Optimal Cheating in Algorand's Proof-of-Stake Protocol

Catherine Yu

7 May 2021

Abstract

In this paper, we analyze Algorand's Proof of Stake protocol, specifically analyzing the distribution of expected rewards for different strategies that an adversary can play to optimize their earnings. We define two strategies and simulate their expected rewards using exponential distributions to better understand how well they perform given stake α, and we compared them using graphical analysis. While these strategies can be played for an infinite amount of rounds and the adversary owns an infinite number of winners in practice, we made two simplifications in our model by truncating the number of rounds played and the number of winners considered per round. Thus, we proved theoretically that the error induced by these two simplifications is extremely small, which guarantees the correctness of the simulation's results.

1 Introduction

Bitcoin was started in 2008 by Satoshi Nakamoto as one of the first peer-to-peer networks for electronic transactions [1]. Nakamoto introduced a decentralized currency system where there is no administrator in charge but rather a public ledger of transactions that leaves out the role of a third party intermediary. In this system, data is structured and stored in blocks, such as transactions, and as blocks fill up in storage they are chained onto the previous block, which forms the "blockchain". Mining blocks is the process of creating new blocks to add onto the blockchain, which is achieved by solving a computational puzzle in Bitcoin. Users that participate in the protocol to mine new blocks are called block producers. A reward is given to miners of new blocks, which incentivizes block producers to participate in mining blocks in the blockchain. Different cryptocurrencies guarantee security and efficiency in different ways. In the following section, we will discuss two different protocols that govern cryptocurrencies.

1.1 Proof of Work vs. Proof of Stake

Bitcoin is governed by a protocol called Proof of Work, meaning that block producers are rewarded in proportion to the amount of computational power, or "work". By increasing the amount of computing power they own in their network, notably with supercomputers that are specially built to mine blocks, block producers can claim larger rewards. However, because the chance of mining the next Bitcoin block is dependent on the amount of "work" power one owns, it has led to the consolidation of miners into large mining pools, which defeats the purpose of being decentralized system and original goal of being "for the people". Moreover, Bitcoin has been criticized for energy wastefulness due to the high amounts of electricity needed to power and maintain mining computers. Thus, many other systems have proposed alternatives to the Proof of Work protocol which aim to combat the protocol's weaknesses.

One of these alternatives is called Proof of Stake, which means that block producers are rewarded in proportion to the amount of their stake in the system, or how much coin they own. Because mining the next block is dependent on how much coin the block producer owns in the system, the need to increase computational power and the resources necessary to do so simply disappears. Proof of Stake cryptocurrencies also provide stronger security guarantees and promise a truly decentralized system without incentives for pooling of power. Algorand is one such cryptocurrency that is governed by the Proof of Stake protocol [2]. By analyzing Algorand's implementation of the protocol, we can better understand Proof of Stake's strengths and make sure that they are actually attained by the system.

2 Background and Setup

In this section, we provide important background and setup details for the rest of the paper. We first discuss Algorand's procedure of leader selection to see how Proof of Stake is implemented in practice in 2.1, and then we explain how we model this procedure using exponential distributions in 2.2. Lastly, we define strategies that a dishonest player can play in order to improve their expected rewards in the system in 2.3.

This paper builds on Sally Hahn's independent work titled "Incentive to Cheat: Algorand's Proof of Stake Protocol" submitted for the Program in Applied and Computational Mathematics [3]. Sally's paper analyzed two possible strategies for cheating in Algorand's protocol by using simulations, showing that there is indeed incentive for block producers to cheat because the extent of the rewards they can gain increases with larger stake in the system. Results and work that we carry over from her paper are in section 2.3.1.

2.1 Algorand's Implementation of Proof of Stake

We can look at Algorand's leader selection to understand how Proof of Stake is implemented in the system [2]. We can represent the rth block, or the block mined in the rth round, as $B^r = (r, PAY^r, Q^r, H(B^{r-1}))$. PAY^r is the set of payments or transactions in the system for that round. At the start of round r, all players know the chain of blocks so far, $B^0, B^1, \ldots, B^{r-1}$. A potential leader of round r is a user i such that

$$H[Sig_i(r, 1, Q^{r-1})] < p$$

Let us explain the parts of this criterion.

1. r is the round number.

2. H is a Hash function that outputs a random but deterministic 256 bit string as a function of $Sig_i(r, 1, Q_{r-1})$. In this paper and in line with literature, H is treated as a random oracle, meaning that H maps inputs to a uniformly random element in the interval $[0, 1]$.

3. $Sig_i(r, 1, Q^{r-1})$ is the digital signature of coin (user) i. It outputs a binary string as a function of i and $(r, 1, Q^{r-1})$. Every coin's digital signature will output a different string sequence with the same inputs, and the signature is private to the coin (owner). This means that it is not feasible for users other than i to compute the signature without knowing the private key of coin i.

4. Q^{r-1} is a quantity determined from the previous block B^{r-1}. Therefore, the potential leaders of a round is influenced by the block of the previous round and not anticipated beforehand.

5. p is chosen to be a small enough value so that there are not too many potential leaders, but large enough that there will be potential leader for most rounds. Algorand sets p to be sufficiently large so that the probability of a stall is tiny, i.e. they expect thousands of potential leaders each round.

The actual leader of the round is the coin with the lowest value of $H[Sig_i(r, 1, Q^{r-1})]$. If we assume that there is at least one potential leader every round, the leader coin i_r^* is $i_r^* = \arg\min_i\{H[Sig_i(r, 1, Q^{r-1})]\}$.

2.2 Modeling with exponential distributions

In [3], the process of leader selection to determine the next block in each round was modeled by drawing a uniformly random number for each coin with seed the coin that won the previous round, and the coin that has the lowest value wins the round. However, in practice, this is not the most efficient setup. In this paper, we choose to model the process of leader selection using **exponential distributions**.

First, we define a coin in the context of this paper.

Definition 2.1 (Coin c_i)**.** For each coin c_i owned by user i, there is a list of values $\{c_{i_j}^1, c_{i_j}^2, \ldots, c_{i_j}^k, \ldots\}$, for all integers k and indices $i_1, i_2, \ldots, i_j$, which denotes what is the value of the jth smallest coin in the kth round that follows when you selected coin $c_{i_j}^1$, then $c_{i_j}^2$, and so on during previous rounds.
Thus, we represent coins in this system as their hash values and not necessarily the coins themselves. In other words, $c_{i_k}^r$ is the kth smallest hash of user i during round r after hashing user i's infinitely many coins. Thus, $c_{i_k}^r$ is not really a coin, but rather the hash of some coin. We will abuse notation here to refer to $c_{i_k}^r$ as the kth smallest coin in round r.

We represent $H[Sig_i(r, 1, Q^{r-1})]$ as simply c_i^r, the coin of user i for round r. For every account i, each round c_i^r is drawn from an exponential distribution with rate equivalent to the balance of the account i. Then, if the adversary A has α stake in the network, the A's coins are drawn from an exponential distribution with rate α, where α is some fraction, i.e. $0 < \alpha < 1$. For the rest of the network, N, all of their coins can be represented by a single exponential distribution with rate $1 - \alpha$.

Definition 2.2. Let X_A represent the adversary A's coins, and let X_N represent the rest of the network N's coins. Then, if A has α stake and N has $1 - \alpha$ stake,

$$X_A \sim Exp(\alpha), c_A^r \in Exp(\alpha)$$
$$X_N \sim Exp(1 - \alpha), c_N^r \in Exp(1 - \alpha)$$

That is, A's coins are drawn from an exponential distribution with rate α, and N's coins are drawn from an exponential distribution with rate $1 - \alpha$.

Modeling with exponential distributions has the same result as picking a uniformly random coin, but is more efficient and removes complications associated with drawing many uniformly random values. For example, in practice, drawing many values from a uniform distribution does not capture fractional stake, and not everything is necessarily divisible by integers so we cannot draw randomly well to $0.00\ldots1$ places. Using exponential distributions provides our model with two important properties: the model is **Sybil-proof** and **collusion-proof**, meaning that it does not help participants to spread their coins out among many accounts or merge all their coins together into one account. Moreover, the exponential distribution is the only distribution that satisfies these properties. Therefore, Algorand actually uses exponential distributions in practice to implement their system.

2.3 Strategies

There are many different ways that an adversary can participate in the system given the information they have and the actions they can take. Each player in the system has a certain stake, which determines their coins. In every round, the player can choose to broadcast a coin to the other players, or not broadcast anything. We now discuss different strategies that the adversary can take to possibly cheat in the system.

2.3.1 Work from Sally's Paper

Sally considered two strategies, which she called the j-min strategy and the j-min under uncertainty strategy [3]. The first strategy is when the adversary has knowledge of all coins in each round, and the second is when the adversary only has knowledge of their own coins in each round.

In the **j-min strategy**, the adversary owns an alpha fraction of all coins in network, and can see all coins at round r. Adversary A can calculate coin values for all the coins A owns in round $r + 1$ if A knows what wins round r.

The result of analyzing the j-min strategy is that total returns increase with respect to alpha, implying that the adversary has an incentive to cheat. Specifically, if the adversary has the j minimum coins in round r, they can choose to broadcast one of the j coins in round r that will maximize chances of winning in the future, rather than always broadcasting their smallest coin, which is what an honest player does.

In the **j-min under uncertainty strategy**, the adversary does not have public knowledge of all the coins in a round, and thus can only consider their j lowest coins. They must choose one coin among j coins with the highest score.

Sally proves two propositions in her paper, which we restate here with proofs using exponential distributions.

Proposition 2.1. $\mathbb{P}(|c| = j) = \alpha^j \cdot (1-\alpha)$. *If adversary A owns α fraction of the coins in the market, then this is the probability that adversary A owns the j-smallest coins.*

Proof. This simply follows from the fact that the probability that the lowest, second lowest, etc. to the jth lowest coin is owned by A is each α, and the probability that the $j+1$th lowest coin is not owned by A is $1-\alpha$. □

Proposition 2.2. *A owns j-min coins. A can select which to broadcast. A will win the next round if their coin is lower than everyone else's coin.*
Then, $\mathbb{P}(A \text{ wins the next round} \mid \alpha, j) = \frac{j\alpha}{1+(j-1)\alpha} = \frac{j\alpha}{1+j\alpha-\alpha}$.

Proof. In round r, A has the j smallest coins. The seed for A's coins in the next round, $r+1$, is an exponential distribution with rate $j \cdot \alpha$, which we denote as the random variable X_1. Everyone else in the network's seed in the next round is an exponential distribution with rate $1-\alpha$, which we denote as the random variable X_2. The probability that A wins round $r+1$ is thus $\mathbb{P}(X_1 < X_2)$, and by properties of exponential distributions we know that $\mathbb{P}(X_1 < X_2) = \frac{j\alpha}{1-\alpha+j\alpha}$. □

2.3.2 Strategies in this paper

There are three strategies that we discuss and analyze in this paper.

Definition 2.3 (HONEST strategy)**.** In the HONEST strategy, the adversary acts honestly, meaning that they always broadcasts their smallest coin each round without considering any other factors, such as everyone else's coins. Then, if the adversary plays only the honest strategy and they own α stake in the network, they are expected to win α future rounds in expectation.

Definition 2.4 (OPT strategy)**.** The OPT strategy represents the optimal strategy that the adversary can play in the system given their stake α, and it is the strategy that we mainly analyze in this paper. For this strategy, the adversary tries to win as many future rounds as possible, because if they win a round, their block will seed the next round's hash function, giving them more information to continue winning. Moreover, the adversary has the ability to calculate all of their future coins, assuming that they continue winning rounds with their coins.

If, after the rest of the players in the system broadcast their coins, the adversary knows it has j potential winners, then the adversary can compute the expected future chain of rewards for each of its j possible seeds, and pick the seed which gives them the most expected future rewards. This means that the adversary may choose a coin that does not necessarily give them the best chance of winning the next round, but does give them the best chance of winning 5 consecutive rounds.

Definition 2.5 (COIN strategy)**.** The COIN strategy is another strategy that we analyze that does not require the adversary to know the rest of the network's coins in each round. In each round, if the adversary has j winners, they will choose the coin that has the highest expected reward from winning the current round. The COIN strategy should not give as much expected rewards as the OPT strategy because the adversary does not have as much information to strategize and calculate all of their future earnings each round.

It is important to note that all of these strategies assume the adversary is playing for an infinite number of rounds and has an infinite number of coins, which is what happens in practice. We will be discussing these strategies in the next two sections of the paper.

3 Simulation

We approximated the distribution of expected rewards of the OPT strategy and the COIN strategy by simulating an adversary in the system who continuously improves on their expected rewards by learning from previous rounds played. The simulation eventually reaches a point where the expected rewards no longer changes by a significant amount, at which the distribution of expected rewards of a strategy has been attained, and also implying that rewards gained from cheating the system are finite. We analyzed the strategies by plotting different values of α's, i.e. proportions of stake in the system, against the expected number of future rounds won with that amount of adversary power.

We first formalize a set of definitions used to model the simulation in 3.1. Then, we explain motivation and high-level design for the simulation in 3.2, and we provide the details and pseudocode in 3.3. The analysis and graphs for the OPT strategy are in 3.4, while the analysis and graphs for the COIN strategy are in 3.5. Lastly, we compare the OPT strategy with the COIN strategy in 3.6.

3.1 Definitions

In Section 2, we defined c_i^r as the coin of user i for round r. We extend the definition of a coin here for the purposes of our simulation.

Definition 3.1 (Coin c_i, extension from Definition 2.1)**.** For each coin c_i owned by user i, there is a list of values $\{c_{i_j}^1, c_{i_j}^2, \dots, c_{i_j}^k, \dots\}$, for all integers k and indices $i_1, i_2, \dots, i_j$, which denotes what is the value of the jth smallest coin in the kth round that follows when you selected coin $c_{i_j}^1$, then $c_{i_j}^2$, and so on during previous rounds.

In each round r, user i with stake α draws its coins $c_{i_k}^r$ from the exponential distribution $Exp(\alpha)$. Because exponentials are memoryless, we get the $k+1$th coin by adding to the kth coin, in other words, the first coin is $c_{i_1}^r$, the second coin is $c_{i_1}^r + c_{i_2}^r$, and the ith smallest coin is $\sum_{j \leq i} c_{i_j}^r$.
There are infinitely many future "paths" that a coin c can have because the adversary knows all the coins that they own in the future, given that they win with this coin c and win with all the following coins. This represents **all the information the adversary has** when they make a decision, i.e. when they must decide which coin to broadcast to everyone else.

Recall that in each round, every player can choose to broadcast a coin (or not broadcast), and the lowest coin "wins" the round, meaning that it determines the following round's seed for coins and the player who owns the lowest coin gains the reward for that round. In this paper, expected rewards are synonymous with the expected number of rounds won in the future, while in practice there may be some reward W given to each player for winning a round. Thus, the expected rewards in this paper could be multiplied by this W reward to represent the real world rewards of block mining.

Definition 3.2 ($R(c)$)**.** Given a coin c, we can compute $R(c)$, which we define as the expected number of rounds you will win in the future given that you just won with coin c, or the **expected reward** we gain from coin c.

Definition 3.3 ($D(\alpha)$)**.** We define $D(\alpha)$ to be the distribution of expected rewards given that your stake in the system is α. In other words, $D(\alpha)$ the distribution of $R(c)$. For notation purposes, D is interchangeable with $D(\alpha)$. There is always a given α stake used in the simulation.

Let us denote S as the **strategy** used to decide which coins to play. Then, we have the following definition.

Definition 3.4 ($R^S(c)$ and $D^S(\alpha)$)**.** $R^S(c)$ is defined as the expected number of rounds you will win in the future, given that you just won with coin c using strategy S; and $D^S(\alpha)$ is defined as the distribution of $R^S(c)$, where $c \leftarrow Exp(\alpha)$ and strategy S is used.

3.2 Motivation and Design

We want to know $D^S(\alpha)$ to better understand what the rewards look like in the system we are studying, so we simulate the process of drawing coins and computing expected rewards based on increasingly optimal strategies.

In this simulation, we begin with D_0 and S_0, an initial distribution and strategy, and we apply a process called f in which we first draw coin values, then we draw each coin's expected reward from D_0, and lastly we output the total expected reward from each of these coins, which becomes $f(D_0) = D_1$, the next distribution of expected rewards. In each following iteration i, we improve upon the strategy S_{i-1} and draw expected rewards from D_{i-1} to get the next best strategy S_i and distribution of rewards D_i. The optimal strategy (OPT) has $f(D) = D$, so the goal of this simulation model is to find a D that is invariant under f.

After we go through the simulation, we should get a D^S that is invariant under a certain set of conditions, specifically, the expected reward does not differ by more than some value for some number of consecutive rounds. This represents D^{OPT}.

Definition 3.5 (E^S)**.** We define E^S to be the expected number of future rounds won in a row using strategy S, assuming you just won the previous round. This is slightly different from $R^S(c)$, which is a function of a coin c. E^S represents the overall expected reward from the distribution of rewards D^S.

At the end of the simulation, we analyze the distribution D^S by calculating and plotting E^S.

3.3 Pseudocode

Initializing D_0. There are two ways we can initialize D in our simulation. First, we can define D_0 to be a point mass at 0 and S_0 to be the strategy where we never broadcast any coins. Second, we can initialize D_0 to D^{HONEST}, which is the honest distribution, and S_0 to be the HONEST strategy (see Definition 2.2). When we play honestly, we always output the smallest coin in every round. Thus, we can compute $R^{\text{HONEST}}(c)$ as follows:

- Probability that you win at least 1 round is equal to the probability that their exponential of rate $(1-\alpha)$ exceeds c_1
- Probability that you win at least 2 rounds is equal to the probability that their exponential of rate $(1-\alpha)$ exceeds c_1, AND their second draw exceeds c_{11}
- Continue on by calculating the probability that you win at least k rounds.

We chose to use the former initialization of simply defining D_0 to be a point mass at 0 for simplicity, and because it outputs honest expected rewards after one iteration.

Modifiable parameters. There are five parameters that can be set in the code for the simulation.

1. α, the fraction of the network owned by the adversary.
2. k, the number of coins that the adversary owns (which is also the size of rewards array).
3. n, the size of the distribution D, which represents the number of rounds in the future that the adversary plays or takes into account.
4. The two conditions used for determining invariance of D^S, which is a certain difference in expected reward between iterations and the number of consecutive iterations that the reward does not change by.

The two parameters k and n are the finite cutoffs we decide on to simplify the system model, and we will use them again in Section 4. We chose to use $k = 100$ and $n = 10,000$ based on empirical results of running the simulation multiple times. With larger k and n, runtime is longer but results are more precise. Setting k and n to these two values allowed for reasonable runtime and gave relatively stable results.

Simulation procedure. In the procedure, c_i represents the adversary's ith coin of the first round. To draw from D_{i-1}, we should follow the following process f.

1. Draw c_1 from $Exp(\alpha)$. For all $1 < i \leq k$, set $c_i := c_{i-1} + Exp(\alpha)$.
2. For all $1 \leq i \leq k$, draw r_i from D_{i-1} i.i.d., i.e. sample expected future rewards from the previous distribution D_{i-1}.
3. Output reward: $R^{S_i} = \sum_{i \geq 1} \mathbb{P}(\text{first } i \text{ coins win}) \cdot \max_{j \leq i}\{1 + r_j\}$. This reward becomes part of D_i. The 1 in $1 + r_j$ represents 1 more future round, but could be easily replaced with W, some amount that the adversary would gain for winning the round. See line (3) for the mathematical expression.

Let us now explain each step of the procedure.
Step 1: Draw the values of the "level 1" coins, or the coins that the adversary starts with. Here, c_i represents the value of coin, which determines winning the round or not.

Step 2: Determine the expected future reward corresponding to each of the adversary's coins by drawing from the previous distribution of rewards. Here, r_i is the expected number of rounds you win in future playing optimally if you win with c_i. This step is a representation of how the adversary can compare the future expected earnings among all of their coins and choose the coin that gives them the most expected reward.

Step 3: Calculate the reward R^{S_i}, or the number of rounds the adversary can expect to win in the future based on the coins they drew this round. We can write out the full equation in this step knowing that

$$\mathbb{P}(\text{first } i \text{ coins win}) = e^{-(1-\alpha)c_i} - e^{-(1-\alpha)c_{i+1}}$$

so we have

$$\begin{aligned} R^S &= \sum_{i \geq 1} \mathbb{P}(\text{first } i \text{ coins win}) \cdot \max_{j \leq i}\{1 + r_j\} & (1) \\ &= \sum_{i \geq 1} (e^{-(1-\alpha)c_i} - e^{-(1-\alpha)c_{i+1}}) \cdot \max_{j \leq i}\{1 + r_j\} & (2) \\ &= e^{-(1-\alpha)c_1} + \sum_{i \geq 1} (e^{-(1-\alpha)c_i} - e^{-(1-\alpha)c_{i+1}}) \cdot \max_{j \leq i}\{r_j\} & (3) \end{aligned}$$

and the last line is the value that we calculate and save as the next optimal distribution, D_i, with a more optimal strategy S_i, which represents the strategy in which you select the best winning coin, assuming you use S_{i-1} for the future.

Expected rewards. After going through the procedure, we have a distribution $D_i^{S_i}$, and we can calculate the expected number of rounds the adversary wins by playing optimally, $\mathbb{E}[D_i^{S_i}]$. This is simply the average of $D_i^{S_i}$, the distribution of rewards from optimal play.

Determining $D^{\mathbf{OPT}}$. Starting with $D_0 = 0$ and S_0 being the strategy where the adversary does not broadcast any coins, run $f(D_i^{S_i})$ until $\mathbb{E}[D_i^{S_i}]$ doesn't change by the specified difference for the specified number of times in a row. At that point, we have D^{OPT} and we can use D^{OPT} to calculate E^{OPT}.

In order to run a reasonable simulation, the number of rounds played by the adversary, which is represented by the size of the distribution D, and the number of coins that the adversary owns are both finite. However, in the real world they are both infinite – the adversary can play forever and have an infinite number of coins. Thus, the correctness of the simulation, i.e. how closely it predicts the optimal strategy's rewards, will be discussed in the following section.

3.4 Plotting the results

We defined E^S earlier as the expected number of future rounds you win in a row using strategy S, assuming you just won the previous round. To better understand D^S as we vary stake or α, we use D^S to determine E^S and then plot stake versus expected rewards as well as stake versus proportional expected rewards.

The following pseudocode was used for generating the plots.

- For α, starting at 0.01 and going up to 1.0 in multiples of 0.01:
  - Run $f(D(\alpha))$ until $\mathbb{E}[D(\alpha)]$ does not change by some specified amount (e.g. 0.01) some number of times in a row (e.g. 4).
  - For each α, this produces a $D^S(\alpha)$.
- For each of these 100 α's:
  - Calculate E^S using $D^S(\alpha)$ for each α.
- Plot the results, specifically:
  - Plot α vs. E^S for stake versus expected rewards.
  - Plot α vs. $\frac{E^S}{1+E^S}$ for stake versus proportional rewards.

The first bullet point is addressed by the previous subsection 3.3. For the second bullet point, we state a new definition for E^S.

Definition 3.6 (Mathematical definition of E^S)**.** Mathematically, we can define

$$E^S = \sum_{i \geq 1} \mathbb{P}[\text{I have exactly } i \text{ winners}] \cdot (\text{Expected max of } i \text{ draws from } D^S)$$

See line (4) for the full mathematical expression.

For the first term in this product, we know that $\mathbb{P}[\text{I have exactly } i \text{ winners}] = \alpha^i(1-\alpha)$. For the second term in the product, the expected max of i draws from D^S, we first attempted to draw i random values from D^S and take their max. However, this gave relatively imprecise results, and thus instead, we derived an expression for directly calculating E^S.

Remark. For initial iterations of graphing, details on sanity checks, and what we expect E^S to look like based on different strategies, see the appendix section.

3.4.1 Calculating E^S

We can explicitly calculate E^S by determining the CDF of the expected max of i draws, as $E^S = \int \text{CDF}$, then discretize based on the distribution we have at the end of the simulation. The CDF of the expected max of i draws is equivalent to $\mathbb{P}((\text{Draw } i \text{ winners, draw wins for each, take max of } i \text{ rewards}) > x)$.

Let X represent the expected max of i draws. Let Y_i be the max of (the rewards of) i i.i.d. draws from D (a given distribution).

We know that

$$\text{CDF of } Y_i = F_{Y_i}(x) = (F_D(x))^i$$

i.e. the probability that the max of i draws is $< x =$ the probability that each of the i draws is $< x$ and

$$\text{CDF of } X = F_X(x) = \textstyle\sum_{i\geq 1} \alpha^i(1-\alpha)F_{Y_i}(x).$$

Because $\mathbb{P}(X < x|i) = F_{Y_i}(x)$, this implies that

$$\begin{aligned}\mathbb{P}(X < x) &= \sum_{i\geq 1} \mathbb{P}(X < x|i) \cdot \mathbb{P}(i) \\ &= \sum_{i\geq 1} \mathbb{P}(X < x|i) \cdot \mathbb{P}(i \text{ winners}) \\ &= \sum_{i\geq 1} \mathbb{P}(X < x|i) \cdot \alpha^i(1-\alpha).\end{aligned}$$

With X as a continuous variable we have that the expectation is

$$\begin{aligned}E^S &= \int_0^\infty \sum_{i\geq 1} \alpha^i(1-\alpha)\mathbb{P}(X > x)\, dx \\ &= \int_0^\infty \sum_{i\geq 1} \alpha^i(1-\alpha)(1-(F_D(x))^i)\, dx.\end{aligned}$$

Now, let's discretize this based on the distribution D. Let n be the size of the distribution. Let a_j be the jth highest value in the array. We know that for $x \in (a_j, a_{j+1}]$,

$$F_D(x) = \mathbb{P}(\text{draw from } D < x) = \frac{j}{n}$$

i.e. for values in an interval the CDF is just uniform in the size of the interval relative to the size of the distribution. For example, for $j = 1, x \in (a_1, a_2]$ a draw from D must be a_1 to be $< x$ so the CDF is $\frac{1}{n}$.

Then, we have that

$$\begin{aligned}
E^S &= \sum_{j=1}^{n} (a_{j+1} - a_j) \cdot \sum_{i \geq 1} \alpha^i (1-\alpha) \Big(1 - \Big(\frac{j}{n}\Big)^i\Big) \\
&= \sum_{j=1}^{n} (a_{j+1} - a_j) \cdot \sum_{i \geq 1} \alpha^i (1-\alpha) - (a_{j+1} - a_j) \cdot \sum_{i \geq 1} \Big(\frac{\alpha j}{n}\Big)^i (1-\alpha) \\
&= \sum_{j=1}^{n} (a_{j+1} - a_j) \cdot \Big(\alpha - \frac{(1-\alpha)\alpha j}{n - \alpha j}\Big)
\end{aligned}$$

As we worked through this part of the project, we found that resulting graphs showed curves that quickly dropped off after around $\alpha = 0.93$, which is unexpected because rewards should continue increasing up until $\alpha = 1.0$, at which point rewards would be infinite.

We made two adjustments after reviewing our work for calculating E^S. First, we added an extra term for the very first "interval" at the beginning or the very first coin, which is simply $a_1 \cdot \alpha$. Second, we realized that our definitions for D^S and E^S are conditioned on winning the immediate past round, and thus we are determining future expected rewards based on the assumption that we just won the round. This implies that our calculation for E^S is off by α because we should have won α rounds already based on our stake. Then, E^S should initially start at α rather than at 0.

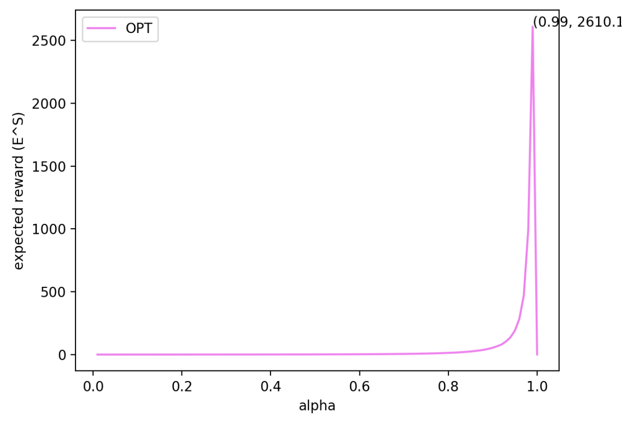
Thus, we should modify our definition for E^S by 1) starting E^S at $j = 0$ and defining $a_0 = 0$, and 2) adding α. In other words, we have

$$E^S = \alpha + \sum_{j=0}^{n} (a_{j+1} - a_j) \cdot \Big(\alpha - \frac{(1-\alpha)\alpha j}{n - \alpha j}\Big) \tag{4}$$

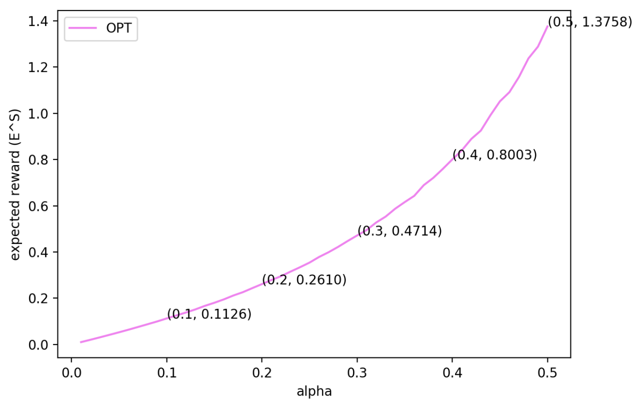
which is the formula we used to calculate E^S for each distribution $D^S(\alpha)$.

3.4.2 Final plots

The final plots for the OPT strategy we generated are below. We ran the simulation until $\mathbb{E}[D_i^{S_i}]$ did not change by 0.01 for 4 times in a row. In Figure 1, we have plots for E^{OPT}, the expected rewards of the optimal strategy. Because the expected rewards have an exponential increase in about the last 10 α's, i.e. the range $0.9 \leq \alpha < 1.0$, as we can see in Figure 1(a), we have provided Figure 1(b), which is zoomed in on the left side of the graph for $0 < \alpha \leq 0.5$, in order to see the graph more clearly.



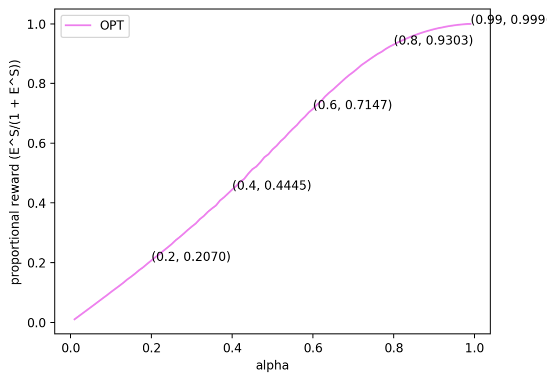
(a) α vs. E^S



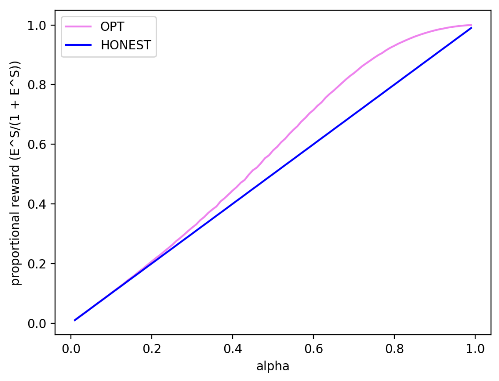
(b) Zoomed in version, $0 < \alpha \leq 0.5$

Figure 1: Optimal strategy, expected rewards

In Figure 2, we have plots for $\frac{E^{\text{OPT}}}{1+E^{\text{OPT}}}$, the proportional expected rewards of the optimal strategy. Figure 2(b) is a comparison of the OPT strategy with the HONEST strategy.



(a) α vs. $\frac{E^S}{1+E^S}$



(b) OPT vs. HONEST comparison

Figure 2: Optimal strategy, proportional expected rewards

In both Figures 1 and 2, we see that we have our expected upward trend up to $\alpha = 0.99$. There are a few things to note here. First of all, we chose to graph up to and including $\alpha = 0.99$ because at $\alpha = 1.0$, rewards are infinite and thus cannot be represented on the graph. Second, in Figure 1(a), we see that even at $\alpha = 0.99$, the expected rewards reaches approximately 2610, which implies that rewards are finite – the adversary cannot win infinitely forever. Lastly, in Figure 2 we see that the optimal strategy does a little better than the honest strategy for most of the left half of the graph and gets a significant uptick when $\alpha > 0.5$.

3.5 COIN strategy analysis

So far, we have discussed the simulation's results for the optimal strategy's rewards. Recall that the OPT strategy is dependent on the assumptions that the adversary knows all of their coins and all of the other players' coins for that round, and the adversary uses this information to calculate each of their coins' expected future wins, which allows them to choose the coin that gives them the most expected rewards and thus the most overall earnings. We represented this by updating the simulation based on calculating the adversary's expected rewards based on the coins they drew in each round.

To understand how other strategies perform in this simulation, we also ran the simulation for the COIN strategy, in which the adversary knows all of their coins but does not necessarily know all of their opponent's coins for that round. Instead of calculating all of their future coins, the adversary simply chooses the coin that gives them the highest expected reward from winning the round. (See Definition 2.5.)

COIN simulation procedure. Like in the previous simulation procedure, c_i represents the adversary's ith coin of the first round. To draw from D_{i-1}, we should follow the following process f.

1. Draw c_1 from $Exp(\alpha)$. For all $1 < i \leq k$, set $c_i := c_{i-1} + Exp(\alpha)$.
2. For all $1 \leq i \leq k$, draw r_i from D_{i-1} i.i.d., i.e. sample expected future rewards from the previous distribution D_{i-1}.
3. Choose the coin that has the highest expected reward from winning the round, i.e. c^*, to be part of D_i. See line (5) for the mathematical expression.

Steps 1 and 2 are the same as the optimal simulation's procedure. However, in Step 3, instead of calculating the future expected reward given all of the coins drawn in that round, we choose a single coin that has the highest expected reward of winning the round. We can calculate the expected reward of each coin i by multiplying the probability of winning the round by the reward of winning the round plus the future reward of that coin. Recall that

$$\mathbb{P}(c_i \text{ wins}) = e^{-(1-\alpha)c_i}$$
$$\mathbb{E}(c_i | c_i \text{ wins}) = \mathbb{P}(c_i \text{ wins}) \cdot (1 + r_i)$$

and thus the coin we choose is

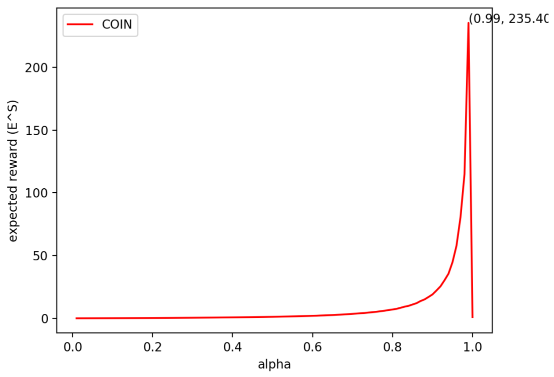
$$c^* = \arg\max_i \{e^{-(1-\alpha)c_i}(1 + r_i)\} \tag{5}$$

Remark. The final distributions D^{OPT} and D^{COIN} for $0 < \alpha < 1.0$ in increments of 0.01, generated by the pseudocode above and Section 3.3, are saved as npy files and can be made available for examination.

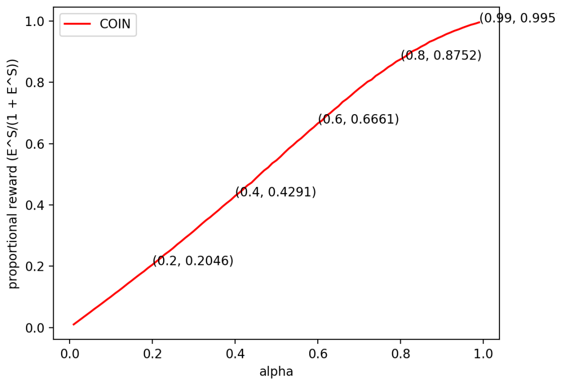
3.5.1 Plots for COIN strategy

As with the earlier plots, we followed the same graphing pseudocode but specifically changed the simulation part with the work above to generate the plots for the COIN strategy below. We ran the simulation until $\mathbb{E}[D_i^{S_i}]$ did not change by 0.01 for 4 times in a row to get D^{COIN}.

In Figure 3(a), we have the plot for E^{COIN}, the expected rewards of the COIN strategy, and in Figure 3(b), we have the plot for $\frac{E^{\text{COIN}}}{1+E^{\text{COIN}}}$, the proportional expected rewards of the COIN strategy.



(a) α vs. E^S



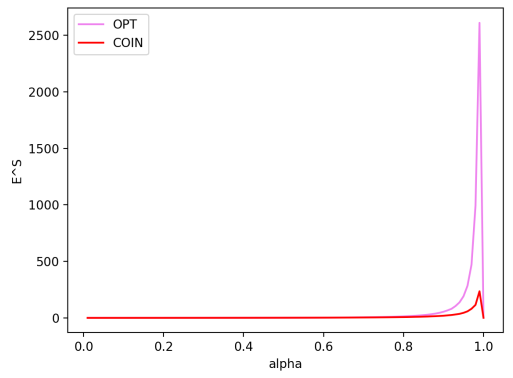
(b) α vs. $\frac{E^S}{1+E^S}$

Figure 3: COIN strategy, expected and proportional rewards

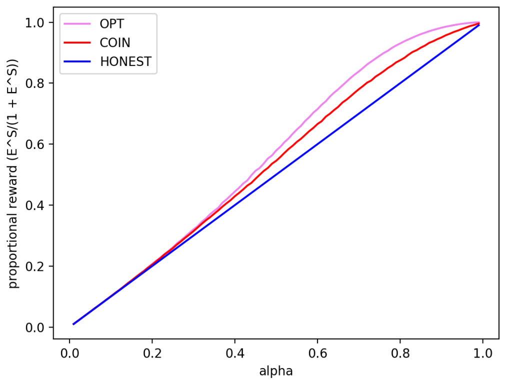
We see that there is an exponential increase in the last 10 α's in Figure 3(a), which is similar to the OPT strategy but not as much of an increase as E^{COIN} only reaches about 235 compared to E^{OPT}'s expected rewards of 2610. We also see in Figure 3(b) that the COIN strategy does a little better than HONEST in general with a slight uptick after $\alpha > 0.5$. As expected, the COIN strategy gives similarly shaped curves as the previous plots in Figure 2 for the OPT strategy, but at every α in the range from 0.01 to 1.0, COIN has less expected rewards than OPT. This is more clear in the next subsection.

3.6 Comparison graphs

We also have two graphs to compare the OPT, COIN, and HONEST strategies in terms of their expected rewards and proportional expected rewards in Figure 4.



(a) Comparison of E^S of OPT and COIN



(b) Comparison of $\frac{E^S}{1+E^S}$

Figure 4: Strategy comparisons of expected and proportional rewards

In Figure 4(a), we have graphed E^{OPT} and E^{COIN} on the same plot, and we can see that the COIN strategy clearly performs worse than the OPT strategy, which is expected because the adversary has less information in the COIN strategy than in the optimal strategy and makes decisions based on the expected reward of a single coin rather than updating over all of their future expected rewards. However, this is mainly only obvious for $\alpha > 0.5$, which is exemplified by Figure 4(b), where we have graphed proportional expected rewards for OPT, COIN, and HONEST. Both OPT and COIN strategies see improvements over the HONEST strategy, but the OPT strategy does better than the COIN strategy at all α's.

4 Supporting Theory

We made two simplifications in our simulation by truncating the number of rounds played and number of coins owned by each player, which contributes some amount of error in the results of the simulation. In this section, we prove that this amount of error has a minimal impact on the correctness of the simulation and we define the error caused by the simplifications.

We first provide more details on the motivation for this section in 4.1, specifically defining another strategy – the omniscient strategy (OMNI). We do an analysis on this omniscient strategy in 4.2, which always performs better than the three strategies we defined earlier in Section 2.3.2 (HONEST, OPT, and COIN), to show that the expected number of future rounds is finite and can be upper bounded for stake $\alpha < 0.5$. We use this analysis to quantify the amount of error caused by truncating the number of rounds played in 4.3, and to show

that the amount of error caused by truncating the number of coins owned is only a small proportion of the total expected rewards in 4.4.
We bring together our work in these two sections in 4.5 to state the overall claim that the optimal strategy truncated in the number of rounds and number of coins is almost as good as the rewards for the optimal strategy, and we can quantify what the error is. This allows us to prove our simulation's correctness and claim that any error in the simulation's results are not due to the simplifications we made to the model.

4.1 Motivation and Setup for Analysis

In the real world, the number of rounds that any player can participate in has no limit – they can draw coins every round and choose to play by a certain strategy forever. There is also no limit on the number of possible winners or coins that a player can have because they can draw infinitely many coins. However, the simulation sets finite cutoffs for the number of rounds played and the number of possible winners that a player can have.

Definition 4.1 (OMNI strategy)**.** We define the omniscient strategy here for the analysis in this section. In the OMNI strategy, the adversary knows all of their future coins and all of their opponent's future coins; in other words, they have all the knowledge in the system. Recall that in the OPT strategy, the adversary does not know all of the opponent's future coins.

In order to show that our simulation is as accurate as the real world optimal strategy to a certain small error, we use the omniscient analysis and specifically the upper bound on possible future rewards provided by the analysis to help us upper bound and clarify the error caused by the simplifications in our model.

For the rest of this section, we define the following random variable.

Definition 4.2 ($X^S(\alpha)$)**.** $X^S(\alpha)$ is defined as the expected rounds won using strategy S given stake α.

4.2 Omniscient Strategy Analysis

In this section, we show that even while the adversary is omniscient, they cannot play for an infinite amount of rounds – the probability that they win k rounds into the future is upper bounded by an expression dependent on k and their stake α. Ultimately, because the omniscient strategy performs better than the optimal strategy, we would like to use this upper bound to define error between the finite optimal strategy described by the simulation and the (infinite) optimal strategy used in practice.

Definition 4.3 ($W(\alpha, k)$)**.** Let $W(\alpha, k) := \mathbb{P}$(get at least one winner k rounds in the future, while omniscient, with stake α).
Our base case is $W(\alpha, 0) = 1$. For $k \geq 1$, we define

$$\begin{aligned} W(\alpha, k+1) &= \sum_{i=1}^{\infty} \mathbb{P}(i \text{ winners this round}) \cdot (1 - \mathbb{P}(\text{all winners this round fail})) \\ &= \sum_{i=1}^{\infty} \alpha^i (1-\alpha) \cdot \left(1 - (1 - W(\alpha, k))^i\right) \end{aligned}$$

where $\mathbb{P}$(all winners this round fail) = $\mathbb{P}$(each winner this round fails to produce a winner in the next round) = (1 - $\mathbb{P}$(winner this round produces at least one winner next round))^i, which is equal to $(1 - W(\alpha, k))^i$.

Proposition 4.1. *For $\alpha \leq 0.5$, we claim that $W(\alpha, k+1) \leq \frac{\alpha}{1-\alpha} W(\alpha, k)$. This is equivalent to $W(\alpha, k+1) \leq (\frac{\alpha}{1-\alpha})^{k+1}$.*

Proof. First, we show that we can upper bound the probability of having winners k rounds in the future. We take a union bound with $i \cdot W(\alpha, k)$ such that

$$\begin{aligned} W(\alpha, k+1) = \sum_{i=1}^{\infty} \alpha^i (1-\alpha) \cdot \left(1 - (1 - W(\alpha, k))^i\right) &\leq \sum_{i=1}^{\infty} \alpha^i (1-\alpha) \cdot i \cdot W(\alpha, k) \\ &= \frac{\alpha}{1-\alpha} W(\alpha, k) \end{aligned}$$

Now, we want to show that this holds for $\alpha \leq 0.5$. We first show by induction that $W(\alpha, 2k) = \frac{\alpha^{2k}}{P^{2k}(\alpha)}, a_{2k} = 1$ for even numbers $2k$, where $P^{2k}(\alpha), a_{2k} = 1$ represents a polynomial of degree $2k$ with leading coefficient 1, and that $W(\alpha, 2k+1) = \frac{\alpha^{2k+1}}{P^{2k}(\alpha)}$ for odd numbers $2k+1$ where $P^{2k+1}(\alpha)$ represents a polynomial of degree $2k+1$.

Our base case is $W(\alpha, 0) = 1$. We first claim that the even form holds, i.e. that $W(\alpha, 2k) = \frac{\alpha^{2k}}{P^{2k}(\alpha)}$, and we want to show that the odd form holds, i.e. that $W(\alpha, 2k+1) = \frac{\alpha^{2k+1}}{P^{2k}(\alpha)}$. Simply plugging into the definition for $W(\alpha, k+1)$, we have

$$\begin{aligned} W(\alpha, 2k+1) &= \sum_{i=1}^{\infty} \alpha^i (1-\alpha) \Big(1 - \Big(1 - \frac{\alpha^{2k}}{P^{2k}(\alpha)}\Big)^i\Big) && (6) \\ &= (1-\alpha) \sum_{i=1}^{\infty} \alpha^i - (1-\alpha) \sum_{i=1}^{\infty} \Big(\alpha^i \cdot \Big(1 - \frac{\alpha^{2k}}{P^{2k}(\alpha)}\Big)^i\Big) && (7) \\ &= \alpha - \frac{(1-\alpha)\alpha(P^{2k}(\alpha) - \alpha^{2k})}{P^{2k}(\alpha) - \alpha(P^{2k}(\alpha) - \alpha^{2k})} && (8) \\ &= \frac{\alpha^{2k+1}}{P^{2k}(\alpha)} && (9) \end{aligned}$$

where we use the formula for infinite geometric series summation to get line 3, and in the denominator we use the fact that P^{2k} has leading coefficient 1 to cancel out the α^{2k+1} terms

which leaves us with another polynomial of degree $2k$, and we simplify the results to get our desired answer.
Next, because we have shown that the odd form holds, we want to show that the even form holds, i.e. that $W(\alpha, 2k+2) = \frac{\alpha^{2k+2}}{P^{2k+2}(\alpha)}, a_{2k+2} = 1$. Again, we plug into the definition for $W(\alpha, k+1)$ and simplify to get

$$\begin{aligned}W(\alpha, 2k+2) &= \sum_{i=1}^{\infty} \alpha^i (1-\alpha)\Big(1 - \Big(1 - \frac{\alpha^{2k+1}}{P^{2k}(\alpha)}\Big)^i\Big) \\ &= \alpha - \frac{(1-\alpha)\alpha(P^{2k}(\alpha) - \alpha^{2k+1})}{P^{2k}(\alpha) - \alpha(P^{2k}(\alpha) - \alpha^{2k+1})} \\ &= \frac{\alpha^{2k+2}}{P^{2k+2}(\alpha)}\end{aligned}$$

As much of the simplifying is the same as earlier, we can quickly see that we indeed have the desired form. Note that in the denominator, we have $P^{2k} - P^{2k+1} + \alpha^{2k+2}$, which is a polynomial of degree $2k+2$ and leading coefficient 1. Thus, we have shown using induction that $W(\alpha, 2k) = \frac{\alpha^{2k}}{P^{2k}(\alpha)}, a_{2k} = 1$ for even numbers $2k$, and $W(\alpha, 2k+1) = \frac{\alpha^{2k+1}}{P^{2k}(\alpha)}$ for odd numbers $2k+1$.
Lastly, we now use these forms by plugging them into the inequality upper bound. For even $2k$, we have the following

$$W(\alpha, 2k+1) \le \frac{\alpha}{1-\alpha} W(\alpha, 2k) \tag{10}$$

$$\frac{\alpha^{2k+1}}{P^{2k}(\alpha)} \le \frac{\alpha}{1-\alpha}\frac{\alpha^{2k}}{P^{2k}(\alpha)} \tag{11}$$

$$P^{2k}(\alpha) \le (1-\alpha)P^{2k}(\alpha) \tag{12}$$

$$\alpha^{2k} \le \alpha^{2k} - \alpha^{2k+1} \tag{13}$$

$$\alpha \le 1-\alpha \implies \alpha \le 0.5 \tag{14}$$

where we treat the polynomials as their largest term in line 8 because the condition must hold in order for the entire polynomial as $k \to \infty$, which gives us the desired condition that α must be less than 0.5. $\square$

The analysis we have done with the omniscient analysis implies that the probability that we keep playing any k rounds into the future is upper bounded by a certain amount dependent on k and the player's stake α. We can now use this upper bound to quantify the amount of error induced by truncating the number of rounds played, and to define the amount of error induced by truncating the number of winners owned by the player in each round.

4.3 Truncating Number of Rounds

In this section, we show that truncating the number of rounds played in the simulation that approximates the optimal strategy causes only a certain small amount of error in the expected number of wins compared to the optimal strategy in practice.

Definition 4.4 ($X_n^S(\alpha)$)**.** Recall Definition 4.2 for $X^S(\alpha)$, the expected rounds won using strategy S given stake α. We denote $X_n^S(\alpha)$ as the expected rounds won using strategy S when it immediately loses after n rounds, given stake α. In other words, the number of rounds played is cut off after n rounds. For this subsection, we define $X^S(\alpha)$ and $X_n^S(\alpha)$ as

$$X^S(\alpha) = \sum_{i=0}^{\infty} \mathbb{P}(S \text{ wins} > i \text{ rounds won in a row}) \tag{15}$$

$$X_n^S(\alpha) = \sum_{i=0}^{k} \mathbb{P}(S \text{ wins} > i \text{ rounds won in a row}) \tag{16}$$

The strategy OPT_n represents the optimal strategy cut off after n rounds, as well as the strategy approximated by the simulation. Then, we claim the following about X_n^{OPT}, the expected number of wins using the strategy OPT_n.

Proposition 4.2. *For all $n \in \mathbb{N}, \alpha \leq 0.5$,*

$$X^{OPT}(\alpha) \leq X_n^{OPT}(\alpha) + RoundErr(\alpha, n) \tag{17}$$

where we have

$$RoundErr(\alpha, n) = \Big(\frac{\alpha}{1-\alpha}\Big)^{n+1}\Big(\frac{1-\alpha}{1-2\alpha}\Big). \tag{18}$$

That is, the strategy OPT_n is at least as good as the strategy OPT in terms of expected rounds won, plus some error term $RoundErr(\alpha, n)$ due to truncating the number of rounds played at n.

Proof. We first claim that for all i,

$$\mathbb{P}(\text{OPT wins} > i \text{ rounds in a row}) \leq \mathbb{P}(\text{OMNI wins} > i \text{ rounds in a row})$$

This is because in order for the optimal strategy win i rounds in a row, there must be a path the strategy can take to do so. If there is any such path that wins i rounds in a row, the omniscient strategy will take it. This means that the optimal strategy can do no better than the omniscient strategy.

We showed in Proposition 4.1 that for any i, $\mathbb{P}[\text{OMNI wins} > i \text{ rounds in a row}] \leq (\frac{\alpha}{1-\alpha})^i$. This implies that the true expected number wins can be upper bounded as follows:

$$\begin{aligned} X^{\text{OPT}}(\alpha) &\leq X^{\text{OMNI}}(\alpha) \\ &= \sum_{i=0}^{n} + \sum_{i=n+1}^{\infty} \Big(\frac{\alpha}{1-\alpha}\Big)^i \\ &= \sum_{i=0}^{n} + \Big(\frac{\alpha}{1-\alpha}\Big)^{n+1} \cdot \frac{1-\alpha}{1-2\alpha} \\ &= X_n^{\text{OMNI}}(\alpha) + RoundErr(\alpha, n) \end{aligned}$$

Moreover, we know that the strategy OPT_n is certainly at least as good as OPT, the optimal strategy in the real world, but only counting the first n rounds, and OPT_n is also certainly not as good as the omniscient OMNI strategy. Therefore, we have the inequality

$$X^{\text{OPT}}(\alpha) \leq X_n^{\text{OPT}}(\alpha) + \mathit{RoundErr}(\alpha, n)$$

which is exactly what the proposition states. $\square$

As n grows, this error term shrinks exponentially quickly in n, and we can actually provide a quantitative bound for the maximum amount of error coming from cutting off our analysis at a finite number of rounds. For example, if $\alpha = 0.2$ and $n = 10000$, which is the n used in the simulation results, the error term is $\mathit{RoundErr}(0.2, 10000) = (\frac{\alpha}{1-\alpha})^{n+1}(\frac{1-\alpha}{1-2\alpha}) \approx 0$, which is an insignificant amount of error.

Therefore, this work shows that the gap between the real world optimal strategy and the simulated optimal strategy that cuts the number of rounds played off at n is exponentially small in n. Then, the simulation results are correct up to exponentially small error, which we can provide a formula for: $\mathit{RoundErr}(\alpha, n) = (\frac{\alpha}{1-\alpha})^{n+1}(\frac{1-\alpha}{1-2\alpha})$.

Remark. This proposition only holds for $\alpha \leq 0.5$, which is sufficient for the purposes of this analysis as the range $0 < \alpha \leq 0.5$ is still an interesting range to study. Moreover, we know that when α is greater than $\frac{1}{3}$, the adversary can subvert the entire protocol anyway [2].

4.4 Truncating Number of Winners

In this section, we show that cutting off the number of winners owned by the adversary in the simulation that approximates the optimal strategy does not make a significant difference in the expected wins of the optimal strategy in practice, because the winners beyond the cutoff only contribute a small amount to the total expected wins. In the real world, the optimal strategy takes into account an infinite number of coins, but in the simulation we make a simplification by setting the number of coins owned by the adversary at a finite number.

Definition 4.5 (S^k)**.** Let S^k be the same strategy as S, except that S^k only considers the first k winners. In other words, S^k considers all coins past the kth coin to be losers, or that they have 0 future expected rewards and thus will never be chosen.

Definition 4.6 ($X^{S^k}(\alpha)$, $F^S(\alpha, i)$)**.** Recall Definition 4.2 for $X^S(\alpha)$, the expected rounds won using strategy S given stake α. We denote $X^{S^k}(\alpha)$ as the expected rounds won using strategy S^k which truncates the number of winners in each round at k, given stake α. In addition, let $F^S(\alpha, i)$ denote the expected future wins using strategy S given α and i winners to choose from. Then, for this subsection, we define $X^S(\alpha)$ and $X^{S^k}(\alpha)$ as

$$X^S(\alpha) = \sum_{i=1}^{\infty} \mathbb{P}(i \text{ winners in first round}) \cdot (1 + F^S(\alpha, i)) \tag{19}$$

$$X^{S^k}(\alpha) = \sum_{i=1}^{k} \mathbb{P}(i \text{ winners in first round}) \cdot (1 + F^S(\alpha, i)) \tag{20}$$

As shorthand from now on, let $\mathbb{P}(i \in R1)$ denote the probability of having i winners in the first round, or $\mathbb{P}(i \text{ winners in first round})$. Now, we first prove the following lemma to get us started with the proofs in this section.

Lemma 4.1 (Upper bound on $X^S(\alpha)$). *We can place an initial upper bound on the expected number of rounds won using strategy S given α as defined in this subsection.*

$$X^S(\alpha) \leq \sum_{i=1}^{\infty} \mathbb{P}(i \textit{ winners in first round}) \cdot (1 + i \cdot F^S(\alpha, 1)) \tag{21}$$

Proof. First, we want to show that for very large i, $F^S(\alpha, i)$ is not very large. Let us set a union bound on $F^S(\alpha, i)$ as

$$F^S(\alpha, i) = \mathbb{E}[\max_i \{i\text{th draw from } F^S(\alpha)\}] \leq i \cdot F^S(\alpha, 1)$$

This is because if the adversary has i winners, they should pick the best one among the i winners, so the expected number of future wins is equivalent to the expected number of wins from the best of i draws from future wins. Then, we can upper bound the expected max of i draws by the sum of i individual draws. This implies that we have the following inequality:

$$\begin{aligned} X^S(\alpha) &= \sum_{i=1}^{\infty} \mathbb{P}(i \in R1) \cdot (1 + F^S(\alpha, i) \\ &\leq \sum_{i=1}^{\infty} \mathbb{P}(i \in R1) \cdot (1 + i \cdot F^S(\alpha, 1)) \end{aligned}$$

which is the desired inequality. □

At this point, we differ in our analysis for $\alpha \leq 0.5$ and $\alpha > 0.5$. We will apply the omniscient analysis from Section 4.2 for the former case with similar logic as in truncating the number of rounds, and we will have a slightly different approach for the latter case.

4.4.1 $\alpha \leq 0.5$

The strategy OPT^k represents the optimal strategy that truncates the number of winners at k each round. Then, we claim the following about X^{OPT^k}, the expected number of wins using the strategy OPT^k, for $\alpha \leq 0.5$.

Proposition 4.3. *For all $n \in \mathbb{N}, \alpha \leq 0.5$,*

$$X^{OPT}(\alpha) \leq X^{OPT^k}(\alpha) + WinnerErr(\alpha, k) \tag{22}$$

where we have

$$WinnerErr(\alpha, k) = \frac{\alpha^{k+1}(1-\alpha)(2+k)}{1-2\alpha}. \tag{23}$$

That is, the strategy OPT^k is at least as good as the strategy OPT in terms of expected rounds won, plus some error term $WinnerErr(\alpha, k)$ due to truncating the number of winners per round at k.

Proof. From our omniscient strategy analysis, we showed that there is an upper bound on the number of future wins that any user can gain, which is

$$\sum_{i=0}^{\infty} \mathbb{P}(\text{OMNI wins} > i \text{ rounds in a row}) = \sum_{i=0}^{\infty} \Big(\frac{\alpha}{1-\alpha}\Big)^i = \frac{1-\alpha}{1-2\alpha}$$

Thus, because the OMNI strategy accounts for a single winner each round, we have that

$$F^{\text{OMNI}}(\alpha, 1) \leq \frac{1-\alpha}{1-2\alpha}. \tag{24}$$

We know that $\mathbb{P}(i \text{ winners in first round}) = \alpha^i(1-\alpha)$, so we can substitute it and line (24) into the inequality we proved in Lemma 4.1 (see line (21)) from above to have

$$\begin{aligned}
X^{\text{OPT}}(\alpha) &\leq \sum_{i=1}^{\infty} \alpha^i(1-\alpha) \cdot \Big(1 + i * \frac{1-\alpha}{1-2\alpha}\Big) \\
&= \sum_{i=1}^{k} + \sum_{i=k+1}^{\infty} \alpha^i(1-\alpha) + \alpha^i \cdot i \cdot \Big(\frac{(1-\alpha)^2}{1-2\alpha}\Big) \\
&= \sum_{i=1}^{k} + \Big(\alpha^{k+1} + \alpha^{k+1} \cdot \frac{k+1-k\alpha}{1-2\alpha}\Big) \\
&= \sum_{i=1}^{k} + \frac{\alpha^{k+1}(1-\alpha)(2+k)}{1-2\alpha} \\
&= X^{\text{OMNI}^k}(\alpha) + WinnerErr(\alpha, k)
\end{aligned}$$

Moreover, we know that the strategy OPT^k is certainly at least as good as OPT, the optimal strategy in the real world, but only considering k winners in each round, and OPT^k is also certainly not as good as the omniscient OMNI strategy. Therefore, we have the inequality

$$X^{\text{OPT}}(\alpha) \leq X^{\text{OPT}^k}(\alpha) + WinnerErr(\alpha, k)$$

which is exactly what the proposition states. □

Similar to our analysis for truncating the number of rounds, as k grows, the error term also shrinks exponentially quickly in k, and we can provide a quantitative bound for the maximum amount of error coming from cutting off our analysis at a finite number of winners. For example, if $\alpha = 0.2$ and $k = 100$, which is the k used in the simulation results, the error term is $WinnerErr(\alpha, k) = \frac{\alpha^{k+1}(1-\alpha)(2+k)}{1-2\alpha} \approx 3.45 \times 10^{-69}$, which is extremely small.

Therefore, this work shows that the gap between the real world optimal strategy and the simulated optimal strategy that cuts the number of coins in each round off at k is exponentially small in k. Then, the simulation results are correct up to exponentially small error, which we can provide a formula for: $WinnerErr(\alpha, k) = \frac{\alpha^{k+1}(1-\alpha)(2+k)}{1-2\alpha}$.

4.4.2 $\alpha > 0.5$

We want to show that we can ignore sufficiently large coins for any alpha, not just $\alpha \leq 0.5$. However, we cannot use the omniscient analysis to show this, because the omniscient strategy only holds for $\alpha \leq 0.5$. Instead of directly upper bounding the value as we did in the previous section, we instead show that the number of future wins we gain from having more than some k winners only accounts for a tiny fraction of the entire sum that makes up the expected number of our future wins.

Definition 4.7 ($FracWinnerErr^S(\alpha, k)$)**.** Let us first generally define $FracWinnerErr^S$ as the error derived from truncating number of winners at k in strategy S, i.e.

$$FracWinnerErr^S = \sum_{i=k+1}^{\infty} \mathbb{P}(i \text{ winners in first round}) \cdot (1 + F^S(\alpha, i)) \tag{25}$$

which simply comes from subtracting the first k terms in the sum definition of $X^S(\alpha)$ or $X^S(\alpha) - X^{S^k}(\alpha)$, i.e. line (19) minus line(20).

Again, we have the strategy OPT^k, which represents the optimal strategy that truncates the number of winners at k each round. Then, we claim the following about X^{OPT^k}, the expected number of wins using the strategy OPT^k, for $\alpha > 0.5$.

Proposition 4.4. *For all $\alpha > 0.5$,*

$$X^{OPT^k}(\alpha) \geq (1 - \varepsilon) X^{OPT}(\alpha) \tag{26}$$

where we have

$$\varepsilon_F = \frac{FracWinnerErr^{OPT}}{X^{OPT}(\alpha)} \tag{27}$$

and k satisfies the condition

$$k \gg \alpha^{-k} - \frac{\alpha - 2}{1 - \alpha}. \tag{28}$$

That is, the strategy OPT^k is at least as good as the strategy OPT in terms of expected rounds won multiplied by $1 - \varepsilon_F$, where ε_F is an extremely small number that represents the fraction of expected rounds won by the winners after the kth winner. This ε_F is guaranteed to be a tiny fraction as long as k satisfies the inequality given above, which is derived from the probability that the adversary will even have more than k winners, a probability that decreases exponentially in k.

Proof. First, we observe that $F^S(\alpha, 1) \leq F^S(\alpha, j)$ for all $j \geq 1$. Let us pick a cutoff k, which will be the finite number of winners we choose to do our analysis over. Then, recalling our expression for S^{S^k} from line (20), we have the lower bound

$$X^{\mathrm{OPT}^k}(\alpha) = \sum_{i=1}^{k} \mathbb{P}(i \in R1) \cdot (1 + F^{\mathrm{OPT}}(\alpha, i)) \geq \sum_{i=1}^{k} \mathbb{P}(i \in R1) \cdot (1 + F^{\mathrm{OPT}}(\alpha, 1)) \tag{29}$$

Next, we apply similar logic as in the proof for Lemma 4.1 with setting a union bound on $F^S(\alpha, i)$ such that $F^S(\alpha, i) \leq i \cdot F^S(\alpha, 1)$. Then, we can use this inequality to set an upper bound on $\mathit{FracWinnerErr}^S$, which is also the contribution made by the winners after the kth winner to the expected number of rounds won. The left side of the inequality comes from line (25).

$$\sum_{i=k+1}^{\infty} \mathbb{P}(i \in R1) \cdot (1 + F^{\mathrm{OPT}}(\alpha, i)) \leq \sum_{i=k+1}^{\infty} \mathbb{P}(i \in R1) \cdot i \cdot (1 + F^{\mathrm{OPT}}(\alpha, 1)) \tag{30}$$

As we have explained earlier, we know that the strategy OPT^k is certainly at least as good as OPT, the optimal strategy in the real world, but only considering k winners in each round. Therefore, we can rearrange:

$$\begin{aligned}
X^{\mathrm{OPT}}(\alpha) &\leq X^{\mathrm{OPT}^k}(\alpha) + \mathit{FracWinnerErr}^{\mathrm{OPT}}(\alpha, k) \\
X^{\mathrm{OPT}^k}(\alpha) &\geq X^{\mathrm{OPT}}(\alpha) - \mathit{FracWinnerErr}^{\mathrm{OPT}}(\alpha, k) \\
X^{\mathrm{OPT}^k}(\alpha) &\geq (1 - \varepsilon_F) X^{\mathrm{OPT}}(\alpha)
\end{aligned}$$

which is the desired inequality when we let $\varepsilon_F = \frac{\mathit{FracWinnerErr}^{\mathrm{OPT}}(\alpha,k)}{X^{\mathrm{OPT}}(\alpha)}$.
However, this is only guaranteed when the two bounds are satisfied – the lower bound for $X^{\mathrm{OPT}^k}(\alpha)$ and the upper bound for $\mathit{FracWinnerErr}^{\mathrm{OPT}}(\alpha, k)$:

$$\begin{aligned}
X^{\mathrm{OPT}^k}(\alpha) &\geq X^{\mathrm{OPT}}(\alpha) - \sum_{i=k+1}^{\infty} \mathbb{P}(i \in R1) \cdot i \cdot (1 + F^{\mathrm{OPT}}(\alpha, 1)) \\
X^{\mathrm{OPT}^k}(\alpha) &\geq \sum_{i=1}^{k} \mathbb{P}(i \in R1) \cdot (1 + F^{\mathrm{OPT}}(\alpha, 1))
\end{aligned}$$

Because $\mathbb{P}(i \in R1) = \alpha^i(1 - \alpha)$, we simply must choose k that satisfies the condition

$$\sum_{i=k+1}^{\infty} i \cdot \alpha^i (1 - \alpha) \ll \sum_{i=1}^{k} \alpha^i (1 - \alpha)$$

which essentially makes $\mathit{FracWinnerErr}^{\mathrm{OPT}}(\alpha, k)$ and thus ε_F extremely small. Both sums can be simplified in their closed forms to derive the inequality in the proposition statement,

$$k \gg \alpha^{-k} - \frac{\alpha - 2}{1 - \alpha}$$

which completes the proof. □

Again, because we cannot use the omniscient analysis when $\alpha > 0.5$, we cannot define a specific formula for the amount of error caused by truncating the number of possible winners. However, what we are able to show here is that even when $\alpha > 0.5$, the amount lost from considering only k coins only accounts for a tiny fraction of the entire sum of expected rewards when k is large enough, specifically only a small ε_F as defined in Proposition 4.4.

Remark. The only assumption necessary for this section is that the expected number of wins is finite. If expected reward is infinite, it may never be true that truncating the number of winners only contributes a small amount of the total expected wins.

4.4.3 Combining 4.4.1 and 4.4.2

We would like to write an expression that applies for all $\alpha, 0 < \alpha < 1$, for strategy OPT^k by combining Propositions 4.3 and 4.4. First, we rephrase Proposition 4.3 so that it is in the same structure as Proposition 4.4:

Proposition 4.5 (Restating Prop 4.3)**.** *For all $n \in \mathbb{N}, \alpha \leq 0.5$,*

$$X^{OPT^k}(\alpha) \geq (1 - \varepsilon_W)X^{OPT}(\alpha) \tag{31}$$

where we have

$$\varepsilon_W = \frac{WinnerErr(\alpha, k)}{X^{OPT}(\alpha)} \tag{32}$$

and we define

$$WinnerErr(\alpha, k) = \frac{\alpha^{k+1}(1-\alpha)(2+k)}{1-2\alpha}. \tag{33}$$

That is, the strategy OPT^k is at least as good as the strategy OPT in terms of expected rounds won multiplied by $1 - \varepsilon_W$. ε_W is an extremely small number that is derived from the upper bound of the OMNI strategy on the number of expected rounds won by the winners after the kth winner, and ε_W decreases exponentially in k.

Now, we can make the following claim.

Proposition 4.6. *For all α, $0 < \alpha < 1$,*

$$X^{OPT^k}(\alpha) \geq (1 - \varepsilon)X^{OPT}(\alpha) \tag{34}$$

where we have

$$\varepsilon = \begin{cases} \varepsilon_W = \frac{WinnerErr(\alpha,k)}{X^{OPT}(\alpha)} & \alpha \leq 0.5 \\ \varepsilon_F = \frac{FracWinnerErr^{OPT}}{X^{OPT}(\alpha)} & \alpha > 0.5 \end{cases} \tag{35}$$

and k satisfies the condition

$$k \gg \alpha^{-k} - \frac{\alpha - 2}{1 - \alpha}.$$

Proof. The $\alpha \leq 0.5$ case was proved in Proposition 4.3, and the $\alpha > 0.5$ case was proved in Proposition 4.4. See lines (23) and (25) for definitions of $WinnerErr(\alpha, k)$ and $FracWinnerErr^{\mathrm{OPT}}$. □

This proposition allows us to claim that the error between the strategy that truncates the number of winners owned by the adversary at each round, which is the strategy approximated by the simulation, and the real world optimal strategy is a ε fraction of the real world optimal strategy that is exponentially small in k.

4.5 Simulation Correctness

We finally combine our work in Subsections 4.3 and 4.4 on the strategy that truncates the number of rounds played and the number of winners considered per round to make an overarching claim about the correctness of the simulation. Specifically, the strategy OPT_n^k represents the optimal strategy that immediately loses after n rounds and considers no more than k coins per round. Then, we claim the following about $X_n^{\text{OPT}^k}$, the expected number of wins using the strategy OPT_n^k.

Proposition 4.7. *For all $\alpha \leq 0.5$ and $n \in \mathbb{N}$, let k such that $k \gg \alpha^{-k} - \frac{\alpha-2}{1-\alpha}$. Then,*

$$X_n^{OPT^k}(\alpha) \geq (1-\varepsilon)X^{OPT}(\alpha) - RoundErr(\alpha, n) \tag{36}$$

with ε and $RoundErr(\alpha, n)$ respectively defined in lines (35) and (18).
That is, the strategy OPT_n^k is at least as good as the strategy OPT in terms of expected rounds won with very small error ε which comes from the truncation of winners at k, with some small, quantifiable error $RoundErr(\alpha, n)$ which comes from the truncation of rounds at n.

Proof. This claim is the combination of Propositions 4.2 and 4.6, where we have the following two inequalities from those propositions:

$$X_n^{\text{OPT}}(\alpha) \geq X^{\text{OPT}}(\alpha) - RoundErr(\alpha, n)$$
$$X^{\text{OPT}^k}(\alpha) \geq (1-\varepsilon)X^{\text{OPT}}(\alpha)$$

We can apply both of these lower bounds to the strategy $X_n^{\text{OPT}^k}$. This strategy must be at least as good as $(1-\varepsilon)X^{\text{OPT}}(\alpha)$, which takes into account the error from truncating the number of winners each round, minus the $RoundErr(\alpha, n)$, which is the error from truncating the number of rounds played. Then, we would get the desired inequality. □

This proposition is crucial because it allows us to claim that the optimal strategy truncated at k and n is almost as good as the the optimal strategy, and we can quantify what the error is in terms of ε and *RoundErr*. The simulation we worked through in Section 3 precisely runs the strategy OPT_n^k and approximates the strategy's expected rewards in terms of expected rounds won. At the end of the day, a simulation is still only an approximation of what happens in practice and is not a perfect calculation. However, if the simulation is incorrect, the work in this section shows that it is not because of truncation, i.e. only a very small value is lost from truncating the number of rounds and number of coins, so there is not much error between the optimal strategy approximated by the simulation and the real life optimal strategy.

Remark. Proposition 4.7 only holds for $\alpha \leq 0.5$, and there is still an open question for what happens when $\alpha > 0.5$.

5 Conclusion

In this paper, we have worked through an analysis of one strategy that an adversary can play to optimize expected rewards and improve their earnings over acting honestly, which we called the OPT strategy. We created a simulation that approximates the expected rewards of the OPT strategy with two simplifications: truncate the number of rounds played, and truncate the number of winners considered per round. In order to guarantee the correctness of the simulation's results, we then proved theoretically that the error induced from these two simplifications is extremely small by comparing the OPT strategy with the omniscient (OMNI) strategy.

We also used our simulation to analyze a less optimal strategy, which we named the COIN strategy, and we compared the strategy's expected rewards with the OPT strategy's expected rewards plotted for a range of possible α's or stakes in $0 < \alpha < 1$. We found it was very easy to modify the simulation in accordance with a different update rule to represent the COIN strategy, and we could use the same simulation structure and pseudocode to approximate the distribution of expected rewards for many other possible strategies.

There are two areas of further work that were considered in the course of this project. The first area is quantifying or bounding the error induced from sampling in the simulation, which comes from Step 2 of the pseudocode, where we sample the previous distribution to represent the expected rewards of each coin. However, other than this sampling error, there should be no other potential sources of error from the procedure of the simulation. The second area is using the results of the simulation and theoretical work to write progress towards a closed form for the variables involved in the process. For example, can we make some claim about the expected rewards of the OPT strategy? Do they have to satisfy some conditions or can we describe it with a closed form distribution?

6 References

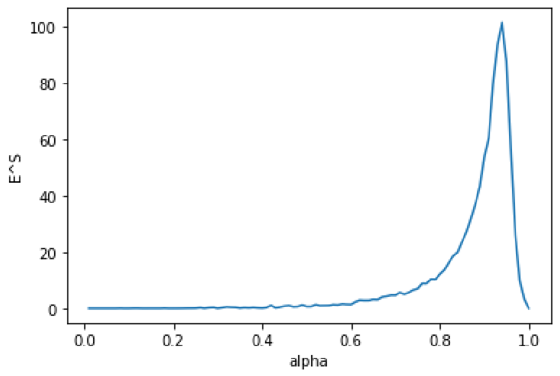
[1] Satoshi Nakamoto. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf
[2] Jing Chen, Silvio Micali. ALGORAND. https://arxiv.org/pdf/1607.01341.pdf
[3] Sally Hahn. Incentive to Cheat: Algorand's Proof of Stake Protocol.

Thank you to Professor Matt Weinberg for advising this paper and Matheus Venturyne Xavier Ferreira for guidance and advice throughout the project. Thank you to Professor Mark McConnell for being a second reader for this paper.

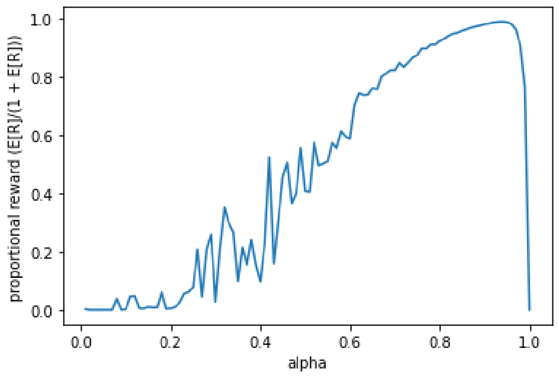
Appendix

Initial plots from sampling max draws from D^S

These plots were the very first plots we generated, in which we calculated E^S and specifically the (expected max of i draws from D^S) term by sampling i random draws from D^S and taking their max. We ran the simulation until R^S did not change by 0.01 for 4 times in a row.



(a) α vs. E^S



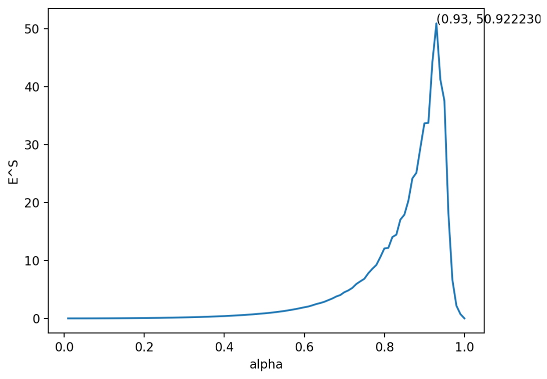
(b) α vs. $\frac{E^S}{1+E^S}$

Figure 5: First plots, sampling max draws from D^S

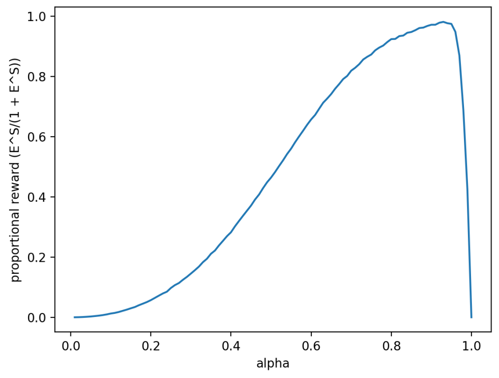
The instability of the sampling method is very obvious in Figure 1(b) and implies that a more precise method is necessary for both better visualization and analysis. It is also important to point out here that the drop off at around 0.9 is abnormal and is an issue that will be addressed in later plots.

Second iteration of plots from calculating E^S

We decided to directly calculate E^S (see section 3.4.1) and we reran the graphing code with this implementation. This resulted in the following plots.



(a) α vs. E^S



(b) α vs. $\frac{E^S}{1+E^S}$

Figure 6: Second plots, directly calculating E^S

As we noted in the initial plots, these plots are abnormal because there is a drop off in expected rewards after $\alpha = 0.93$, while we would expect that expected rewards should continue increasing up until $\alpha = 1.0$, at which point rewards would be infinite and thus cannot be represented on the graph.

Sanity checking E^S

We also talked about sanity checks for the E^S computation. Initially, we expected that $E^S = \frac{\alpha}{1-\alpha}$ for the honest strategy. However, when we ran the simulation for a single iteration to get $D_1^{S_1}$, we found that the E^S calculation on the distribution gave higher than expected E^S. This can be explained by the design of the simulation such that when it is given the honest strategy and starting distribution, it will use the distribution to optimize future rewards in the very first iteration and thus output a distribution that is better than honest. Moreover, recall that D^{OPT} is the distribution of rounds won using the optimal strategy starting from scratch, and E^S is the distribution of rounds I win in the future assuming I just won with a coin right now. The simulation maintains the invariant: if I go through the computation that a draw from D exceeds x, then I know that that computation is supposed to give me back the probability that a draw from D exceeds x. Thus, we would expect that $\mathbb{E}[D^{\text{OPT}}] = E^{\text{OPT}}$, and when we compare these two values for each $0 < \alpha < 1$ in increments of 0.01, we indeed find that they are the same.