FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig

swisscom | Blockchain

FEES IN BITCOIN COIN SELECTION

AUTHOR
MARIUS GIGER
PIBS 2015
SWISSCOM AG
marius.giger@swisscom.com

SUPERVISORS
PROF. MARTIN KLAPER
SUPERVISOR TERM PAPER
FFHS
martin.klaper@ffhs.ch

KAMAL YOUSSEFI
INDUSTRY LEAD ENTERPRISE BLOCKCHAIN
SWISSCOM BLOCKCHAIN AG
kamal.youssefi@swisscom.com

ZÜRICH, JANUARY 14, 2019

ABSTRACT

With the ever-decreasing block reward for Bitcoin miners, transaction fees are getting more and more attention since they will be a key factor in guaranteeing the network security in the future by incentivizing the miners. However, the research on this topic is still fairly underdeveloped. In this paper we give an overview of different fee estimation algorithms for Bitcoin and compare them using a prediction score. We observe that many algorithms require a substantial warm-up time and are therefore not always suited. For this reason, we introduce a new algorithm based on the current memory pool of pending transactions without the need for a warm-up. We show that the algorithm performs well for short-term estimates and conclude that more elaborate algorithms should be used once the warm-up is complete. Additionally, we give an outlook on coin-selection algorithms and the influence transaction fees are having on them. Our research provides a basis for understanding and further improving fee estimation algorithms as well as coin selection algorithms.

TABLE OF CONTENTS

PREFACE

This section serves as an overview over the context of the paper, its aspirations and relevance.

A. Author

At the time of writing the author of this paper is employed at "Swisscom Blockchain AG", a Swisscom venture with the goal to get Switzerland up to speed in the blockchain space. The author is part of a SCRUM team developing a digital asset custody product for financial intermediaries to send Bitcoin and other crypto-currencies. The problem of estimating and minimizing transaction fees is part of the solution and caught the author's attention.

B. Scope

The scope of this work changed during the elaboration from coin selection in Bitcoin to fees in Bitcoin and their influence on the coin selection. Therefore, the title of the paper was changed from: *"Coinselection - Input selection strategies for sending a Bitcoin transaction"* to *"Fees in Bitcoin coin selection"*.

C. Questions

The following questions should be answered in this paper:

i. Which factors contribute to the current transaction fee?
ii. How can the transaction fee for sending Bitcoin be determined considering the current mempool of pending transactions?
iii. Which influence does the fee have on the selection of coins?

D. Audience and Boundaries

The topics of this paper are rather advanced in the vast domain of blockchains hence basic knowledge about blockchain technologies is assumed. The following sources proved to be valuable in understanding the Bitcoin ecosystem and are recommended in case of uncertainties:

- the "Bitcoin Whitepaper" of Satoshi Nakamoto [1]
- the book "Mastering Bitcoin" by Andreas Antonopoulos [2]
- the Bitcoin Wiki: https://en.bitcoin.it/wiki/Main_Page
- Glossary at the end of this paper

This term paper focuses exclusively on fees and transactions in UTXO-based crypto currencies and therefore will not cover account-based implementations, e.g. known from Ethereum.

I. Introduction

Blockchain-technologies such as Bitcoin [1] and other crypto currencies like Ethereum [3] or ZCash [4] are currently entering a more mature stage and are already being spun out into commercial applications [5]. However, the user experience is still lacking in several areas. One crucial point regarding payments is the optimization of transaction fees. The foremost problems in Bitcoin are the facts that fees are difficult to predict[1] and have skyrocket in the recent past to over 30$ per transaction[2]. The main practical problems that confront us are the facts that *"A fee estimator is essentially trying to answer a question about the future based on an incomplete picture of the past"* [6] and that the concrete wallet implementations in many cases implement the basic functionality such as *"estimating fees"* or *"selecting coins"* in a suboptimal naïve way. In this paper, approaches to estimate fees and select coins for the Bitcoin network are challenged and improvements are discussed. Specifically, a method for estimating transaction fees based on the approach of the Bitcoin Core[3] and btcsuite[4] is analyzed and compared to a naïve algorithm based on go-ethereum's suggest gas price method. The problem with these approaches is that they need a certain warm-up time (e.g. btcd needs at least 30 minutes). In order to improve that, we introduce a new fee estimation algorithm based on the current state of pending transactions. The algorithm provided is compared to existing algorithms.

In the context of Swisscom Blockchain AG this paper provides a valuable research for a product currently in development used for financial intermediaries to send Bitcoin. The algorithms and findings discussed in this paper should subsequently be implemented and reflected for different financial products of Swisscom Blockchain AG.

The rest of this paper is structured as follows: Section II provides an overview of the Bitcoin Blockchain and the main theoretical concepts needed for the context of this paper. Section III presents previous work in this area of research. Section IV introduces different fee estimation algorithms. The results are reflected in section V and section VI and VII conclude the paper and identify areas for future work.

II. Bitcoin

Bitcoin is best-known as the world's leading cryptocurrency. But rather than simply being an electronic currency, it also incorporates the protocol, the network and the software itself. It was introduced 2008 by the unknown person Satoshi Nakamoto [1] and has since had a major impact on the digital era [7]. Bitcoin is implemented as a decentral peer-to-peer network aimed at creating consensus in an adversarial system for which it applies an approach called *Proof of Work*. Bitcoin is built on top of an append-only ledger - the *Blockchain* - which is shared amongst all users. This ledger is an accurate representation of all users' claims to Bitcoins.

The following section will introduce the Bitcoin ecosystem based on the whitepaper of Satoshi Nakamoto [1], the book "Mastering Bitcoin" of Andreas Antonopoulos [2] and the introduction of Mark Erhardt [8].

A. Blockchain and Mining

The Bitcoin blockchain is made up of blocks containing a list of transactions. The blocks are chained together by referencing the cryptographic hash of the predecessor (parent block). To achieve consensus on the order of transactions the network designates a random node to extend the blockchain by one block approximately every ten minutes. In order to do so, a subset of nodes, called *miners*, continuously search for new block candidates by trying to solve a hash-puzzle which consists of applying a cryptographic hash function to different block candidates until the resulting hash is lower than a global difficulty level.

[1] https://www.coindesk.com/bitcoin-low-fees-why-happening-why-matters [Last access: 07.01.2019]
[2] https://bitinfocharts.com/comparison/bitcoin-transactionfees.html#1y [Last access: 07.01.2019]
[3] https://bitcoin.org [Last access: 12.10.2018]
[4] https://github.com/btcsuite/btcd/blob/master/mempool/estimatefee.go [Last access: 07.01.2019]

Once a solution is found by a miner, a new block is discovered and can be broadcasted to the peers. This process of reaching consensus is commonly denoted as *mining*. Wood refers to it as follows: *"Mining is the process of dedicating effort (working) to bolster one series of transactions (a block) over any other potential competitor block. It is achieved thanks to a cryptographically secure proof."* [9]. It is important to note that two blocks may be found at the same time which could lead to conflicting transactions. To ensure that only one set of transactions is agreed upon, the longest chain of blocks is considered to be valid. To incentivize miners, Bitcoin pays a block reward for finding new blocks (e.g. 12.5 BTC at the time of writing) which is decreasing over time[5]. Furthermore, a miner receives the fees of the processed transactions.

B. Transactions

Transactions in Bitcoin are the mechanism of transferring value to other users. Once signed by the user the unconfirmed transaction gets broadcasted to the network. Each node validates the properties of the transaction and adds it to its memory pool of unconfirmed transactions (mempool). As soon as a miner finds a new block including the transaction, the transaction is confirmed. Every node updates the mempool accordingly and the value transfer can be considered to be settled. However, a user is advised to wait another 60 minutes (6 blocks) to be sure that the chain including the block with the transaction will remain in the longest chain. Transactions are composed of outputs of previous transactions called *unspent transaction outputs* (UTXO).

C. UTXOs and coin selection

UTXOs are used as inputs for a payment (transaction) and the associated fees. UTXOs are often referred to as *coins*. When a user wants to send Bitcoin, she has to choose exactly which "inputs" (UTXOs) to use. This process of choosing UTXOs as transaction inputs is called *coin selection*. Furthermore, a user has to determine how much she is willing to pay for the transaction to be processed contemporary (refer to Transaction Fees). The selection of coins is in most cases implemented by the wallet. If this selection is done right, the UTXO management can be simplified as well as the transaction fees can be reduced. However, the problem of choosing the right parameters is considered NP-hard which means that if the user has many coins it is not efficiently solvable. Additionally, the goals for choosing the right coins are conflicting to a certain extent. The different goals include:

- **low transaction fees**
- **avoidance of dust:** If a UTXO is too small, it is considered dust which means it is not worth spending anymore.
- **reduction of the UTXO pool**: every UTXO has to be stored on the blockchain. If the UTXO pool grows, additional data has to be stored on the blockchain, which automatically leads to higher storage costs for the network participants, particularly for the full nodes[6], which store all the information.
- **privacy:** every transaction includes a certain amount of change which is going back to the sender. If the sender is using a hierarchical deterministic wallet[7], the change address is different to the sending address for the sake of privacy. However, if the algorithm for selecting coins is always producing the same output, it is inherent which output of a transaction is the change and therefore the change address can be associated with the sender which leads to a lack of privacy.

For example, the avoidance of dust might conflict with low transaction fees.

D. Currency and Wallets

Bitcoin's currency is as well called *Bitcoin* and is identified by the shortcut "BTC". It can be subdivided into smaller units called Satoshi where $1\ BTC\ = 1 * 10^8\ Satoshi$. For sending Bitcoin, a user has to

[5] https://en.bitcoin.it/wiki/Controlled_supply [Last access: 07.01.2019]
[6] https://en.bitcoin.it/wiki/Full_node [Last access: 07.01.2019]
[7] https://en.bitcoin.it/wiki/Deterministic_wallet [Last access: 07.01.2019]

possess a so-called *wallet* which is a software application providing a gateway to the Bitcoin system. The wallet holds the keypairs to one or multiple addresses and is able to create and broadcast transactions on behalf of the user. Therefore, the wallet controls access to the user's money. When a wallet is created at least one private key is generated from which a public key and an address is derived. The private keys are used to prove ownership of the UTXOs a user holds, and addresses are pseudonymous identifiers of the users in the network which are used to designate the recipient of a transaction.

E. Transaction fees

In the beginning of Bitcoin, transaction fees were optional and considered a donation to the miners[8]. This was implemented by the wallet developers to whatever value they thought was appropriate. With the increasing demand however the transaction fees were soon a mandatory part of a transaction.

Bitcoin treats fees implicitly meaning that the fee is not explicitly stated when creating a transaction. The fee is defined as the unassigned remainder of the difference of inputs and outputs of a transaction (see Definition 1). The fees may be claimed by the miner of the confirming block.

$$transaction\ fee := \sum inputs - \sum outputs$$

Definition 1 - fee calculation (remainder of the unassigned difference of values)

In comparison to credit card payments the transaction fee in the Bitcoin network does not relate to the amount of Bitcoins sent but rather depends on the following two factors:

- The **fee-per-byte ratio** in the mining network at time t,
- the **size** of the transaction

a. Fee-per-byte-ratio

Once issued, a transaction must be included in a block. Since block space in the Bitcoin blockchain is a limited resource, Bitcoin has a fee market to ensure this block space is granted to those who value it the most, which means the ones who pay the most for it [7, p. 127]. This leads to a dynamic competitive market with changing fees based on capacity. An unconfirmed transaction gets added to the mempool which can be compared to a decentralized public sale - users can place bids for block space in the form of transactions with a fee attached and miners will place them in the next block based on this fee. Assuming a miner prioritizes unconfirmed transactions with higher fee-per-byte-rates, this results in a transaction to be processed faster by the mining network since it gives more incentive to do so. Furthermore, the transaction fees are an incentive for the miners to participate in the network as well as a protection against malicious abuse such as Denial-of-Service attacks or spam.

b. Transaction size

The second crucial factor in calculating the transaction fees is the actual transaction size measured in bytes which depends on the number of inputs and outputs of a transaction and the scripting language[9] the inputs and outputs are protected with.

F. Fee estimation

Fee estimation describes the process determining a fee-per-byte rate which can consecutively be used to create a new transaction. Hence fee estimation is part of the payment process in the Bitcoin network. The fee estimation is in most cases implemented by the wallet application. The goal of the fee estimation is to determine how much to pay for a transaction without constantly overpaying the network but still for a transaction to be processed contemporary. For high frequency wallets a good prediction on the current market price (fee-per-byte-rate) can lead to substantial savings over time.

[8] http://bitcoinfees.com [Last access: 07.01.2019]
[9] https://en.bitcoin.it/wiki/Script [Last access: 07.01.2019]

Fee estimation is a challenging problem (also refer to: [6]) due to the following facts:

- **supply is unpredictable**: Approximately every 10 minutes a new block is found in the Bitcoin network having space for 2MB of transactions. But because of the Poisson distribution of the block discovery it can also be 7 seconds or 45 minutes in one of hundred times [6]. In the first case the mempool is suddenly emptied of high fee-paying transactions in the second case the mempool gets filled with higher fee-paying transactions.
- **demand is unpredictable**: There are certain patterns when it comes to transaction flow (e.g. during the week there are more transactions). However, it is difficult to predict the short-term demand.
- **different requirements for different users**: some users might want a transaction to be confirmed as fast as possible others might want it to confirm within the next 6 hours or even later which means that there exists no one-size-fits all approach for the speed in which a transaction should be picked up rather should the user be able to determine this value (e.g. safe low, standard, fast).

The data used for estimating fees can come from various sources, such as:

- current mempool: the current state of pending transactions
- history of blocks: the history of recent prices
- transaction flow: the time transactions needed to be confirmed for a given estimate

It is worth mentioning that neither a full node nor a third-party service can guarantee having an exhaustive list of pending transactions which is due to the nature of Bitcoin being a decentralized network. Transactions are being broadcasted through the network in a peer-to-peer manner. Consequently, some latency might exist.

III. Related Work

The building blocks for Bitcoin as well as for many of the major blockchains that exist today were introduced by the whitepaper of Satoshi Nakamoto in 2008 [1]. This paper shows a way to build a digital currency without relying on a trusted third party. Antonopoulos provides in his recently published book [2] a comprehensive introduction to the Bitcoin ecosystem.

Since the blockchain technology is quite young, there is still a certain lack of profound literature in certain domains such as the subject of fee estimation or coin selection.

A. Fee estimation

Huoy examines the economics of Bitcoin transaction fees based on a static equilibrium model [10]. He observes that the security of Bitcoin heavily relies on the mining power which depends on the fees getting higher over time as the mining reward gets smaller. In a similar way, Huberman et al. [11] analyze the economics of Bitcoin and capture their findings in a simplified economic model that provides answers to the underlying model for the determination of fees. They point out that *"transaction fees and infrastructure level are in fact determined in an equilibrium of a congestion queueing game derived from the system's limited throughput"* [11, p. 1]. Most importantly, they show that such a system requires congestion to raise revenue and fund infrastructure. Li et al. [12] analyze the transaction queuing game of Bitcoin in prospect of the decreasing block reward and its influence on the transaction fees. They conduct a theoretical analysis on a queuing game with non-preemptive priority and retrieve five types of Nash equilibria of the game, showing that the equilibrium transaction fees are determined by the waiting time and time cost of a user. They give an outlook on what they plan to do in their future work including a more realistic approach based on ACP (Artificial Systems + Computational Experiments + Parallel Execution) to study the transaction fee problem in a more realistic setting.

B. *Coin selection*

Delgado-Segura et al. [13] introduce STATUS a tool to analyze the set of UTXOs in Bitcoin. In particular they provide a detailed analysis of the set including the proportion of dust (unprofitable UTXOs), the distribution of the block height in which outputs were included and the use of non-standard outputs. Pérez-Sola et al. pick up this work and show in [14] a detailed analysis of the current UTXO set and give an in-depth overview on the proportion of dust by providing an additional measure of unprofitability. Furthermore, they compare the three currencies Bitcoin, Bitcoin Cash and Litecoin in terms of the UTXO set. They show that 35-45% of Bitcoin's and Bitcoin Cash's and 67% of Litecoin's UTXOs are in fact economically not worth spendable and conclude that the actual state of the UTXO set is far from ideal for the three currencies.

Erhardt provides in his master thesis [8] an overview of different coin selection algorithms and introduces a new algorithm based on the knapsack-solving strategy "Branch and Bound" which was recently adopted by Bitcoin Core as the main coin selection algorithm[10]. He uses a simulation based on transaction data of the moneypot[11] hot wallet to measure the performance of the different algorithms and shows that for simple wallets random selection of coins produces reliable but not optimized results. His newly introduced algorithm outperforms the other algorithms in almost all aspects. Erhardt points out that the performance of the algorithms heavily relies on the simulation scenario and that it is difficult to provide a one-size-fits-all solution.

One of the tough challenges for researches in this domain is the difficulty of simulating the underlying system. Bitcoin embodies many complex concepts such as different scripting languages[12], multiple change addresses[13] or segwit[14] that make a simulation of the real-world environment virtually unaffordable. Furthermore, the network is used by different users with different requirements which makes it difficult to gather data for one particular use case since only looking at the blockchain does in most cases not reveal the users' intention and background.

IV. Fee Estimation Algorithms

The following section introduces different algorithms for estimating fees.

A. *A naïve approach*

The following algorithm (see Algorithm 1 in the appendix) is a basic fee estimation algorithm that makes use of the data of the latest block and is inspired by the go-ethereum *suggest gas price* method[15]. The inputs and outputs for every transaction of the last block are retrieved and the respective fee-per-byte-rate is calculated. The fee-per-byte-rates are sorted, and a fee rate based on a percentile is selected (a percentile of 50 means the median fee rate of the last block is used). To get a safe estimate a percentile value of 60 is used.

The algorithm is resistant to adversarial conditions - for example, against users paying exorbitant fees or from outlier data (e.g. outliers caused by programming errors in a wallet implementation) - since it does not take the average fee but relies on a value close to the median.

Analysis

However, the algorithm has the following limitations:

- it relies solely on historical data, meaning that it does not consider the current state of pending transactions

[10] https://github.com/bitcoin/bitcoin/pull/10637 [Last access: 08.01.2018]
[11] https://www.moneypot.com [Last access: 08.01.2018]
[12] https://en.bitcoin.it/wiki/Script [Last access: 08.01.2018]
[13] https://en.bitcoin.it/wiki/Address [Last access: 08.01.2018]
[14] https://en.wikipedia.org/wiki/SegWit [Last access: 08.01.2018]
[15] https://github.com/ethereum/go-ethereum/blob/master/eth/gasprice/gasprice.go [Last access: 28.11.2018]

- it does not track how long the transactions have been in the mempool before they were included in the block
- it only includes data of one block meaning that a misbehaving miner can easily influence the fee estimation by stuffing a block with high fee-rate private transactions
- it does not provide a measure on how fast a transaction with the given estimate will take to be confirmed nor a way of controlling this parameter
- if everyone would be using this algorithm fees would go up over time, since a value above the median is used
- running the algorithm apart from a full node is quite slow due to the nature of the Bitcoin blockchain which requires that for every transaction input the corresponding transaction generating that input is queried (e.g. if a block contains 2000 transactions, at least another 2000 requests for the inputs of these transactions have to be made or more if transactions have multiple inputs)

B. Bitcoin Core's approach

The Bitcoin Core fee estimation algorithm is based on historic data as well as the current mempool. The algorithm has been improved various times in the recent releases of Bitcoin Core[16] and is used by the rpc method *estimatesmartfee*[17]. The algorithm requires one input parameter to specify the number of blocks within which a transaction should be included in the blockchain and returns a fee rate that should achieve this target.

Under the hood [6] it works by grouping transaction fee rates into buckets where each bucket consists of a range of fee rates. The buckets are exponentially-spaced from 1 Satoshi per Byte to 10'000 Satoshi per Byte. Additionally, the algorithm keeps track of the target representing the number of blocks between a transaction entering the mempool and being accepted in a block as well as various low-level subtleties which are documented in the code itself[18] such as thresholds for making sure enough data is available. For any target-bucket pair the algorithm finds the probability that a transaction would have been included within the target number of blocks. Furthermore, it makes use of an exponentially weighted moving average to weight more recent blocks over older blocks which means that a block in the past doesn't influence the estimate as much as a more recent block. The estimation is done by looking at the lowest bucket with a probability of being confirmed within the target number of blocks of at least 95% and taking the median fee rate of that bucket.

Estimates are being tracked on three different time horizons (short: target 12 blocks, medium: target 48 blocks, long: target 1008 blocks) to adapt more quickly to changes in conditions and allow targets that are further in the future. A user can additionally specify if the estimate should be economical or conservative. Economical estimates are designed to be lower during periods of low transaction activity but may lead to unconfirmed transactions if the fees increase rapidly. In the case of conservative estimates longer time horizons are used which are less susceptible to rapid changes in fee conditions (e.g. temporary network congestions).

Analysis:

This algorithm is rather complex and entails various tuning parameters. The user is able to control the confirmation of a certain estimate.

However, the algorithm has the following limitations:

- it requires a full node to have access to all the required data
- it has a substantial warm up time during which a default fee is returned

[16] https://bitcointechtalk.com/whats-new-in-bitcoin-core-v0-15-part-2-41b6d0493136 [Last access: 19.12.2018]
[17] https://bitcoincore.org/en/doc/0.16.0/rpc/util/estimatesmartfee/ [Last access: 19.12.2018]
[18] https://github.com/bitcoin/bitcoin/blob/master/src/policy/fees.h [Last access: 19.12.2018]

C. *BTCSuite's approach*

Under the collective name BTCsuite various tools related to Bitcoin implemented in go are available one of which is btcd an alternative implementation of the Bitcoin protocol. The fee estimation algorithm of btcsuite keeps track of how long it takes for a transaction in the mempool to be included in a block. It can be seen as a much more simplified version of the Bitcoin Core algorithm. The algorithm is considered in this work to make statement about the complexity of the Bitcoin Core algorithm and its justification or if a much simpler algorithm can produce equally good predictions. Note that such a statement only holds for short term predictions since the btcsuite algorithm only allows predictions with a target of up to 25 blocks.

Analysis:

Similar to Bitcoin Core's algorithm, the user is able to control the confirmation of a certain estimate by specifying a target of blocks.

However, the algorithm has the following limitations:

- it requires a full node or a smart client to have access to all the required data
- it has a substantial warm up time (at least 3 blocks) during which an error is returned
- it does not track long term stats, nor does it allow conservative estimates

D. *New Mempool-based approach*

The limitations of the aforementioned algorithms include that they need some warm-up time by inspecting the history and flow of transactions. In a productive scenario this might be an issue. Therefore, we propose the following naïve algorithm based on the current pool of pending transactions (see algorithm 2 and figure 6 in the appendix).

Using the information from the current mempool the algorithm estimates a fee rate with the goal to be included in the next few blocks without overpaying the miners. It does so by making timing assumptions about the block discovery. Firstly, it retrieves the average number of transactions per block for the past few blocks, then it estimates the current progress in block discovery. Secondly, the fee rates are extracted from the mempool and sorted in ascending order. Only the top x rates are inspected where x is equal to the average number of transactions of the past blocks. Thirdly, a percentile based on two constant parameters and the computed block discovery progress is calculated. Resulting in a value that is between the constant parameters *Percentile* and *Percentile - Range* depending on the estimated progress of block discovery.

> *For example, if the latest block has just been found the diff between now and the time mined is 0, in this case a percentile of* $80 - 0.0 * 60 = 80$ *would be used, if the diff is 10 or more minutes a percentile of* $80 - 1.0 * 60 = 20$ *would be used. This is to make sure that transactions arriving at a later point with a higher fee are anticipated.*

Lastly, the fee rate from the top x rates at this percentile is taken as the proposed estimate.

Analysis:

One of the major advantages of this algorithm is that it can be recalculated for every estimate without having to rely on previously gathered data. Furthermore, it can quickly adapt to a changing environment (e.g. due to network congestion).

- Miners are already working on the next block which means that the top x transactions will be included most probably in the next block.
- The wait time is at least 10 minutes in which many transactions may come in
- As pointed out before, the block discovery follows a Poisson-distribution which means that there might be lucky runs of blocks and slow runs of blocks, signifying that one in hundred blocks is only found after 45 minutes.

E. Prediction Scores

In previous work we introduced a measure to determine the goodness of a certain estimate [15] and adapted their use for Bitcoin in this work. The *prediction score* determines the percentage of transactions having a bigger fee per byte ratio in a certain block. 0 means that all transactions in the block have a lower ratio than the estimate, 100 means that all the transactions have a bigger ratio. Consequently, a lower score means more chance for a transaction using an estimate to be included in a certain block. We will be reusing this score to evaluate the estimates in this work. See algorithm 3 in the appendix for more details.

V. Evaluation of Fee Estimation Algorithms

The data to carry out the fee rate estimations was retrieved using a go[19] application querying a previously setup Bitcoin node which was allowed to fully synchronize with the network. The application makes repeated calls to the JSON rpc interface of the Bitcoin Core blockchain client to retrieve the latest blocks, information about transactions and the current state of the mempool. The introduced algorithms were (re-)implemented in a go[20] application except for the Bitcoin Core algorithm for which the respective json rpc interface of the Bitcoin Core node was used. Subsequently, the estimations and prediction scores were analyzed using a python script. In this section we examine the results of the analysis and provide a comparison of the algorithms. The sampling of the algorithms was carried out over a timespan of 82 blocks[21]. During this period for every block a fee-per-byte-rate prediction was made using the different algorithms. The values used for targets using the Bitcoin core and the btcsuite algorithm are the following:

- Economical: a target of 10 blocks
- Standard: a target of 6 blocks
- Fast: a target of 2 blocks

A. Estimations

The following illustration shows the predicted fee-per-byte-rate in Satoshi for the sampling period. The btcutil (btcutil) and the Bitcoin Core algorithm (core) appear to be rather constant whereas both naïve algorithms - the one based on go-ethereum's suggest gas price method (denoted as naïve) and the newly introduced mempool-based algorithm (mempool) - are showing high fluctuations.

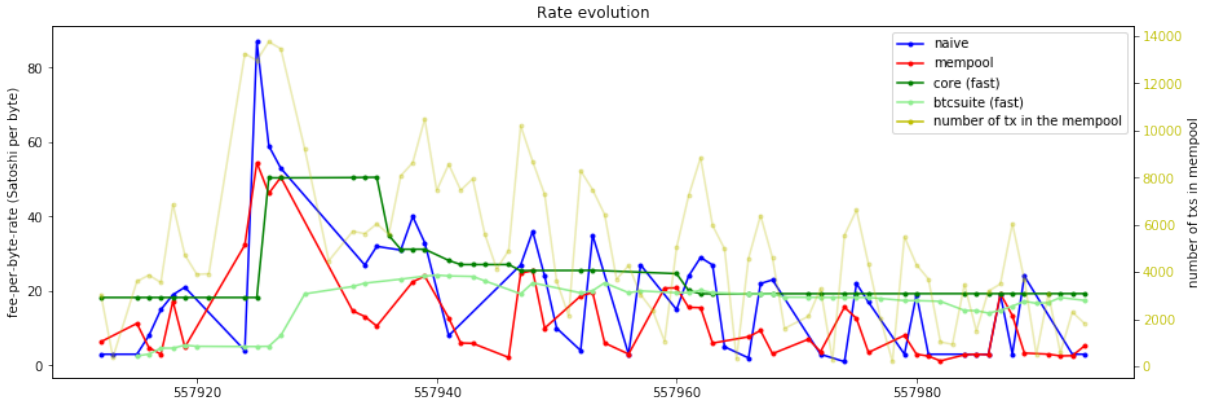

Figure 1 - fee rate evolution

It can be observed that the peaks of the naïve algorithm apparently follow the peaks of the number of transactions in the memory pool (yellow line) and also for the mempool algorithm such a tendency is visible. It was soon clear that due to the fact that the naïve as well as the mempool algorithm try to get an estimate for a transaction to be processed contemporary, the estimates could only be compared to the

[19] https://golang.org [Last access: 19.12.2018]
[20] See Appendix for the link to the repositories
[21] block height 557912 - 557994

fast estimates of the Bitcoin Core and the btcsuite algorithm. In appendix B further rate evolutions are displayed.

Table 1 shows an overview of the different average fee rates and gives an indication of how big an actual transaction fee would be for a standard transaction for an exchange rate of 3652.91$ per Bitcoin.

	naïve	mempool	Bitcoin Core (fast)	Bitcoin Core (standard)	btcsuite (fast)	btcsuite (standard)
average fee-rate (Satoshi per byte)	18.5	16.87	24.573	7.724	16.508	3.256
std. deviation of average fee-rate (Satoshi per byte)	17.429	13.966	9.628	6.653	6.1	1.254
average fee for a transaction[22] 0.169$	0.154$	0.224$	0.071$	0.151$	0.03$	

Table 1 - overview average fee rates

B. Prediction Scores

To investigate the goodness of a fee rate estimation the prediction scores [15] were calculated for every estimate for the next 10 blocks. Figure 2 and 3 show the percentage of the next 10 blocks in which a given estimate would not have been included. Inclusion was defined as 5% of transactions of a block

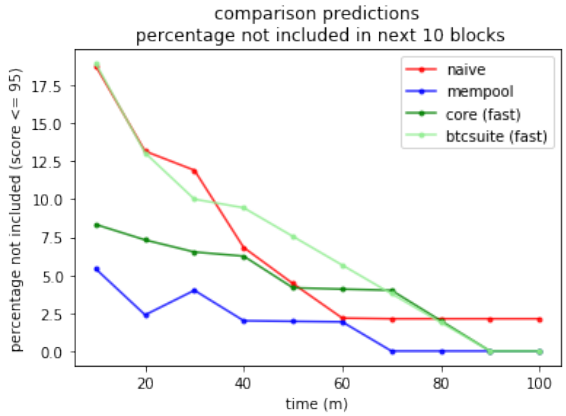

Figure 2 - comparison of prediction scores for fast rates (percentage scores < 95%)

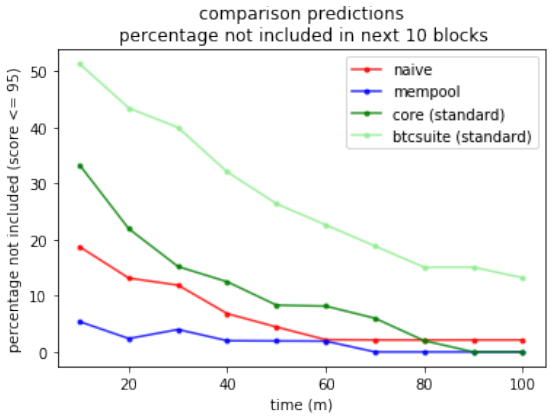

Figure 3 - comparison of prediction scores for standard rates (percentage scores < 95%)

having a fee rate below or equal to the estimate which means that transactions with that estimate would have been included and would have made out at least 5% of the transactions in the block. It is important to mention that this means that certain transactions with the predicted fee-per-byte-rate would have been included but it does not indicate the exact amount of transactions. Thus, there is no guarantee that a particular transaction would have been included

According to figure 2, the mempool algorithm produces estimates that in the next block would have been included in ~95% of all cases and in 70 minutes in all cases. The inclusion of the naïve algorithm is worse, with an inclusion of ~80% in the next block and roughly ~2% of estimates that would not even have been included after 60 minutes. Similarly, the btcsuite algorithm shows an inclusion of about 80% for fast estimates for the next block and an inclusion of 100% after 90 minutes. The fast estimates retrieved from the Bitcoin core algorithm show an inclusion of 92% for the next block and have the same point of an inclusion of 100% as the btcsuite algorithm. For standard rates (fig. 3) the Bitcoin core algorithm clearly outperforms the btcsuite algorithm with an in inclusion of 90% after 50 minutes whereas the btcsuite algorithm only has an inclusion rate of roughly 80% after 100 minutes.

[22] $fee = feeRate * size$ where transaction size: 250 bytes and exchange rate: 11.01.2019 20:50: 1 BTC = 3652.91 USD https://www.xe.com/currencycharts/?from=XBT&to=USD

Comparing these observations to the average fee rates of the algorithms in table 1 hints that the differences between the Bitcoin core algorithm and the btcsuite algorithm can be accounted to the amount of the rate which show that for both standard and fast rates the Bitcoin core algorithm provides higher rates.

Figure 2 indicates that the mempool algorithm performs well while also having the second lowest average fee-rate. This means that the simple mempool-based estimation algorithm provides relatively good short-term results which are clearly better that the results of the naïve estimation algorithm.

C. Coin-Selection

Previous research commonly treated the fee rate as a never changing constant in their coin selection algorithms [8]. But actually, the fee rate fluctuates depending on the network usage which is confirmed by the previous findings in this paper. As was observed by Pérez-Soì et al. [14] a substantial portion of the UTXO pool (between 35-45%) is actually not worth spending for a fee rate of 100 Satoshi per byte. However, based on different fee rates different UTXOs are labeled as dust. The same analysis as in [14] was carried out again to inspect the current set of UTXOs in Bitcoin by using their tool called STATUS[23].

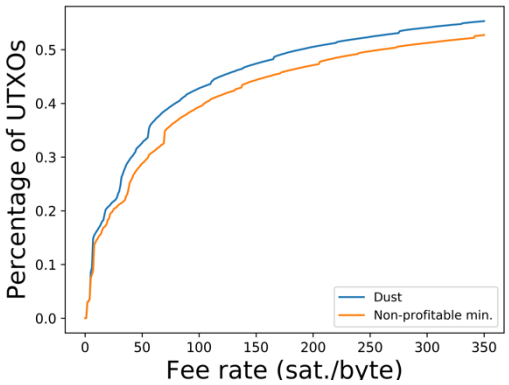

Figure 4 - Dust percentage of UTXOs by fee rate retrieved in this work (Bitcoin height 557947)

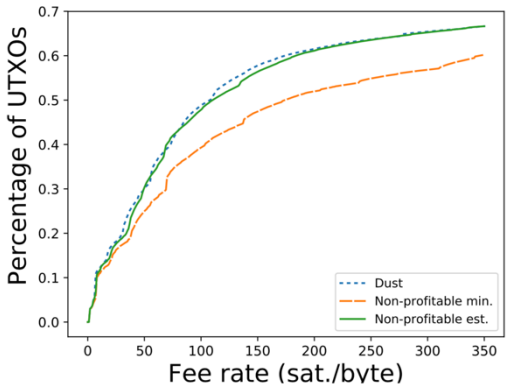

Figure 5 - Dust percentage of UTXOs by fee rate by Pérez-Soì et al. [14]

The findings of Pérez-Soì et al. [14] that between 35-45% of all UTXOs are dust for a fee rate of 100 Satoshi per byte are confirmed by figure 4 which shows that the proportion of dust is still comparable to their results and has neither increased nor decreased significantly. The fact that the curve has a rather steep slope means that depending on the current fee rate which is often between 3 and 40 Satoshi per byte (see rate evolutions in appendix B) either 8% or 30% of the UTXOs are labelled as dust. This disparity suggests that coin selection algorithms should also consider the changing fee rates and be smart enough to manage UTXOs based on the current fee rate. Ideally, such an algorithm would keep track of which fee rates are common and select UTXOs accordingly. This could for example be done using a measure of "*how worth spendable*" a UTXO for a given fee rate is (not just by labelling it as dust or not) and probably delay the spending based on this factor. More concretely, this measure could be included as an additional criteria in the knapsack-solver of the Bitcoin Core coin-selection algorithm introduced by Erhardt [8].

VI. Discussion

In this paper we have analyzed three different algorithms for estimating fee rates in Bitcoin based on a naïve approach using an adapted version of the suggest gas price method from go-ethereum, the advanced and widely used Bitcoin Core algorithm and an algorithm originating from btcsuite implemented in their Bitcoin node software btcd. We observed that each of the algorithms need a certain warm-up time but provide good estimates with the Bitcoin Core algorithm outperforming the other algorithms in terms of the time-cost tradeoff and also by allowing to specify a much higher target of

[23] https://github.com/sr-gi/bitcoin_tools/tree/master/bitcoin_tools/analysis/status [Last access: 11.01.2019]

blocks for the desired inclusion of a transaction. To overcome the shortcomings of the warm-up time we introduced a new algorithm based on the current pool of pending transactions and showed that the algorithm provides useful results in times where there is no immediate estimation from a more elaborate algorithm available. However, the algorithm does not allow a planned delay of a transaction which suggests that once the warm-up is complete a more elaborate algorithm (e.g. the Bitcoin Core algorithm) should be used for a desired target of blocks.

From the review of the fee estimation algorithms, the following findings emerge:

- **Complexity**: Many of the existing fee estimation algorithms such as the Bitcoin Core algorithm are rather complex and not formally documented. They make use of various tuning parameters which are most probably retrieved via trial and error.
- **Warm-up**: Many fee estimation algorithms need a warm-up time and either return an error or a default value during that time.
- **Need for a full node**: It proved difficult to write fee estimation algorithms without directly integrating or extending the software of a full node. A better solution could be to write a software to index the blockchain and then use an interface (preferably abstracting the complexities) to the provided data. This goes in line with the fact that the Bitcoin blockchain is growing constantly (currently the full blockchain takes up to 240GB of disk space) which makes exhaustive analyses very time consuming.
- **Different use cases**: Different users have different requirements which on one hand means that the data on the blockchain reflects different realities and on the other that fee estimation algorithms need to be able to handle different requirements such as targets of more than one day. Both points should be considered when trying to build fee estimation algorithms and perform analyses.
- **Estimation instead of prediction**: All of the introduced algorithms use current or past data to retrieve an estimate and are not very smart when it comes to a statement about the future. Therefore, talking about a prediction is not fully correct. A prediction would rather include more assumptions about the future and might consider other factors such as the time of day or the current exchange rates to other currencies.
- **Changing dynamic of the fee market**: Especially, when fees become more important, due to the fact that the block reward for miners diminishes, the fee market will change which means that most probably existing algorithms will need to be reworked to adapt to the new dynamics.

Instead of blindly creating another coin selection algorithm, we became aware of the need of a more fundamental understanding of the factors that compose the coin selection and the influence that fee rates have. We therefore conducted another analysis on the current set of UTXOs and verified prior studies in that field. We speculate that due to the fact that many coin selection algorithms treat the fee rate as a constant value the UTXO management is suboptimal and propose a measure on how to improve this drawback. The implementation and verification of these assumptions are part of future work. One of our key findings is that current implementations of coin selection algorithms were realized without properly testing their assumptions in a real-world scenario. Even the research from Erhardt [8] which proved to be the only scientific paper in that regard made use of one single data set which was retrieved from a hot wallet and could skew the results. This deficit may be explained due to the fact that the Bitcoin ecosystem is very complex and cannot be simulated properly without an enormous effort and indicates that there is still a lot to be investigated.

The approach for evaluating the estimates suffers from a very naïve way of calculating the prediction scores which do not indicate if a particular transaction would be included, only that certain transactions are included with a fee rate less or equal to the estimated fee rate. Future research should therefore further develop the concept of scores to include a probability that a certain transaction for a given

estimate will be included in a block. However, this requires that the mempool is monitored at every prediction.

An apparent limitation of the method is the relatively small sample size of roughly 80 blocks which made the differences between the algorithms clear but may exclude special cases such as the behavior in times of heavy network congestion and the performance during a longer time span. Another restriction of this paper involves that transactions using segwit[24] were not included in the naïve algorithm and the score calculation which may skew the results. This is due to the fact that segwit imposes technical difficulties when trying to retrieve the value of a UTXO. Therefore, future research should be conducted to get around this challenge and include transactions using segwit in the results.

The listed limitations above show that future investigations are necessary to validate the kinds of conclusions that can be drawn from this study and possibly should further improve the mempool algorithm as well as the prediction scores.

VII. Conclusion

In conclusion, the research done in this paper has only scratched on the surface of fee estimation. It has shown that predicting Bitcoin transaction fees has still a lot of potential but that the existing algorithms to estimate fee rates are already rather elaborate with the Bitcoin Core algorithm as the prime example. However, we acknowledge the fact that they were most certainly implemented without deep prior observation of the determining factors and systematical derivation of the tuning parameters. Our newly introduced mempool-based algorithm adds another piece to the puzzle of fee estimation by offering a solution that works without prior warm-up. We observe that coin selection algorithms commonly leave out the changing fee rates and show a possible way to improve this drawback. Despite all the limitations our results provide a solid basis for future research and implementations.

[24] https://en.bitcoin.it/wiki/Segregated_Witness [Last access: 11.01.2019]

GLOSSARY

The following Glossary is composed of the definitions of Erhardt [8] and the Bitcoin wiki[25]:

Term	Definition	Link
address	A hash representation of a user's public key.	https://en.bitcoin.it/wiki/Address
Bitcoin Core	The reference implementation of Bitcoin, currently, the defacto pro- tocol definition. Written in C++. Formerly known as Bitcoin-Qt. Maintained by the Bitcoin Core Developers.	http://github.com/bitcoin/bitcoin
change output	A transaction output that returns to the sender what's left after paying for spending target and fee	https://en.bitcoin.it/wiki/Change
doublespend	The act of authoring two transactions which both reference the same UTXO as input. Since UTXOs can only be spend once, the transactions are in competition and only one of them can be confirmed	https://en.bitcoin.it/wiki/Irreversible_Transactions
dust	A UTXO whose spending would cost more than a third of its own value in fees	https://en.bitcoinwiki.org/wiki/Cryptocurrency_dust
full node	A "full node" refers to Bitcoin clients that validate the complete transac- tion history of Bitcoin and enforce all rules in Bitcoin. Most full nodes store a complete copy of the Bitcoin blockchain and provide thin clients with lookup services.	https://en.bitcoin.it/wiki/Full_node
height	The blockchain height refers to the index of a block in the longest valid chain derived from the Genesis Block. The Genesis Block hereby is at height zero, while the fifth block after the Genesis Block is at height five.	https://bitcoin.org/en/glossary/block-height
input	A script that references a previous UTXO to be used to fund a transaction .	https://en.bitcoin.it/wiki/Transaction
memory pool	A node's storage of unconfirmed transactions. Memory pools be- tween different nodes may differ in content. A limit for the memory pool's size can be set with a mininum fee ration (fee per byte) to admit transactions, or a maximum size of memory set aside for it. When the memory pool is full it will evict the transactions with the smallest fee ratio to admit better endowed transactions.	https://en.bitcoin.it/wiki/Vocabulary#Memory_pool
miner	Depending on the context, either a person, an entity or a device that con- tributes computing power to secure the Bitcoin network.	https://en.bitcoin.it/wiki/Vocabulary#Miner
mining	The competitive process to determine the author of the next block. Re- sults in synchronization of the network's state and securing of the network's transactional history.	https://en.bitcoin.it/wiki/Mining

[25] https://en.bitcoin.it [Last access: 11.01.2019]

output	A script that designates an amount of bitcoins to be spendable by the owner of the recipient address, or an alternative authentication token.	https://en.bitcoin.it/wiki/Transaction
Satoshi	A Satoshi is the smallest subunit of a bitcoin. Bitcoins are divisible into 10e8 Satoshi, i.e. one Satoshi is a hundred millionth of a bitcoin.	https://en.bitcoin.it/wiki/Satoshi_(unit)
Segregated Witness	A Bitcoin Improvement Proposal to resolve transaction malleability and rectify inefficiencies, see BIP 141	https://en.bitcoin.it/wiki/Segregated_Witness
transaction	An order to the network to transfer bitcoins from UTXO referenced in inputs to newly created UTXO referenced in a list of outputs.	https://en.bitcoin.it/wiki/Transaction

Table 2 - Glossary

APPENDIX

The source code created for this paper can be found in the following repositories:

- Algorithms: https://github.com/swisscom-blockchain/bitcoin-feeestimator [Access on demand: Marius.Giger@Swisscom.com]
- Analysis: https://github.com/mariusgiger/bitcoin-feeestimator-analysis

A. *Algorithms*

Algorithm 1: naive fee estimation

input : A block B with transactions
output: The estimated fee-per-byte-rate e

```
1  Retrieve all transaction with inputs and outputs of this block;
2  feeRates ← [];
3  transactions ← B.Transactions;
4  for i ← 0 to transactions.length() do
5      Compute the fee-rate-per-byte for every transaction
6      inputSum ← Sum(transactions[i].Inputs);
7      outputSum ← Sum(transactions[i].Outputs);
8      fee ← inputSum-outputSum;
9      feeRate ← fee/transactions[i].Size();
10     feeRates.Add(feeRate);
11 Sort(feeRates);
12 percentile← 60;
13 index← (feeRates.length()-1)*percentile / 100;
14 e← feeRates [index];
```

Algorithm 1 - naive fee estimation

Algorithm 2: mempool-based fee estimation

input : Percentile := 80, Range := 60
output: The estimated fee-per-byte-rate e

```
1  avgTxNum ←getAvgTxsPerBlock()
2  pool ←getCurrentMempool()
3  lastMined ←getTimeLastMined()
4  diff ←time.now().sub(lastMined)
5  powProgress ← (1/10) * diff.minutes()
6  if powProgress > 1 then
7      powProgress ← 1
8  poolRates ←[]
9  foreach tx ∈ pool do
10     extract fee rates.
11     feeInSatoshi ← tx.Fee * 1e8
12     ratePerByte ← feeInSatoshi/tx.Size
13     poolRates.Add(ratePerByte)
14 poolRates.Sort()
15 idx ← poolRates.length() − avgTxNum
16 if idx < 0 then
17     idx ← 0
18 only consider rates starting at idx.
19 filteredRates ← poolRates[idx :]
20 verificationPercentile ← Percentile − Range * powProgress
21 targetIndex ← (filteredRates.length() − 1) * verificationPercentile/100
22 e ← filteredRates[targetIndex]
```

Algorithm 2 - mempool-based fee estimation

Algorithm 3: calculate prediction score

input : *block* := A block with transactions. *estimate* := An estimated fee-per-byte-rate.
output: *score* := The score for *estimate* for *block*.

```
block.Transactions ← block.Transactions.OrderBy(tx => tx.FeeRate)
for i ← 0 to block.Transactions.length()-1 do
    tx ← block.Transactions[i]
    if tx.FeeRate > prediction then
        score← (1.0 − (i/block.Transactions.length())) ∗ 100.0
        return
score← 0
```

Algorithm 3 - adapted prediction score for Bitcoin estimates

```go
func (e *Estimator) estimateFee() (float64, error) {
    info, err := e.client.GetBlockChainInfo()
    if err != nil {
        return 0, err
    }

    pool, err := e.mempoolCache.GetCacheAt(info.Blocks)
    if err != nil {
        return 0, err
    }

    avgBlockSize, lastMined, err := e.getAverageBlockSize(int(info.Blocks))
    if err != nil {
        return 0, err
    }

    diff := time.Now().Sub(lastMined)
    powProgress := (float64(1) / float64(10)) * float64(diff.Minutes())
    if powProgress > 1 {
        powProgress = 1
    }

    poolRates := e.getPoolRates(pool)
    sort.Float64s(poolRates)

    idx := len(poolRates) - avgBlockSize
    if idx < 0 {
        idx = 0
    }

    blockWindowRates := poolRates[idx:]
    verificationPercentile := float64(Percentile) - float64(Range)*powProgress
    estimate := blockWindowRates[(len(blockWindowRates)-1)*int(verificationPercentile)/100]
    return estimate, nil
}
```

Figure 6 - Implementation of mempool algorithm in Golang

B. Rate Evolutions

Figure 7-10 fee rate evolutions of the different algorithms

C. *Mempool evolution*

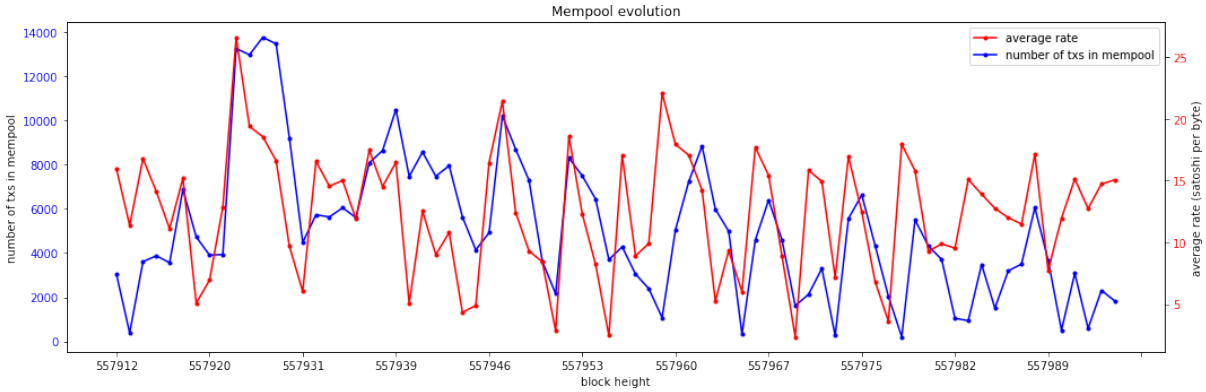

Figure 11 - Number of transactions in the mempool and average fee rate of those transactions

D. *Prediction scores*

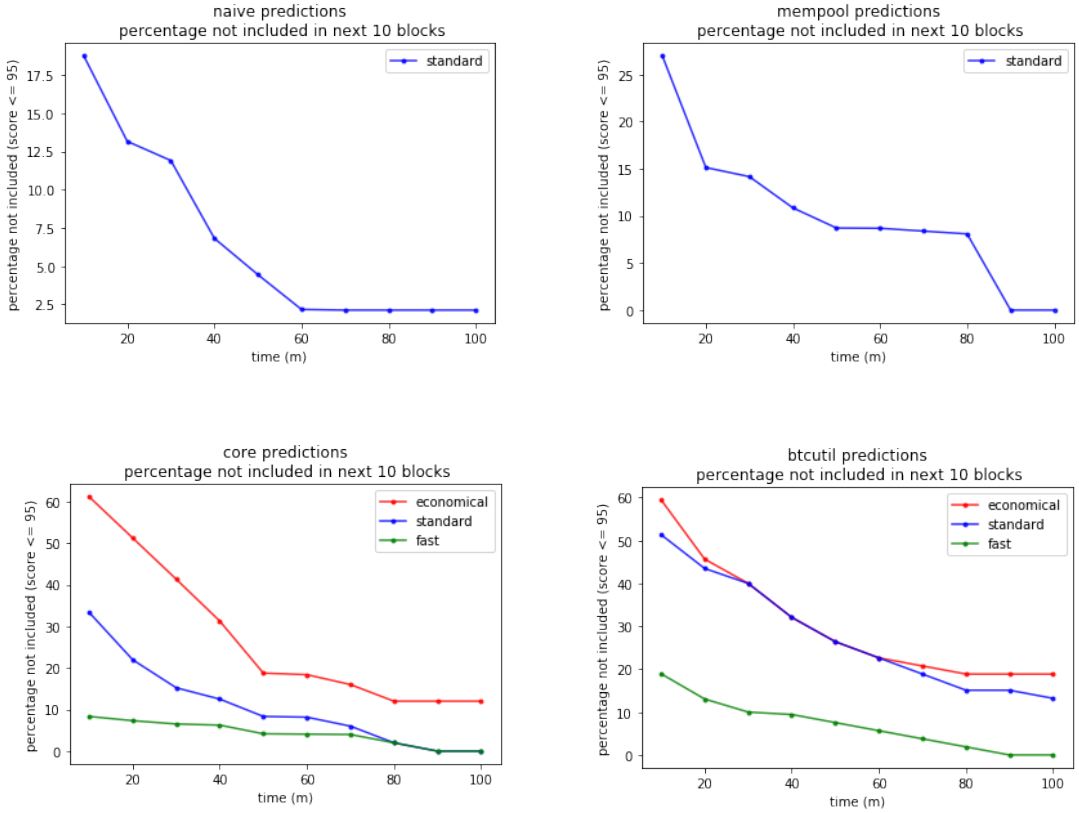

Figure 12-15 - Percentage of estimates without inclusion in the next blocks for the different algorithms

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system.,” 2008.

[2] A. Antonopoulos, *Mastering Bitcoin*, 2nd ed. 2017.

[3] V. Buterin, “A Next-Generation Smart Contract and Decentralized Application Platform,” 2014.

[4] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, “Zcash Protocol Specification,” pp. 1–53, 2017.

[5] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, “SoK: Research perspectives and challenges for bitcoin and cryptocurrencies,” *Proc. - IEEE Symp. Secur. Priv.*, vol. 2015–July, pp. 104–121, 2015.

[6] J. Newbery, “An introduction to Bitcoin Core fee estimation,” 2017. [Online]. Available: https://bitcointechtalk.com/an-introduction-to-bitcoin-core-fee-estimation-27920880ad0. [Accessed: 09-Jan-2019].

[7] K. Fanning and D. P. Centers, “Blockchain and Its Coming Impact on Financial Services,” *J. Corp. Account. Financ.*, no. 27.5, pp. 53–57, 2016.

[8] M. Erhardt, “An Evaluation of Coin Selection Strategies.” 2016.

[9] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger.,” 2014.

[10] N. Houy, “The economics of Bitcoin transaction fees,” 2014.

[11] G. Huberman, J. D. Leshno, and C. C. Moallemi, *Monopoly without a monopolist: An economic analysis of the bitcoin payment system.* 2017.

[12] J. Li, Y. Yuan, S. Wang, and F. Wang, “Transaction Queuing Game in Bitcoin BlockChain,” *2018 IEEE Intell. Veh. Symp.*, vol. 1, pp. 114–119, 2018.

[13] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomart\i, “Analysis of the Bitcoin UTXO set,” *Proc. 5th Work. Bitcoin Blockchain Res. Res. (in Assocation with Financ. Crypto 18), Lect. Notes Comput. Sci.*, 2018.

[14] C. Pérez-Soì, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, “Another coin bites the dust: An analysis of dust in UTXO based cryptocurrencies.” 2018.

[15] M. Giger, K. Youssefi, and U.-M. Künzi, “Predicting the Ethereum Gas Price,” no. January, 2019.