://www.bitcoinblockhalf.com/ (2020), accessed: 2020-07-22
11. Heilman, E.: One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In: International Conference on Financial Cryptography and Data Security. pp. 161–162. Springer (2014)
12. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Annual Symposium on Theoretical Aspects of Computer Science. pp. 404–413. Springer (1999)
13. Kwon, Y., Kim, D., Son, Y., Vasserman, E., Kim, Y.: Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 195–209. ACM (2017)
14. Kwon, Y., Kim, H., Yi, Y., Kim, Y.: An eye for an eye: Economics of retaliation in mining pools. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 169–182 (2019)
15. Lavi, R., Sattath, O., Zohar, A.: Redesigning bitcoin's fee market. In: The World Wide Web Conference. pp. 2950–2956. ACM (2019)
16. Luu, L., Saha, R., Parameshwaran, I., Saxena, P., Hobor, A.: On power splitting games in distributed computation: The case of bitcoin pooled mining. In: 2015 IEEE 28th Computer Security Foundations Symposium. pp. 397–411. IEEE (2015)
17. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 706–719 (2015)
18. Mankiw, N.G.: Principles of economics. Cengage Learning (2014)
19. Mohsen Minaei, T.G.: Source code of the blockchain simulation and undercutting experiments, https://github.com/haas256/UP
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
21. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 305–320. IEEE (2016)

22. Pass, R., Shi, E.: Fruitchains: A fair blockchain. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 315–324. ACM (2017)
23. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint arXiv:1112.4980 (2011)
24. Rosenfeld, M.: Analysis of hashrate-based double spending. arXiv preprint arXiv:1402.2009 (2014)
25. S.A., B.L.: Bitcoin blockchain api. `https://www.blockchain.com/api` (2020)
26. S.A., B.L.: Bitcoin miners mining power. `https://www.blockchain.com/en/pools` (2020), accessed: 2020-02-27
27. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)
28. Tom, S.M., Fox, C.R., Trepel, C., Poldrack, R.A.: The neural basis of loss aversion in decision-making under risk. Science **315**(5811), 515–518 (2007)
29. Won, D.: 2020's best monero pools. `https://www.exodus.io/blog/best-monero-pools/` (2020)
30. Zhang, R., Preneel, B.: Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In: Cryptographers' Track at the RSA Conference. pp. 277–292. Springer (2017)

A Giving Up After Two Blocks Behind

Now, we discuss $D = 2$. Rational miners make decisions at states $S^* = \{(1,1), (1,2), (2,1), (2,2), ...\}$. The probability p now comprises infinite series. Without loss of generality, we let $F_{0,1} = 1$, $F_{0,2} = F_{0,3} = \gamma$, $F_{1,1} = a, F_{1,2} = b$ and $F_{1,3} = 1 + 2\gamma - a - b$ (where $a \in [0,1], \gamma \geq 0$). $F_{0,2}, F_{0,3}$ can be of different values in reality but here we use the same value to highlight the wealthiness of $F_{0,1}$. Following the intuition from $D = 1$ case, we let $F_{1,1} = a$ and the remaining fees from the first two blocks on C_1 is $(1 + \gamma - a)$. Suppose eventually we derive a condition T for setting $D = 2$ as well, then the undercutter would want to set a and b to satisfy $\frac{1+\gamma-a}{a} > T$ and $\frac{1+2\gamma-a-b}{b} > T$ to avoid being undercut.

We take the same route as in $D = 1$ case. We know that if there is no attack, the undercutter expects to receive

$$E[R'_u] = 2\beta_u\gamma$$

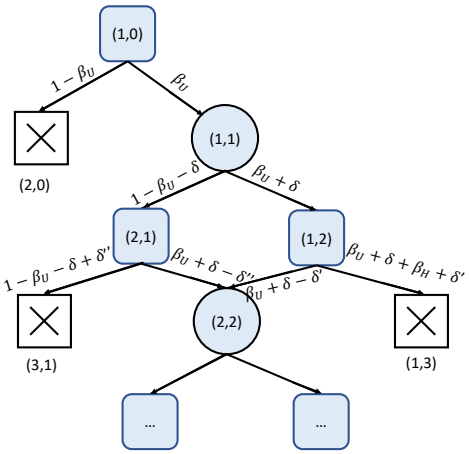


Fig. 4: State transition for $D = 2$. Notations are the same as Figure 1. Now we have infinite state transitions. δ' and δ'' are the amount of rational mining power shifting from one chain to another. We do not label the probabilities of state transitions at the end of the graph to emphasize that: (i) the game can go on infinitely, and (ii) the corresponding probabilities are different when traveling from different paths.

If it starts the attack, its expected
return from the rightmost branch (shown in Figure 4) when no miners shift is

$$E[R_u] = \beta_u(2\gamma+1)\sum_{i=0}^{\infty}\beta_u^{i+2}(1-\beta_u)^i$$
$$= \beta_u^3(2\gamma+1)\cdot\lim_{n\to\infty}\frac{1-\beta_u^n(1-\beta_u)^n}{1-\beta_u(1-\beta_u)} = \frac{\beta_u^3(2\gamma+1)}{1-\beta_u(1-\beta_u)}$$

When $\gamma < \frac{\beta_u^2}{2(1-\beta_u)}$ (with limited bandwidth set), the undercutter can expect to successfully start the attack without rational miners joining C_1. This bound is more demanding than the one for $D = 1$. For $\beta_u = 0.5$, the upper bound is now 0.25 instead of 1. For $\beta_u = 0.2$, the bound is 0.025 instead of 0.25. Overall, for weak attackers, the condition is way more demanding than before.

Next, we consider when $\gamma \geq \frac{\beta_u^2}{2(1-\beta_u)}$ (with sufficient bandwidth set) and the undercutter needs rational miners to join C_1. Same as before, rational miners allocate their mining power among the two chains to maximize their expected returns. We solve for x in

$$\max_{x\in[0,1]} E[R_r] = \max_{x\in[0,1]}\left(\frac{\beta_r}{\beta_r+\beta_h}p_0 + \frac{(1-x)\beta_r}{\beta_h+(1-x)\beta_r}p_0\cdot 2\gamma\right.$$
$$\left. + \frac{x\beta_r}{x\beta_r+\beta_u}p_1\cdot b + \frac{x\beta_r}{x\beta_r+\beta_u+\beta_h}p_1\cdot(1+2\gamma-a-b)\right) \quad (6)$$

where $p_0 = \beta_u(1-\beta_u-x\beta_r)^2$ is the probability of C_0 leading by 2 blocks first and $p_1 = \beta_u(\beta_u+x\beta_r)(\beta_u+x\beta_r+\beta_h)$ is the probability of C_1 leading by 2 blocks first. Here we only consider the leftmost and rightmost branch in Figure 4 because they are the two most significant paths. We can observe that the coefficient for the second-degree term in the quadratic function is positive. The expected returns for rational miners may reach maximum at either of the two ends.

Again we let $E[R_{r|x=0}] < E[R_{r|x=1}]$.

$$2(\beta_h-\beta_u)\gamma < \frac{\beta_h^2}{\beta_h+\beta_r} + \beta_h b - (\beta_u+\beta_r)a + \beta_u - \beta_h$$

Here optimal choices of a and b depend on γ as described in the beginning of this subsection. We know that $a \in [\frac{1+\gamma}{2}, \frac{1+\gamma}{1+T}]$. The lower bound is because the undercutter does not need to claim less than $1/2$ of the fees in the first two blocks on C_0 because the avoidance ratio would be 1 for the first two blocks on C_1. It's safe except when the undercutter has majority mining power. Similarly, b has lower bound $\frac{1+2\gamma-a}{2}$. The exact solution for γ in the above inequality can be obtained numerically for different sets of variables. We provide an approximate solution that can be provided in closed form, using the lower bounds for a, b. We can obtain

$$\gamma > \frac{\frac{4\beta_h^2}{\beta_h+\beta_r} - \beta_h + 4\beta_u - 2}{3\beta_h - 8\beta_u + 2}$$

where $3\beta_h - 8\beta_u + 2 < 0$ or $\beta_h < \frac{8\beta_u - 2}{3}$. For $\beta_u < 0.25$, there is no feasible solution.

With rational miners joining, the expected return for undercutter on the rightmost branch is now

$$E[R_{u|r}] = \big(a + \frac{\beta_u}{\beta_u + \beta_r} b + \beta_u(1 + 2\gamma - a - b)\big)\beta_u(\beta_u + \beta_r)$$

We let $E[R_{u|r}] > E[R'_u]$ to arrive at a tighter bound, and we have

$$\gamma < \frac{4\beta_u + 2\beta_r - \beta_u\beta_h}{8\beta_h + 6\beta_r + 3\beta_u\beta_h} \tag{7}$$

Overall in sufficient mempool scenario, we need $\beta_h < \frac{8\beta_u - 2}{3}$, Equation 7 and

$$\gamma > \max\{\frac{\beta_u^2}{2(1-\beta_u)}, \frac{\frac{4\beta_h^2}{\beta_h + \beta_r} - \beta_h + 4\beta_u - 2}{3\beta_h - 8\beta_u + 2}\}$$

Note that a solution might not exist for certain parameter sets.

In conclusion, for $D = 2$, the limited bandwidth set bound is now $\gamma < \frac{\beta_u^2}{2(1-\beta_u)}$. This criterion can be hard to meet for weak miners with less than 30% mining power ($\gamma < 0.065$ for $\beta_u = 0.3$). But for strong attackers with 40%-50% mining power (0.13 - 0.25), the conditions are not rare to satisfy. In the sufficient bandwidth set scenarios, attackers also have tighter bounds on γ, especially for weak attackers.

We present algorithms for $D = 2$ below. Figure 5 demonstrates the undercutting conditions for $D = 2$ graphically.

Part 1. **undercutter** decides whether to undercut:

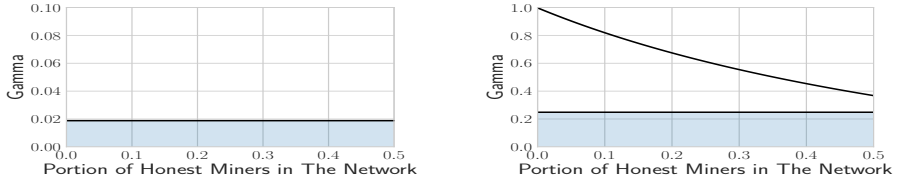
1. Check if $\gamma < \frac{\beta_u^2}{2(1-\beta_u)}$. If Yes, start undercutting. Calculate $T = \frac{\beta_{u_2}^2}{2(1-\beta_{u_2})}$ if $\beta_h \geq \frac{8\beta_{u_2} - 2}{3}$ and $T = \frac{4\beta_{u_2} + 2\beta_{r_2} - \beta_{u_2}\beta_{h_2}}{8\beta_{h_2} + 6\beta_{r_2} + 3\beta_{u_2}\beta_{h_2}}$ otherwise. Set the first block on C_1 to contain $a \leq \frac{1+\gamma}{1+T}$ of the fees in C_0 chain head, exit; else, Continue.
2. Check if $\beta_h < \frac{8\beta_u - 2}{3}$, if Yes, Continue; else, Exit.
3. Check if $\gamma < \frac{4\beta_u + 2\beta_r - \beta_u\beta_h}{8\beta_h + 6\beta_r + 3\beta_u\beta_h}$ and $\gamma > \frac{\frac{4\beta_h^2}{\beta_h + \beta_r} - \beta_h + 4\beta_u - 2}{3\beta_h - 8\beta_u + 2}$. If Yes, start undercutting. Calculate $T = \frac{4\beta_{u_2} + 2\beta_{r_2} - \beta_{u_2}\beta_{h_2}}{8\beta_{h_2} + 6\beta_{r_2} + 3\beta_{u_2}\beta_{h_2}}$. Set the first block on C_1 to contain $a < \min\{\frac{\beta_h}{\beta_r + \beta_h}, \frac{1+\gamma}{1+T}\}$ of the fees in C_0 chain head. Exit.

Part 2. **rational miners** decide whether to join a chain: Solve for x (the proportion of mining power to shift to the chain) in Equation 8.
Part 3. **miners** avoid being undercut:

1. Calculate T:
   - If $\gamma < \frac{\beta_u^2}{2(1-\beta_u)}$, Calculate $T = \frac{\beta_u^2}{2(1-\beta_u)}$;

– else if $\beta_h < \frac{8\beta_u-2}{3}$ **and** $\gamma < \frac{4\beta_u+2\beta_r-\beta_u\beta_h}{8\beta_h+6\beta_r+3\beta_u\beta_h}$ **and** $\gamma > \frac{\frac{4\beta_h^2}{\beta_h+\beta_r}-\beta_h+4\beta_u-2}{3\beta_h-8\beta_u+2}$, calculate $T = \frac{4\beta_u+2\beta_r-\beta_u\beta_h}{8\beta_h+6\beta_r+3\beta_u\beta_h}$;
– otherwise, exit.

2. Check if $\gamma < T$. If Yes, include in the current block $a < \frac{1+\gamma}{1+T}$ of the fees in bandwidth set; else, use the bandwidth set.



(a) $\beta_u = 17.6\%$ **(b)** $\beta_u = 49.9\%$

Fig. 5: Undercutting conditions in $D = 2$ setting as summarized in Part 1 of the $D = 2$ algorithm.

Suppose C_1 extends by one, rational miners decide the amount of mining power allocated on C_0 to shift to C_1 by solving x in

$$\max_{x\in[0,1]} E[R_r] = \max_{x\in[0,1]} \Bigg((\text{fees own on } C_0)p_0 + (\text{fees own on } C_1)p_1 + (\text{claimable fees on } C_0)\frac{(1-x)\beta_r}{1-O-x\beta_r}p_0 + (\text{claimable fees on } C_1)\frac{x\beta_r}{O+x\beta_r}p_1 \Bigg) \tag{8}$$

where $p_1 = (O+x\beta^*_{R|m})^{D-\tilde{D}}$ and $p_0 = (1-O-x\beta^*_{R|m})^{D+\tilde{D}}$ ($\beta^*_{R|m}$ is the rational mining power on C_0). Claimable fees are total fees that a miner can expect to obtain in the unconfirmed transaction sets of each chain within size limit $(D \mp \tilde{D}) \cdot B$ (B is the block size limit). When C_0 extends by one, the processing is similar to the previous scenario (conceptually equivalent). The only difference is that in calculation, $p_1 = (O-x\beta^*_{R|f})^{D-\tilde{D}}$ and $p_0 = (1-O-x\beta^*_{R|f})^{D+\tilde{D}}$ ($\beta^*_{R|f}$ is the rational mining power on C_1).

Since we are solving for approximate conditions by considering only significant paths instead of infinite many paths, we do not give an exact theorem as Theorem 1. The attacking conditions when $D = 2$ are tighter. This means that when one can expect a profitable attack in $D = 2$, it's also feasible to undercut and give up after being one block behind.

B Rearranging of Past Blocks in Extreme Case

Let's consider an extreme case where there are only negligible fees unclaimed in the unconfirmed transaction set for a sufficiently long time (greater than multiple block generation intervals I). As we have noticed, when there is only a limited amount of fees left in the mempool, a rational miner can avoid undercutting by claiming only a part of the bandwidth set. But if the situation continues to worsen and no new transactions enter into the system, the remaining transaction fees become negligible at a certain point. In this extreme case, rational miners may look back to the previous blocks and start undercutting at certain block height and rearrange blocks from there onwards. Suppose an undercutter goes back C blocks. As long as there are only negligible transaction fees flowing into the system during $\frac{C \cdot I}{\beta_u}$, it's more desirable for an attacker who earned less than its fair share in the past C blocks to attack. The previous analysis does not apply to this extreme case. However, the good news is that Bitcoin, Monero, and possibly other important altcoins currently do not suffer from this problem and may remain this way. If this is no longer the case, undercutting may not be a major concern.

C Experiment Details

C.1 Experiment Setup

Initial setup. We initialize the system's time to the earliest timestamp (t_0) of the collected transaction. Then we create the empty genesis block B_0 and create the chain C_0 by appending B_0 to it. Next, we insert the tuple (C_0, t_0) to an empty list `chainsTime`. The tuples inside this list indicate when the next block for each of the chains will be generated. Alg. 1 provides an overview of the simulation after the initial setup. The simulation takes the transactions (`txSet`), miners (`minerSet`) and the tuple list (`chainsTime`) as inputs. We consider these inputs as global for all functions in the simulation. Each iteration of the while loop indicates a new event.

Block creation (line 2-4 in Alg. 1). In the first step of each iteration, the chain to be extended, `extChain`, is selected using the `nextChainToExtend` function. It sorts all the tuples in `chainsTime` and picks the chain with the smallest next block creation time. Next, the algorithm selects the new block's miner using the function `selectNextBlockMiner`. This function randomly selects miner `m`, from all the miners that are working on `extChain`, weighted by their mining power. Finally, the selected miner `m` publishes the next block using the transactions in its mempool.

Chain updates (line 5 in Alg. 1). After the creation of the new block B_i, all the chains in the system are updated via the procedure depicted in Alg. 2. In the first step, block B_i is appended to the current extending chain `extChain`. Next, all other chains are checked against the `extChain` to see if they are in a non-winning situation (`extChain` has at least D blocks from the forking point).

If a chain is non-winning, it will be removed from the system, and `chainsTime` will be updated. Finally, the `chainsTime` list is updated with the new time for the next block on `extChain`.

Miner updates (line 6 in Algo. 1). Following the chain updates, miners update their working chains. Each miner based on its type (undercutter, honest, rational) decides whether to change its working chain (shown in `updateMiners` function in Algo. 2).

1. If `extChain` is an competing chain of a undercutter miner, the miner checks whether `extChain` is D blocks ahead of its own chain and switches to `extChain` if Yes.
2. If `extChain` is not a forked chain created by the undercutter, and miner `m` (miner of block B_i) is not the undercutting miner itself, the undercutter begins condition checking routine. If the condition is ripe, it forks block B_i. Otherwise, it continues to extend `extChain`.
3. All honest miners check whether `extChain` is the longest chain in the system. If Yes, they switch to chain `extChain`.
4. Rational miners that are not on `extChain`, compare the length of `extChain` with their current chain and calculate their expected returns as described in Section 4.1 and Appendix A, and decide whether to switch to `extChain`.

Mempool update (line 7 in Algo. 1). The last system update before moving to the creation of the next block (B_{i+1}) is the mempool update. In this step, all transactions in `txSet` with a timestamp between the creation time of B_{i-1} and B_i are added to the mempool of the miners on `extChain`.

Simulation run. In a normal run, we repeat the above steps until all transactions have been consumed (`txSet` is empty). To moderate fluctuations caused by the random selections (of block generation time and block owners), we repeat the experiments 10 times (for each parameter set) for both Bitcoin and Monero and report the mean values of profit proportions along with the 95% confidence intervals. We maintain two parameters in the simulation: (i) safe margin D: the undercutter gives up on the forked chain as soon as the competing chain is $D(= 1 \textit{ or } 2)$ blocks ahead; (ii) honest miners β_h in the system: we consider 0% to the maximum feasible honest mining powers, with increment being 5%.

We also simulate undercutting avoidance enabled. The difference from a normal run is that when miners extend a chain, they implement "Part 3" described in the two summarized algorithms in Section 4.1 and Appendix A. Undercutting avoidance, implemented by other miners, makes the original attacking conditions no longer satisfied. We do not carry out the exact avoidance technique for experiment purposes and potentially give some advantage to the undercutter. To be more specific, the fees left unclaimed for avoidance purposes may not fit into the next block and the effectiveness of avoidance is weakened.

D Definitions

Definition 3. ***(Near Bandwidth Set.)*** *Given block size limit B and an unconfirmed transaction set A composed of N transactions, $\tilde{S} \in P(A)$ is a near*

bandwidth set of A w.r.t B if $\tilde{S}.size \leq B$ and $S.fee - \tilde{S}.fee \geq \epsilon$ where S is a bandwidth set of A w.r.t B, $\epsilon > 0$ is a small distance parameter.

Remark 3. Near bandwidth set appears more often than bandwidth set according to history block statistics, due to propagation delay, miners' balancing between block size and bandwidth usage, reluctance to give up work already spent on the current block, and potentially other causes. To be more specific, when a miner forms a block, it balances the computation of the puzzle and the inclusion of new transactions into the block being constructed. It also takes into account the propagation speed of the new block which depends on its size. As a result, miners often have near bandwidth set in their new blocks. We do not distinguish between the exact bandwidth set and near bandwidth set in analysis.

E Proofs

E.1 Choice of D

We have looked into safe margins $D = 1$ and $D = 2$ in the analysis. We do not explore into $D > 2$ because (i) first, from $D = 1$ to $D = 2$, we observe a change towards smaller profitability space. Intuitively, there are more uncertainties when safe margins increase as undercutters rely on future bandwidth set to be less wealthy than the undercutting target block. Larger safe margins potentially reduce the premiums from each attack, and profitability conditions become tighter, which diminishes the total number of attacks for the same dataset. (ii) Second, when we apply avoidance techniques as a defense against undercutting, the avoidance for $D = 1$ is the strongest as profitability conditions for higher margins D are tighter. In other words, when we defend against $D = 1$ attacker, we defend against others as well. (iii) Third, the probability of catching up after being more than 2 blocks behind is small. We give more details about this argument below.

Problem Statement. Let m, n be the relative height of chain 1 and chain 2 after the forking point, α be the mining power on chain 2 and $\alpha \leq 0.5$. Let $\tilde{D} = n - m$ ($|\tilde{D}| < D$). Show that for $D > 2$, the probability of chain 2 winning when $\tilde{D} \leq -2$ is sufficiently small.

Proof. Let X denotes the waiting time for chain 1 until it extends by one and Y denotes the waiting time for chain 2 until it creates a new block. We calculate the probability of chain 2 winning as follows:

$$p = \Pr[\text{Chain 2 wins}] = \sum_{i=0}^{\infty} \Pr[(D - \tilde{D} + i) \cdot Y < (i + 1) \cdot X]$$

We know that $D - \tilde{D} \geq 5$.

$$p = \sum_{i=0}^{\infty} \alpha^{D-\tilde{D}+i}(1 - \alpha)^i < \sum_{i=0}^{\infty} \alpha^{5+i}(1 - \alpha)^i$$

We know α^5 and $(\alpha(1-\alpha))^i$ both take maximum at $\alpha = 0.5$ for $\alpha \in (0, 0.5]$. Let $\alpha = 0.5$, we have

$$p < \sum_{i=0}^{\infty} (\frac{1}{2})^{5+2i} = \frac{\sum_{i=0}^{\infty} 4^{-i}}{32} = \frac{1}{32} \lim_{n\to\infty} \frac{1-4^{-n}}{1-1/4} = \frac{1}{24}$$

We consider this probability to be relatively small. For a more practical $\alpha = 0.2$, $p < \frac{1}{2625}$ and we consider it to be small.

□

F Related work

First, we give more details to the example where undercutting is not rational and "petty compliant" may not be rational for a miner. Note that petty compliant miners break ties by selecting the chain with more unclaimed tokens (mainly because they can claim all of them in a setting without a block size limit). Let there be 33% honest mining power and 100 total tokens at this timestamp with at most 20 claimable tokens. For a 17% potential undercutter, there is 50% remaining PC mining power. If the undercutter does not attack, it expects to receive at most 3.4 from the next block. If it attacks and claims 10 in the first block, with a 17% probability this block appears before the main chain extends. If other PC miners stay on the main chain, they expect profits of $0.5/0.83 * 0.83 * 20 = 10$; if they follow the fork, they expect premiums of $0.5/0.67 * 0.67 * 20 = 10$. Shifting is not rational for the PC miner who owns the first block and may not be rational for others since they have already started working on the next block on the main chain for some time. Then the undercutter's expected returns are either $0.17 * (0.17 * 10 + 0.17 * 20) = 0.867 < 3.4$ (PC miners not joining) or $0.17 * (0.67 * 10 + 0.67 * 0.17/0.67 * 20) = 1.717 < 3.4$ (PC miners joining). Undercutting is not rational in this example.

Together with other non-compliant mining strategies. There have already been rigorous discussions on attacks related to mining strategies. Most notable attacks are selfish mining [8,27,21], block withholding [23,17,6,16,7], and fork after withholding [13]. Defenses against these game-theoretic attacks have also been studied [11,30,22,14,15]. It is possible to combine undercutting with other mining strategies like selfish mining [8,27,21] and block withholding [23,17,6,16,7]. For block withholding, because undercutters prefer larger mining power, the two attacks have opposite goals, so one needs to balance the computation resource allocation. Selfish mining purposely hides discovered blocks, while undercutting intends to publish a block and attract other miners. They do not share the same rationale, but we can schedule the two strategies and apply the one with higher expected returns at a certain time. In this work, we put our focus on the profitability and mitigation of undercutting, which affects the undercutting part of this strategy scheduler.

F.1 Undercutting and Avoidance in the wild

Optimization. In real-world implementations of undercutting, optimized conditions can be applicable based on mempool states, miner type composition, network latency, and possibly others. The performance can be improved if one can predict future transaction arrival. Avoidance parameters should be adjusted accordingly based on observations or more sophisticated models. To differentiate between normal forks and undercutting, one can examine the timestamp and differences of embedded transactions inside competing chain heads, and whether this happens regularly. Undercutter often starts attacking after the target block has been created and includes only part of claimable transactions.

Latency. Network and propagation latency have large impacts on mempool states, thus affecting the feasibility of undercutting. If some transactions are not broadcast across the network, undercutting is hard to implement, and it can be difficult to differentiate between normal and undercutting forks. One extreme case is when each transaction is submitted to only one mining pool and each pool holds a distinct mempool. Undercutting becomes infeasible.

Other Mitigations. There also exist other mitigation techniques. One such proposal is to implement the rule that transactions include the height of the latest chain head block, and they can only be included in blocks of higher height. For example, at time t_1, a new block is appended to the block at height h. Then transactions appearing during height h and $h+1$ are only allowed to be included in blocks of height higher than h. We note that the effectiveness of this defense technique is discounted by the size of the mempool.

Changes to Bitcoin Core codebase. The changes are mainly contained in "*miner.cpp*" with the parameter for threshold value T being set in consensus file "*consensus.h*" and "*policy.h*". Major changes specifically reside in $addPackageTxs()$ and a utility function $RemoveFromBlock()$ is added. We show the code snippet we add to the transaction selection function as follows:

Listing 1.1: Changes to function BlockAssembler::addPackageTxs

```
// This transaction selection algorithm orders the mempool based
// on feerate of a transaction including all unconfirmed ancestors.
// Since we don't remove transactions from the mempool as we select them
// for block inclusion, we need an alternate method of updating feerate
// of a transaction with its not-yet-selected ancestors as we go.
// This is accomplished by walking the in-mempool descendants of selected
// transactions and storing a temporary modified state in mapModifiedTxs.
// Each time through the loop, we compare the best transaction in
// mapModifiedTxs with the next transaction in the mempool to decide what
// transaction package to work on next.
void BlockAssembler::addPackageTxs(int &nPackagesSelected, int
    &nDescendantsUpdated)
{
   // until line 331
   while (mi != m_mempool.mapTx.get<ancestor_score>().end() ||
       !mapModifiedTx.empty()) {
```

```
// ... until line 395
if (!TestPackage(packageSize, packageSigOpsCost)) {
    // ... until line 406
    if (TestPackageForBS(packageSize, packageSigOpsCost)) {
        // add the failed transaction to next bandwidth set
        CTxMemPool::setEntries ancestors;
        uint64_t nNoLimit = std::numeric_limits<uint64_t>::max();
        std::string dummy;
        m_mempool.CalculateMemPoolAncestors(*iter, ancestors,
            nNoLimit, nNoLimit, nNoLimit, nNoLimit, dummy, false);

        onlyUnconfirmed(ancestors);
        ancestors.insert(iter);

        // Test if all tx's are Final
        if (TestPackageTransactions(ancestors)) {
            // Package can be added. Sort the entries in a valid
                order.
            std::vector<CTxMemPool::txiter> sortedEntries;
            SortForBlock(ancestors, sortedEntries);
            for (size_t i=0; i<sortedEntries.size(); ++i) {
                nFeesNext += sortedEntries[i]->GetFee();
                nNextBlockWeight +=
                    sortedEntries[i]->GetTxWeight();
                nNextBlockSigOpsCost +=
                    sortedEntries[i]->GetSigOpCost();
            }
        }
    }

    if (nConsecutiveFailed > MAX_CONSECUTIVE_FAILURES &&
        nBlockWeight >
            nBlockMaxWeight - 4000 && nNextBlockWeight >
            nBlockMaxWeight - 10000 ) {
        // Give up if we're close to full and haven't succeeded in
            a while
        assert(!nFees);
        float gamma = nFeesNext/nFees;
        float T = 0.63; // as an example, 30% strongest rational
            miner min{0.5,0.63}
        // T needs to be in consensus file
        CAmount nFeesCurrent = 0;
        if (gamma < T) {
            nFeesCurrent = std::floor(nFees* (1+gamma) / (1+T));
            // iterate over transactions at the end of the current
                block
            CTxMemPool::setEntries::iterator iit = inBlock.end();
            // CTxMemPool::txiter iit = inBlock.end();
            while (nFeesCurrent < nFees) {
                failedTx.insert(*iit);
```

```
                    RemoveFromBlock(*iit);
                    iit--;
                }
            }
            break;
        }
        continue;
    }
    // until the end
    }
}

void BlockAssembler::RemoveFromBlock(CTxMemPool::txiter iter)
{
    pblocktemplate->block.vtx.pop_back();
    pblocktemplate->vTxFees.pop_back();
    pblocktemplate->vTxSigOpsCost.pop_back();
    nBlockWeight -= iter->GetTxWeight();
    --nBlockTx;
    nBlockSigOpsCost -= iter->GetSigOpCost();
    nFees -= iter->GetFee();
    inBlock.erase(iter);

    bool fPrintPriority = gArgs.GetBoolArg("-printpriority",
        DEFAULT_PRINTPRIORITY);
    if (fPrintPriority) {
        LogPrintf("fee %s txid %s\n",
                  CFeeRate(iter->GetModifiedFee(),
                      iter->GetTxSize()).ToString(),
                  iter->GetTx().GetHash().ToString());
    }
}

// in miner.h : class BlockAssembler
    uint64_t nNextBlockWeight;
    uint64_t nNextBlockSigOpsCost;
    CAmount nFeesNext = 0;
```