Open Blockchain-based Local Energy Market Simulation Platform

Master Thesis

Author: Niklas Reinhold (Student ID: 5377277)
Supervisor: Univ.-Prof. Dr. Wolfgang Ketter
Co-Supervisor: Philipp Kienscherf

Department of Information Systems for Sustainable Society
Faculty of Management, Economics and Social Sciences
University of Cologne

June 18, 2019

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs. 1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Niklas Reinhold

Köln, den xx.xx.20xx

Abstract

[Abstract goes here (max. 1 page)]

Contents

1 Research Motivation **1**

2 Literature Review **3**
2.1 Blockchain-based Energy Markets 3
2.2 Distributed Resource Optimization 4
2.3 Competitive Benchmarking 4

3 Research Design **6**

4 Expected Contribution **7**

5 Simulation Platform Architecture **8**

6 About Blockchain **9**
6.1 General Purpose 9
6.2 Architecture 9
6.2.1 World State 10
6.2.2 Block 11
6.2.3 Transaction 13
6.2.4 Transaction Life Cycle 14
6.2.5 Mining 15
6.2.6 Ethereum Clients 18
6.2.7 Smart Contract 19

7 Linear Programming **21**
7.1 Duality Theory 22
7.1.1 Weak Duality Theorem 23
7.1.2 Strong Duality Theorem 23
7.1.3 Simplex-Method 23

A Appendix **25**

References **26**

List of Figures

1 Research Design following Ketter, Peters, Collins and Gupta . . . 6
2 Platform Design . . . 8
3 World State: Transition of States . . . 10
4 Example of a Patricia Merkle tree . . . 11
5 Example of a data change in a leaf node . . . 11
6 Node network demonstrating a transaction broadcast . . . 14
7 Scenario of Blockchain Branches . . . 17

List of Tables

1 Transaction Pool of Mining Node 15

Listings

1 Pseudocode for PoW mechanism 16

1 Research Motivation

The generation from distributed renewable energy sources (RES) is constantly increasing (Mengelkamp, Gärttner, et al., 2018). In contrast to power plants which run by non-renewable fossil fuels, distributed RES produce energy in a decentralized and volatile way, which is hard to predict. These characteristics of the distributed RES challenge the current energy system (Ampatzis et al., 2014).

The existing electric grid is build for centralized generation by large power plants and the design of the current wholesale markets is not able to react in real-time to a significant amount of distributed RES (Mengelkamp, Gärttner, et al., 2018) (Ampatzis et al., 2014). Moreover, this way of energy generation is economically not ideal because of energy losses due to long physical distances between generation and consumption parties.

Therefore, new market approaches are needed, to successfully integrate the increasing amount of distributed RES (Mengelkamp, Notheisen, et al., 2018). A possible solution to the technical and market problems is Peer-to-Peer (P2P) energy trading in local energy markets (LEM) (Long et al., 2017).

LEM, also called microgrid energy markets, consist of small scale prosumers, consumers and a market platform which enables the trading of locally generated energy between the parties of a community. Due to the trading of locally generated energy within the related communities, LEM support sustainability and an efficient use of distributed renewable energy sources. Likewise, the need of expensive and inefficient transportation of energy through long physical distances can be reduced. The concept of LEM strengthens the self-sufficiency of communities and enables possible energy cost reductions. Moreover, profits remain within the communities whereby reinvestments in additional RES are promoted (Mengelkamp, Gärttner, et al., 2018).

However, P2P energy trading in LEM requires advanced communication and data exchanges between the different parties, which makes central management and operation more and more challenging. The implementation of LEM needs local distributed control and management techniques. (Andoni et al., 2019). Therefore, a new and innovative information communication technology (ICT) is required. A possible solution provides the emerging distributed ledger technology (DLT). It is designed to enable distributed transactions without a central trusted entity. Furthermore, an Ethereum-based blockchain allows the automated execution of smart contracts depending on vesting conditions, which suits the need of LEM for decentralized and autonomous market mechanisms. This offers new approaches and market designs. Accordingly, DLT can help addressing the challenges faced by decentralized energy systems. However, DLTs are not a matured

technology yet and there are several barriers in using them, especially for researcher who do not have a technically background.

In addition, Ketter, Peters, Collins and Gupta introduce the approach of Competitive Benchmarking (CB) (Ketter et al., 2015). This approach describes a research method which faces a real world wicked problem that is beyond the capacity of a single discipline. This is realized by developing a shared paradigm which is represented in a concrete open simulation platform. In detail, it consists of the three principal elements *CB Alignment*, *CB Platform* and *CB Process*. The *CB Alignment* refers to the constant synchronization process between the shared paradigm and the wicked problem. Further, the *CB Platform* represents the medium, in which the shared paradigm is technically illustrated. In addition, the *CB Platform* provides the infrastructure for the third element *CB Process*. It describes the iterative development of new theories and design artifacts through independent researchers, which influence each other and improving their work in direct sight of each other.

Due to the plurality of involved parties and the interdisciplinary requirements for the implementation of new energy market approaches, the accessibility is of major importance. Therefore, the presence of such a open simulation platform depicting the shared paradigm of a LEM, would ensure the accessibility and could help to gain new valuable outcomes in the research field of LEM or new energy market designs in general.

Further, Guo, Koehler and Whinston developed a market-based optimization algorithm, which solves a distributed system optimization problem by self-interested agents iteratively trading bundled resources in a double auction market run by a dealer. The authors called it bundle trading market framework or short BTM (Guo et al., 2007). The central problem of the stated market-based optimization algorithm can interpreted as the welfare optimization of all participants in a LEM. The dealer, which runs the double auction market, maximizes the welfare through allowing agents to trade their preferable bundles of energy. Hence, the stated BTM implemented on basis of a blockchain as underlying ICT can depict the concept of a LEM.

Consequently, this research will bring all the introduced approaches together and develop an open blockchain-based LEM simulation, which enables the research approach based on the three stated elements of CB. The platform will be realized through the introduced optimization algorithm with a blockchain as the underlying ICT. That means, the smart contract takes the role of the market dealer and the self-interested agents represent the individual participants in a LEM. The focus of this paper will be on the implementation and software design of the open blockchain-based LEM simulation platform.

2 Literature Review

2.1 Blockchain-based Energy Markets

To start off, Mihaylov et al. were the first who adressed blockchain technology in energy markets. They present a new decentralized digital currency with the aid of which prosumers trade locally produced renewable energy (Mihaylov et al., 2014). In their introduced concept, the generation and consumption of renewable energy is direct transferable into virtual coins. However, the market value of the virtual currency is determined centrally by the distributed system operator. Further, Al Kawasmi et al. introduce a blockchain-based model for a decentralized carbon emission trading infrastructure (Al Kawasmi et al., 2015). Their model based on the bitcoin protocol and focus on privacy and system security goals. Besides, they provide a solution to the problem of anonymous carbon emission trading. Equally, Aitzhan and Svetinovic address the issue of transaction security in decentralized smart grid energy trading and implemented a proof-of-concept for a blockchain-based energy trading system including anonymous encrypted messaging streams (Aitzhan & Svetinovic, 2018). Concluding, they show that blockchains enable the implementation of decentralized energy trading and that the degree of privacy and security is higher than in traditional centralized trading platforms. Furthermore, Sikorski et al. present an proof-of-concept where a blockchain enables machine-to-machine (M2M) interactions depicting an M2M energy market (Sikorski et al., 2017). They pointed out that the blockchain technology has significant potential to support and enhance the 4th industrial revolution. Moreover, Mengelkamp et al. reveal the concept of a blockchain-based local energy market without the need of a thrusted third entity (Mengelkamp, Gärttner, et al., 2018). In addition, they deduce seven market components as a framework for building efficient microgrid energy markets. Consequently, the Brooklyn Microgrid project is introduced and evaluated according the market components. As a result, the Brooklyn Microgrid also shows that blockchains are a suitable technology to implement decentralized microgrid energy markets, though current regulation does not allow to run such a local energy markets in most of the countries. Later on, Mengelkamp et al. present an initial proof-of-concept of a simple blockchain-based concept, market design and simulation of a local energy market consisting of hundred households (Mengelkamp, Notheisen, et al., 2018). Finally, they conclude that the real-life realisation and technological limitations of such blockchain-based market approaches need to be investigated by further research. In addition, it is mentioned that regulatory changes will play an important role in the future of blockchain-based LEM.

2.2 Distributed Resource Optimization

To begin with, Fan et al. outline a new approach for the development of an information system which can be used for the problem of a supply chain. The concept demonstrate a decentralized decision making process that is realized through the design of a market-based coordination system which incite the pariticipants to act in a way that is beneficial to the overall systems (Fan et al., 2003). Further, Guo et al. revive this concept and develop a market-based decomposition method for decomposable linear systems, that can be easily implemented to support real-time optimization of dsitributed systems (Guo et al., 2007). They prove that the system optimality can be achieved under a dynamic market-trading algorithm in a finite number of trades. Moreover, the outlined alogrithm can be operated in synchronous and as well in asynchronous environments. Later on, Guo et al. extend their stated concept to a dynamic, asynchronous internet market environment (Guo et al., 2012). Additionally, they examine how various market design factors like dealer inventory policies, market communication patterns, and agent learning strategies affect the computational market efficiency and implementation. Finally, also this time, Guo et al. prove finite convergence to an optimal solution under all these different schemes.

2.3 Competitive Benchmarking

Firstly, March and Smith describe a two dimensional framework for research in IS (March & Smith, 1995). The two dimensions can be distinguished into behavioral-science and design-science. The behavioral-science paradigm is based on explaining or predicting human or organizational behavior. Whereas, the design-science paradigm is based on broad types of outputs produced by design research. It strives to extend the boundaries of human and organizational capabilities through the creation of new artifacts.

Furthermore, the research framework presented by Hevner et al. combines the both stated IT research paradigms and illustrates the interaction between these two (Hevner et al., 2008). They present that the technology and behavior are inseparable in an information system. Therefore, they argue that considering the complementary research cycle between design-science and behavioral-science is crucial to address fundamental problems faced in the productive application of information technology.

In addition, Ketter, Peters, Collins and Gupta introduce the IS research approach called Competitive Benchmarking (CB), which includes varios basic principles of the design science approach of Hevner et al. (Ketter et al., 2015). This IS research concept is designed for so called wicked problems and address the

problems and needs of interdisciplinary research communities. CB focus on the interconnection of problems at the same and different levels to imitate real world. In repeated competitions, in which individual teams compete against each other, the developed environment and models can be tested and evaluated. This results in a diversity of outcoming designs, which are difficult to achieve in traditional design science frameworks.

3 Research Design

This research is inspired by the stated CB research approach and focus on the development of the second CB element, an open simulation platform depicting the shared paradigm of a LEM. The proposed design is presented in Figure 1. In contrast to the inital CB research design, the third element *CB Process* differs. The paltform enables research groups to design and test their artifacts in the field of the shared paradigm, using the developed platform. However, the designed artifacts in the form of autonomous software agents do not compete against each other in competitions. Rather, research groups are able to design and develop the behavior of all agents and the autonomous market mechanism. Hence, it is also possible to analyze and evaluate the designed artifacts and use them as a benchmark to compare different designs.

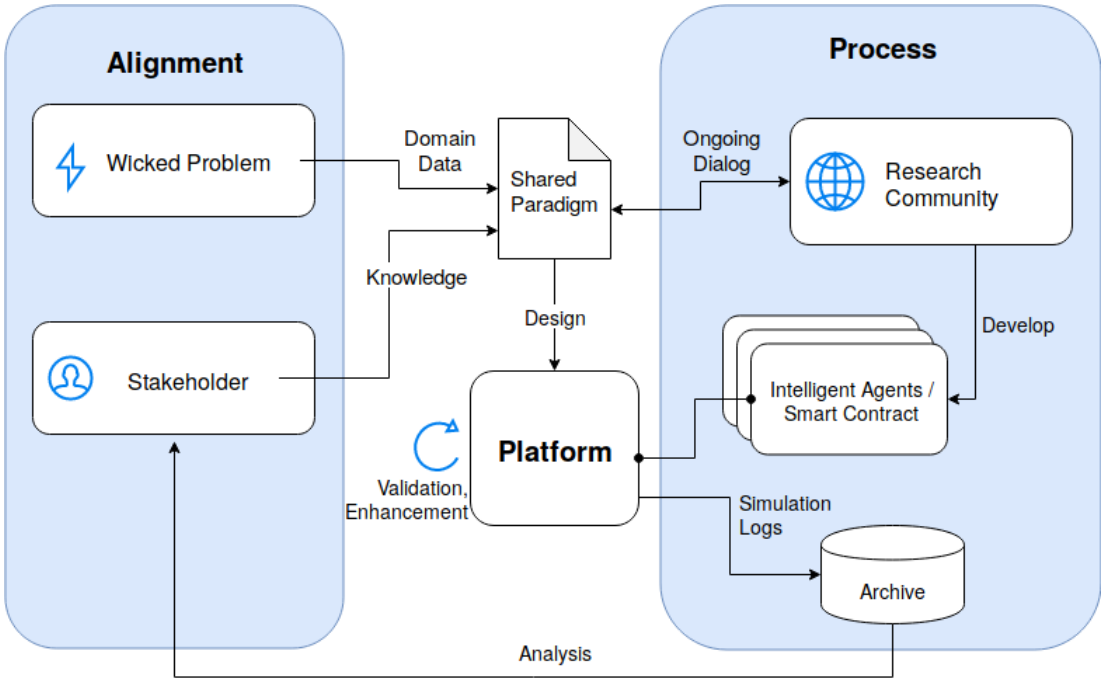

Figure 1: Research Design following Ketter, Peters, Collins and Gupta

According to the IS Design Science principles introduced by Hevner, March, Park, and Ram (Hevner et al., 2008), the presented research design produce a viable artifact in the form of an open simulation platform, which depict a relevant real-world wicked problem. Due to the outcoming data produced by the simulation platform, it is possible to develop methods to evaluate the utility, quality and efficacy of the artifact. Moreover, the research provide a varifable contribution through the artifact, the open simulation platform, itself. The platform enables the investigation of possible solutions of unsolved problems and further, it removes the technical barriers for the use of the complex distributed ledger technology.

Besides, the implementation of the platform is based on an appropriate selection of techniques. As stated in 2.1, blockchains are a suitable technology to implement decentralized microgrid energy markets. In addition, the in 2.2 introduced distributed optimization algorithm achieve evidentially system optimality under a dynamic market-trading algorithm. Therefore, this research relies upon the application of rigorous methods and comply the requirement of the fifth guideline by Hevner et al. (2008). In addition, the platform enables the iterative search for an optimal design through the comparison of different produced solutions and is valuable to technology-oriented as well as management-oriented audiences. As a result, these research design also fulfil the seven research guidlines of the research framework introduced by Hevner et al. (2008).

4 Expected Contribution

This research provides two different contributions. First, the fully decentralization of the introduced distributed optimization algorithm (Guo et al., 2007). The bundle-trading market grant access of any given trade to the market dealer. This, again, necessitates trust of the agents that it will use those resources according to the over-arching organizational goal. Due to the implementation of the market dealer by a smart contract, the technical implementation is public accesible. Moreover, all transactions of the market dealer to allocate the resources are transparent. Therefore, the behavior of the market dealer is comprehensible for every participant. Furthermore, it provides a high degree of security due to cryptographic encryption methods which are essential parts of the blockchain. Second, the open simulation platform itself. From the scientific perspective, the platform is relevant due to the removal of existing technical barriers for practicioners in using complex distributed ledgers technologies. Therefore the platform facilitates researchers without a deep technically background to use those DLTs and incentivises to design and test their artifacts by using this platform. On the other hand, from a business perspective, this research is relevant to policy makers and energy suppliers. The platform allows stakeholders to get a better understanding of the dynamics of decentralized LEM and enables the testing and evaluating of policy options and implications.

5 Simulation Platform Architecture

This chapter gives a short representation of the first platform design and the chosen technologies and frameworks. First, the Ethereum blockchain is implemented through Ganache. It is a personal local blockchain for the Ethereum development, which can be used to deploy smart contracts, develop decentralized applications, and run tests. Further, Ganache is available as a desktop application as well as a command-line tool. Next, the role of the market dealer is realized by a smart contract which is written in Solidity. This object-oriented programming language was especially designed for developing smart contracts that run on a Ethereum blockchain. The Solidity code is compiled to bytecode, which can be deployed into the blockchain. Moreover, the clients which constitute the different agents are implemented through the programming language Python. Python is a object oriented, high-level programming language with a easy and simple to learn syntax. Therefore, the hurdles in modifying the client behavior are minimized. Finally, the clients using the Web3.py python library for the interaction with the Ethereum blockchain.

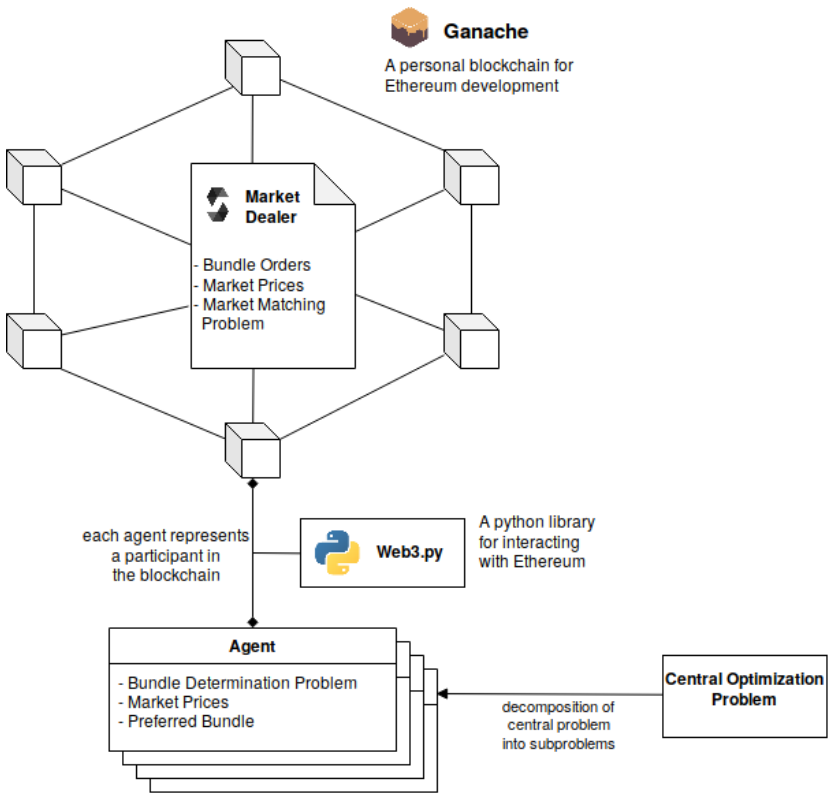

Figure 2: Platform Design

6 About Blockchain

To begin with, this chapter will give an overview of the general purpose, the contained components and the fundamental functionality of a blockchain.

6.1 General Purpose

In general, a blockchain can be described as a digital data structure that can be understand as a shared and distributed database, containing a continuous expanding and chronological log of transactions (Andoni et al., 2019). Besides, various types like digital transactions, data records and executables can be stored in this digital data structure. The data transmission in a blockchain is comparable with copying data from one computer to another. However, the resulting challenge is that the system needs to ensure that the data is copied just once (Andoni et al., 2019). For example, in the domain of cryptocurrencies, this is equal to sending a coin from one wallet to another. In this case, the system needs to validate that this coin is spended just once and there is no double-spending. A conventional solution for this problem is a third intermediary. To come back to the stated example, the third intermediary is represented by a traditional bank, which store, protect and continuously update the valid state of the ledger (Andoni et al., 2019). But, in some cases central management is not practicable or reasonable. Reasons for this are possible intermediary costs or a high degree of trust of the users into the intermediary who operates the system. Further, central management has a significant disadvantage because of a single point of failure. Hence, the centralized system is fragile to technical problems as well to external malicious attacks (Andoni et al., 2019). Consequently, the main reason of bockchain technologies is the removal of such third trusted intermediaries through a distributed network of various users, who cooperating together to verify transactions and protect the validity of the ledger

6.2 Architecture

This subsection covers the architectural design of a blockchain and presents the contained components in detail. Due to the plurality of the blockchain technologies, each of the technology slightly differs in design and components. The following explanations are oriented torwards the Ethereum blockchain implementation, which is also used as the underlying ICT to implement the open simulation platform.

6.2.1 World State

Referring to the *Yellow Paper* by Wood (2014), Ethereum can be seen as a transaction-based state machine. What does that mean? At the beginning, Ethereums state machine starting with a so called *"genesis state"*. This is analogous to a blank sheet. On this state, no transactions have happened on the network. Next, transactions are executed and the state of the Ethereum world changes into a new state. Further, transactions are executed incrementally and morph it into some final state. Consequently, the final state is accepted as the canonical version of the world of Ethereum and represents at any times the current state.

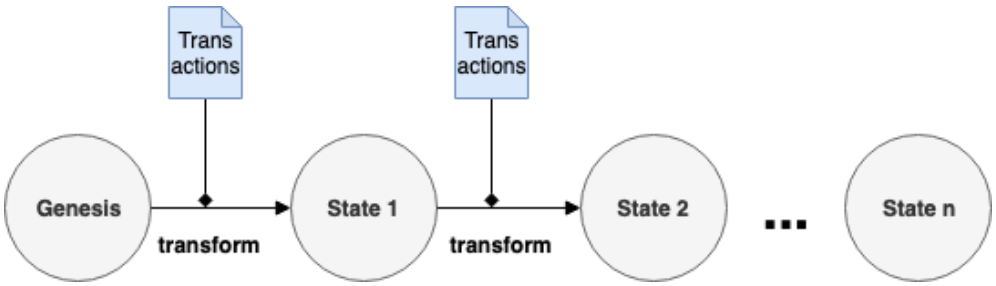

Figure 3: World State: Transition of States

In more detail, the world state arises out of a mapping of a key value pair for every account which exists on the Ethereum network (Wood, 2014). The key constitutes the address of an Ethereum account and the value presents the account's state, which contains detailed information of this account. However, the world state is not stored on the blockchain itself. This mapping is stored and maintained in a modified data structure called a *Merkle Patricia tree*. This tree is stored off-chain in a simple database backend (i.e. on a computer running an Ethereum client), also known as the *state database* in the Ethereum world (Wood, 2014). To get a better understanding of the operating principles of the blockchain, it is necessary to get an idea of how a *Merkle Patricia tree* works. A *Merkle Patricia tree* is a type of binary tree, which consists of a set of nodes. It has a large amount of *leaf nodes*, containing the underlying data. Further, a set of intermediate nodes, where each node is the hash of its two children, and finally, one single root node, representing the top of the tree which is also build out of its two child nodes (Buterin et al., 2013) (Wood, 2014). As mentioned before, the *leaf nodes* contain the stored data by splitting these data into chunks. Afterwards, these chunks are splitted into buckets. Then, each bucket gets hashed and the same process repeats, traversing upwards the tree, until the total number of hashes remaining becomes only one and the root node is reached (*Merkling in Ethereum*, 2015).

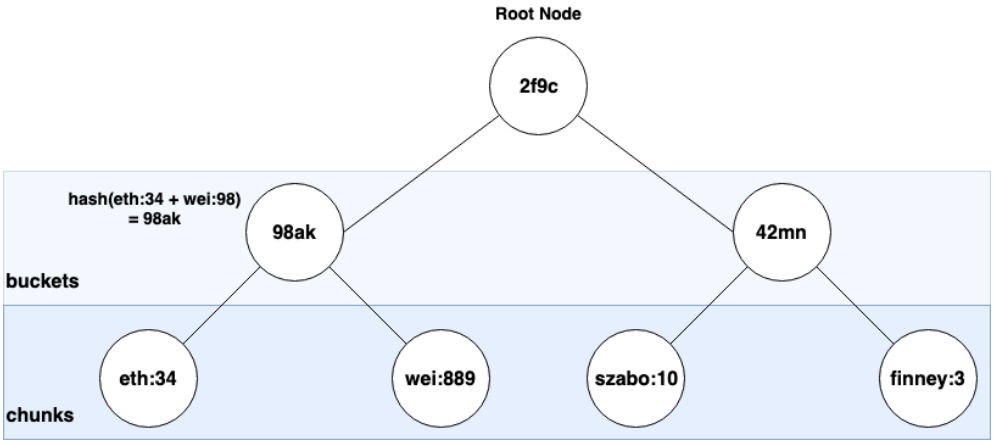

Figure 4: Example of a Patricia Merkle tree

Therefore, any change to the underlying data, stored in a *leaf node*, causes a change of the hash of the node. Each parent's node hash depend on the data of its children. Due to this, any change to the data of a child node causes the parent hash to change. This procedure repeats traversing upwards until the root node. Hence, any change to the data at the leaf nodes effects the root hash (*Modified Merkle Patricia Trie Specification*, 2015).

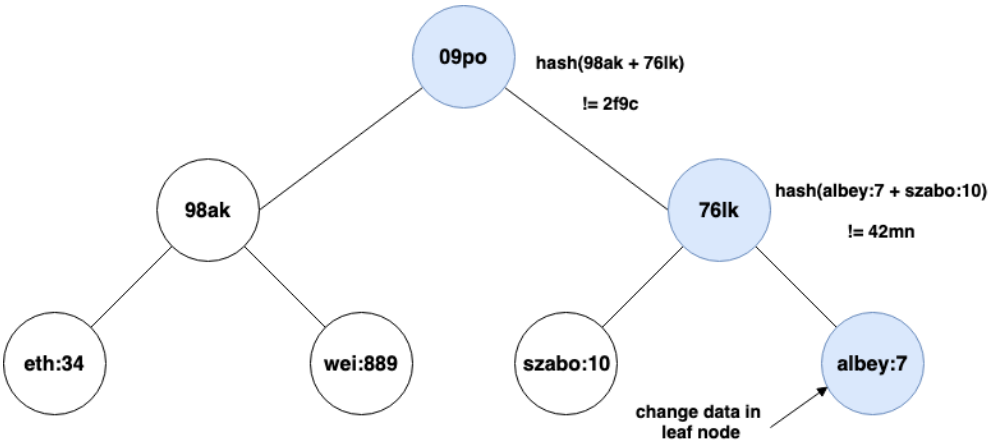

Figure 5: Example of a data change in a leaf node

Because of this characteristic, it is not necessary to compare the data of the entire tree. It is sufficient to compare the single root hash to ensure that all the data are the same. This property is very important, because it makes it possible to store only the hash of the root node to represent a state of the Ethereum world.

6.2.2 Block

As described in section 6.2.1, whenever transactions are executed, the state of the Ethereum world changes into a new state. Every state and the belonging

transactions, which transform the prior state into the new state, is cumulated in a so called block. That means, states are represented by blocks. As you can see in figure 3, the history of the Ethereum world is a linkage of states, or in other words, blocks. That is where the name blockchain comes from. A blockchain is a sequence of blocks, which holds a complete list of transaction records (Zheng et al., 2017). Moreover, a block is a collection of different relevant informations and consists of the *block header* and the *block body*, which contains the list of transactions. Following *Ethereums Yellow Paper* by Wood (2014), the subsequent pieces of information are contained in the *block header*:

Parent Hash: This is the hash of the parents block's header. Therefore, every block points to his ancestor. Due to this contained attribute, a chain arises out of the single blocks.

Beneficiary: The miners address to which all block rewards from the successful mining of a block are transferred.

State Root: This is the hash of the root node of the state tree, after a block and its transactions are finalized. As mentioned in section 6.2.1, the state tree is the one and only global state in the Ethereum world. It is used as a secure unqiue identifier for the state and the state root node is cryptographically dependent on all internal state tree data.

Transactions Root: This is the hash of the root node of the transaction tree. This tree contains all transactions in the block body. In contrast to the state tree, there is a separate transactions tree for every block.

Receipts Root: Every time a transaction is executed, Ethereum generates a transaction receipt that contains information about the transaction execution. This field is the hash of the root node of the transactions receipt tree and like the transaction tree, there is a seperate receipt tree for every block.

Difficulty: This is a measure of how hard it was to mine this block – a quantity calculated from the previous block's difficulty and its timestamp

Number: This is a quantity equal to the number of blocks that precede the current block in the blockchain.

Gas Limit: This is a quantity equal to the current maximum gas expenditure per block. Each transaction consumes gas. The gas limit specifies the maximum gas that can be used by the transactions included in the block. It is a way to limit the number of transactions in a block.

Gas Used: This is a quantity equal to the total gas used in transactions in this block.

Timestamp: This is a record of Unix's time at this block's inception.

Nonce: This is an 8-byte hash that verifies a sufficient amount of computation has been done on this block. Further, it is a number added to a hashed block that, when rehashed, meets the difficulty level restrictions. The nonce is the number that blockchain miners are solving for.

So far, we introduced Ethereum as a transaction-based state machine, transforming one state into another through the execution of transactions. Further, we explained how these individual states are stored and emphasises the meaning of these storage. Moreover, we pointed out that an Ethereum state is represented by blocks and described all relevant informations contained in it. However, we didn't introduce how many transactions be part of a block and who build and validate a block? To answer this, we introduce the transaction object and his life cycle in depth.

6.2.3 Transaction

To start with, a transaction is the basic method for Ethereum accounts to interact with each other. Further, in the Ethereum world exists two different types of transactions. Those, which result in a so called *message call*, which can be seen as a traditional transaction, and those which result in a *contract creation* (Wood, 2014). Though, both different types of transaction share the following common attributes (Wood, 2014):

Nonce: This attribute is discontiguous of the block attribute. In contrast to the block attribute, the nonce of a transaction keep track of the total number of transactions that an account has executed.

Gas Price: As mentioned in section 6.2.2, each transaction consume gas. Gas can be seen as fees. This attribute presents the price per unit of gas for a transaction [1].

Gas Limit: This attribute is similiar to the block attribute of the same name. In this case, it is a quantity equal to the current maximum gas expenditure per transaction.

To: In case of a *message call*, this attribute contains the address of the recipient. Otherwise, in case of a *contract creation* this value remains empty.

[1] A website to see current gas prices: https://ethgasstation.info/index.php

Value: In case of a *message call*, this attribute contains the amount of Wei, which will be transferred to the recipient. In the case of a *contract creation*, this value contains the amount of Wei as a intial endowment of the contract.

6.2.4 Transaction Life Cycle

Next, the life cycle of a transaction will be outlined. We will take a simple *message call* as an example and will pass through the entire flow of how this transaction gets executed and permanently stored on the blockchain. During this process, many basic concepts of a blockchain will be introduced.

To begin with, someone want to send an abitrary amount of ether to someone else. Consequently, a transaction with the respective attributes, which are stated in section 6.2.3, will be created. As a first step, this transaction will be signed. That means, the one who wants to execute the transaction needs to proof the ownership of the account. Otherwise, someone else could execute a transaction on your behalf. The way to proof this is by signing the transaction with the corresponding private key of this account (*Medium, Life Cycle of Transactions*, 2017). Next, the signed transaction is submitted to the related Ethereum node. Then, the node will validate the signed transaction. After a successful validation, the node will send the transaction to it's peer nodes, who again send it to their peer nodes and so on (*Medium, Life Cycle of Transactions*, 2017). As soon as the transaction is broadcast to the network, the related node will declare a transaction id, which is the hash of the signed transaction and can be used to track the status of the transaction [2].

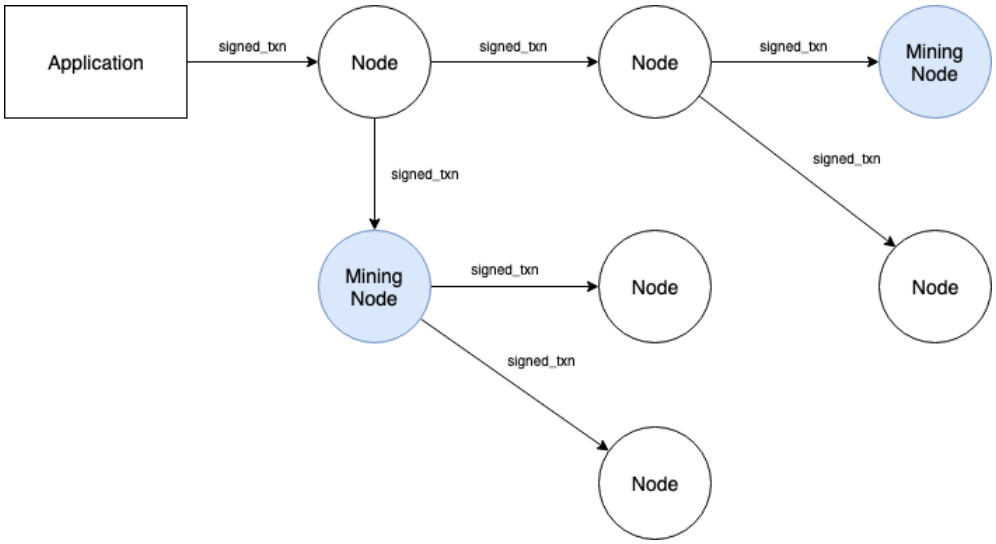

Figure 6: Node network demonstrating a transaction broadcast

[2] A Website to track transactions: https://etherscan.io/

Further, figure 6 describes a Ethereum network. As you can see, it contains a mix of miner nodes and non miner nodes. The non miner nodes can be distinguished into different types of nodes with different behavior, which are explained in detail in section 6.2.6. The miner nodes are the ones who processing transactions into blocks (*Medium, Life Cycle of Transactions*, 2017).

Mining nodes contain and maintain a pool of transactions where they collect incoming transactions. They sort the transactions by gas price. However, there are no specific rules how the nodes sort the transactions. A common configuration is to sort the transaction by gas price in a descending order to optimize for a higher pay (*Medium, Life Cycle of Transactions*, 2017).

Signed Transaction	Gas Price
hashOfTransaction1	15 Gwei
hashOfTransaction2	13 Gwei
hashOfTransaction3	11 Gwei
hashOfTransaction4	10 Gwei
hashOfTransaction5	7 Gwei

Table 1: Transaction Pool of Mining Node

Further, the mining node takes transactions from the pool and processes them into a pending block. Moreover, a block can only contain a certain number of transactions due to the block attribute *gas limit*, mentioned in section 6.2.2. That means, the mining node can only process so many transactions into one block until the gas limit is reached [3]. For example, imagine that the current *gas limit* of a block is 28 Gwei. Next, we look at the transaction pool of a mining node in table 1. As you can see, two blocks are required to process all transactions from the pool. The first two transactions are contained in the first block. Furhtermore, the second block contains the remaining three transactions. So far, it is described how a pending block will be created and the transaction limit of a block is explained. In the next section 6.2.5, the process of a pending block to a valid block is outlined in depth. Finally, after the validation process of a block is finished successfully, the life cycle of a transaction is terminated. In other words, the transaction is anchored in the blockchain and modified the world state as mentioned in section 6.2.1.

6.2.5 Mining

In this section, the stteps of the validation process of a pending block will be described. In general, each mining node in the network can generate or propose a

[3] A website to see current attributes: https://ethstats.net/

block. The emerging question is, which node build the new block and how will the newly generated block accepted by the remaining network members? This process is called *consensus mechanism*. It exists a lot of various consensus algorithms, each of them provides different features, advantages and disadvantages.

To a large extent, the key perfomance drivers of a blockchain like transaction speed, scalability and security depending on the embedded consensus algorithm (Andoni et al., 2019). In this section, we will expound the *Proof of Work (PoW)* mechanism, which is at the time of this writing the current strategy in the Ethereum blockchain implementation for reaching consensus. *Proof of Work* enables a high scalability in terms of the number of nodes and clients (Vukolić, 2015), whereas it provides a poor transaction speed and a high power consumption due to the solving of a cryptographical puzzle, which requires significant computational effort (Andoni et al., 2019).

The *PoW* is a random process which is not predictable and therefore only solvable through a trial and error approach (*Proof of Work*, 2019). All mining nodes compete with each other and the goal is to achieve a hash output that is lower than a given specified target (Andoni et al., 2019). The hash output includes and comprises out of the *Parent Hash*, *Transaction Root* and *Nonce*, which are all part of the block headers data, mentioned in 6.2.2. However, the *Parent Hash* and *Transaction Root* are given and immutable, thus the *Nonce* is the value that all the mining nodes are solving for [4]. Consequently, the mining nodes modify the *Nonce* until the hash output is lower than the required target (Andoni et al., 2019). The following pseudocode in listing 1 will illustrate the *PoW* procedure:

```
1  targetValue = 00005cdf6d384113165841052dfd4638eaf756ac
2  parentHash = abbecf2d59eacbde676c3f1f0bfcf124f8c68209
3  transactionRoot = 917c631e5ee59596859910759c8ed76a21252010
4  nonce = 1
5  valid = False
6
7  while(not valid):
8      hashOutput = sha256(parentHash+transactionRoot+nonce)
9
10     if(hashOutput <= targetValue):
11         valid = True
12     else:
13         nonce += 1
```

Listing 1: Pseudocode for PoW mechanism

[4] A interactive blockchain demo: https://anders.com/blockchain/

Finally, when one mining node successfully calculated the *Nonce* and therefore reaches the target value, it broadcasts the block to all other mining nodes in the network. Now, all other nodes mutually validate the correctness of the hash value by recalculating the hash output and verifying if it is lower then the target value. If the block is validated through all other mining nodes, all nodes will append this new block to their own blockchains (Zheng et al., 2017).

Considering, the network is decentralised and the simultaneous generation of valid blocks is possible when multiple nodes find the suitable *Nonce* at a nearly same time (Zheng et al., 2016). In this case, branches will be generated as shown in figure 7

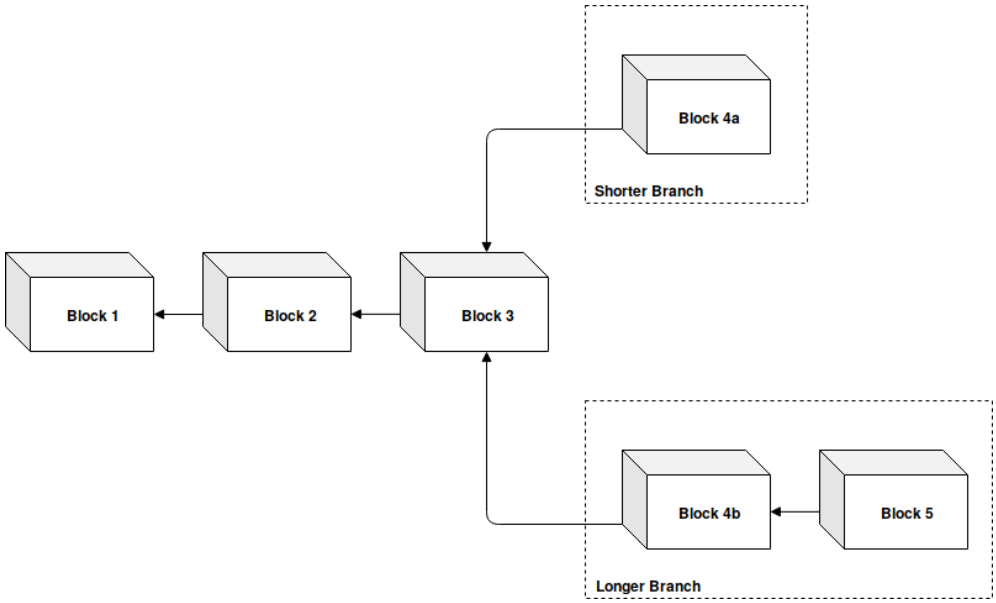

Figure 7: Scenario of Blockchain Branches

Nevertheless, it is unlikely that two competing branches will generate the next block simultaneously again. In the *PoW* protocol, the longer branch is judged as the authentic one. In figure 7, two branches arised out of two simultaneous valided blocks *Block 4a* and *Block 4b*. After that, all mining nodes accept both branches and start working on the next consecutive block until a longer branch is found (Andoni et al., 2019). As you can see in figure 7, *Block 4b* and *Block 5* forms the longer chain, therefore all miners on the branch with *Block 4a* will switch to the longer branch (Zheng et al., 2017).

To sum up, this section joins and completes the previous section 6.2.4. The consensus mechanism *PoW* is described and outlined in pseudocode, whereby the validation process of new generated blocks is explained and illustrated.

6.2.6 Ethereum Clients

First of all, the expressions Ethereum client and Ethereum node can be used interchangeable. As mentioned earlier in the previous section 6.2.4 and illustrated in the figure 6, clients can be distinguished into different types. This section will describe the concept of an Ethereum client in general and will give a brief overview of the different types.

In general, an Ethereum client is a software application that implements the Ethereum specification, which is specified in the Ethereum yellow paper (Wood, 2014). A client communicates over the peer-to-peer network with other Ethereum clients. Although these clients can be implemented in different programming languages, they all communicate through the standardized Ethereum protocol, wherefore they are able to operate and interact with the same Ethereum network. (Antonopoulos & Wood, 2018).

Since the Ethereum blockchain is not officially implemented in a particular programming language, following a list of the current main implementations of the Ethereum protocol:

- `Parity`, written in Rust
- `Geth`, written in Go
- `cpp-Ethereum`, written in C++
- `pyethereum`, written in Python
- `Harmony`, written in Java

However, while all clients differ from each other, they share some fundamental features (*Medium, Ethereum-Clients*, 2019).

First, each is able to join the peer-to-peer Ethereum network. Next, they all synchronizing a local copy of the blockchain. There are a few modes of synchronizing, which comes with miscellaneous advantages and disadvantages, which will be outlined later on. Moreover, it makes a significant difference for the health and resilience of the decentralised network. Lastly, each client is also capable of broadcasting new transactions to the network and creating and managing accounts (*Medium, Ethereum-Clients*, 2019).

As mentioned above, there are differences in client behavior regarding to the synchronizing modes. In general, the clients can be distinguished into two different types, the so called *full node* and *light node*.

Full Node: A full node will download all history data peer-to-peer from another full node. This requires a significant amount of hardware and bandwith resources (Antonopoulos & Wood, 2018). Afterwards, it will simulate every transaction in the ledger and execute the whole deployed source code to recalculate the state of each existing block (*Medium, Ethereum-Clients*, 2019). Therefore, the amount of independently operating and geographically dispersed full nodes is a crucial and very important indicator for the health, resilience and censorship resistance of a blockchain (Antonopoulos & Wood, 2018). However, a full node is not necessarily a mining node. It authoritatively validates all transactions, but to participate in the mining competition, an additional software called *ethminer* is needed [5]

Light Node: To begin with, a light node is not a separate piece of software from the clients that run a full node. The difference lies in the *light* mode for synchronization. In contrast to a full node, a light node only synchronize the block headers and the current state of the chain (*Light-Client-Protocol*, 2019). Anyway, a light node lacks the ability to run transactions throughout the history of the whole blockchain. For this reason, it does not contribute to the health and resilience of the network like a full node and traditionally cannot act as a mining node (*Medium, Ethereum-Clients*, 2019). In addition, a light node has some further limitations. For instance, they are not capable to monitor pending transactions from the network and the hash of a transaction is not sufficient to locate the transaction. Moreover, to perform certrain types of operations, a light node relies on requests to full nodes, whereby they can be far slower to query the chain (*Light-Client-Protocol*, 2019).

6.2.7 Smart Contract

This section deals with smart contracts in the Ethereum world and based entirely on the book *Mastering Ethereum* by Antonopoulos and Wood (2018). First of all, the term smart contract is a bit misleading, since Ethereum smarts contract are neither smart nor legal contracts. However, how can a Ethereum smart contract be defined? A Ethereum smart contract is an immutable comuputer program that runs deterministically in the context of an Ethereum Virtual Machine (EVM) as part of the networkprotocol. To go into the definition in more detail, immutable means, that once the contract is deployed into the blockchain, it is not possible to make any changes in the source code. The only way to modify the contract is to deploy a whole new instance. Further, deterministic means that the result of the execution is the same for all who run it, depending on the context of the transaction that initiated its execution and the state of the blockchain at the

[5]GPU Mining Worker: https://github.com/ethereum-mining/ethminer

moment of execution. As already mentioned in section 6.2.3, in the Ethereum world exists two different types of transactions. Those which result in a message call, and those which result in a contract creation. That means, smart contracts are deployed through special contract creation transactions into the blockchain. The special thing about these transactions is that the recipient of the transaction remains empty, whereby the transaction will be send to a special destination address called *zero address (0x0)*. Each contract is identified and reachable via a common Ethereum address, which can be used in a transaction as the recipient to send funds to the contract or to call one of the contract functions. Anyway, in contrast to an externally owned account, there are no keys associated with an account of a smart contract. For this reason, the creator of a smart contract does not have any special rights at the protocol level. Although, it is possible to get those special rights if you explicitly specified them in the source code of the contract. Furhtermore, contracts are only active and run if they called by a transaction. A contract is able to call another contract and so on, but nevertheless, the first call have always come from an externally owned account. A contract will never run on his own or in the background. Additionally, contracts are not executed in parallel. They will always executed consecutively, hence the Ethereum blockchain can be considered as a single-threaded machine. Finally, it should be noted that transactions are atomic regardless of how many contracts they call. Transactions execute in their entirety and any changes in the global state are only recorded if all executions terminates successfully. If the executions fails, all the changes in state are reverted as if the transaction never ran. Anyhow, a failed execution of a transaction is recorded as having been attempted. Moreover, the gas fees spent for the execution is withdrawn from the originating account.

7 Linear Programming

This section will give an introduction into the basics of mathematical linear programming and linear programming duality. It will present the purpose and necessity of optimization models and will give an example how an optimization model is constructed.

To begin with, optimization models try to define in mathematical terms, the goal of solving a problem in the best or optimal way. This can be applied in many different areas, for example, it might mean running a business to maximize profit, designing a bridge to minimize weight or selecting a flight plan for an aircraft to minimize time or fuel use (Griva et al., 2009). The use case of solving an optimization problem optimally is so ubiquitous, that optimization models are used in almost every area of application (Griva et al., 2009). Often it is not possible or economically feasible to make decisions without the help of such a model. Due to the excellent improvements in computer hardware and software in the last decades, optimization models became a practical tool in business, science and engineering (Griva et al., 2009). Finally, it is possible to solve optimization problems with a huge set of variables.

Further, linear optimization, also known as a *linear program*, refers to the optimization of a linear function with the decision variables, $x_1, ..., x_n$ subject to linear equality or inequality constraints. As presented by Bichler (2017), in canonical form a linear optimization model is given as:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\ & x_j \geq 0, \qquad\qquad j = 1, ..., n \end{aligned}$$

However, a linear program can be modeled in different forms. It is also possible to write a model in the matrix-vector notation. To represent the above model in this form, letting $x = (x_1, ..., x_n)^T$, $c = (c_1, ..., c_n)^T$, $b = (b_1, ..., b_m)^T$ and name the matrix of the coefficients a_{ij} by A. As introduced by Griva et al. (2009), the model becomes:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

In general, the objective function of a linear model is either minimized or maximized. Additionally, the constraints may include a combination of inequalities and equalities and the variables are unrestricted or restricted in sign (Bichler, 2017).

7.1 Duality Theory

In addition, for every linear program exists another problem, which is called the *dual* of the original linear problem, also known as the *primal* (Bichler, 2017). In the *dual*, the roles of variables and constraints are reversed. That means, every variable of the *primal* becomes a constraint in the *dual*, and every constraint in the *primal* becomes a variable in the *dual* (Griva et al., 2009). For instance, if the *primal* has n variables and m constraints, the *dual* will have m variables and n constraints. Further, the *primal* objective coefficients are the coefficients on the right-hand side of the *dual*, and vice versa. Finally, the transposed coefficient matrix of the *primal* constitutes the matrix in the *dual* (Griva et al., 2009). To give an example of a *primal* and the corresponding *dual* linear program, the representation by Bichler (2017) is considered:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \qquad \text{(primal)}$$

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & A^T y \leq c \\ & y \geq 0 \end{aligned} \qquad \text{(dual)}$$

Next, the motivation of the duality theory can explained by the issue of trying to find bounds on the value of the optimal solution to a linear programming problem. If the *primal* is a minimization problem, the *dual* represents a lower bound on the value of the optimal solution, otherwise, if the *primal* is a maximization problem, the *dual* represents an upper bound (Bichler, 2017). To illustrate the concept and motivation of duality theory more tangible, an example stated by Bichler (2017) is given. In an application, the variables in the *primal* linear problem stand for consumer products and the objective coefficients represent the profits linked to the production of those consumer products. In this case, the objective in the *primal* describes directly the relation between a change in the production of the products and profit. Whereas, the constraints in the *primal* describe the availability of raw materials. Consequently, an increase in the availability of raw materials needed for the production of the consumer products, enables an increase in production, and finally an increase in the overall profit. However,

the *primal* problem does not outline this relationship reasonable. Hence, one of the benefit of the *dual* is to properly show the effect of changes in the constraints on the value of the objective. Due to this, the variables in a *dual* linear program often called *shadow prices*, since they describe the hidden costs associated with the constraints.

7.1.1 Weak Duality Theorem

As stated in the section before (7.1), the *dual* problem provides, depending on maximization or minimization problem, lower and upper bounds for the *primal* objective function. Referring to Vanderbei et al. (2015), this result is true in general and is described as the *Weak Duality Theorem*:

Theorem 1. *If x is feasible for the primal and y is feasible for the dual, then:*

$$c^T x \leq b^T y$$

Regarding to Griva et al. (2009), there are some logical corollaries of the *Weak Duality Theorem*:

Corollary 1. *If the primal is unbounded, then the dual is infeasible. If the dual is unbounded, then the primal is infeasible.*

Corollary 2. *If x is a feasible solution to the primal, y is a feasible solution to the dual, and $c^T x = b^T y$, then x and y are optimal for their respective problems.*

7.1.2 Strong Duality Theorem

Concerning to corollary 2, it is possible to verify if the points x and y are optimal solutions, without solving the corresponding linear programs. Furthermore, this corolary is also used in the proof of strong duality (Griva et al., 2009). The *Strong Duality Theorem* describes, that for linear programming there is never a gap between the *primal* and the *dual* optimal objective values (Vanderbei et al., 2015).

Theorem 2. *If the primal problem has an optimal solution x^*, then the dual also has an optimal solution y^*, such that:*

$$c^T x^* = b^T y^*$$

7.1.3 Simplex-Method

This subsection will give a brief introduction to the *simplex-method*. To start with, in the field of linear programming, the *simplex-method* is the most widely used

algorithm. This method was developed around 1940 and had several competitors at that time. However, the algorithm was able to assert itself due to its efficiency. (Griva et al., 2009) In general, the *simplex-method* is an iterative algorithm, used to solve linear programming models (Griva et al., 2009). To apply the algorithm to a model, it must be in standard form (Luenberger, 2008). According to Griva et al. (2009), we consider the following linear progamming model:

$$\begin{aligned} \min \quad & z = -x_1 - 2x_2 \\ \text{s.t.} \quad & -2x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 7 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Next, we give an example to illustrate how to convert a linear program into standard form. To achieve that, the so called *slack variables* will be added, which represents the difference between the right-hand side and the left-hand side.

$$\begin{aligned} \min \quad & z = -x_1 - 2x_2 \\ \text{s.t.} \quad & -2x_1 + x_2 + x_3 = 2 \\ & -x_1 + 2x_2 + x_4 = 7 \\ & x_1 + x_5 = 3 \\ & x_1, x_2, +x_3, +x_4, +x_5 \geq 0 \end{aligned}$$

Finally, with the linear progamming model in standard form, the idea of the *simplex-method* is to iteratively proceed from one basic feasible solution to another (Luenberger, 2008). Thereby, it always looks for a feasible solution which is better then the one before. The definition of better in this case depends on the nature of the linear problem. The process is performed until a solution is reached which cannot be improved anymore. In the end, this final solution is then an optimal solution (Vanderbei et al., 2015).

A Appendix

References

Aitzhan, N. Z., & Svetinovic, D. (2018). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, *15*(5), 840–852.

Al Kawasmi, E., Arnautovic, E., & Svetinovic, D. (2015). Bitcoin-based decentralized carbon emissions trading infrastructure model. *Systems Engineering*, *18*(2), 115–130.

Ampatzis, M., Nguyen, P. H., & Kling, W. (2014). Local electricity market design for the coordination of distributed energy resources at district level. In *Innovative smart grid technologies conference europe (isgt-europe), 2014 ieee pes* (pp. 1–6).

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., ... Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, *100*, 143–174.

Antonopoulos, A. M., & Wood, G. (2018). *Mastering ethereum: building smart contracts and dapps*. O'Reilly Media.

Bichler, M. (2017). *Market design: A linear programming approach to auctions and matching*. Cambridge University Press.

Buterin, V., et al. (2013). A next-generation smart contract and decentralized application platform. *Ethereum White Paper*.

Fan, M., Stallaert, J., & Whinston, A. B. (2003). Decentralized mechanism design for supply chain organizations using an auction market. *Information Systems Research*, *14*(1), 1–22.

Griva, I., Nash, S. G., & Sofer, A. (2009). *Linear and nonlinear optimization* (Vol. 108). Siam.

Guo, Z., Koehler, G. J., & Whinston, A. B. (2007). A market-based optimization algorithm for distributed systems. *Management Science*, *53*(8), 1345–1358.

Guo, Z., Koehler, G. J., & Whinston, A. B. (2012). A computational analysis of bundle trading markets design for distributed resource allocation. *Information Systems Research*, *23*(3-part-1), 823–843.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2008). Design science in information systems research. *Management Information Systems Quarterly*, *28*(1), 6.

Ketter, W., Peters, M., Collins, J., & Gupta, A. (2015). Competitive benchmarking: an is research approach to address wicked problems with big data and analytics.

Light-Client-Protocol. (2019). Retrieved from `https://github.com/ethereum/wiki/wiki/Light-client-protocol`

Long, C., Wu, J., Zhang, C., Cheng, M., & Al-Wakeel, A. (2017). Feasibility of peer-to-peer energy trading in low voltage electrical distribution networks. *Energy Procedia*, *105*, 2227–2232.

Luenberger, D. G. (2008). *Linear and Nonlinear Programming 4TH* (Tech. Rep.). Retrieved from `http://www.springer.com/series/6161`

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, *15*(4), 251–266.

Medium, Ethereum-Clients. (2019). Retrieved from `https://medium.com/@eth.anBennett/ethereum-clients-101-beginner-geth-parity-full-node-light-client-4bbd87bf1dee`

Medium, Life Cycle of Transactions. (2017). Retrieved from `https://medium.com/blockchannel/life-cycle-of-an-ethereum-transaction-e5c66bae0f6e`

Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L., & Weinhardt, C. (2018). Designing microgrid energy markets: A case study: The brooklyn microgrid. *Applied Energy*, *210*, 870–880.

Mengelkamp, E., Notheisen, B., Beer, C., Dauer, D., & Weinhardt, C. (2018). A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science-Research and Development*, *33*(1-2), 207–214.

Merkling in Ethereum. (2015). Retrieved from `https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/`

Mihaylov, M., Jurado, S., Avellana, N., Van Moffaert, K., de Abril, I. M., & Nowé, A. (2014). Nrgcoin: Virtual currency for trading of renewable energy in smart grids. In *European energy market (eem), 2014 11th international conference on the* (pp. 1–6).

Modified Merkle Patricia Trie Specification. (2015). Retrieved from `https://github.com/ethereum/wiki/wiki/Patricia-Tree`

Proof of Work. (2019). Retrieved from `https://en.bitcoin.it/wiki/Proof_of_work`

Sikorski, J. J., Haughton, J., & Kraft, M. (2017). Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Applied Energy*, *195*, 234–246.

Vanderbei, R. J., et al. (2015). *Linear programming.* Springer.

Vukolić, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security* (pp. 112–125).

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*, 1–32.

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In *Big data (bigdata congress), 2017 ieee international congress on* (pp. 557–564).

Zheng, Z., Xie, S., Dai, H.-N., & Wang, H. (2016). Blockchain challenges and opportunities: A survey. *Work Pap.–2016*.