Agent-Based Computational Modeling of Cryptocurrency Design

Model Documentation

Student: Ude, Felix
Advisor: Chen, Shu-Heng

June 29, 2019

Contents

List of Tables

In order to get more insight into the design of a cryptocurrency and its calibration, an agent-based model is created. The goal of the model is to find a better way to design a cryptocurrency, that allows for a high security and high wealth simultaneously in a pareto optimal fashion. In the first, step a model was created that tries to simulate the real cryptocurrency Bitcoin. After the model has been created, it will get calibrated and optimized which is described in section 4. The model is based on an existing one by Cocco & Marchesi (2016), which was then adapted and extended. The core features of the original model include a Bitcoin market, where agents can sell and buy Bitcoin in a realistic fashion. Furthermore, agents are able to create (i.e. mine) new Bitcoins. The full list of the features of the original model is:

- Three kinds of agents, including chartists, random traders/users and miners;
- A realistic order book to imitate a real cryptocurrency market;
- Agents join over time and decide to engage in the trade and or to mine Bitcoins;
- A power law wealth distribution for existing and later added agents;
- Miners are all in mining pools, meaning that they have a steady stream of income. They also can decide at given time points to invest in or divest their hardware.
- A transaction system, allowing users to send a transaction and attaching a transaction fee. It also has a capacity limit. All transaction fees of the transactions within this limit will be distributed among the miners.

The inclusion of the transaction system allows to analyze the development of the transaction fees as well as a better representation of the mining network hashing power. The model simulates each day between February 23rd in 2014 and April 3rd 2018 in a single time step, totaling 1500 simulation steps.

1 Types of Agents

The model used in this thesis includes three types of agents: Chartists, users and miners. These will be explained in the following sections in more detail.

1.1 Chartist

A chartist is an agent who trades for profit. S/he will place a buy order when s/he speculates that prices will rise and a sell order otherwise. Every chartists has a time window T, that has a mean of 20 and a standard deviation of 1. This approach has been firstly introduced by Arifovic (2002). If the price increases in T was over the threshold of 0.01, s/he will place a buy order and a sell order if otherwise. 10% of the chartists follow a contrary strategy, meaning

the chartist will sell when the price increase is over 0.01 and buy if it is not the case. Every chartists entering the market always places a buy order. This approach can also be found in Raberto et al. (2003).

1.2 User

A user is an agent who participates in the market either for diversification of his portfolio and/or for the usage of Bitcoin as a form of currency. S/he does not follow any trends with his/her orders. In fact, the probability for sell and buy order is the same for a user at any given time. Every user who is entering the market, issues a buy order. Since one of the reasons for his/her participation in the market is the usage of Bitcoin as a currency, s/he also has the ability to make a transaction at any given time step. Since the capacity of transactions per time step is limited, s/he will always add a fee that allows the transaction to be picked up by miners. More on transaction in section 2.2.

1.3 Miner

Miners are agents that participate in the model by generating new Bitcoins with their mining hardware. They only participate in the market, if they require more cash in order to invest or pay their electricity bills. At the beginning of the simulation, the total hash power of the total Bitcoin network was 28,314 giga hash/s (GH/s). (Blockchain.com, 2019) These numbers, as well as other following variables, have been divided by the factor of 5000. This was done as to adjust all the model, so that it could be run within computational limits. This hash rate was divided by the expected initial amount of miners (i.e. 9) to calculate the initial hashing capability of one miner, resulting in a hashing capability of 0.00041 GH/s per Miner. The hashing power of available hardware is getting cheaper over time, as the technology continues to improve, whereas the power consumption per hashing capability is decreasing over time. Cocco & Marchesi (2016) collected the average hashing power for the existing hardware with corresponding prices and electricity consumption in the time frame between 2011 and 2015. They estimated two curves through their aggregated data with the following forms:

$$R(t) = 8.635 \cdot 10^4 \cdot e^{0.006318 \cdot t}$$

where $R(t)$ is the average hash rate per US-Dollar with $\frac{H}{s \cdot \$}$.

$$P(t) = 4.649 \cdot 10^{-7} \cdot e^{-0.004055 \cdot t}$$

where $P(t)$ is the average electricity consumption per hash with $\frac{W}{H/s}$.
$P(0)$ times the initial 0.00041 GH/s per Miner gives the initial electricity consumption per miner with 1.10586 W. New miners entering the market as well as existing miners, who are able to upgrade their hardware use these two equations to determine their new purchased hashing

power $r_{i,u}(t)$ and its corresponding electricity consumption $e_{i,u}(t)$. The total hashing power of a miner is consequently the sum of all his/her purchases or

$$r_i(t) = \sum_{s=t_i^E}^{t} r_{i,u}(t)$$

where t_i^E is the entry time of a given miner. The corresponding electricity consumption can be expressed as:

$$e_i(t) = \sum_{s=t_i^E}^{t} \epsilon \cdot P(s) \cdot r_{i,u}(s) \cdot 24$$

where ϵ is the electricity price per W/h. It is fixed to 0.07 US$, which was the average during the model time frame in China, where the majority of miners today are located. (CEIC, 2019) The 24 refers to the hours of the day, since one time step in the model is a whole day. Miners who are already in the model and make the decision to invest, use a fraction $\gamma_{1,i}(t)$ of their fiat cash. Additionally, they will sell a fraction $\gamma_i(t)$ of their Bitcoin and use the revenue as well. Hence, the new purchased hardware will have a hashing power of:

$$r_{i,u}(t > t_i^E) = [\gamma_{1,i}(t)c_i(t) + \gamma_i(t)b_i(t)p(t)]R(t)$$

where c_i is the fiat cash and b_i the Bitcoins held by Miner i and $p(t)$ is the current Bitcoin price. Miners entering the market will always make a purchase of mining hardware, this time only using the fraction $\gamma_{1,i}(t)$ of their fiat cash. Hence, the hashing power of the initial bought hardware for Miners is equal to:

$$r_{i,u}(t = t_i^E > 0) = \gamma_{1,i}(t)c_i(t)R(t)$$

The values for $\gamma_{1,i}(t)$ and $\gamma_i(t)$ are the same as in Cocco & Marchesi (2016) and both have a log normal distribution. $\gamma_{1,i}(t)$ has a mean of 0.15 and a standard deviation of 0.15, whereas $\gamma_i(t)$ has a mean of 0.175 and a standard deviation of 0.15. Since Miners are not allowed to take a credit, their γ are always set to 1 in case that $\gamma > 1$.

Miners are not able to purchase at any given time. Instead, they make a decision whether to invest or divest on average every 60 days, which is another value taken from Cocco & Marchesi (2016). This average has a normal distribution and a standard deviation of 6 days and the next decision time step is determined by each individual miner every time an investment decision has been made by him/her. Since by assumption all miners belong to a mining pool, their generated revenue is proportional to their hashing power in comparison to the total networks hashing power. The total daily Bitcoin income is the sum of the newly mined Bitcoins per day plus the total sum of transaction fees within the transaction limit submitted at that time step,

which can be expressed as:

$$b_{Tot}(t) = B(t) + \sum_{j=1}^{tlimit} f_j(t)$$

where $B(t)$ is the amount of Bitcoins mined per day and $\sum_1^{tlimit} f_{j=1}(t)$ is the total sum of the highest fees within the transaction limit $tlimit$.
This revenue is now distributed among the miners in proportion to their hashing power.

$$b_i(t) = \frac{r_i(t)}{r_{Tot}(t)} b_{Tot}(t)$$

where $r_{Tot}(t)$ is the networks total hashing power.
At any decision time step, every miner decides whether the revenue of the new hardware outweighs the connected costs. This constraint is expressed as follows:

$$e_{i,u}(t) < \frac{r_{i,u}(t)}{r_{Tot}(t)} b_{Tot}(t) p(t)$$

where $r_{i,u}(t)$ is the hashing power and $e_{i,u}(t)$ the electricity consumption of hardware u of miner i. If s/he decided against the investment, s/he will issue a sell order equal to $\frac{\gamma_i(t)}{2} b_i(t)$ to cover his/her electricity expenses.

Additionally, every miner is able to divest his/her hardware for the price of $R(t)$. S/he will do so, if the electricity expenses are 20% larger than the revenue associated with that specific hardware. The constraint for divest is fulfilled if:

$$1.2 e_{i,u}(t) \geq \frac{r_{i,u}(t)}{r_{Tot}(t)} b_{Tot}(t) p(t)$$

2 The Model

The model used in this thesis has two main components:

- A Bitcoin market which is similar to a real life cryptocurrency exchange, where traders can place buy and sell orders.
- A transaction system, which allows users to send Bitcoins to other users and attach a transaction fee to it.

The following sections will explain these components in greater detail.

2.1 The Bitcoin Market

The first main component of the model is the Bitcoin market. It is modeled after real life Bitcoins exchanges or other trading exchanges in general. More specifically, an order book is

created. This approach has also been used previously in Raberto et al. (2005) and Cocco & Marchesi (2016).

Order Book

The order book keeps record of all the buy and sell orders with their respective amount in Bitcoin, the remainder of the transaction for partial transactions, the corresponding limit prices and the expiration time of an order. The following will explain these terms in more detail.

Buy Order: The amount of a buy order is proportional to the available fiat cash $c_i(t)$ of a trader i. That excludes any fiat cash that is currently used in pending orders. The amount of a buy order is defined as:

$$b_\alpha = c_i(t) \cdot \beta \tag{1}$$

where β is a variable pulled from a log normal distribution with average 0.25 and standard deviation of 0.2. In the case that $\beta > 1$, β is set to 1, since a trader is not able to take any credit.

Sell Order: Similar to equation 1 is the amount for a sell order determined.

$$s_\alpha = b_i(t) \cdot \beta \tag{2}$$

β is the same as in equation 1. Short selling is also not possible.

Limit Prices: The limit price is a price at which a trader wishes to perform his/her order. This mechanism works similar to a limit price from a real world exchange platform. A limit price can also have a value of zero, in which case it is regarded as a market price, meaning that a trader wishes to perform his/her order with the best available price. Each agent category has their own probability to issue an order with a market price. 0.2 for users, 0.7 for chartists and 1 for miner. These numbers are chosen and picked by Cocco & Marchesi (2016), who argue, that chartists and miner are the types of agent who have the greatest desire to trade at the best price, chartists for profit and miners for liquidity reasons.

An agent is willing to perform his/her buy order only if the sell limit is below or equal to his/her buy limit. The limit price for a buy order is given as:

$$buy_i(t) = p(t) \cdot N_i(\mu, \sigma_i) \tag{3}$$

Similarly, an agent is only willing to perform his/her sell order if the buy limit is greater or equal to his/her sell limit. The limit price for a sell order is given as:

$$sell_i(t) = p(t)/N_i(\mu, \sigma_i) \tag{4}$$

For both equations 3 and 4, $p(t)$ is the current Bitcoin price and $N_i(\mu, \sigma_i)$ is a Gaussian distribution with a mean of 1.05. This distribution with this mean is supposed to represent limit

prices, that are not fully rational. Additionally, agents are on average willing to pay a small amount more than the market price would suggest. σ_i is supposed to represent the volatility of the model. This approach has been used in Raberto et al. (2001). The standard deviation of the Gaussian distribution is given by:

$$\sigma_i = K\sigma(T_i) \tag{5}$$

where K is a constant set to 2.5 and $\sigma(T_i)$ is the standard deviation of the price in the time window T which is set to 20. Furthermore, has σ_i the following constraint:

$$\sigma_{min} = 0.003 \leq \sigma_i \leq 0.01 = \sigma_{max} \tag{6}$$

Expiration Time: Every order has an associated expiration time. It has the following form:

$$Exp_i = t + pat_i \tag{7}$$

where t is the simulation step of the order and pat_i is the patience of the ordering agent. It is 0 for chartists, meaning their order will expire if not performed in that time step. This is the case since chartists wish to follow market trends. Users have a log normal distribution from which they retrieve their patience for each time they issue an order. It has a mean of 3 and a standard deviation of 1. Miners who always issue market price orders, have a patience of infinite, meaning their order will stay in the market until it is executed.

Price Clearing

The price clearing mechanism used in this model is similar to the one from Raberto et al. (2005).

In every simulation step, after all agents were able to issue their order, the order book is sorted in respect to the limit prices. For buy order, the list is sorted in descending order, and for sell order in ascending order. The model now compares the two top orders from the buy and the sell order list. A match occurs if at least one of the following condition is fulfilled:

$$buy_i(t) \geq sell_j(t) \tag{8}$$

$$sell_j(t) = 0$$

$$buy_i(t) = 0$$

where $buy_i(t)$ is the limit price of the buy order and $sell_j(t)$ is the limit price of the sell order. Paraphrased, this means that either the buy limit price needs to be greater or equal to the sell limit price and/or at least one of the order is a market price order.

In case that the amount of Bitcoins in the buy and sell order does not match, the order with

the smaller amount is fully executed. The fully executed order will be removed from the order book and the non-fully executed one will remain with the remainder of that transaction. This process will be repeated until the top two orders of the order book do not match anymore. After the last match has occurred, all expired orders will be removed.
Price Formation: the price for a matched transaction p_T is formed by the following four rules:
1. If $buy_i(t) > 0$ and $sell_j(t) > 0$:

$$p_T = \frac{buy_i(t) + sell_j(t)}{2}$$

2. If $buy_i(t) = 0$ and $sell_j(t) > 0$:

$$p_T = max(sell_j(t), p(t))$$

3. If $buy_i(t) > 0$ and $sell_j(t) = 0$:

$$p_T = min(buy_i(t), p(t))$$

4. If $buy_i(t) = 0$ and $sell_j(t) = 0$:

$$p_T = p(t)$$

where $p(t)$ is the current Bitcoin price. Every time p_T has been determined, it is henceforth used as the new $p(t)$.

2.2 The Transaction System

The second main component of the model is the transaction system. It aims to replicate the occurrence of transactions fees in the real Bitcoin network.
The transaction system in this model is essentially a list of all transactions that have been broadcasted by the user agents. It is always sorted in descending order by the fees f_i. As the real Bitcoin protocol, the model has a transaction limit called $tlimit$. Only transactions which are within $tlimit$ are able to be executed in that simulation time step.
In every simulation step, a user agent has a fixed probability of 20% to make a transfer. All other transactions are considered to happen over a Bitcoin exchange which therefore are not appended to the transaction system of the Bitcoin network. The amount a user sends within a single transaction is determined by a normal distribution with an average of 143 $ and a standard deviation of 20. The value of this mean was taken from the median of the transaction volume of all transaction in the modeled time frame. (BitInfoCharts, 2019) Even though all

the transaction fees are expressed in US-Dollar, the model internally uses the corresponding amount of Bitcoin. The recipient of the transaction is randomly chosen from the pool of users.
Fee determination: A user will attach a fee to his transaction based on the length of the transaction list $L(t)$ and the following rules:

$$\text{1. If } L(t) \geq tlimit: \qquad f_i = (f_{last}(t) + \alpha) \cdot N(\mu, \sigma) \tag{9}$$

$$\text{2. If } L(t) < tlimit: \qquad f_i = (f_{last}(t) - \alpha) \cdot N(\mu, \sigma) \tag{10}$$

$$\text{3. If } L(t) = 0: \qquad f_i = (f_{last}(t-1) - \alpha) \cdot N(\mu, \sigma) \tag{11}$$

where $N(\mu, \sigma)$ is a normal distribution with the mean of 1 and standard deviation of 0.01. $f_{last}(t)$ is the fee of the last transaction of this time step still within $tlimit$, whereas $f_{last}(t-1)$ was the last fee within $tlimit$ of the last time step. α is a constant set to 0.01$. As mentioned, all transactions have their fees attached in Bitcoin, however a transaction also holds the information of a fee expressed in US-Dollar, which is actually the information used for $f_{last}(t)$.
As by Bitcoin protocol design, a fee is always required. The hard coded absolute minimum fee for a transaction is set to $1 \cdot 10^{-8}$ Bitcoin. Therefore:

$$\text{For } \frac{f_i}{p(t)} < 10^{-8}: \qquad f_i = \frac{p(t)}{10^8} \tag{12}$$

As with orders, transaction have an expiration date, which is set to:

$$Exp_i = t + pat_i \tag{13}$$

where t is the current simulation step and pat_i is a log normal distribution with a mean of 1 and standard deviation of 0.01.
At the end of every time step, all transactions within $tlimit$ are executed. The recipient receives his/her designated amount. The total sum of all fees within $tlimit$ will be distributed among all miners in proportion to their hashing power. All executed and expired transactions will be removed, all other transactions remain in the list, still in sorted order for the next time step.

3 Initialization

3.1 Agent Distribution and Type

The amount of agents entering the model at any simulation time step is determined by a generic exponential function, which was taken from Cocco et al. (2017). The authors estimated the

amount of participants in the Bitcoin network based on observations over time. The amount was slightly adapted to fit with the time frame used in this model and has the following form:

$$N(t) = 1.77 \cdot 10^4 \cdot exp(0.002465 \cdot (t + 1878)) \tag{14}$$

where $N(t)$ is the total amount of agents at simulation step t.

There is little data available for the distribution of the kind of participants in the Bitcoin network. As for 2019, there were nearly 35 million Bitcoin wallet users. (Statista, 2019) It is estimated that the current amount of miners in Bitcoin is around one million. (Buybitcoinworldwide, 2019) Therefore, the probability for an agent at any time step to be a miner $p_M(t)$ is set to 0.025.
Cocco & Marchesi (2016) stipulated that of all the agents which are not miners, 70% are users (in their paper generally referred to as a random trader) and 30% are chartists. Therefore, there are the following percentages for the three different agent types:

Table 1: **Agent Type Probability**

Agent Type	**Probability**
Miner	2.5%
User	68.25%
Chartist	29.25%

3.2 Agent's Wealth

The wealth distribution of the agents is following a power law distribution. This approach has also been used in Cocco & Marchesi (2016) and Raberto et al. (2003). A power law has the general form of:

$$p(x) = \frac{C}{x^\alpha} \qquad \text{(Newman, 2005)}$$

Rewritten for the purpose of this thesis, it has the following form:

$$b_i(t_i^E = 0) = \frac{b_{max}}{i^\alpha} \tag{15}$$

where $b_i(t_i^E = 0)$ are the initial Bitcoins of initial agent i, b_{max} the Bitcoins of the wealthiest agent and α is a constant for the power law. The total amount of Bitcoins for $t = 0$ is equal to:

$$B(0) = \sum_{i=0}^{n_0} \frac{b_{max}}{i^\alpha} = \sum_{i=0}^{n_0} b_i(t_i^E = 0) \tag{16}$$

where n_0 is the initial amount of agents.

Equation 15 is used several times. Once for the Bitcoins of the agents at the beginning of the simulation. Here, α is set to 1.
b_{max} is calculated by solving equation 16 for b_{max} with $B(0)$ set to 60% of the actual amount of the day at the beginning of the simulation, since like in Cocco & Marchesi (2016), it is assumed that only 60% of the Bitcoins are in circulation. n_0 was retrieved from equation 14. This gives a b_{max} of 1,156,094 Bitcoins, which is then divided by the computational factor of 5000.
The second time it was used to calculate the cash for the initial agents with the same alpha and a maximum cash of 20,587. The last time the equation was used, was for the cash of all the agents which are entering at a later point. For that purpose α was set to 0.6 and the cash of the richest trader was set 10 times higher as the cash from the richest initial trader. Their initial Bitcoins are zero. For the agents who entered later on, a pool of cash values was generated with the size of the total amount of agents entering during the whole simulation process. A value was then pulled each time a new agent entered. This approach and the parameters used are all similar to Cocco & Marchesi (2016).

4 Calibration

This thesis aims to analyze optimal design possibilities of a cryptocurrency. For that reason, the two parameter, the transaction limit $tlimit$ and the block mining reward or generation $B(0)$, were picked. These two variables were chosen, because both of these are part of the original protocol of Bitcoin and were never really justified by their creator Satoshi Nakamoto. Additionally, these two variables can be adjusted comparatively easily, by a new consensus of all Bitcoin users, implementing a hard fork. The following sections will describe the three different calibration approaches of this thesis.

4.1 Realistic Parameters

The first calibration process involved using the real Bitcoin parameters to have a baseline value to compare the other calibration processes against. As described in section **??**, the current Bitcoin block has a size limit of 1 MB. Since an average transaction has a size of 250 bytes, this results in a block transaction size limit of 4000. As the model simulates on a daily basis, this number is multiplied by 144 (6 blocks per hour and 24 hours a day). Afterwards, it is divided by the computational factor of 5000. Therefore, $tlimit$ is set to 115 for this calibration. The amount of Bitcoins mined per block was already described in section **??**. As described, the block mining reward halves every 210,000 blocks, which equals approximately 4 years. Therefore, the mining reward does not stay the same during the whole simulation. Table 2 shows the values used for the mining reward during the simulation runs.

Table 2: **Daily Mining Reward in Simulation**

Date	Real Bitcoin Reward	Simulated Bitcoin Reward
2014.02.23 - 2014.11.11	3600	0.72
2014.11.12 - 2018.04.03	1800	0.36

4.2 Optimization for Wealth

The second calibration step is the optimization for wealth in the model. This optimization objective is similar to the one from Chiu & Koeppl (2017). In their paper, Chiu & Koeppl used a theoretical model to optimize the welfare of a market which uses Bitcoin as a currency. They defined welfare as the aggregated surplus of the market minus the expenditures, i.e. the mining costs. The aim of this optimization process is now to verify their findings, which are that the optimal calibration of a blockchain to maximize the welfare is a low mining block reward and no transaction fees. This thesis uses a similar welfare function, which is equal to the overall wealth and it is defined as:

$$W_{tot}(t) = \sum_{i=1}^{n} b_i(t) \cdot p(t) + c_i(t) + \frac{r_{tot}(t)}{R(t)} \tag{17}$$

where $b_i(t)$ are agent i's Bitcoins, $c_i(t)$ his fiat cash, $p(t)$ the Bitcoin price at time t and $\frac{r_{tot}(t)}{R(t)}$ the current worth of the miners' hardware in US-Dollar.

Since the relation between the parameters and the wealth is not known and might not be linear, a brute force approach would be very computation-intensive and therefore unfeasible. Instead, a multi-objective evolutionary algorithm (MOEA) is used to maximize the wealth. This approach has been used for multi-objective optimization of agent-based model before by Narzisi et al. (2006) and is still used in this case here with only one objective due to its very efficient nature.

The genetic algorithm used in this thesis is called Non-dominated Sorting Genetic Algorithm II (NSGA-II) and will be briefly explained in the following:

Generally NSGA-II, just like any other MOEA minimizes f with m objectives and the following form:

$$\textbf{Minimize: } f(s) = (f_1(s), ...f_m(s)) \tag{18}$$

$$\textbf{Subject to: } s_l < s < s_u$$

where s is a vector of optimization variables with the lower and upper limit of s_l and s_u. (Seah et al., 2012) A solution s_i dominates solution s_j if the following conditions are fulfilled: 1. s_i is better or equal in all objectives as s_j. 2. s_i is better than s_j in at least one objective. Thus, a solution that is not dominated by any other solution is called non-dominated.

Just like most other genetic algorithm, NSGA-II starts by picking a random set of parameters

(their genetic code here) and uses the outcomes for the creation of a new generation, that is a new set of parameters. The new offspring is created using mutation and crossover including a tournament selection to determine which parameters to use for the crossing. NSGA-II has the following special features: First off all, it follows an elitist principle, i.e. some of the best performing populations will be used in the following generation. Secondly, for the fitness evaluation, all results are sorted into classes of pareto-fronts, that is by how many other populations they are being dominated. For populations of the same front, a crowding distance is used which tries to diversify the offspring and prefers results that are further away from other solutions (in terms of their euclidean distance of their parameters to that of other). (Calle, 2017)

To make the optimization process feasible, some compromises have to be made. First of all, the NSGA-II have been restricted to 1000 simulation steps. Additionally, a number of runs for the Monte Carlo simulation has been picked, that allows for a robust mean, but could decrease the overall computational time. This approach has also been used in Narzisi et al. (2006). Table 3 shows the average for both the wealth and the overall hashing power, as the latter will be used in the last calibration process. 25 runs were used for the NSGA-II, as the average wealth and hashing power appeared to sufficiently stabilize at that amount of Monte Carlo runs.

Table 3: **Model's stochasticity per amount of Monte Carlo Runs**

No. of Monte Carlo Runs	**Average Wealth**	**Average Hashing Power**
1	14794	13,277,608
5	13476	11,885,676
25	12672	11,073,409
50	12904	11,168,782
100	12795	11,047,399

The meta-parameters for the NSGA-II in this calibration are listed in table 4.

Table 4: **Meta-Parameters for NSGA-II of Wealth Optimization**

Objective	Maximize: $W_{tot}(t)$
$\mathbf{B_l(t)}$	0
$\mathbf{B_u(t)}$	36,000
$\mathbf{tlimit_l}$	0
$\mathbf{tlimit_u}$	8,050
No. of Monte Carlo Runs	25
No. of Simulation Steps per Run	1000
No. of population	20
No. of generations	20

4.3 Optimization for Wealth and Hashing Power

The last calibration step is the optimization for wealth and hashing power. The optimization of a cryptocurrency just for the overall wealth might be not a feasible solution, since the network's security is neglected. In the case of Bitcoin and similar cryptocurrencies, relies the network's security completely on its hashing power. (Pagnotta, 2018) It is therefore a vital aspect of the design and should also given a high degree of attention. The network's hashing power, introduced in section 1.3, is defined as:

$$r_{tot}(t) = \sum_{i=1}^{j} r_i(t) \tag{19}$$

where $r_i(t)$ is the hashing power of miner i and j is the amount of all miners at t. This time the NSGA-II algorithm optimizes for two objectives, which is the maximization of $W_{tot}(t)$ and $r_{tot}(t)$, where wealth and NSGA-II both have been introduced in the previous section. The optimization in this step will not return one optimal value, but instead an optimal pareto-front of values, that are all non-dominated by each other. This is expected to result in a function that represents the trade-off between wealth and hashing power of a cryptocurrency network. The final parameters for the NSGA-II are the following:

Table 5: **Meta-Parameters for SNAG of Wealth and Hashing Power Optimization**

Objective	Maximize: $W_{tot}(t)$ Maximize: $r_{tot}(t)$
$\mathbf{B_l(t)}$	0
$\mathbf{B_u(t)}$	36,000
$\mathbf{tlimit_l}$	0
$\mathbf{tlimit_u}$	8050
No. of Monte Carlo Runs	25
No. of Simulation Steps per Run	1000
No. of population	40
No. of generations	20

Bibliography

Arifovic, J. (2002). Exchange rate volatility in the artificial foreign exchange market. In *Evolutionary computation in economics and finance* (pp. 123–134). Springer.

BitInfoCharts. (2019). *Bitcoin median transaction value historical chart.* Retrieved 2019.04.23, from https://bitinfocharts.com/comparison/mediantransactionvalue-btc-sma90.html

Blockchain.com. (2019). *Hash rate; the estimated number of tera hashes per second (trillions of hashes per second) the bitcoin network is performing.*

Buybitcoinworldwide. (2019). *How many bitcoins are there?* Retrieved 2019.04.25, from https://www.buybitcoinworldwide.com/how-many-bitcoins-are-there/#

Calle, P. (2017). NSGA-II explained. *Analytics lab of University of Oklahoma*. Retrieved 2017-10-24, from http://oklahomaanalytics.com/data-science-techniques/nsga-ii-explained/

CEIC. (2019). *China electricity price.* Retrieved 2019.04.13, from https://www.ceicdata.com/en/china/electricity-price

Chiu, J., & Koeppl, T. V. (2017). *The economics of cryptocurrencies - bitcoin and beyond.* doi: 10.2139/ssrn.3048124

Cocco, L., Concas, G., & Marchesi, M. (2017). Using an artificial financial market for studying a cryptocurrency market. *Journal of Economic Interaction and Coordination*, *12*(2), 345–365. doi: 10.1007/s11403-015-0168-2

Cocco, L., & Marchesi, M. (2016). Modeling and simulation of the economics of mining in the Bitcoin market. *PLoS ONE*, *11*(10), 1–42. doi: 10.1371/journal.pone.0164603

Narzisi, G., Mysore, V., & Mishra, B. (2006). Multi-objective evolutionary optimization of agent-based models: an application to emergency response planning. *International Conference on Computational Intelligence*(Ci), 224–230.

Newman, M. E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, *46*(5), 323–351. doi: 10.1080/00107510500052444

Pagnotta, E. (2018). Bitcoin as decentralized money: Prices, mining rewards, and network security. *Mining Rewards, and Network Security (October 26, 2018)*.

Raberto, M., Cincotti, S., Dose, C., Focardi, S. M., & Marchesi, M. (2005). Price formation in an artificial market: Limit order book versus matching of supply and demand. *Lecture Notes in Economics and Mathematical Systems*, *550*, 305–315. doi: 10.1007/3-540-27296-8_20

Raberto, M., Cincotti, S., Focardi, S. M., & Marchesi, M. (2001). Agent-based simulation of a financial market. *Physica A: Statistical Mechanics and its Applications*, *299*(1-2), 319–327. doi: 10.1016/S0378-4371(01)00312-0

Raberto, M., Cincotti, S., Focardi, S. M., & Marchesi, M. (2003). Traders' Long-Run Wealth in an Artificial Financial Market. *Computational Economics*, *22*(2-3), 255–272. doi: 10.1023/A:1026146100090

Satoshi Nakamoto. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. , 1–9. doi: 10.1007/s10838-008-9062-0

Seah, C. W., Ong, Y. S., Tsang, I. W., & Jiang, S. (2012). Pareto rank learning in multi-objective evolutionary algorithms. *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 10–15. doi: 10.1109/CEC.2012.6252865

Statista. (2019). *Number of blockchain wallet users worldwide from 1st quarter 2016 to 1st quarter 2019.* Retrieved 2019.04.25, from `https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/`