TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

# The Bitcoin Miners' Game: A Theoretical and Simulation Work

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Statistik–Wirtschaftsmathematik**

eingereicht von

**Tim Crailsheim B.Sc.**

Matrikelnummer: 01325407

Ausgeführt am Institut für Stochastik und Wirtschaftsmathematik
der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuer: ao. Prof. Dr. Gernot Tragler

Wien, am 10. Jänner 2023

| Tim Crailsheim | Gernot Tragler |
|---|---|

# Kurzfassung

Die folgende Arbeit beschäftigt sich mit der Fragestellung, warum sich die Bitcoin Mining Leistung langfristig und dauerhaft steigert, warum Mining Unternehmer praktisch dazu gezwungen sind, immer mehr Leistung ins Netzwerk einzubringen, und welche Strategien diese beim Equipmentnachkauf verfolgen können.

Da dieses Thema, insbesondere der damit verbundene Energieverbrauch, oft in den Medien präsent ist, sich die meisten Leute aber nicht über die Hintergründe bewusst sind, soll diese Arbeit eine Erklärung von Seiten der Miner liefern. Nach dem Lesen der Arbeit ist ersichtlich, dass die Miner sowohl strategisch als auch immer profitabel handeln und jede Chance zur Optimierung ergreifen müssen und somit auch nichts verschwendet werden darf.

Zur Veranschaulichung davon wird eine Kombination aus spieltheoretischer Analyse und Modellierung mit anschließender Simulation am Computer verwendet.

# Abstract

We use a compact model to analyse the important incentives for “proof of work” miners, in the case of bitcoin. In the next step we create a game between the miners where they optimize their cost/profit functions under a given budget constraint by choosing a certain technology level, comparable with the amount of mining computers. Just as in the real bitcoin protocol, after a certain time the difficulty adjusts and a new computational power might be chosen. We make use of different strategies and change the payout from a ”winner takes it all” to a shared but smaller constant one.

We simulate this game and find that most miners have to quit over time, due to their personal budget constraints and the irregular payout as assumed, in the first version.

As a result, we study the benefits of mining pools under a game theoretic aspect and why they are so common. We will also motivate this aspect in our situation game.

# Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich bei dieser Arbeit sowie auch das ganze Studium hinweg unterstützt und begleitet haben.

Mein größter Dank gilt hier meiner Mutter, die mich durchgehend darin bestärkt und motiviert hat weiter zu machen, egal wieviel Zweifel ich auch zwischendurch hatte. Auch für deine finanzielle Unterstützung möchte ich mich auf diese Weise nochmal bedanken. Ohne dich wäre ich sicher nie so weit gekommen!

Als nächstes möchte ich mich bei meiner Freundin Elena bedanken, die mir jeden Tag, an dem ich sonst nur allein gelernt oder diese Arbeit geschrieben hätte, gerettet hat. Auch vielen Dank für deine durchgehende Unterstützung, die Versuche, meine Erklärungen nachzuvollziehen, und auch ganz besonders für die ganzen Korrekturvorschläge für diese Arbeit.

Nicht unerwähnt will ich auch alle meine Studienkolleginnen und Studienkollegen lassen, speziell Babsi, Sarah und Peter, mit denen ich oft unzählige Stunden täglich verbracht habe. Egal ob es jetzt um gemeinsame Lerneinheiten oder das Feierabendbier danach ging, egal ob gemeinsam gelacht, geraged oder verzweifelt wurde, ihr wart immer da, danke.

Zuletzt will ich auch meinem Betreuer, Gernot Tragler, danken. Du hast mir die Freiheit gegeben, mir ein Thema selbst auszusuchen, bei dem wir beide nicht wussten, was herauskommen wird. Du hast dich dafür auch in ein komplett neues Thema eingelesen und meine Arbeit sogar bei einem internationalen Kongress vorgestellt. Dieses Vertrauen bedeutet mir wirklich viel, danke.

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 10. Jänner 2023

Tim Crailsheim

# Contents

# 1 Introduction

The following thesis deals with the question of why the Bitcoin mining performance increases in the long run, why miners are practically forced to bring in more and more power into the network and which strategies they can use to decide when to buy more power.

This topic, in particular the associated energy consumption, is often present in the media. Since most people are not aware of the background, this work should provide an explanation of the miners' view. After reading this thesis, it can be seen that the miners always have to think strategically profitably, that every chance to optimize must be taken, and therefore nothing can be wasted - as often presumed.

One of the main problems is that seemingly very few researchers work on that topic. That's why one of the few papers, written by June Ma, Joshua S. Gans and Rabee Tourky in 2018 [6], is taken as a base. It is a very simple and theoretical model and leaves a lot of room for improvement. In the following thesis this base model is taken and tried to be modelled on the computer. It turned out quickly that nearly nothing of that theoretical base paper was usable for a simulation, which is why the model was created newly. So in short the paper gave a good incentive to create a more realistic model.

Additionally, the white paper of the father of bitcoin Satoshi Nakamoto [8] was taken, as well as the book from Evans and Pritzker [4] for better technical understanding.

For computerised modelling the language Python was used. Additionally, aspects of game theory analysis played an important role.

To get a rough idea about the thesis, it can be described as a logical step-by-step path from the beginning of mining to why mining pools are the logical consequence. A better insight can be obtained by reading Chapter 5, "Interpretation and Results".

# 2 Theoretical Background

## 2.1 What is Bitcoin?

As a result of the big financial crisis in 2008, Satoshi Nakamoto published the bitcoin[1] white paper in the same year [8]. Since then bitcoin is regularly present in the media for positive just as for negative aspects. Some say it will revolutionize the money system, while others say it will destroy the world with its energy consumption.

To start at the beginning, bitcoin was the first electronic peer-to-peer payment network. Its basic idea is that monetary transactions should be accessible for anybody and independent from a central authority, like (central-)banks or states. A completely safe transaction between strangers that cannot be stopped should be guaranteed. Additionally, it should not be possible to freeze or hack any accounts. The monetary value should be saved in a limited good, that's why there will never be more than 21 million bitcoins. Therefore inflation is prevented.

The core of bitcoin is the "blockchain", a decentralized public ledger where all transactions are written and saved on. In more technical words, it consists of blocks of information, which are connected, and with every adding the previous blocks get more difficult to change. Nakamoto compared this with a Gambler's Ruin problem.

## 2.2 The mining process

Bitcoins are mined or created with the Proof-of-Work[2] algorithm. The function of the so called miners is to verify every new block and get bitcoins as a reward for that. In such a block there is usually a lot of information. For a basic understanding the most important parts are, the hash of the previous block[3], the list of all transactions and the nonce[4]. All

[1] abbr. BTC
[2] abbr. PoW
[3] an explicit value, explained later
[4] a randomly chosen value, also explained later

this is put into the sha256 function, which generates a specific unpredictable output (a hash value in cryptography). The sha256 function is known for taking any input and converting it into a 256-bit (32 bytes) hash value. It is usually represented as a hexadecimal number of 64 digits, that means a combination of digits and letters.

The goal for the miners is to get a specific hash output, which is represented by a number of zeros at the start, for instance:[5]

$$000000000000000000094bfa4edb1245c347e42452e4418e9fe5a1d24e335b16$$

For accomplishing that, the miners try different nonces, the third input beside the hash of the last block and the transaction list. They put different inputs and check if they get a suitable output. The moment they find that proper nonce, it is sent to the network. Everyone can check if the block works with the nonce, the miner gets a reward and can add the block to the blockchain. Afterwards everything starts again with the hash just found. The following Figure 2.1 illustrates that process.

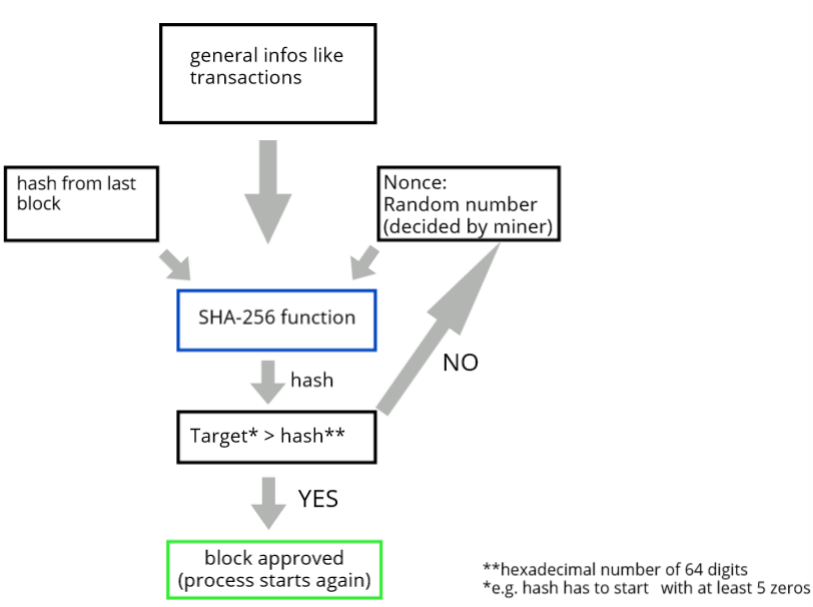


Figure 2.1: mining process

Simply put, it is like participating in a lottery and every nonce, so every try of getting the right output, is a ticket. Every miner has to pay for the ticket in the form of hardware and

[5] for getting a feeling for sha256 try: https://emn178.github.io/online-tools/sha256.html

energy.

One can come quickly to the consideration that the more tickets are bought, the faster the blocks are mined. This is of course true, because the more tries the higher the probability to get a right nonce. Therefore bitcoin has a built-in difficulty check. Every 2016 blocks (approximately every 2 weeks) the average time between two consecutive blocks is checked and if it is not around 10 minutes, the difficulty is adjusted. This happens via changing the necessary number of zeros in front of the target hash. For the system it does not matter how many miners are competing. The more miners there are, the harder and hence more expensive to cheat in any way. One general idea behind bitcoin was always that it should be more rewarding to help the system, rather than betraying it.
For more technical knowledge the book by Evans and Pritzker [4] is recommended.

## 2.3 An insight into the current numbers

At the moment the highest estimated hash rate of bitcoin was 265 EH/s[6], that means there were 265 000 000 000 000 000 000 tries per second to find a valid block [2]. Figure 2.2 can be contemplated for a good visualisation of the development over the last 3 years.

New good ASIC miners nowadays have a power of about 100 TH/s[7], costing about $16 000 each. That means, about 2,65 million of these high-level machines would be necessary to provide that computational power while also old less powerful and less efficient machines are used.

That leads to the widely known topic that these computers consume a lot of energy. The University of Cambridge publishes the estimated data about the yearly bitcoin power demand [10]. The estimation for 2022 is about 148TWh[8] per year, which is a lot in comparison to Austria's energy consumption of 65TWh in 2020 [9]. To be fair, the comparison to a country is not suitable. It is better to compare it to a similar use case like the yearly gold production (estimation around 165TWh) or the banking system (with an estimation around 700TWh) [7].

For classification, the last all-time-high price was around $66 000 per Bitcoin in October

[6] ExaHash/second
[7] 1.000.000 TeraHash = 1 ExaHash
[8] Terawatt hours

2021, while in the summer of 2022 the price was around $20 000. Therefore a big fluctuation can be seen, which is typical for cryptocurrencies [5]. The development of the price is also depicted in Figure 2.2. However, in general it can be said that the prices constantly go up in the very long run. In 2008 one bitcoin was nearly worthless, 2017 it broke $1 000. If we take the year 2017 as a start, it made an average revenue of about 110% p.a., considering the years before 2017 even more. The reward for a single miner at the moment would be 6,25 BTC per block, corresponding to approximately $125 000.

These numbers are important to understand the motivations of miners to enter the game and to understand what monetary values are involved in this business in general.

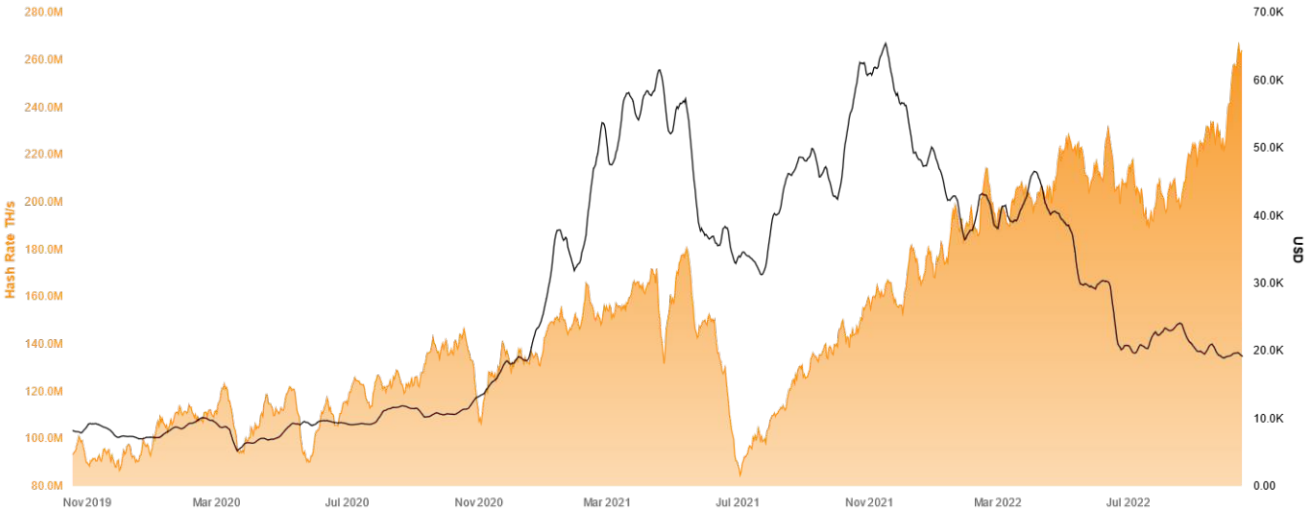


Figure 2.2: Hash Rate vs. Price [2]

## 2.4 Alternatives to Proof-of-Work

A valid question would be if this effort is really necessary. There is an alternative to Proof-of-Work called Proof-of-Stake[9]. The idea behind PoS is that consensus is reached by selecting validators in proportion to their quantity of holdings in the associated cryptocurrency (the stake). That means, the more stake someone has, the more influence s/he has over the network. This makes sense, because if these people betray the system, they have the most to lose. PoS of course needs a lot less energy, because of the unnecessary

[9] abbr. PoS

computational power.

That firstly sounds like a good alternative to solve the energy discussion of the bitcoin industry. In reality however, it is not an option when considering the basic idea of bitcoin. There has to be a 100% save peer-to-peer transaction, where nobody has to be trusted. This applies for PoW since it is pure math, in comparison to PoS where the stakeholders have to be trusted.

## 2.5 Mining pools

A problem for miners can occur if they can't solve a block for a long time, even if they have great computational power. This firstly doesn't sound like a problem, because probability says that in the long run they will solve blocks and get their profits. This, however, doesn't pay current bills, like those for energy costs. That's why the miners started working together very early in so called "mining pools" where they combine their resources and split their, in that way more frequently earned, profits depending on their share of the pool. The following table provides an overview of the biggest current mining pools [3].

| **Name** | **share in general mining** |
|---|---|
| AntPool | 15,5% |
| F2Pool | 14,5% |
| Poolin | 11,5% |
| ViaBTC | 11,3% |
| Foundry USA | 10,7% |
| Binance Pool | 10,1% |

It can be seen that only the 6 biggest pools are responsible for over 70% of all mining activity. They only represent a small fraction of the total number of pools. This shows that there are hardly any 'solo miners', which will be also explained in the following thesis.

# 3 Different Approaches

The author decided to start with a simulation that is as realistic as possible. In the first approach the payout is probabilistic which means that it is pretty risky for the miners, just as in reality. After that, the payout of each miner will be a share of the overall mining power, so that they can improve their future planning.

It can be seen that this little change in payouts will make a huge difference in the surviving probability of the miners. In the following pages different intern strategies of the miners and a more general game theoretic approach will be presented, explaining why miners would participate in upgrading equipment in the first place.

## 3.1 General introduction of the simulation

At this point a rough description of the simulation program will be given, for more details please check Chapter 7; also the full code will be given in the Appendix (Chapter 8).

### General basic conditions

Firstly there are some general basic conditions of the simulation program, that are used for all approaches. Of course these numbers can be changed, but experience has shown that these numbers lead to stable outcomes. In other words different numbers don't make a big change considering the outcome. With the following setting, interesting results can be observed:

- number of miners: $N = 20$
- block rewards per round: 10
- rounds: 100 or 500
- iterations of the whole process: first 1, then 100
- additional exogenous variables: energy cost, Bitcoin price, amount of Bitcoins paid every round, initial budget, etc.

For reasons of simplicity only the most basic variables are mentioned in the list above. For a complete list of all variables, please refer to Chapter 7.

### Flow description

In the following overview a description of the algorithm for every round is given:

(0.) every miner $i \in N$ initially starts with a random technology level $x_i \in [1, 20]$. (This step is just for initializing and only done once)

1. every miner's share of the overall mining power $s_i = \frac{x_i}{X}$ [1] is calculated and so a vector of intervals for everyone is created
2. then random values are calculated and it is checked in which miner's interval they are and thus winners are drawn
3. finally the new budgets are calculated by adding the BTC wins and subtracting energy and acquisition costs (if a budget becomes negative, the miner is out of the game)
4. now the miners can use different "strategy functions" to buy more or sell their equipment and the flow starts again at step 1. with the new technology level (if a technology level is set 0 or negative, the miner is out of the game)

### How to read the graphical output

There will be two different types of output, depending on how many iterations will be done. If there is only one iteration, two graphics will be shown. On the x-axis, they both have the simulation rounds and on the y-axis one has the budget of the miner's, and the other one the corresponding technology levels. With this setup the correlation between technology level and budget changes can be seen. All miners should have a unique colour.

The second type of output can be seen when there are 100 iterations. Here are three graphics. Since the pathway is not important any more, just the outcome at the end is presented. On the x-axis the numbers of the surviving miners can be seen. On the y-axis, the average technology level and the average budget of these survivors can be seen (arithmetic mean). At last, there is a further graphic with the sum of the budgets of all surviving miners to check if there is a difference in the overall value depending on how many miners survive until the end.

[1] $X = \sum_{i=1}^{N} x_i$

### Different strategies

This can be considered the most interesting part of the simulation. This is the part where miners decide if they want to invest in more equipment (= technology level up), if they just want to keep what they have, or if they want to sell. For a better visualization one can imagine the number of technology levels as the number of mining devices a miner owns. There are an unlimited number of strategies. The author started with some simple strategies and increased their complexity step by step. In this thesis every miner uses the same strategy. In the following enumeration the ideas are presented:

1. "The random strategy $S_r$": This strategy can be seen as a benchmark, so providing a control if the other strategies are better than complete randomness.
   After every round, every miner chooses with a probability of $\frac{1}{3}$ to do nothing, with a probability of $\frac{1}{3}$ to sell 5 technology levels and with a probability of $\frac{1}{3}$ to buy 5 new technology levels.
   $z[i] = random.randint(0, 2) \Longrightarrow$ equally distributed
   $$S_r(x_i, z[i]) = \begin{cases} x_i, & z[i] = 0 \\ x_i + 5, & z[i] = 1 \\ x_i - 5, & z[i] = 2 \end{cases}$$
   Note: In case $x_i < 5$ the miner will be thrown out of the game

2. "just buy, $S_{jb}$": Miner $i$ will buy equipment if s/he has money, meaning if her/his budget ($B$) was positive in the last round $(t-1)$:
   $$S_{jb}(x_i, B_{i,t-1}) = \begin{cases} x_i + 1, & B_{i,t-1} > 0 \\ x_i, & B_{i,t-1} \leq 0 \end{cases}$$

3. "just buy advanced $S_{jba}$": Miner $i$ will buy equipment if s/he has money **and** if her/his budget has increased in the last round:
   $$S_{jba}(x_i, B_i) = \begin{cases} x_i + 5, & B_{i,t-1} > 0 \wedge B_{i,t-1} > B_{i,t-2} \\ x_i, & else \end{cases}$$

4. "thought out, $S_{to}$": Miner $i$ will buy equipment if his expected profit is positive, i.e.if their share of the overall mining power ($\sim$ their expected win) is bigger than their running and acquisition cost, or in simple words *if the expected long-term profit is positive, buy.* On the other hand, if the budget decreased in the last rounds, equipment has to be sold to get positive profit again.
   t* = time steps needed for winning;
   ec = energy cost

$$S_{to}(x_i, B_i) = \begin{cases} x_i + 5, & \underbrace{\frac{x_i}{X_{t-1}} * BTC_p * 6,25}_{expected\ win} - \underbrace{(x_i - x_{i,t-1}) * 100}_{acquisition\ cost} - \underbrace{t^* * ec * x}_{running\ cost} + \underbrace{B_{t-1}}_{current\ Budget} > 0 \\ x_i - 5, & B_{t-3} \geq B_{t-1} \\ x_i, & else \end{cases}$$

**Side note**

The budget of the miners is calculated in a currency, for example € or $. That is because of two reasons: first, being at the moment mining equipment, energy costs, etc. cannot be paid in Bitcoin and second, the program is built in a way that the Bitcoin price can change over time to become more realistic.

## 3.2 Probabilistic payout

In the first version of the simulation (V1), the payout follows "the winner takes it all" concept, just like in reality. The first miner who found the suitable Nonce gets the whole Bitcoin payout and all the others go away empty-handed and lose money because of their energy costs. In this simulation there are 10 winners chosen per round, which means that 10 block rewards are distributed. A single miner can also win multiple rewards per round. That comes from the fact that miners won't change their strategies every 10 minutes[2]. This also leads to a faster process and more results.

The problem with this "the winner takes it all" approach is that if miners lose some rounds in a row, they will probably soon have no capital left to pay their energy costs and hence they will have to resign. With every quitting miner the others have a better chance to win. This conclusion obviously changes if some of them change their mining hardware stock which would change the proportions.

Generally it can be observed that with this approach after some time, there are just a few to one single miner left in the game.

### 3.2.1 Simulation V1 - 100 rounds - 1 iteration

In what follows, two different strategies are presented:

1. The random strategy $S_r$ (Strategy 1): After every round, each miner has a probability of $\frac{1}{3}$ to do nothing, to sell 5 technology levels, or to buy 5 new technology levels.

   In Figure 3.1 it can be seen that in the first ten rounds, 17 miners have to resign. The miners have two problems. Firstly, they have to be lucky and win the payout in order to be able to pay the running cost. Secondly, there is a $\frac{1}{3}$ chance that they have to sell their equipment, which would obviously lead to a problem. Remember, they start with a technology level $x_i \in [1, 20]$, so the chance to sell 5 or 10 could happen fast and end everything.

   The second interesting aspect are the orange miners. Both of them seem to win a lot, their budget is constantly rising. However, around round 50 the first one is forced to sell his equipment to zero (this also leads to setting the budget to zero). Shortly

[2] 1 block $\hat{=}$ 10 minutes

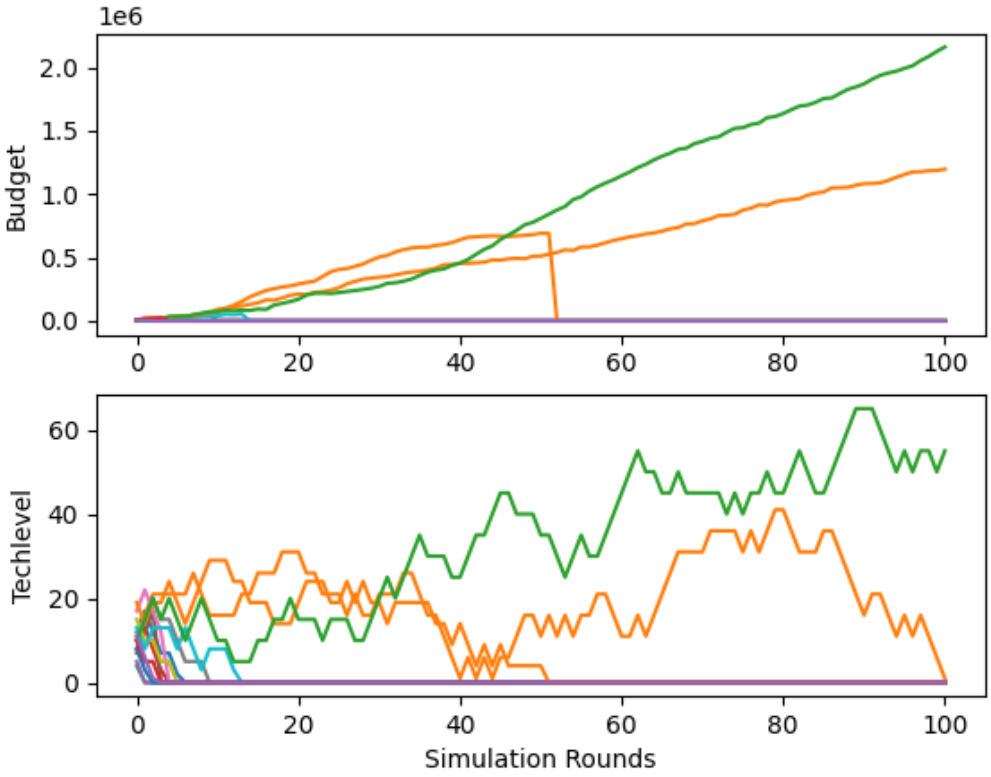


Figure 3.1: The random strategy

before the end at round 100, the second orange miner is causelessly forced to sell everything, too, although they would have had enough budget to survive. It is hard to see, but in the last round $x_{orange2} = 1$, which means they were really close to getting kicked out, too.

In general no logical strategy can be seen, even a few counter-intuitive decisions are made.

2. The thought-out strategy $S_{to}$ (Strategy 4): In a short summary it means that, if the expected long-term profit is positive, buy. On the other hand, if the budget decreased in the last rounds, sell equipment.

   By comparing Figure 3.2 to Figure 3.1, a completely different picture can be seen. In

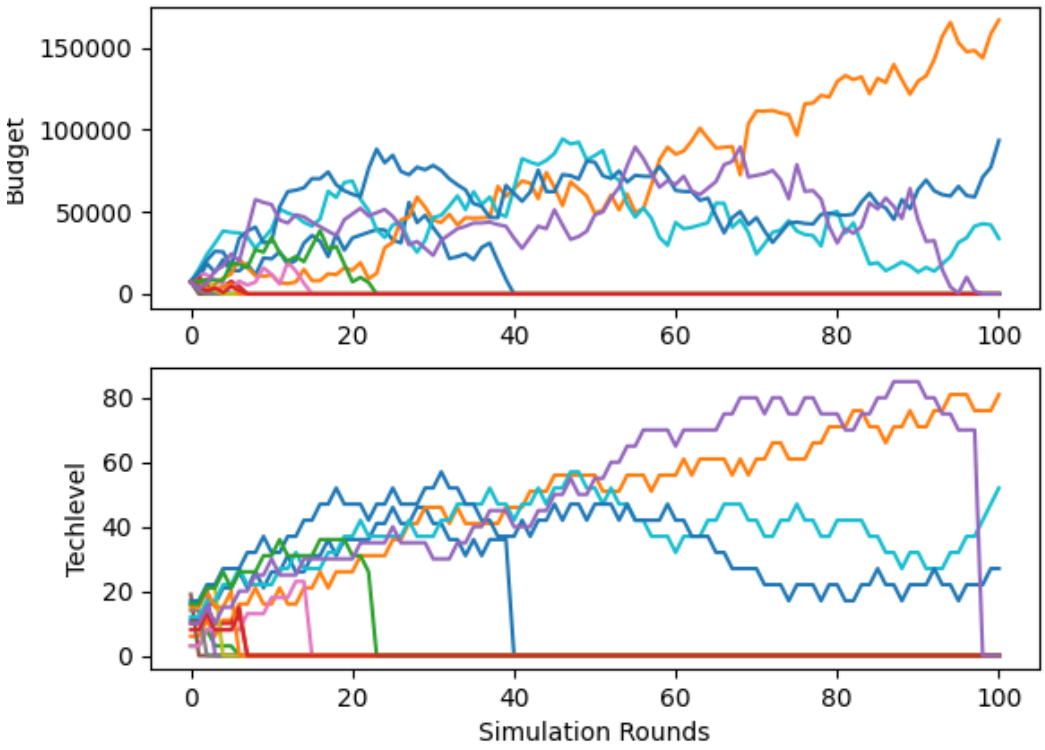


Figure 3.2: thought-out strategy

the new graphic, 7 miners survive the first 10 rounds and 3 survive all 100 rounds. There is no more approximately linear rising of the budget. The miners have to buy/sell very often which leads to a lot of movement in the graphs. It can be observed that with a more advanced strategy, the miners resign because of the capital loss, in contrast to being forced to do so by some random strategy.

It is important that even though the overall share of a single miner might be pretty high, if they don't get lucky and win, they slowly lose money and have to quit after some time, even if they should have a good standing in the long run.

The purple and the orange miners illustrate that perfectly. Their history is pretty equal until round 70. After that, the purple miner wins significantly less and even before round 100 they have to quit for budget reasons. Interestingly, most of the time between round 70 and 100 the purple miner has a higher technology level, so the odds

should have been better for them.

### 3.2.2 Simulation V1 - 100 rounds - 100 iterations

This part of the thesis summarizes the first, probabilistic, version of the simulation (Figure 3.3). The previous simulation is done 100 times in a row and the different outputs are recapped in one result. In the following only the results for the thought-out strategy (Strategy 4) are presented.

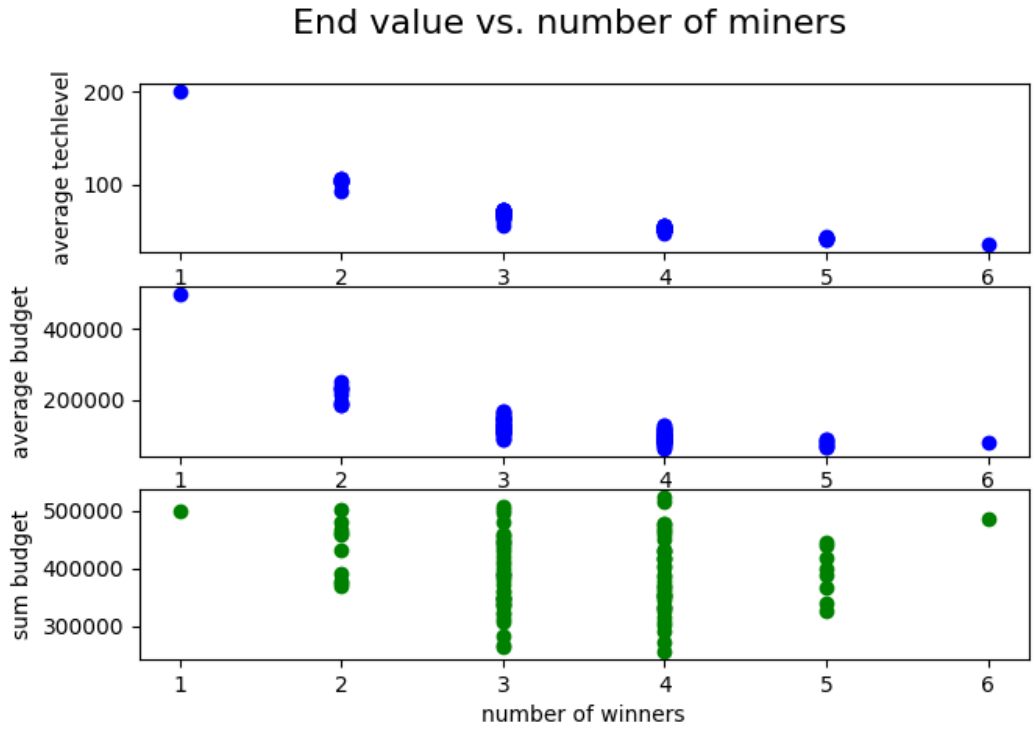


Figure 3.3: Thought-out strategy

When taking a look at the first "average technology level" graphic, an unlogical result can be observed. Although only one miner is left in the end, their technology level is very high (200) even though they have no more competitors. This results from the used strategy which forces the single miner to "fight until the end" as they don't know they are already alone. It can be seen as a future improvement, that for instance the last miner should sell all their equipment except one, to maximize their profit. However, this case will never happen in reality.

When summing up, 200 is roughly the technology level, that is reached in all iterations. If two miners survive, their average technology level is around 100 each, if three survive it's around 70, and so on.

A related pattern can be observed for the average budget. This correlation can be seen best in the third (green) graphic where all the average budgets are multiplied by the surviving miners.

The most frequently reached value is around 500 000, although a bunch of meanderings can be observed. There could be multiple reasons for that, however an explanation could only be given by looking at every single iteration. Some explanations could be that sometimes miners survive until shortly before the end or they are nearly bankrupt and therefore just pull down the average.

As a conclusion, one can say that it doesn't matter much how many miners survive, it doesn't change the overall monetary value in the end and stays roughly the same on average.

## 3.3 Shared payout

In the second version of the simulation (V2), the miners always get a share of the payout, depending on their share of the overall technology level. This means if a miner has 10% of all mining equipment, they will get 10% of the total payout every round.

For obvious reasons, this makes the future planning much easier for the miners. In this version, 500 rounds are simulated, the reason for which can immediately be seen by looking at the graphics.

### 3.3.1 Simulation V2 - 500 rounds - 1 iteration

In the following two different strategies are presented again:

1. First, the simplest strategy (Strategy 2 - "just buy") is used. It implies that if the budget is positive, 1 technology level has to be bought. For the sake of a better interpretation of the output this slow increase is chosen. If the increase is set higher every round (e.g. 5), all miners, except one (random), would go bankrupt extremely fast. This remaining one will have to extend their technology level to about 2500[3],

[3] 5*500 rounds

although it would make no sense without any competitors and would therefore lead to a senseless result.

That is why in the lower graphic of figure 3.4 a slow linear increase can be seen. From time to time miners fail because they go bankrupt, but all others raise their equipment stocks every round.

The more interesting graphic is the "Budget". In the first 100 rounds it can be seen that most miners increase their budget by a lot. Everyone profits nearly equally. However, there is a certain point when nobody can work profitably any more, because they have too much equipment with too high running cost compared to the existing bitcoin wins. Nevertheless, they still have to buy equipment.

Hence the following fall is as fast as the preceding increase and the fastest falling miners go bankrupt. As all of them still have to buy more equipment no matter what, it is more like a "bankrupt-interval". Afterwards the reallocation of the winning share starts again. That is why those who survive start their big budget increase again, until the crucial "non-profitability" point reappears and they start falling again.

2. For Figure 3.5, the more advanced Strategy 3 is used. It implies that if there was a budget increase in the last round, a miner should buy 5 more technology levels.

   That is why in the first graphic of Figure 3.5 a very fast increase of the budget can be seen, followed by a little drop and then a linear decrease. This results from the stop of buying new equipment, due to it not being profitable anymore and so causing smaller losses every round. However, after some time these constant losses lead to the first bankruptcies, followed by all other miners working profitably again, for a short time, which leads them to start buying equipment again. One can see that in the first 250 rounds there are just too many miners so that a sustainable increase is not possible. After that, more dynamics occur because the bankruptcy of one additional miner makes a greater difference for the surviving ones.

   It can be concluded that the possibility to "not buy more equipment" is for sure a huge advantage.

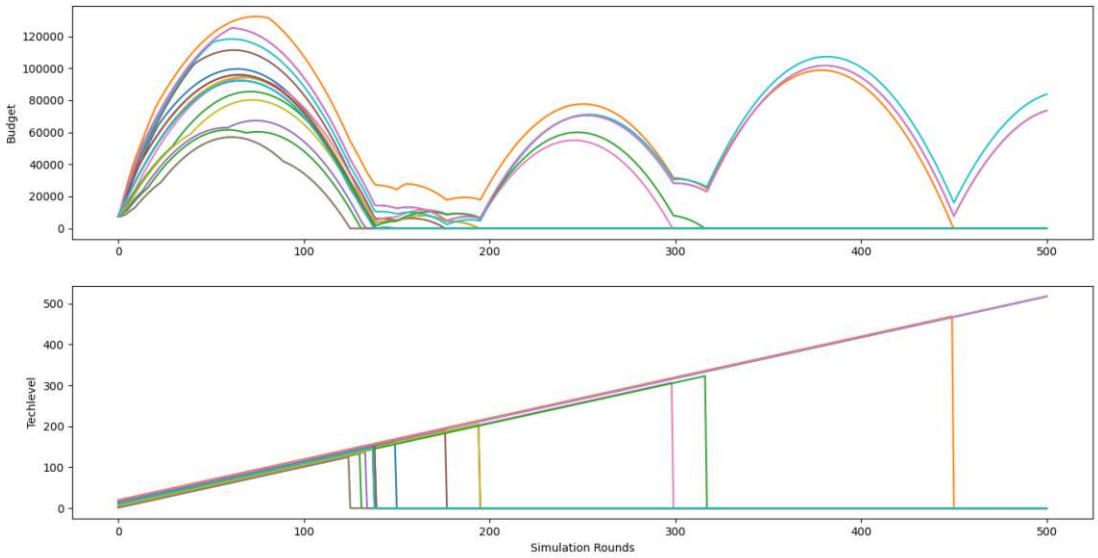


Figure 3.4: "Just buy" strategy

### Comparing the two strategies with share payout

When comparing the two strategies it is worth mentioning that there is a similar "crucial tipping point". In both figures one can observe that when all miners are around technology level 100, they can't work profitably anymore. This makes sense, because of the same payout of bitcoins and the equal price for them. The difference in increasing 1 vs. 5 levels per buy can also be seen very clearly in the amount of time steps to the tipping point.

As a last side note the height of the budgets is worth a look. At a small increase like "1 step" (cf. Figure 3.4) the wealthiest miner has about 120 000 Budget whereas in the "5 steps" case they don't even get to 30 000 (cf. figure 3.5).

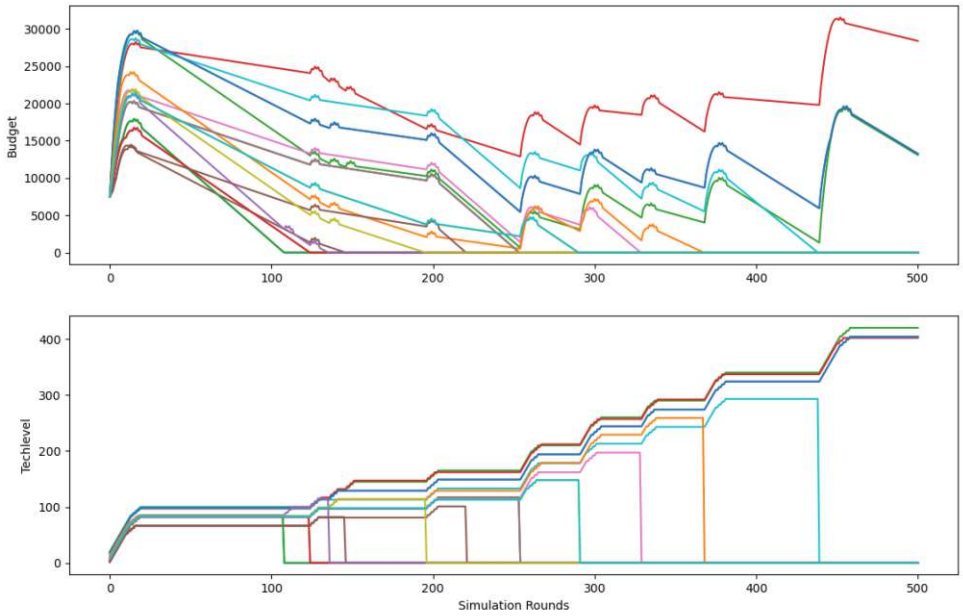


Figure 3.5: "Just buy advanced" strategy

### 3.3.2 Simulation V2 - 100 rounds - 100 iterations

In the following the shared payout simulation (cf. Figure 3.6) is done again. This time it is done just 100 rounds (vs. 500 rounds before), but for 100 iterations. Although the solution is pretty straight forward, it is important for a comparison.

First of all, it can be mentioned that in the first 100 rounds nobody goes bankrupt. Secondly, the "just buy" strategy is used, so every round 1 is added to the technology level. This leads to an average technology level between 108 and 113, depending on the initial random value. Thirdly, the budgets, are higher caused by the very slow increase of the equipment stock generating a bigger save of money for the miners in the beginning of the simulation.

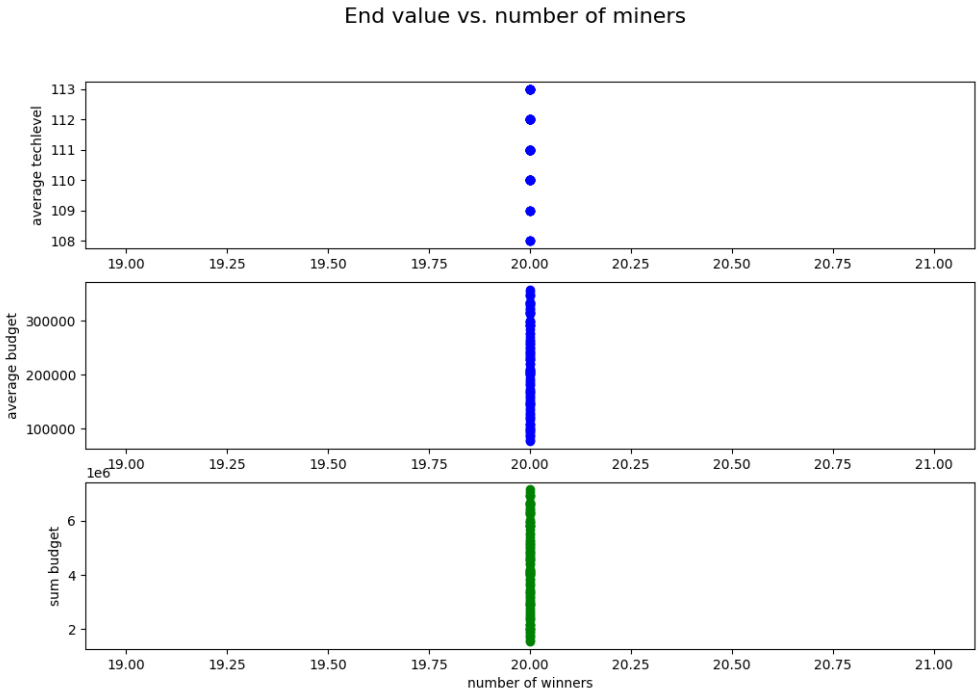


Figure 3.6: "Just buy" strategy

## 3.4 Game theoretic approach

The following chapter deals with the question of the general incentive why miners should buy new equipment after all. For that purpose, a small game theoretic model is considered.

For simplification it is assumed that there are only two miners, with the exact same equipment. Consider that the hardware equipment is also called "a miner". For the next time interval they can choose between keeping their current equipment or buying additional devices (in this case doubling it).

| | | Miner 1 | |
|---|---|---|---|
| | | don't buy more | buy more |
| Miner 2 | don't buy more | 0,5/0,5 | $0,\dot{3}/0,\dot{6}$ |
| | buy more | $0,\dot{6}/0,\dot{3}$ | 0,5/0,5 |

The values result from the idea that they always split the winning probability. The total value is 100%, so when both have 1 mining device the payoff is split at the ratio of 50:50, if

one has 2 mining devices and the other one 1 mining device the ratio is 66:33, and if they both have 2 mining devices it is 50:50 again.

In the following the game is solved for dominant choices.

| | | Miner 1 | |
|---|---|---|---|
| | | don't buy more | buy more |
| Miner 2 | don't buy more | 0,5/0,5 | $0,\dot{3}$/**0,$\dot{6}$** |
| | buy more | **0,$\dot{6}$**/$0,\dot{3}$ | **0,5/0,5** |

$\Longrightarrow$ (buy more/buy more) is the strictly dominant choice

This implies a strong incentive for miners to buy more equipment to raise their probability to win. However, when all miners think that way and raise at the same speed, the probability will stay the same. This also means that if a miner wants to keep their share of the overall mining power, they have to buy new equipment or otherwise the other ones will change the probabilities against them.

This can also be observed in reality, where the hash rate (mining power) is constantly rising (see Chapter 2.3 "An insight into the current numbers" and in particular in Figure 2.2). So in fact it is an endless game for the miners, where they are forced to play and invest, otherwise they just lose slowly. The big changes only happen if miners go bankrupt or if there are big technical innovations, that only some miners can use.

An example of this was observed when China banned bitcoin mining in June/July of 2021 [1]. At first the hash rate went down, comparable with bankruptcy for a short time [2]. Other miners got a bigger share and probably huge profit. However, shortly after these banned miners returned in other countries, the hash rate went up again and then quickly it was even higher than before the banning, which is perfectly visualized in Figure 2.2.

Furthermore, it has to be understood that the miners have to optimize where they can. They always have to use the cheapest energy source (usually renewable or in any way subsidised energy). They also have to use all of their budget for mining equipment. It can be seen that the competition is permanent and a never-ending story. There is no break, and with every moment (or time step) the chances to win respectively their overall share of mining power changes. If they start this game, they must play it until the end.

# 4 The Consequences of an Increasing Bitcoin Price

The following chapter demonstrates the results of the simulation if the Bitcoin price increases over time. In Section 2.3 (An insight into the current numbers) can be seen that since the beginning of Bitcoin its value has increased constantly, when looked at it in the long run. Figure 2.2 shows this increase in a timespan of 3 years.

In the referred section an annual increase of 110% is mentioned. For the simulation in this chapter only 50% was chosen, because a constant increase in every round is implemented. This is in contrast to reality where big variability can be observed. Technically this is implemented by adding 0,1% of value to each bitcoin reward every simulation round[1].

In what follows, two cases are presented. One for the probabilistic payout and one for the shared one. It can be observed that the results are very similar to the original results disregarding some changes in scale, but the structure is kept the same as before.

The case of an increase like 1% or more per simulation round[2] will be mentioned here descriptively in short words. This percentage would make a huge difference. The miners would not be able to spend most of their wins on buying equipment any more due to technical restrictions, and their budget would increase exponentially. They would just be able to buy as many miners per round as they are allowed to. The results would be some exponential curves which don't give any additional or meaningful insight.

[1] 1 block $\hat{=}$ 10min; 1 simulation round $\hat{=}$ 10 blocks; 1 year $\hat{=}$ 52 560 blocks; 50% p.a. $\rightarrow \sim 0,1\%$ per simulation (due to compound interest a bit more than 50%)

[2] This would correspond to a constant Bitcoin value increase of 1% every 100 minutes!

### Probabilistic payout - Bitcoin increase

The setup for this result is exactly the same as for Figure 3.2 in Chapter 3.2.1, so please compare directly. In short words, a budget increase can be observed because the value of each new reward increases. Secondly, the maximum technology level increases from 80 to 100. Aside from that, the structure is very similar.

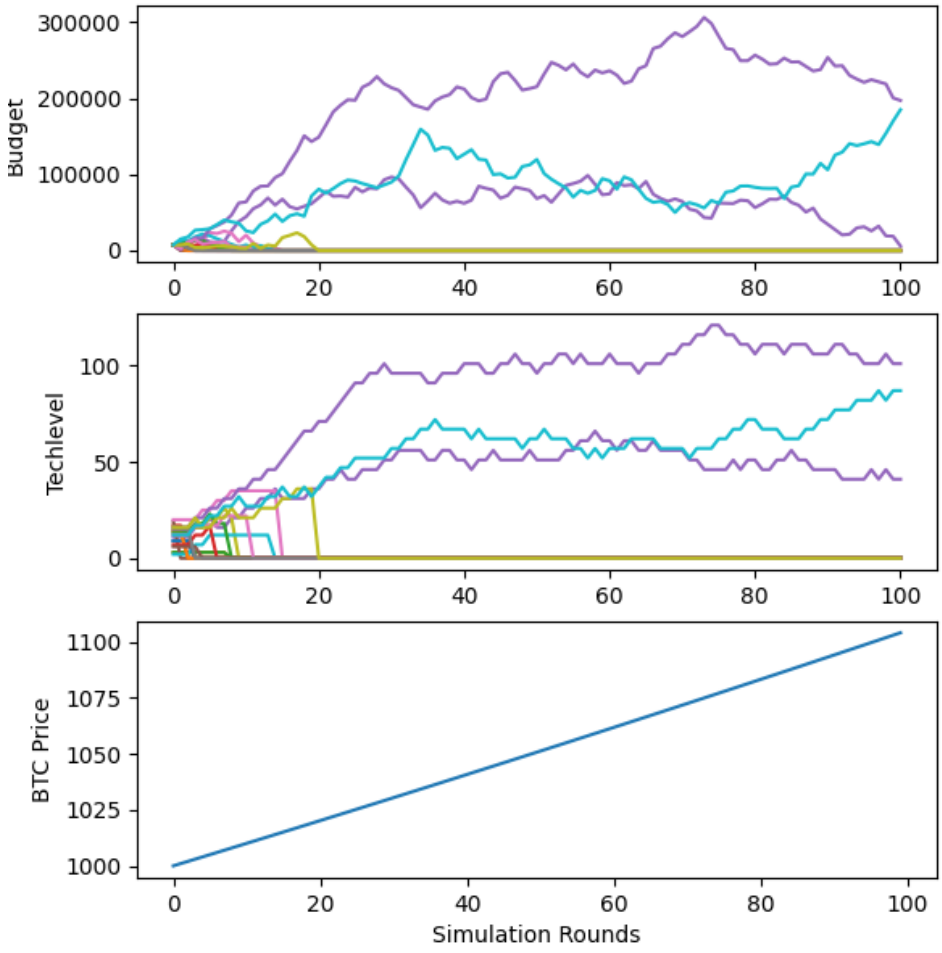


Figure 4.1: "Thought-out" - Strategy 4

### Shared payout - Bitcoin increase

In the following the setup is exactly the same as for Figure 3.5. Again it can be observed that the structure is comparable. Here the main difference is in the process of bankruptcy, which takes longer. Moreover particularly the overall budget is slightly higher. The maximum technology level, on the other hand, stays the same.

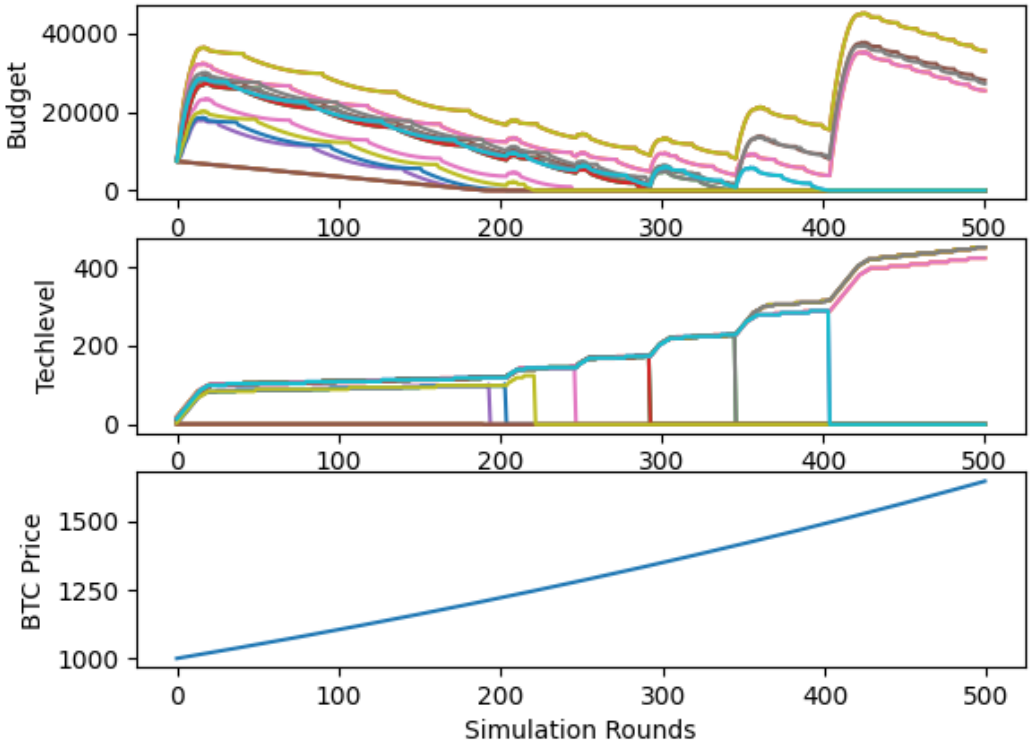


Figure 4.2: "Just buy advanced" - Strategy 3

# 5 Interpretation and Results

In this chapter everything is brought together and the central theme will be discussed.

### The game theoretic approach

Section 3.4 explains why miners are prompted to buy more equipment in general and also why all the other miners have to restock in the following time. There are no constraints, it is a simple thought experiment.

### The probabilistic approach

This part shows how difficult it is for the miners to calculate when the wins/earnings come infrequently and randomly. The problem is that they can only plan with long-term probabilities. However, if they don't have enough budget reserves until they reach their first wins, they will go bankrupt, no matter how good their odds, how large the equipment stock, or how sophisticated their strategy was. This can be well observed in the particular Chapter 3.2.

### The improvement through the shared payout approach

The main problem in the preceding approach was the random payout. This is solved with the shared payout every round depending on the percentage of the overall mining power (Chapter 3.3). This approach gives the miners a much better planning possibility and leads to better analysable results. There is no real randomness any more, everything is explainable.

However, when a longer horizon is observed, a new peculiarity occurs. There appears to be a maximum technology level for the miners, that when reached, leads to unprofitability and some have to go bankrupt. After that, the overall wins are redistributed and the loop starts again until some miners have to go bankrupt again. This is caused by the problem that the used strategies force the miners to buy new equipment or at least give no possibility to sell equipment, so their running costs rise with a capped win.

### Results for an increasing BTC price

This leads to the thought, what a changing Bitcoin price could cause, which is presented in Chapter 4. In short it can be said that it doesn't have a big influence on the structure of the results. Only in the case of extremely high rates of increase, nobody ever goes bankrupt and there is no interesting result in that.

### The way to mining pools

As a final result of this thesis shows, a natural way for miners is to form mining pools.

When someone decides to join the mining game, "The game theoretic approach" explains that they will buy more equipment when they can. In "The probabilistic approach" one can understand the risks of being a single miner, and finally "The improvement through the shared payout approach" leads directly to mining pools. The miners join a group and share a frequent small win and can therefore pay their running cost easier and gain safety.

# 6 Outlook on Future Works

This thesis has to be seen as a basic model and should help to understand the importance of strategies, the balance between the equipment stock and budget, and the high short-term risk of the miners.

There are a lot of possible future developments on the modelling side to include for instance the wear of mining equipment or generally more external effects, such as changing or different energy costs for every miner.

However, the most interesting aspects will definitely be new strategies. There are a lot more interesting approaches that could be implemented as well as letting the miners form groups with different strategies, while simulating and testing which one performs the best.

Another kind of paper or thesis could deal with a new model, where miners have a constant equipment stock, but they are able to switch off their equipment, e.g., when there are a lot of competitors or when the energy costs are too high.

# 7 Technical Explanation of the Simulation

In this Chapter the structure of the simulation program and how functions work together will be explained. The full code can be seen in the Appendix (Chapter 8), where a lot of comments can be found there to help to understand everything.

**How the different main functions work together:**

1. *repeater:* This function uses all other functions, its purpose is to repeat until all 'simulation_rounds' are done. It also collects all the data produced.
   a) *Evaluation*
      i. *rantime:* Here the time is given, which is needed to find a valid block. It is more or less exogenously given, but depending on the overall computational level. The higher the level, the shorter the time needed. All miners have to pay energy costs for this period.
      ii. *winner:* Here the winning miner is selected. Depending on the technology level they have chosen, they have a certain percentage of the overall computational power.
         - *jackpotFKT:* A random integer between 1 and 100 is put out.

         It is tested in whose interval this integer is. A higher technology level leads to a higher probability to win. This also allows small miners to win.
      iii. *CostFKT:* Every miners' running cost is determined. It depends mainly on their technology level, the time it takes to find a solution, and the energy cost. In the second step the Bitcoin reward is paid out to the winning miners.
   b) *acquisition_cost:* For the first round the initial *techchoose* costs are compiled and subtracted from the budget. For the following rounds only the changes of technology level are considered. No depreciation is supposed.
   c) *kickout:* If after one simulation round the budget is negative or zero, the technology level and the budget are set to zero, which means that the corresponding miner is out of the game forever.

d) *optimizer_*:* There are a lot of different optimizers used, from simple to more complex ones. They consider the change in the miners' budgets, the overall and particular technology levels, and so on. They try to find a good "next" technology level.

⇒ After all these compilations the next round starts.

2. *plotter:* Here all the collected data is used to be plotted.

**Initial/Side functions:**

1. *techchoose:* Miners get a random technology level for the first round, it is between 1 and 20, after the first round it is changed by the *optimizer_**.

2. *btc_price_change:* If the value of BTC changes over time it is calculated in this function.

3. *btc_price_changer_for_plot:* For easier plotting a vector with the changing BTC prices is calculated here.

# 8 Appendix

Full simulation program code (Language: Python):

```
#Packages:
-------------------------------------------------------------------------

import numpy as np                          #Math package
import matplotlib.pyplot as plt             #for plots
from scipy.stats import poisson             #for Poisson
import random                               #for random int techchoose-
    fkt

print("Mining model:")
print(" ")
#fix Variables:
-------------------------------------------------------------------------

N = 20                          #Miners
K=1000                          #difficulty level
energy_cost=3                   #Energy cost per compilation
btc_price= 1000                 #BTC price at time t
btc_price_increaser=0           #Percentage of btc price increase every
    simulation round
btc_amount= 6.25                #Amount of BTC the winning
Inibudget= 7500                 #Budget every miner has at the start
NB=10                           #Jackpot rounds = n_BTC that are paid out
simulation_rounds=500             #Number of simulation rounds
tech_acquisition_cost=100     #Cost of each tech level
adder=5                        #tech level increase for optimizer
    optimizer
multi=1                       #Number of iterations

#dependent Variables:
-------------------------------------------------------------------------
```

```
budget=np.full(N, Inibudget)          #Budget
x=np.ones(N)                  #computing technology
jackpot=np.ones(NB)           #winner Vector

#general Functions:
    ------------------------------------------------------------------

def techchoose(x):                                                    #
    random technology
    for i in range(N):
        x[i] = random.randint(1, 20)          #Everyone chooses a
            random technology level between 1-20
    return(x)
def btc_price_change(btc_price):                                      #
    gives an increase in BTC value
    new_btc_price=btc_price*(btc_price_increaser+1)
    #print("new BTC Price:", new_btc_price)
    return(new_btc_price)
def btc_price_changer_for_plot(btc_price):
    btc_price_vector=np.full(simulation_rounds,float(btc_price))
    for i in range(1,simulation_rounds):
        btc_price_vector[i]=btc_price*(1+btc_price_increaser)**i
    #print(btc_price_vector)
    return(btc_price_vector)

#Functions for Calculation:
    ------------------------------------------------------------------

def jackpotFKT(jackpot):        #random int for evaluation of
    winner
    for i in range(NB):
        jackpot[i] = random.randint(1, 100)
    return(jackpot)
def rantime(x):                 #gives a time in which miners solve
    z=np.sum(x)
    if z<=130:
        y=85
    if z<250:
```

```
            y=100
        else:
            y=80
        return(y)

def CostFKT(x, timesteps, winning_miner, budget,btc_price):    #
    costfunction
        cx=budget                                                #
            Budget Vector
        #print("cx", cx)
        for j in range(NB):                                      #NB
            =Jackpot rounds
            for l in range(N):
                cx[l]=cx[l]+(x[l]*timesteps*energy_cost)*(-1)
                            #ChoosenTechnology*100= fixed costs +
                        variable costs -> everyone has to pay
            cx[int(winning_miner[j])]=cx[int(winning_miner[j])]+
                btc_amount*btc_price          #Winner gets the Bitcoin
                & transform to int!
        return(cx)

def CostFKT_FIX_payout(x, timesteps, winning_miner, budget,
    btc_price):   #costfunction
        cx=budget                                                #
            Budget Vector
        m=np.sum(x)+0.0000000000000001                           #
            global Techlevel -> +eps for formal reasons so never x/0
        y=(x/m)                                                  #
            how big is every miners share of the overall tech stock
        print("cx:", cx)              #NB=Jackpot rounds
        for i in range(N):
            y[i]=round(y[i], 2)                                  #
                Rounds the share for readability
            cx[i]=cx[i]+(x[i]*timesteps*energy_cost)*(-1)        #
                ChoosenTechnology*100= fixed costs + variable costs ->
                 everyone has to pay
```

```
            cx[i]=cx[i]+y[i]*btc_amount*btc_price*NB                    #
                Winner gets the Bitcoin & transform to int!
        print("y:", y)
        print("cx:", cx)
        print("
            ------------------------------------------------------------------")
            )
        return(cx)

def acquisition_cost(Tech_Matrix, counter, NewBudget):          #
    acquisition cost, the one from the start and all following
    changes
        if counter <= 1:
            NewBudget=NewBudget-Tech_Matrix[0]*tech_acquisition_cost
                        #In the first round the initial value has to
                be paid
        else:
            NewBudget=NewBudget-abs(Tech_Matrix[counter-1]-
                Tech_Matrix[counter-2])*tech_acquisition_cost        #In
                 all following rounds only the change of the equipment
                 has to be paid
        return(NewBudget)
def winner(x, t, jackpot):                                          #Who
    is winning
        timesteps= np.ceil(K/np.max(t))                             #How
            many timesteps does the fastest miner needs & round them
            to int
        m=np.sum(x)+0.00001                                         #
            global Techlevel -> +eps for formal reasons so never x/0
        y=(x/m)*100                                                 #
            winning percentage Vektor
        z=y                                                         #Vector
             that shows the probability intervalls in [0,100]
        for i in range(N-1):
            z[i+1]=z[i]+z[i+1]                                      #here
                the actual Vector is calculated
```

```
        if z[i+1]>99.99:
            z[i+1]=100                                  #for
                rounding mistakes
    jackpot=jackpotFKT(jackpot)                         #Random
        int to check who won
    xxx=np.zeros(NB)                                    #
        Winning Miners Vekor
    for j in range(NB):
        for i in range(N):                              #in who
            's intervall is the "jackpot" value == who winns
            if z[i] >= jackpot[j]:                      #>=
                for technical reasons
                xxx[j]=i+1
                break
    return(timesteps,m, y, xxx)
def Evaluation(y, jackpot, budget,btc_price):           #Combining
    everything and doing the Evaluation Step
    tx = rantime(y)                                     #Time that
        is needed to find a result
    winner_evaluation=winner(y,tx,jackpot)
    timesteps=winner_evaluation[0]                      #
        Evaluation of the fastest miner in Timesteps
    xxx=winner_evaluation[3]-1                          #winner //
        -1 because python counts from 0
    m=winner_evaluation[1]                              #sum of
        technology used, to calculate "winning probability"
    ########################################
    # insert "CostFKT" or "CostFKT_FIX_payout" down here!
    #-> This is how the changing between probabilistic and fix
        payout is done
    NewBudget=CostFKT_FIX_payout(y,timesteps,xxx, budget,
        btc_price)          #determining new Budget
    return(xxx,timesteps,NewBudget,m)
def kickout(NewBudget, x):                              #checks
    if a miner has used all the butget -> They will be set to 0
    and removed
```

```
    for i in range(N):
        if NewBudget[i] <=0 or x[i] <=0:                    #x[i] is
            for the case that miner starts with <=4 and decreases
            his mining equipment in the first round,
             NewBudget[i]=0                                 #Set
                future Butget to 0
             x[i]=0                                         #Sets
                future tech level to 0 that miner won't play again
    return(NewBudget,x)


#Simulation Functions:
-------------------------------------------------------------------------

def repeater(x,jackpot, budget,btc_price):
                            #recalls the whole process "
    simulation_rounds" times
    print("Doing",simulation_rounds, "Simulation Rounds:")
    print("")
    counter=1
    Tech_Matrix=np.zeros((simulation_rounds+1,N))           #
        Matrix for saving the choosen Tech levels from every round
    Tech_Matrix[0]=x                                        #
        Initializes with starting technology
    Budget_Matrix=np.zeros((simulation_rounds+1,N))         #
        Matrix for saving the development of the Budget of every
        round
    Budget_Matrix[0]=budget                                 #
        Initializes with starting Budget
    while counter <= simulation_rounds:
        #print("Simulation Round:", counter)
        Eval=Evaluation(x,jackpot, budget,btc_price)
        xxx=Eval[0]
        timesteps=Eval[1]
        NewBudget=Eval[2]
        m=Eval[3]                                           #sum
            of technology used, to calculate "winning probability"
```

```
            NewBudget=acquisition_cost(Tech_Matrix, counter,
                NewBudget)
            kick=kickout(NewBudget, x)
            budget=kick[0]
            x=kick[1]
            Tech_Matrix[counter]=x                          #Saves
                the choosen Tech levels from every round
            Budget_Matrix[counter]=NewBudget                #Saves
                the development of the Budget of every round
            ######################################
            #Opt -Fkt below!                  # -----!!!! Input
                Optimzer here !!!!------
            # This is how the different optimizers are used by
                changing the #Number below
            x=optimizer_1(x,counter, Budget_Matrix, Tech_Matrix,
                timesteps,m,btc_price)
            Tech_Matrix[counter]=x
            btc_price=btc_price_change(btc_price)
            counter=counter+1
        if multi==1:
            describer(Tech_Matrix,Budget_Matrix)            #Prints
                the results in words
        return(x,jackpot, budget, Tech_Matrix, Budget_Matrix)
    def plotter(budget, tech, btc_price_plot):      #plots the result
        if btc_price_increaser == 0:                #garantees that
           BTC price plot is only shown, if it changes
            plt.subplot(2, 1, 1)                    #2-Row,1-Column
                ,1-Figure
            plt.plot(budget)
            plt.ylabel('Budget')
            plt.subplot(2, 1, 2)
            plt.plot(tech)
            plt.ylabel('Techlevel')
            plt.xlabel('Simulation_Rounds')
            plt.show()
        else:
```

```
        plt.subplot(3, 1, 1)                    #3-Row,1-Column
            ,1-Figure
        plt.plot(budget)
        plt.ylabel('Budget')
        plt.subplot(3, 1, 2)
        plt.plot(tech)
        plt.ylabel('Techlevel')
        plt.subplot(3, 1, 3)
        plt.plot(btc_price_plot)
        plt.ylabel('BTC Price')
        plt.xlabel('Simulation Rounds')
        plt.show()
def describer(Tech_Matrix,Budget_Matrix):
                            #Prints results in words
    print("
        ----------------------------------------------------------------------")
        )
    for i in range(1,simulation_rounds):
                                #Prints the time when a single
        miner is kicked out
            temp_counter=0
            for j in range(N):
                if Tech_Matrix[i,j]==0 and Tech_Matrix[i-1,j] >
                    0:
                    temp_counter=temp_counter+1
            if temp_counter > 0:
                print("In the ", i, "round", temp_counter, "
                    miners have gone bankrupt")
    for i in range(N):
        #Shows a readable output of the winning miners
        if Budget_Matrix[simulation_rounds,i] >0:
            print("Miner", i, "winns, with an Budget increase of"
                , Budget_Matrix[simulation_rounds,i]-Inibudget, "
                and a tech level of", Tech_Matrix[
                simulation_rounds,i],". Start tech level: ",
                Tech_Matrix[0,i])
```

```
    print("␣")
    return()

#Evaluator Functions:
    ------------------------------------------------------------------

def evaluatorREPEATER(Budget_Matrix, Tech_Matrix):        #
    determining "big results", who survived, with what, average
    Knockoutime etc
    counter=0
    counter2=0
    matrix=np.zeros(N)
    for i in range(N):
        if Tech_Matrix[simulation_rounds,i]==0:
            matrix[i]=i
        else:
            matrix[i]=999
            counter=counter+1
    winner=np.zeros([counter,4])
    for i in range(N):
        if matrix[i]==999:
            winner[counter2,0]=i
            winner[counter2,1]=Tech_Matrix[0,i]
            winner[counter2,2]=Tech_Matrix[simulation_rounds,i]
            winner[counter2,3]=int(Budget_Matrix[
                simulation_rounds,i])
            counter2=counter2+1
    average= np.sum(winner, axis=0)/counter+1
    averageEndTechlvl=average[2]
    averageEndBudget=average[3]
    print("There␣have␣been␣",counter,"winners")
    print("averageEndTechlvl",averageEndTechlvl)
    print("averageEndBudget",averageEndBudget)
    knockout=np.zeros(N)
    for i in range(N):
        for j in range(simulation_rounds):
            if Tech_Matrix[j,i]>0:
```

```
                knockout[i]=j+1                    #for readability
    averageKnockoutTime=sum(knockout)/N
    print("Avergage knockout time: ", averageKnockoutTime)
    return(winner, averageKnockoutTime,counter,averageEndTechlvl,
        averageEndBudget)
def meancalculator(survivors,averageEndTechlvl,averageEndBudget):
    #calculates the average means for evaluatorALL and puts them
    in a suitable vector
    maxSurvivors=int(max(survivors))
    meanTech=np.zeros(maxSurvivors)
    meanBudget=np.zeros(maxSurvivors)
    for j in range(maxSurvivors):                  #to get all
        values to calculate mean
        for i in range(multi):
            if i == survivors[j]:
                meanTech[j]=meanTech[j]+averageEndTechlvl[i]
                meanBudget[j]=meanBudget[j]+averageEndBudget[i]

    print("Sum Tech", meanTech)
    print("Sum Budget",meanBudget)

    count=np.zeros(maxSurvivors)
    for j in range(maxSurvivors):                  #to get how
        often the values are in the vector
        count[j] = int(np.count_nonzero(survivors == j+1))
        print("count",j+1,":",count[j])
    for j in range(maxSurvivors):                  #to calculate
        the means
        meanTech[j]=int(meanTech[j]/count[j])
        meanBudget[j]=int(meanBudget[j]/count[j])
    print("meanTech", meanTech)
    print("meanBudget",meanBudget)

    for j in range(maxSurvivors-1):                #to reshape the
        mean vector
        count[j+1]=count[j]+count[j+1]
```

```
    print("count",count)

    meanVektorTech=np.full(multi, meanTech[0])
    meanVektorBudget=np.full(multi, meanBudget[0])
    for j in range(maxSurvivors):
        for i in range(multi):
            if i>=count[j-1]:
                meanVektorTech[i]=meanTech[j]
                meanVektorBudget[i]=meanBudget[j]
    print("meanVektorTech", meanVektorTech)
    print("meanVektorBudget", meanVektorBudget)
    sorted_survivors=np.sort(survivors)
    print("sorted_survivors",sorted_survivors)
    print("maxSurvivors",maxSurvivors)
    return(meanVektorTech,meanVektorBudget,sorted_survivors)
def evaluatorALL(x,jackpot, budget,btc_price):                    #here
    comes the summary of all evaluations!
    if multi<1:
        print("Not funny!!")                                      #if someone
            inputs an unfeasible variable
    if multi==1:
        x=techchoose(x)                                           #
            Needed for the first round, as initial random values
        SIMULATION=repeater(x,jackpot, budget,btc_price)
        evaluationResult=evaluatorREPEATER(SIMULATION[4],
            SIMULATION[3])
        PLOT=plotter(SIMULATION[4], SIMULATION[3],btc_price_plot)
                #to create graphical output
    if multi>1:
        test=np.zeros([multi,N,4])
        averageKnockoutTime=np.zeros(multi)
        survivors=np.zeros(multi)
        averageEndTechlvl=np.zeros(multi)
        averageEndBudget=np.zeros(multi)
        sumEndBudget=np.zeros(multi)
        for i in range(multi):
```

```
        print("␣")
        print("Round:␣", i+1, "
            ----------------------------------------------------------------------
            ")
        x=techchoose(x)                                                    #
            Needed for the first round, as initial random
            values
        SIMULATION=repeater(x,jackpot, budget,btc_price)
        round_i=evaluatorREPEATER(SIMULATION[4], SIMULATION
            [3])
        #test[i]=round_i[0]
        averageKnockoutTime[i]=round_i[1]
        survivors[i]=int(round_i[2])
        averageEndTechlvl[i]=int(round_i[3])
        averageEndBudget[i]=round_i[4]
        sumEndBudget[i]=averageEndBudget[i]*survivors[i]
print("␣")
print("
    ----------------------------------------------------------------------
    ")
print("Fazit:")
print("
    ----------------------------------------------------------------------
    ")
print("survivors", survivors)
print("averageKnockoutTime",averageKnockoutTime)

plt.subplot(3, 1, 1)                                    #2-Row,1-Column
    ,1-Figure
plt.scatter(survivors,averageEndTechlvl, c="blue")
plt.ylabel('average␣techlevel')
plt.suptitle('End␣value␣vs.␣number␣of␣miners', fontsize
    =16)
plt.subplot(3, 1, 2)
plt.scatter(survivors, averageEndBudget, c="blue")
plt.ylabel('average␣budget')
```

```
        plt.subplot(3, 1, 3)
        plt.scatter(survivors,sumEndBudget, c="green")
        plt.ylabel('sum_budget')
        plt.xlabel('number_of_winners')
        plt.show()

#Optimizer Functions:
    ------------------------------------------------------------------

def optimizer_1(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m,btc_price):                    #1 simple optimzer
    for i in range(N):
        if Budget_Matrix[counter,i] > 0:
            x[i]=x[i]+1                                    #if
                budget is positive, by 5 new techlevel
    return(x)
def optimizer_2(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m,btc_price):                    #2 simple optimzer
    for i in range(N):
        if Budget_Matrix[counter,i] > Budget_Matrix[counter-1,i]:
            x[i]=x[i]+adder                                #if
                budget has increased, increase techlevel by adder
    return(x)
def optimizer_3(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m,btc_price):                    #RANDOM optimizer
    z=np.zeros(N)
    for i in range(N):
        z[i]=random.randint(0,2)
        if x[i] > 0:
            if z[i] == 1:
                x[i]=x[i]+adder
            if z[i] == 2:
                x[i]=x[i]-adder
                if x[i]<=0:
                    x[i]=0
    return(x)
def optimizer_4(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
```

```
    m, btc_price):                    #1 real optimizer
      for i in range(N):
          if Budget_Matrix[counter-1,i] > 0:
              if (x[i]*(btc_amount*btc_price)/m)-((Tech_Matrix[
                  counter-1,i]-Tech_Matrix[counter-2,i])*
                  tech_acquisition_cost+timesteps*energy_cost*x[i])
                  >0:
                   x[i]=x[i]+adder
      return(x)
def optimizer_5(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m, btc_price):                    #2 real optimizer
      for i in range(N):
          if Budget_Matrix[counter-1,i] > 0:
              if (x[i]*(btc_amount*btc_price)/m)-((Tech_Matrix[
                  counter-1,i]-Tech_Matrix[counter-2,i])*
                  tech_acquisition_cost+timesteps*energy_cost*x[i])
                  >0:
                   x[i]=x[i]+adder
              else:
                   if Tech_Matrix[counter-1,i] >= 0:
                       x[i]=x[i]-adder//2
                       print(np.ceil(adder/2))
      return(x)
def optimizer_6(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m, btc_price):                    #3 real optimizer
      for i in range(N):
          if Budget_Matrix[counter-1,i] > 0 and Tech_Matrix[counter
              ,i] > 0:
               if (x[i]*(btc_amount*btc_price)/m)-((Tech_Matrix[
                  counter-1,i]-Tech_Matrix[counter-2,i])*
                  tech_acquisition_cost+timesteps*energy_cost*x[i])
                  >0:
                   x[i]=x[i]+adder
               if Budget_Matrix[counter-3,i] >= Budget_Matrix[
                  counter-1,i]:
                   x[i]=x[i]-adder
```

```
    return(x)
def optimizer_7(x,counter, Budget_Matrix, Tech_Matrix, timesteps,
    m,btc_price):                    #3 simple/advanced
    for i in range(N):
        if Budget_Matrix[counter,i] > 0 and Budget_Matrix[counter
            ,i] > Budget_Matrix[counter-1,i]:
            x[i]=x[i]+adder                                    #if
                budget has increased, increase techlevel by adder
    return(x)

#Output:
    ------------------------------------------------------------


btc_price_plot=btc_price_changer_for_plot(btc_price)      #This is
    needed when the BTC price changes over time

evaluatorALL(x,jackpot, budget,btc_price)

inp = input("ENDE")
```

# Bibliography


[1] BBC. China declares all crypto-currency transactions illegal. https://www.bbc.com/news/technology-58678907. Last checked: 2022-10-11.

[2] blockchain.com. Current hashrate. https://www.blockchain.com/charts/hash-rate. Last checked: 2022-10-16.

[3] BTC.com. Mining pool distribution. https://btc.com/stats/pool?pool_mode=year. Last checked: 2022-03-24.

[4] N. Evans and Y. Pritzker. *Inventing Bitcoin: The Technology Behind the First Truly Scarce and Decentralized Money Explained*. Amazon Digital Services LLC - KDP Print US, 2019.

[5] Kraken.com. Current btc price. https://www.kraken.com/prices/btc-bitcoin-price-chart/eur-euro?interval=1m. Last checked: 2022-03-24.

[6] J. Ma, J. S. Gans, and R. Tourky. Market structure in bitcoin mining. https://www.nber.org/papers/w24242, 2018. Last checked: 2022-02-19.

[7] H. McCook. A comparison of bitcoin's environmental impact with that of gold and banking. https://www.nasdaq.com/articles/a-comparison-of-bitcoins-environmental-impact-with-that-of-gold-and-banking-2021-05-04. Last checked: 2022-05-04.

[8] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/en/bitcoin-paper, 2008.

[9] Statistic Austria GmbH. Power consumption of Austria 2020. https://de.statista.com/statistik/daten/studie/325788/umfrage/stromverbrauch-in-oesterreich. Last checked: 2022-02-19.

[10] University of Cambridge. Current power consumption of bitcoin network. https://ccaf.io/cbeci/index. Last checked: 2022-02-19.