# PolyaChain

Aparna Krishnan, Ashwinee Panda, Sampada Deglurkar, Peter Schafhalter

EECS126 Markov Chain Project

## 1 Introduction

Bitcoin, the titular cryptocurrency that first propelled blockchain into the news, is hailed as a revolutionary advance in the fields of cryptography, distributed systems, and economics. However, Satoshi Nakamoto did not create any advances in these fields -rather, they created one simple innovation: they created the first ever trustless pseudo-anonymous decentralized currency iterating on previous works like B-Money and Hash Cash. The way multiple actors agree on the state of the network in a trustless manner is by expending some resources to solve a puzzle called Proof-of-Work. In reward for the work done, the actors get paid some newly created Bitcoin and transaction fees paid by the users of the network. Proof of Work was a groundbreaking development, but many alternate consensus mechanisms have been proposed such as Proof of Stake, Proof of Burn, etc. While expending resources in Proof-of-Work buys security guarantees for the network in terms of time and money requigreen to attack the system, there is also an extremely high expenditure of electricity. There has been a lot of exploration into alternate systems which provide similar levels of security but are more energy efficient. The stability and convergence of these incentive mechanisms has been difficult to study without baseline mathematical models for how they operate. In our project, we created an efficient simulation for a Proof-of-Stake system by modelling arbitrary distributions and reward functions. We display the accuracy of our simulation by measuring the divergence of a simulated forecast from the real distribution.

A Proof-of-stake or Proof-of-Work reward mechanism can be represented as a Polya's Urn model. In this model, we assume we have $k$ colors and $x_i^0$ number of balls of color $i$ in the urn at time step 0. We start with $\sum x_i^0$ number of balls. At each time step, a ball is randomly selected from the urn and then replaced along with a certain number of balls of the same color as the one chosen. This models the 'winner wins more' or 'rich get richer' nature of Nakamoto Consensus. Although it is most exactly analogous to the Proof of Stake consensus mechanism, it can be argued that this is similar to Proof of Work as well. Miners often 'invest' large amounts of capital into their mining rigs, and use the accrued bitcoin to invest in further GPUs.

The Polya's Urn model is a specific type of Markov Chain, and the abstraction allows us to represent the evolution of wealth on the Bitcoin blockchain as a Markov Chain. By running a Markov Chain on our initial distribution for over a thousand steps, and comparing this to the empirical distribution after the same number of steps, we can gauge the overall accuracy of this simulation in comparison to the real-world process. This in turn allows us to gauge how useful the representation of the Bitcoin blockchain's wealth evolution as a Markov Chain is.

## 2 Methods

We use the Polya's Urn model to simulate the distribution of a player's wealth in the system over time. Here, the model is used to represent a martingale random process that starts with a certain initial distribution of balls of different colors present in the urn. For example, a 2-player system can be modeled with only 2 colors of balls, green and blue, and a possible initial distribution would be 5 green and 5 blue balls present. This would mean that the percentage of balls that the green player owns in the system is initially 50%. Similarly, an n-player system would have n colors present. At each time step, a ball is randomly selected from the urn and then replaced along with a certain number of balls of the same color as the one chosen. The amount of balls that are added to the urn depends on factors such as transaction fees and block rewards.

Based on this definition, we model the Polya's Urn problem using a transient Markov chain with states representing amounts of balls of each color in the urn. The probability of moving from one state to another in which the amount of balls of color $c$ has increased is then equivalent to the number of balls of color $c$ in the first state divided by the total amount of balls. For example, if $X_j$ is the amount of balls of color j, $(X_1, X_2, ..., X_n)$ is a possible state in the chain for an urn with n colors and the next state would be $(X_1 + z, X_2, ..., X_n)$ with probability $X_1/(X_1 + X_2 + ... + X_n)$. Here, $z$ is the amount of balls of color 1 added to the urn.

We determine the initial distribution for the Polya's Urn model by fetching empirical data from blockchain.info/q which exposes an API for fetching various useful information from the blockchain. In particular, we fetched the block reward (12.5 BTC) and the transaction fee (averaged to 25 BTC per day, with 144 blocks a day, comes out to a transaction fee of 1/6 BTC per block on average).

A simulation of system evolution over time involves a random walk along our Markov chain, tracking the proportion of balls owned by a certain player over a large period of time. Since martingales converge to the beta distribution as the time approaches infinity, this proportion should also converge to within an error bound after a certain time period. Thus, the main questions that we examine here are a) what the limiting distribution is and b) how much time it takes to reach this distribution.

Below is some pseudocode for our simulations:

```
def simulation(initial_dist, num_steps, num_balls):
    INIT data
    data[0] = initial_dist
    for epoch in num_steps:
        #copy the previous distribution of balls
        #get the current distribution of balls
        #pick a ball to draw
        #add block_reward(epoch) balls
        #add transaction_fee(epoch) balls
        #set this new distribution into the data
    return data
def block_reward(epoch):
    # taken from the web
    return hashpower(12.5)
def transaction_fee(epoch):
    # empirically estimated from the web
    return hashpower(1/6)
def hashpower(reward):
    # convert BTC to USD
    USD = 8146.70 * reward
    # convert USD to hashpower by buying AntMiner
    miners = USD/3000
    hashpwr = 13 * miners
    # return hashpower as percentage of total hashpower
    return hashpwr / 31000000
```

# 3 Experiments

We first simulated a 2 player Polya's Urn over several time steps and different initial distributions to see what distribution the initial distribution converges to. Since the 2-color case converged to a beta distribution, we hypothesized that the k-color case would converge to a Dirichlet distribution.

We sampled an initial distribution from the proportions of Bitcoin mining pools by hashpower, from April 8th 2018. We sampled the top 12 mining pools from this hashrate chart, which accounted for 98.8% of the total hashing power on the Bitcoin blockchain. We ran the Polya's Urn model on the initial distribution for 1440 timesteps, and ran this overall simulation 100 times.

The amount of balls added to the urn each time is $b + t$ where $b$ is the block reward (generally fixed for every period of 200,000 blocks) and $t$ is the transaction fee (variable, but approximated for our setting using a linear regression model). These are the rewards for successfully validating a block. We add them back to the player's wealth to represent the ability to invest this additional capital into hashing power (in Proof of Work) or into staking power (in Proof of Stake).

# 4 Results and Analysis

We prove that because the proportions of each color evolve according to a martingale, they must converge by the martingale convergence theorem in the case that the reward matrix is constant.

## 4.1 Convergence of the Distribution

**Theorem 4.1.1.** *Let $p_n^{(i)} = \frac{x_n^{(i)}}{\sum_{j=1}^{m'} x_n^{(j)}}$, be the proportion of stake of the $i^{th}$ validator after $n$ rounds. Let $\mathcal{F}_n = \sigma(v_i)$, where $v_i$ is the $i^{th}$ validator. Then $p_n^{(i)}$ is a Martingale w.r.t $\mathcal{F}_{n-1}$*

*Proof.*

$$E[p_n^{(i)}|\mathcal{F}_{n-1}] = E[\frac{x_n^{(i)}}{\sum_{j=1}^{m'} x_n^{(j)}}|\mathcal{F}_{n-1}] = E[\frac{x_n^{(i)}}{\sum_{j=1}^{m'} x_0^{(j)} + n}|\mathcal{F}_{n-1}]$$

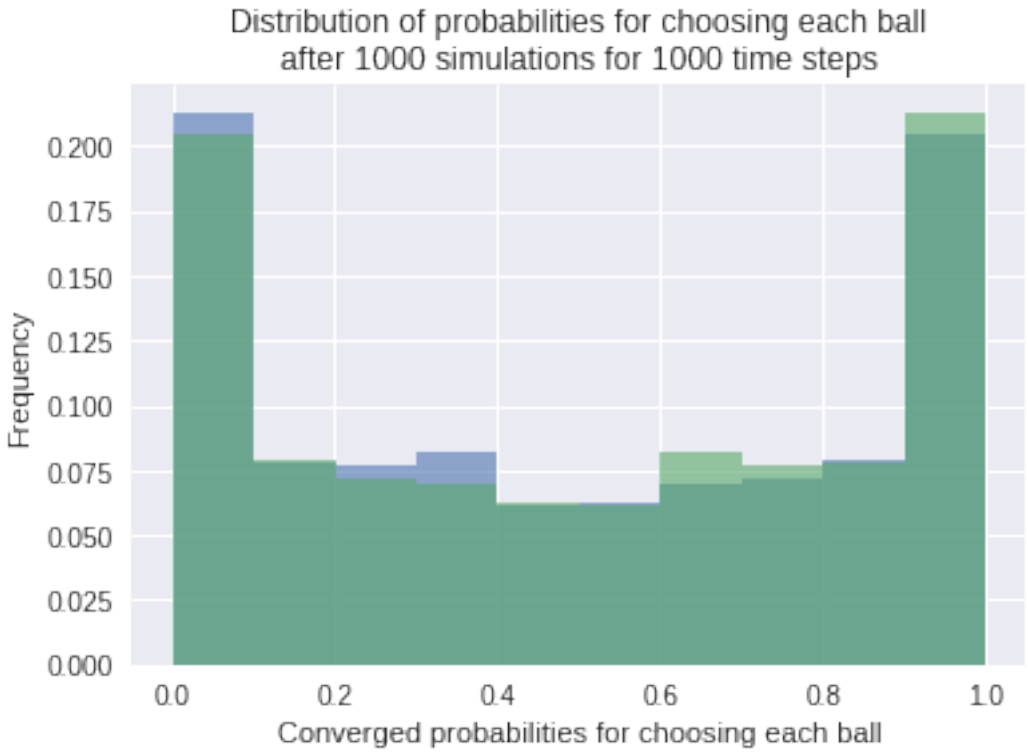
$$= \frac{1}{\sum_{j=1}^{m'} x_0^{(j)} + n} E[x_n^{(i)}|\mathcal{F}_{n-1}] = \frac{1}{\sum_{j=1}^{m'} x_0^{(j)} + n}(x_{n-1}^{(i)} + \frac{x_{n-1}^{(i)}}{\sum_{j=1}^{m'} x_0^{(j)} + n - 1}) = p_{n-1}^{(i)}$$

□

The takeaway from the above theorem is that since $p_n^{(i)}$ is a martingale and is bounded by 1 on the upper end, it must surely converge to some value.

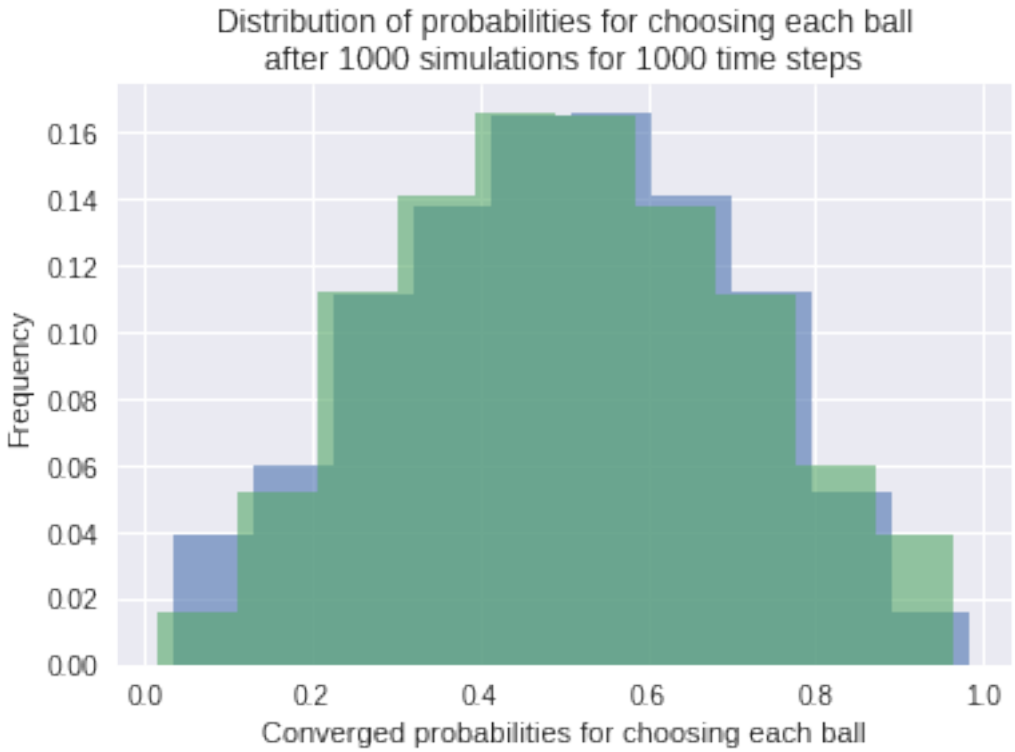
From the results of the experiment, we see that the two color case converges to a Beta Distribution. We see that in:

#### 4.1.1 Simulating 1 blue ball and 1 green ball



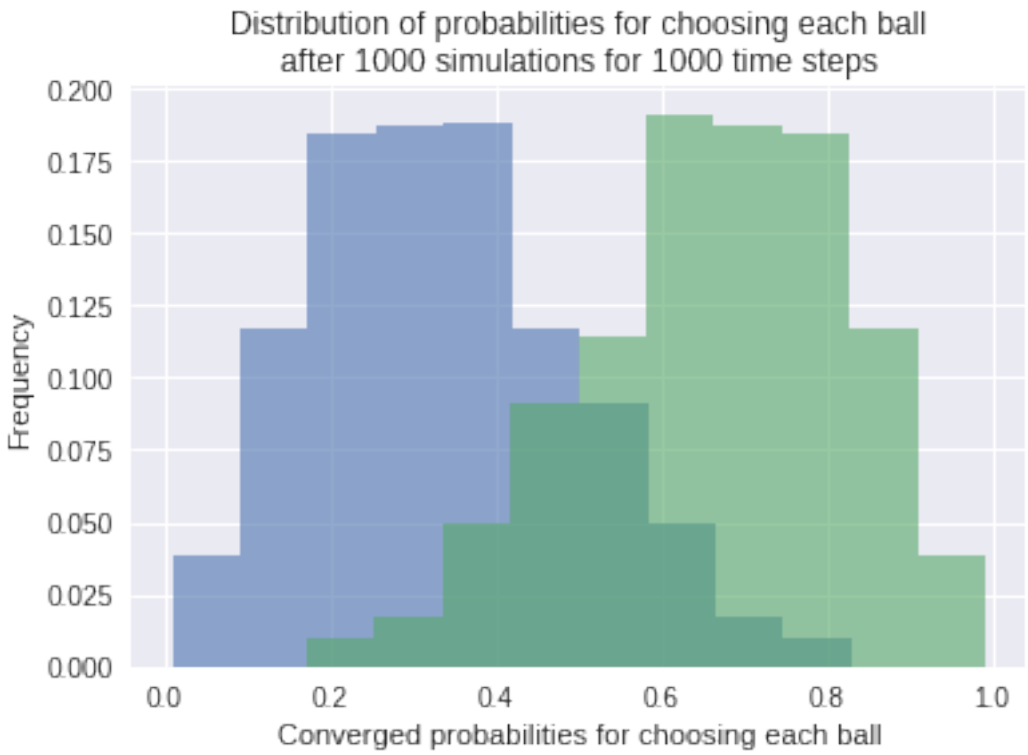

| Color | Mean | Variance |
|---|---|---|
| blue | 0.493211 | 0.123945 |
| green | 0.506789 | 0.123945 |

#### 4.1.2 Simulating 5 blue balls and 5 green balls




| Color | Mean | Variance |
|---|---|---|
| blue | 0.497823 | 0.04041 |
| green | 0.502177 | 0.04041 |

#### 4.1.3 Simulating 5 blue balls and 10 green balls




| Color | Mean | Variance |
|---|---|---|
| blue | 0.338045 | 0.024672 |
| green | 0.661955 | 0.024672 |

For a general $k$ color case we guess it must converge to an extended Beta distribution i.e a Dirichlet distribution. We confirm using the theorem below that the probability density function matches that of the Dirichlet distribution so it converges to a Dirichlet distribution after sufficient time steps.

**Theorem 4.1.2.** *Suppose a Pólya urn starts with an initial composition of $X_0$ and initial total stake $\Gamma_0$. Let $X_n$ be the composition of the Pólya urn after $n$ draws, then*

$$P(x_n^{(j)} = x_0^{(j)} + y_j, j = 1, .., m) = \frac{\prod_{j=1}^{m} \langle x_0^{(j)} \rangle_{y_j}}{\langle \Gamma_0 \rangle_n} \binom{n}{y_1, .., y_m} \tag{1}$$

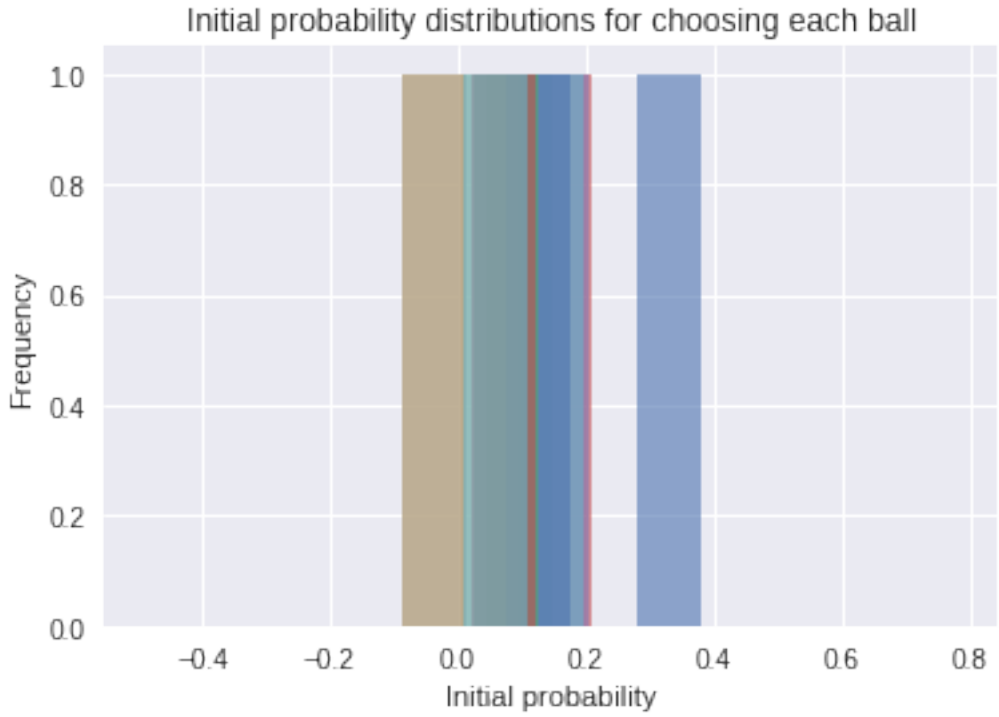
*Proof.* If the $i^{th}$ colored ball is drawn $y_i$ times, then $x_0^{(i)}(x_0^{(i)}+1)..(x_0^{(i)}+y_i-1)$ is in the numerator in some ordering. Thus independent of ordering the numerator

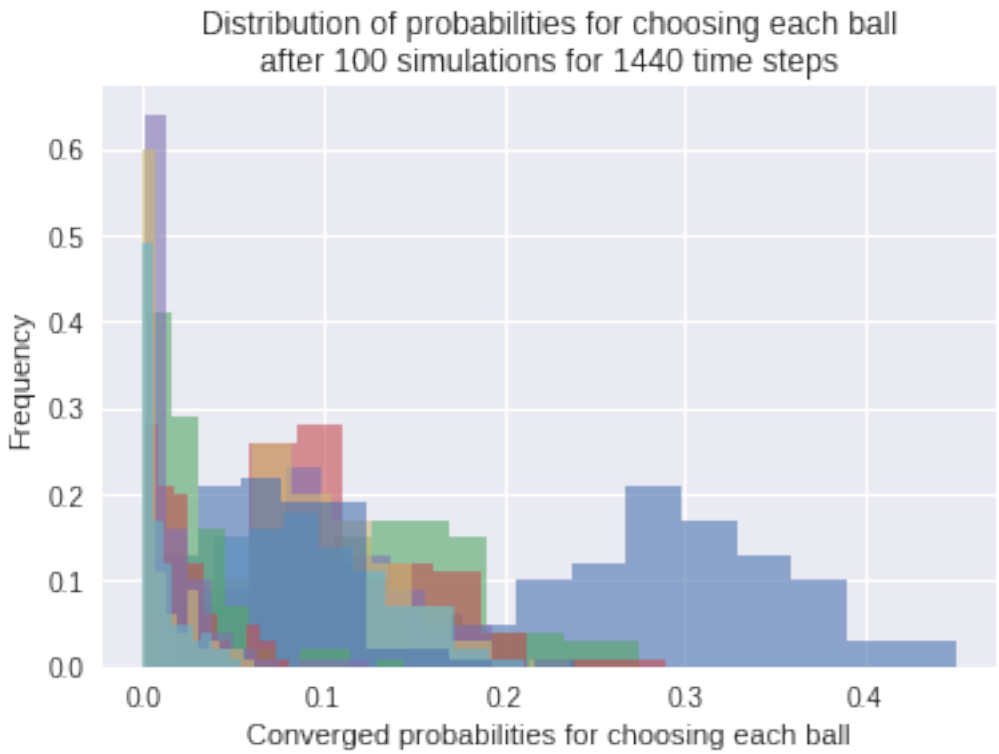
will remain $\prod_{j=1}^{m} \langle x_0^{(j)} \rangle_{y_j}$ and $\langle \Gamma_0 \rangle_n$ will be in the denominator. There are $\binom{n}{y_1,..,y_m}$ ways of ordering the distribution in the numerator. □

## 4.2 Empirical Results

We measured the KL divergence between the hashrate distribution after running our simulation for 1440 steps and the actual hashrate distribution 1440 blocks after the sampled initial distribution, and averaged it over the 100 simulations that we ran. The error remained within a very small bound of $\leq 1\%$, meaning that our simulation was very accurate. This simulation set the convergence time to be 14400 minutes.

### 4.2.1 Simulating on Blockchain data






| Name | Initial | Final | Simulated Mean | Simulated Variance |
|---|---|---|---|---|
| BTC.com | 0.275 | 0.278 | 0.290169 | 0.004196 |
| AntPool | 0.143 | 0.134 | 0.145824 | 0.002120 |
| ViaBTC | 0.106 | 0.118 | 0.111854 | 0.002018 |
| BTC.TOP | 0.103 | 0.092 | 0.099993 | 0.001649 |
| SlushPool | 0.101 | 0.107 | 0.099059 | 0.001553 |
| Unknown | 0.095 | 0.105 | 0.093240 | 0.001759 |
| F2Pool | 0.074 | 0.074 | 0.073369 | 0.001523 |
| BTCCPool | 0.027 | 0.011 | 0.027088 | 0.000588 |
| BitFury | 0.023 | 0.027 | 0.021813 | 0.000325 |
| BitClubNetwork | 0.014 | 0.013 | 0.015812 | 0.000461 |
| BW.Com | 0.011 | 0.015 | 0.011363 | 0.000202 |
| Bitcoin.com | 0.01 | 0.013 | 0.010416 | 0.000152 |

# 5 Discussion

The largest limitation was that our simulation is very accurate for Proof of Stake; a largest stake of cryptocurrency directly corresponds to a higher likelihood of receiving the associated block reward and transaction fees. However, the Bitcoin blockchain does not use Proof of Stake as its consensus mechanism, and the blockchains that currently live using Proof of Stake (NEO, which actually uses Delegated Proof of Stake) do not have enough data or enough nodes for a distribution to be considered stable. Therefore, we had to approximate the addition of hashing power by converting the block reward to USD, then USD to hash power by considering a theoretical purchase of the latest model of Antminer, then the hash power to percentage of total hash power on the Bitcoin blockchain.

This analysis has a few major flaws; the most apparent is that miners rarely withdraw their BTC, preferring to let it accumulate in value over time, and therefore the effect of the rewards of mining can be much larger than we account for. The next is that we were only able to account for the 12 largest mining pools; although these account for nearly 99%.

Regardless, we believe the very high accuracy of this simulation even on Proof of Work, and even not accounting for all of the mining pools, indicates that this simulation would be nearly perfect in estimating the convergence and behavior of the Markov Chain representing the Proof of Stake consensus model, even when accounting for variable block rewards and transaction fees as we did.