Numerical Analysis of Private Mining with Reset

Andre Chen

June 2024

1 Problem Statement

This problem models a situation of two mining parties in a blockchain scenario.

- Party A, the adversary, constructs a private chain from a certain public base block. The height of the private chain (which does not include the public base block) is h_A.
- Party B, the honest miners, extends the public chain from the same public base block. The height of the public chain (which also does not include the public base block) is h_B.

Each party runs its own Poisson point process representing the times at which they mine a block. These point processes have rates λ_A and λ_B for parties A and B respectively. The situation is parameterized by the confirmation depth k, which is the minimum height that a chain must be in order to confirm the first block in that chain.

The goal of the adversary is to create a violation, while the goal of the honest miners is to prevent a violation. A violation is defined by the following condition:

$$h_A \geq k \wedge h_A \geq h_B(t) \tag{1}$$

Note that this differs from the traditional definition of "violation" in that, instead of requiring that both the public and private chain have the *same* height of at least k, it only requires that the private chain have a height of at least k. However, it can be seen that once this definition is satisfied, the adversary can always cause a violation to occur with certainty by contributing to the public chain to create $h_B = k$ before revealing k blocks of their private chain. In a sense, this is a definition for when the adversary has a "guarantee of violation". We choose to go with this definition because it simplifies the analysis while still being useful to the attacker. For the formal analysis, we do not need to consider what happens after a guarantee of violation is met.

The parties do not have control over their rate of mining. Technically speaking, if λ_P is the mining power of party P, then party P can choose any rate of mining between 0 and that rate. However, it is easy to see that in every situation, it is always optimal for both parties to mine at their maximum capable rate, so we assume that they do.

The only degree of freedom in this game is that the adversary may choose, at any time, to "reset" the private chain, meaning that they begin mining at the most recent honest block, with all of the blocks in the current private chain being discarded. This has the effect of setting both h_A and h_B to 0.

The adversary has knowledge of λ_A, λ_B, and the current values of h_A and h_B. The problem is to find a reset policy that will minimize the expected time until the first violation.

2 Simplification Into a Board Game

The problem can be reduced to a simple board game from the perspective of the adversary.

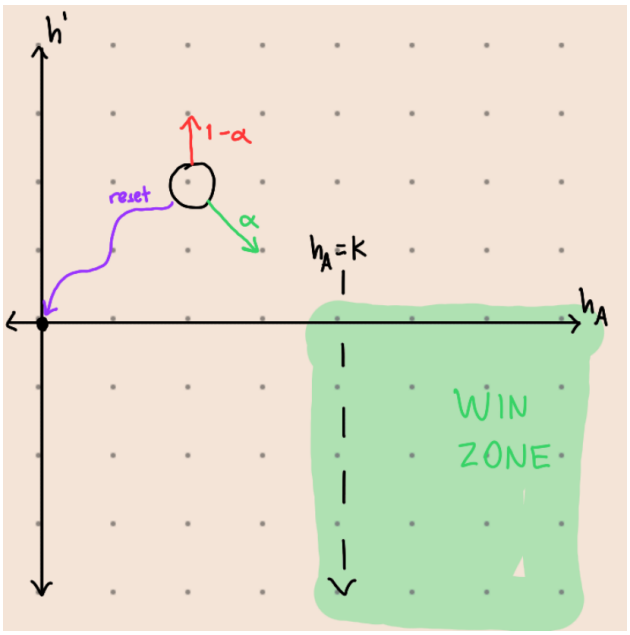

Figure 1: Two-dimensional representation of current state

2.1 Modeling Position

We model the current state of the system as the position of a player in two-dimensional integer coordinate system as shown in Figure 1, where the position along the horizontal axis represents the current h_A and the position along the vertical axis represents the difference between the chain heights $h' = h_B - h_A$. The player position can be updated in the following ways.

- When the adversary mines a block, h_A increases by 1 and the chain height difference h' decreases by 1. This is analogous to the player moving down 1 unit and right 1 unit.
- When the honest miners mine a block, h_A is unaffected and the chain height difference h' increases by 1. This is analogous to the player moving up 1 unit.
- When the adversary chooses to reset, the player is teleported back to the origin.

A violation occurs when the player enters the "win zone", which is the region where $h_A \geq k$ and $h' \leq 0$.

2.2 Modeling Time

We define time in this model as a sequence of discrete steps. In order to create a faithful mapping between the discrete time of the board and the continuous time of the real situation, we define a single Poisson process that captures the mining of both the adversary and honest miners: this is simply the sum of their respective Poisson processes, where events are block mining times occuring with rate $\lambda_A + \lambda_B$ and each block mined has an α chance of being by the adversary (otherwise by the honest miners). α is defined as the fraction of adversarial mining power.

$$\alpha = \frac{\lambda_A}{\lambda_A + \lambda_B} \tag{2}$$

If we are only given in advance all the times that events occur in this combined Poisson process, we can faithfully simulate the entire system; in this sense, the combined process "drives" the system. In modeling the combined process this way, the time it takes for an event to occur is completely independent from what happens once the event occurs. This independence allows us to calculate the expected time until violation as follows.

$$\mathbb{E}[\text{time until violation}] = \mathbb{E}[\text{steps until violation}]\mathbb{E}[\text{time per step}] \tag{3}$$

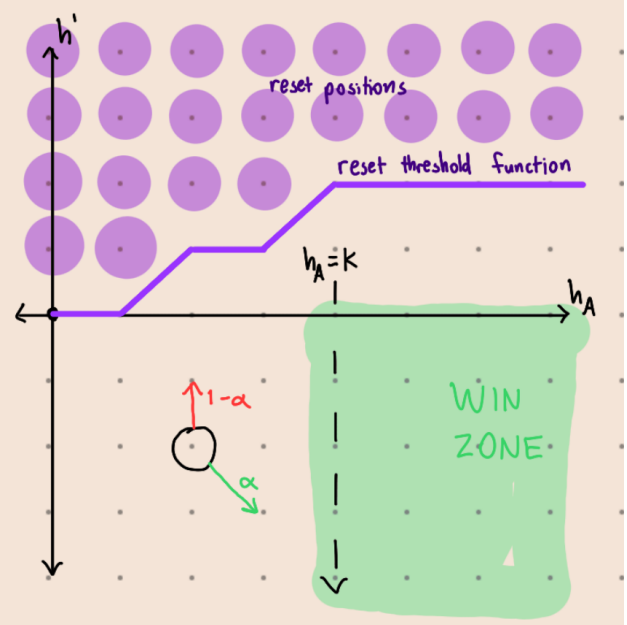

Figure 2: Reset policy representation on the board game

As such, we now only need to calculate the expected steps until violation.

2.3 Reset Policy

The discretization of steps remains faithful even when we introduce the fact that the adversary may reset at any time, because the optimal behavior for the adversary is to reset as soon as the system updates to an unfavorable position. The memoryless property of the driving Poisson process ensures that if no event has occurred, there is no reason for the adversary to change its decision on whether to reset; therefore the decision to reset depends only on the current position on the board.

A reset policy can then be viewed as a set of positions (termed "reset positions") on the board where adversary chooses to reset, as shown by the purple dots in Figure 2. The player moving to one such position will instantly be teleported back to the origin. We can further refine our view of a reset policy by making the following observations about an optimal reset policy.

- For a given h_A, every reset position lies above (i.e. has a greater h' than) every non-reset position. This is because there is no situation in which the adversary would want to reset, but would choose not to if the honest miners had an extra block—honest blocks strictly harm the goal of violation. This means that each h_A value has some h' threshold above which the adversary wants to reset. [1]

- For a given h', every reset position lies to the left of (i.e. has a lesser h_A than) every non-reset position. This is because there is no situation in which the adversary would not want to reset, but would choose to do so if he had an extra block—adversarial blocks strictly benefit the goal of violation. This means that the aforementioned h' threshold is monotonically non-decreasing with h_A.

- For $h_A = 0$, the aforementioned threshold is 0. Above this position, resetting strictly benefits the goal of violation; below it, resetting strictly harms it.

[1] This also addresses the first point of the extra credit. For the goal that we have defined, minimizing time until guarantee of violation, the adversary will never prefer a position that is strictly greater in the number of honest blocks. Resetting to any base block other than the most recent base block (analogous to any point above the origin) is the same as resetting and then giving the honset miners free blocks. This may change if the definition of violation is changed to require that the public and private chains be at the same height; then the attacker may benefit from allowing the honest miners to make some headway first.

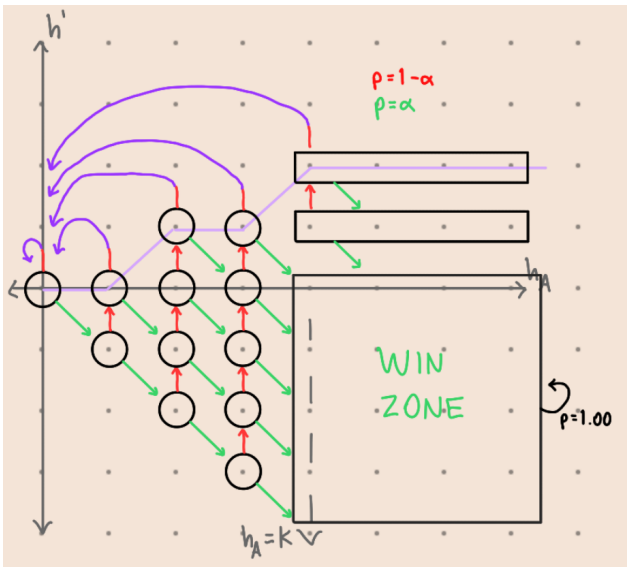

Figure 3: Representation of the game as a Markov chain

- For $h_A \geq k$, the aforementioned threshold is some constant value. This is because at that point, the horizontal axis of the board game stops mattering: the game would be isomorphic to one in which the player only moved up and down, rather than up and down-right. Intuitively, this makes sense as once the private chain reaches the required confirmation height k, the adversary now only cares about catching up to the public chain.

These properties mean that all candidates for the optimal reset policy can be modeled as some threshold function $r : \mathbb{N} \to \mathbb{N}$, which takes as input the current h_A value and outputs the threshold h' value above which the adversary should reset. $r(0) = 0$, r is monotonically non-decreasing over $[0, k]$, and r is constant for $h_A \geq k$. The curve in Figure 2 shows visual representation of r.

2.4 Calculating Expected Time Until Violation

With all these simplfications in hand, we can calculate the expected number of steps until first violation by converting the board into a Markov chain. Instead of having one node in the Markov chain for the infinite number of positions (h_A, h'), we only need a node for every *reachable* position, which is finite. Positions that lie above the curve defined by the reset threshold function r are not reachable, since a player going there will instantly teleport back to the origin: edges that would have gone to these positions go to the node representing the origin instead. Positions that have $h' < -h_A$ are likewise not reachable from the fact that every move downward must also be a move rightward. The constancy of the system after $h_A \geq k$ can be exploited by consolidating all positions where $h_A > k$ into the corresponding position where $h_A = k$. Finally, the entire win zone can be represented as one big absorbing state. Figure 3 shows one such Markov chain.

Define P to be the transition matrix for this Markov chain, with $P[i, j]$ being the chance of transition to state j if the player is in state i. The first state is the origin state, the last state is the aborbing win state, and the other states can be in any order in between. P must take the following form.

$$P = \begin{bmatrix} Q & R \\ \mathbf{0} & 1 \end{bmatrix} \tag{4}$$

According to the Wikipedia article on absorbing Markov chains, the expected number of steps before entering the aborbing state from the first state (the origin state) is given by:

$$((I - Q)^{-1}\mathbf{1})[1] \tag{5}$$

where $v[1]$ denotes the first element of v and $\mathbf{1}$ is the column vector of all ones.

This gives us a method to calculate the expected number of steps it takes to create a violation given a reset policy[2].

Finally, we know that the expected time per step is just the mean of the exponential distribution with rate parameter $\lambda_A + \lambda_B$; this mean is $(\lambda_A + \lambda_B)^{-1}$. We can multiply the expected number of steps by the expected time per step, as per Equation 3, to obtain the expected time until violation.

$$\mathbb{E}[\text{time until violation}] = \frac{((I - Q)^{-1}\mathbf{1})[1]}{\lambda_A + \lambda_B} \tag{6}$$

3 Implementation and Analysis

The function `analysis::calculate_expected_steps` of the provided Rust code is given the fraction of adversarial mining power α and the reset policy as a sequence of integers representing the threshold values of the reset threshold function r (the first element should always be `0`, as $r(0) = 0$, and all elements after the kth element are taken to be equal to the kth element), and outputs the expected number of steps it will take to create a violation. This function uses the method described above: it creates a transition matrix Q based on the Markov chain resulting from applying the given reset threshold function, and outputs $((I-Q)^{-1}\mathbf{1})[1]$. The confirmation depth k is indicated to the function by the length of the `reset_policy` argument: k is always one less than `reset_policy.len()`.

We ran this function for all combinations of $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$ and all reset policies with no threshold greater than 5. We then selected the best reset function for each α and plotted them on a graph, grouping plots by confirmation depth k. The results are shown in Figure 4. Curves in a cooler color represent the best policies for low α; warmer colors correspond to higher α. The best policy for the recommended values of $k = 5$ and $\alpha = 0.3$ is shown in Figure 5; for these parameters the expected number of steps until violation is about 36.88425, making the expected time until violation:

$$\min \mathbb{E}[\text{time until violation}] = \frac{36.88425}{\lambda_A + \lambda_B} \tag{7}$$

We can see that, in general, the best reset curves are those that quickly reach whatever max threshold they are willing to tolerate before tapering off, attaining a shape similar to the log or $\sqrt{\cdot}$ functions. These curves tend to be higher the greater k is, which makes sense as the resetting means losing more work if the confirmation depth is higher. These curves also tend to be higher the greater α is, which makes sense as the adversary is willing to tolerate more height difference if they have a greater chance of catching up from being lucky.

Up to $k = 6$, the best reset policy curves never reach a threshold value of 5. We can expect that, after a certain point, higher curves will lead to worse expected times (because it means that the adversary is waiting too long for the honest miners to run away before resetting). In the absense of there being a local maximum that would make this assumption false, we can trust that these curves really are the best possible policies. At $k = 7$ and above, however, the reset policy curves begin to reach their maximum threshold of 5; this limitation of 5 is artificial and merely a result of not being able to scan the entire infinite parameter space of valid reset policy curves. It therefore may be the case that there exists a better reset policy curve that utilizes a higher threshold than 5, but this data will not show that. Future work could involve coming up with a more clever way of searching the space of all reset policy curves (such as gradient descent, rather than simply enumerating every single one below a certain threshold value), which would allow for faster calculations without an artifical limit.

4 Accounting for Propagation Delay

This model can be modified to determine a lower bound for the expected value when accounting for propagation delay. The effect of propagation delay can be summarized as each honest block having some chance of not being a pacer. If an honest block is not a pacer, then it does not contribute to the height of the public chain. This corresponds to an additional type of move: where previously there were only two possible moves

[2]This analytical method was corroborated by simulations; see the `simulation.rs` module of the code.

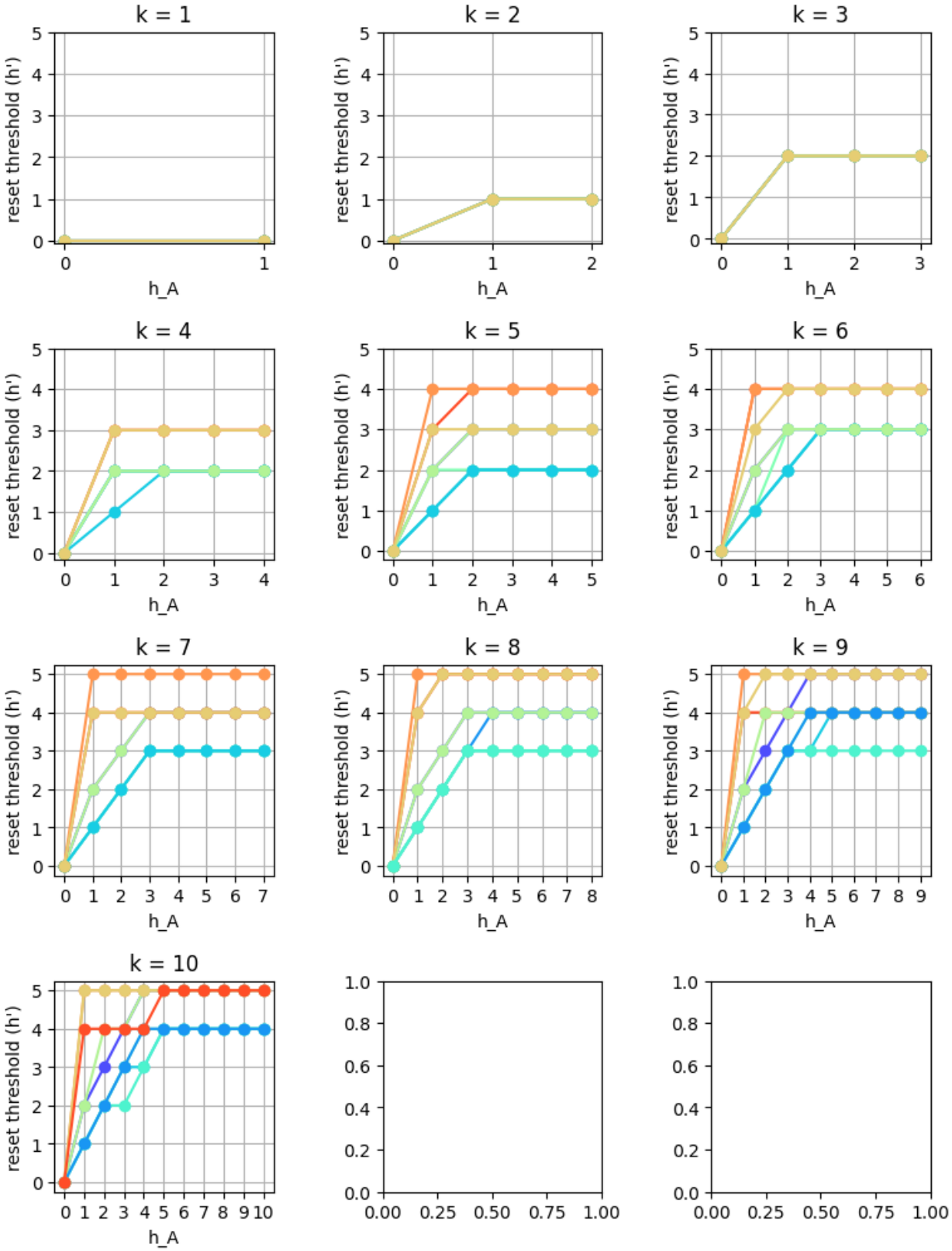

Figure 4: The best reset threshold function for every tested k and α

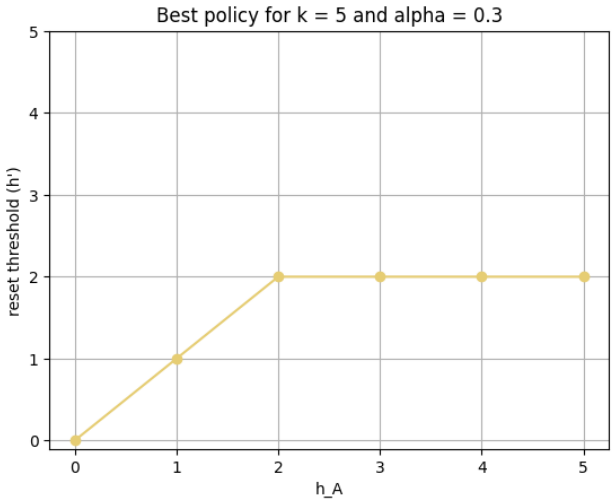

Figure 5: The best reset threshold function for $k = 5$ and $\alpha = 0.3$

(up for an honest block, down-right for an adversarial block), with propagation delay, there is a move where the player doesn't change position at all.

In the Poisson process for the honest miners only, the time between consecutive events follows an exponential distribution, whose CDF is given below.

$$\text{CDF}(t) = 1 - \exp(-\lambda_B t) \tag{8}$$

We consider the situation where the blocks are maximally delayed by Δ; that is, after a pacer block is mined, any block that is mined within Δ time of the pacer does not contribute to the height of the chain (any block mined more than Δ from the previous pacer is a pacer). Thus, given that the previous block is a pacer, the probability that the current block is noncontributory is given by:

$$\text{CDF}(\Delta) = 1 - \exp(-\lambda_B \Delta) \tag{9}$$

Although not every block follows a pacer, every block's chance of being noncontributory cannot exceed this probability. Therefore, if we apply this probability of being noncontributory to every block, we obtain the best-case scenario for the adversary. Now, each state transition in the Markov chain can be one of the following.

- With probability α, the player moves down-right, possibly winning.
- With probability $(1 - \alpha)(1 - \exp(-\lambda_B \Delta))$, the player stays put (self-edge in the Markov chain).
- With probability $(1 - \alpha)(\exp(-\lambda_B \Delta))$, the player moves up, possibly resetting.

The quantity $\lambda_B \Delta$ is the only factor that determines how propagation delay affects the system. We term this the "rate-delay product", and it represents the number of blocks that you expect to be mined within Δ of the previous block. Alternatively, it represents the ratio between the propagation delay Δ and the expected inter-block mining time.

The function `analysis::calculate_expected_steps` of the provided Rust code takes a parameter named `rate_delay_product`; this quantity is incorporated into the transition matrix according to the above transitions. In the previous sections, only calculations with this parameter set to zero were shown, representing no propagation delay. For every parameter set sampled in the previous section, we also ran the function once for rate-delay products in $\{0.0, 0.1, \ldots, 1.0\}$; that is, we ran this function for all combinations of $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$, $\lambda_B \Delta \in \{0.0, 0.1, \ldots, 1.0\}$, and all reset policies with no threshold greater than 5. We then plotted the expected steps until violation as a function of α and $\lambda_B \Delta$, grouping by values of k. The results are shown in Figure 6.

Keep in mind that when the propagation delay is not zero, the curves represent lower bounds. Thus, the actual expected number of steps until violation will be between the value showed in the graph at the desired rate-delay product and the value shown in the graph when rate-delay product is zero.

We see that the attacker benefits as the rate-delay product increases. In addition, the attacker benefits more for higher confirmation depths (and not at all for the lowest confirmation depth of 1). The worst-case effect of rate-product delay on expected steps until violation for the recommended values of $k = 5$ and $\alpha = 0.3$ is shown in Figure 7.

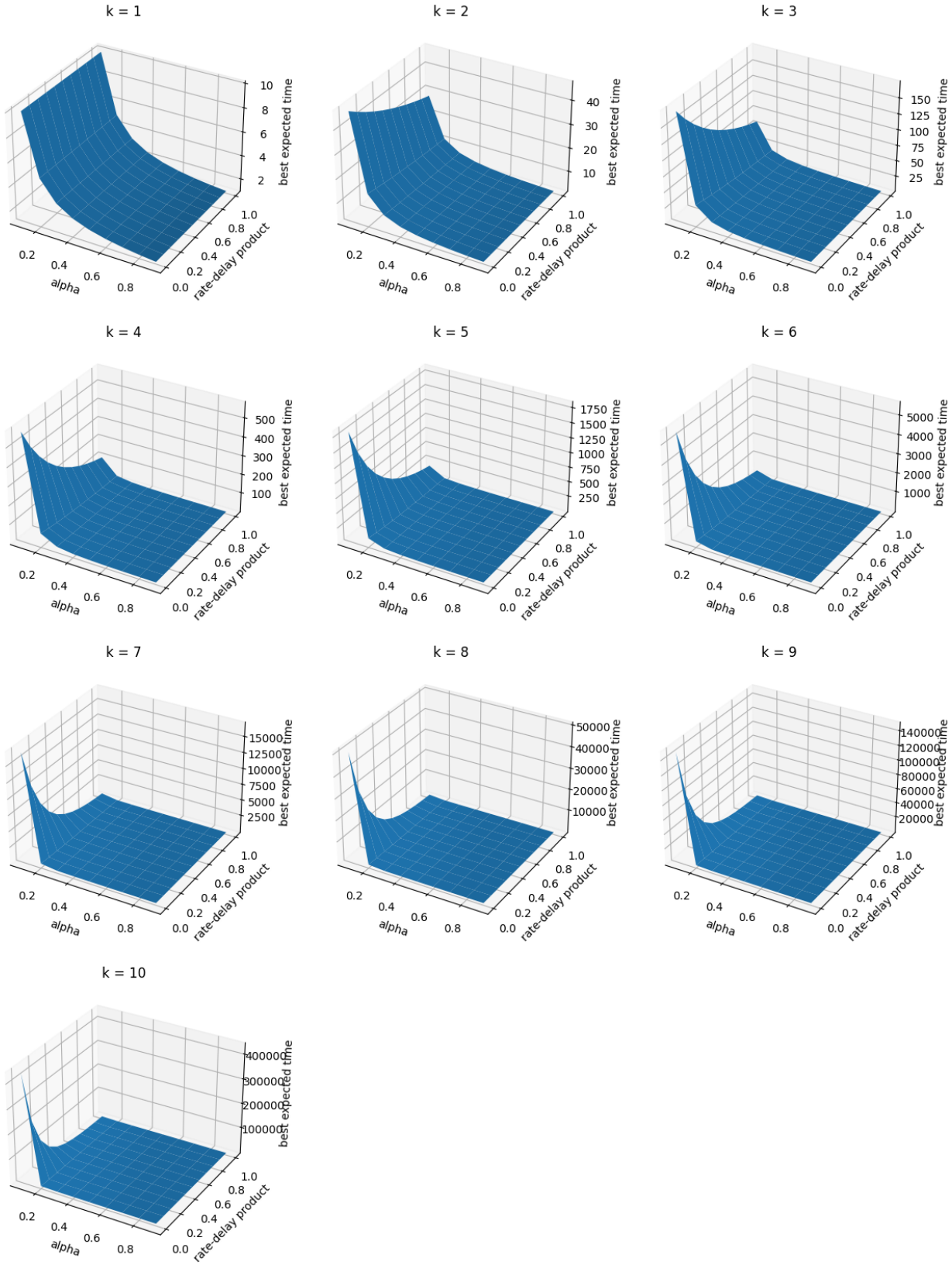

Figure 6: Worst-case effect of delay on expected steps until violation for all tested k, α

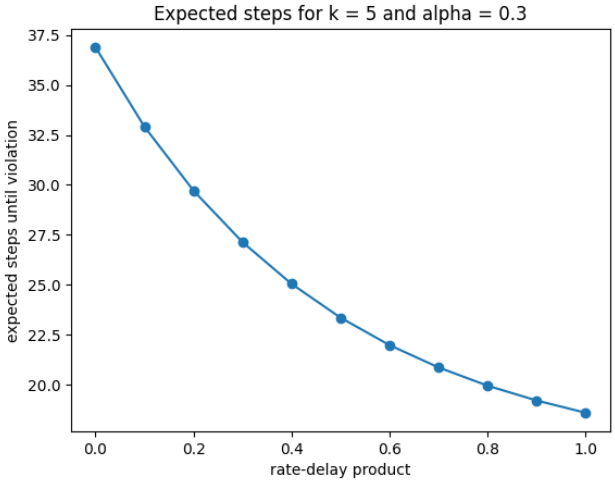

Figure 7: Worst-case effect of delay on expected steps until violation for $k = 5$, $\alpha = 0.3$