A User-Oriented Model for Oracles' Gas Price Prediction

Giuseppe Antonio Pierro, Henrique Rocha, Stéphane Ducasse, Michele Marchesi, Roberto Tonelli

A User-Oriented Model for Oracles' Gas Price Prediction

Giuseppe Antonio Pierro[a,b], Henrique Rocha[c], Stéphane Ducasse[b], Michele Marchesi[a], Roberto Tonelli[a]

[a] *Cagliari University - Università degli Studi di Cagliari, Italy*
[b] *Université de Lille, Inria, CNRS, Centrale Lille, UMR 9189 – CRIStAL, France*
[c] *University of Antwerp, Belgium*

Abstract

The Ethereum blockchain is a distributed database of transactions, where the Gas Oracles suggest the users the *Gas price*'s categories to get a transaction recorded. The paper explores the idea that the Gas Oracles are based on a data-centered model which does not provide users with a reliable prediction. We present an empirical study to test the reliability of the existing Gas Oracles from both the points of view of the *Gas price* predictions and the existing categories.

The study reveals that the Gas Oracles' predictions fail more often than advertised and shows that the Gas price categories do not correspond to the categories set by the users. Therefore we propose a user-oriented model for the Oracles' *Gas price* prediction, based on two *Gas price* categories actually corresponding to the users' interests and a new method to estimate the Gas price. The new method, performing the Poisson regression at smaller intervals of time, predicts the *Gas price* to pay with a lower margin of error when compared to the actual one. The predictions based on the user-oriented model thus provide the users with a more effective *Gas price* to set.

Keywords: `Ethereum`, `Gas`, `transaction fees`, `empirical study`, `transaction pool`, `Gas price categories`

1. Introduction

Ethereum blockchain is a distributed ledger where transactions are recorded into a sequence of ordered blocks. The Gas is a unit of measurement unique to the Ethereum blockchain that measures the computational work required to run transactions within the Ethereum Virtual Machine (EVM). The transactors, i.e. the users or the smart contracts that submit transactions to the blockchain network (henceforth "users"), also propose a fee in terms of Gas price to validate,

[☆] Fully documented templates are available in the elsarticle package on CTAN.

include, and compute the transactions effect, when an executable code, the so called "smart contract", is called [20]. The users especially pay the fee in Ether, the Ethereum cryptocurrency, for the effort required to compute the proof-of-work (PoW). The PoW keeps the network resilient, though requiring a big investment of the miner, i.e. the node that solves the PoW challenge [1]. The PoW challenge indeed consists in a cryptographic puzzle requiring large computational resources [26]. As there is a (weighted) distribution of minimum acceptable Gas prices, the users will have a trade-off to decide between lowering the Gas price and maximising the chance that their transaction will be timely committed to the blockchain [34].

To send a transaction on the Ethereum blockchain, the user needs to specify a Gas limit, which is the maximum amount of Gas that can be consumed by the transaction, and a *Gas price* which is the cost in Ether the user is willing to pay per unit of Gas consumed. If the transaction spends less Gas than the Gas limit, the remaining Gas will be refunded to the user and the miner will earn less than the maximum Gas Limit. Indeed, unlike the Bitcoin blockchain, where the users do not need to set a Gas limit, but just the transaction fee which will be paid to miners, in the Ethereum blockchain there is always the possibility for the miners to receive a minor reward compared to the Gas limit set by the users [4].

There are some main reasons to have a *Gas price* in the Ethereum blockchain: 1) the users must pay for computational costs and resources used (*e.g.,* energy, CPU) to generate and include their transactions into blockchain blocks upon approval; 2) a Gas price regulates and limits the use of blockchain resources; 3) a Gas price incentivizes miners to actually include transactions in the blocks without just mining empty blocks; 4) a Gas price allows the users to express (and pay for) priority; 5) a Gas limit avoids network abuse or misuses, intentional or unintentional (*e.g.,* DoS attacks, infinite loops) [6].

The users, sometimes via an intermediary, send a transaction to an Ethereum node. From there, the transaction is broadcast to other nodes and distributed across the network. When the transaction reaches a miner's node, the miner can add it to the pool of pending transactions (also called "memory pool") and then include it in a new block which may be appended to the last one in the chain [12].

The computational cost of a transaction in Gas units depends only on the computations occurred to process such transaction. The Ethereum documentation provides the different costs of each elementary operation. The users are free to specify any *Gas price* that they wish, however the miners are free to ignore transactions as they choose. Some miners, especially the miners with high computational resources, may seek to make the highest profit and change the source code to evaluate the transactions based on the Gas parameters, i.e. the Gas limit and the Gas price [11]. For instance, "Go Ethereum", a software installed

[1] Ethereum is currently migrating to a Proof of Stake consensus algorithm. However, our analysis targets current transactions on the main network where the PoW is still in use.

in some nodes of the Ethereum network, might be used to set the parameter expressed by the variables "–txpool.pricelimit" and "–txpool.lifetime". In particular, the "txpool.pricelimit" variable defines a baseline transaction price under which the node will simply not accept transactions (not even to forward it to other nodes) [6]. Consequently, on the one hand, if the value set by the user is too low, miners will probably ignore such transactions which risk to be never included in the blockchain. On the other hand, if the transaction fee is too high, miners will be prone to include it in the Ethereum blockchain, but the user will allegedly waste money. To suggest the best trade off for Gas price, the Gas Oracles assign the *Gas price* to categories, which are actually based on four quantiles (50th, 75th, 95th, 99th) determined from past Gas price observations. Section 2.4 explains how the Gas Oracles model the Gas price, based on data from past blocks [1].

In this paper, we extend preliminary results [24] obtained for a single Oracle case, the *EthGasStation Oracle*, to another case, the *Ethchain Oracle*, in a wider time-frame, by analyzing the data of the Oracles that predict the Gas price, along with the Ethereum transactions' and blocks' data. The Ethereum transactions' variables considered in the study are:

- the waiting time calculated as the time elapsing between the time the transaction was seen by the miner we are considering in this research and the time the transaction has been included into the block [16].
- the Gas price, *i.e.,* the amount of Ether the user is willing to pay for every unit of Gas, which is measured in "GWei" [34].

Oracles' data are useful to predict the Gas price a user should pay to make it convenient for a miner to include the transaction into a block. To help the users in deciding the price to pay for the cost of the PoW calculation, Gas Oracles propose the following four price categories: *'safeLow'*, *'average'*, *'fast'*, and *'fastest'*. These categories define the Gas price required to have a transaction included within the next 100, 20, 5, and 2 blocks, respectively. The paper aims to answer the following research questions:

- RQ#1: Are the Oracles' predictions reliable as much as declared?
- RQ#2: Do the Gas price categories provided by the Oracles correspond to the *Gas price* categories the users actually set?
- RQ#3: How could the Oracles provide the users with more reliable predictions?

To answer our research questions, we hypothesized that 1) the predictions made by the Gas Oracles have a margin of error greater than the margin of error declared by them (2%); 2) the categorizations of the Gas price made by two Oracles do not correspond the Gas price the users and/or companies set; 3) it is possible to reduce the Gas Oracles' error margin by calculating the *'recommended*

Gas price' when each block is added instead of every 100 added blocks as the existing Gas Oracles actually do [1].

We collected data in three-months time from two Gas Oracles (Etherchain and EtherGasStation) which predict the *Gas price* every time that 100 blocks are added to the Ethereum blockchain. During the same period, we also collected over 10 million transactions from a transaction pool. We then cross-checked the data collected by the transaction pool and the Oracles, to understand whether the Oracles' estimates fail.

First, the results of the paper show that both Gas Oracles (Etherchain and EtherGasStation) give the *Gas price* prediction with a higher margin of error compared to what they declare (2%). The margin of error ranges from a minimum of 5% for the *'fastest'* category to a maximum of 16% for the *'fast'* category. Second, the results show that the margin of error could be lowered to 2% for all the categories, by performing the Poisson regression at smaller intervals of time. Finally, the results suggest that two of the Gas Oracles categories are not frequently used in practice: *'fast'* and *'average'* categories. It is indeed reasonable to expect that single users or companies aim to save money and thus set some requirements, which are different in terms of waiting time and are not provided by the default categories.

The rest of the paper is organized as follows. Section 2 presents the concepts needed to better understand our research, such as the transaction pool, the Gas Oracles, and the Gas price categories investigated in the paper. Section 3 presents the related work the paper uses as a starting point for a user-oriented model for the Gas Oracles' Gas price prediction. Section 4.1 presents the experimental hypotheses guiding the study. Section 4 describes the methodology used to test the hypotheses, to collect and perform the regression model on the data of the study. Section 5 presents the results of the study. Section 6 discusses the results in the light of the user-oriented model. Finally, Section 7 draws some conclusions and outlines some ideas for future work.

2. Background

This section provides the readers with a brief introduction on the blockchain technology and in particular on the Gas price mechanism sets on the Ethereum blockchain to ensure a balanced use of resources.

2.1. Blocks

The blockchain is an ordered sequence of blocks containing the records of valid transactions as approved by a consensus algorithms shared between a set of computational nodes in a peer-to-peer network. It is a shared ledger where, to keep unchangeable the block sequence and the temporal order of recorder transactions, each block includes a cryptographic hash depending on the information recorded on the previous block. Each block is also identified by progressive number named "height" [4]. Once a block is created and added to the blockchain, the transactions in the block cannot be changed or deleted. This

is to ensure the integrity of the transactions and to prevent the double-spending problem [25].

Block time is the time the network takes to generate one extra block In Ethereum the "median" block-time is about 13 seconds and depends on how long the miners take to find the correct hash to validate a block by brute force computation. In the blockchain there are two units of time measurements: (i) seconds, and (ii) blocks' number or height [31].

2.2. Transaction Pool

The software running in each miner node collects the transaction into a virtual storage named "transaction pool". The miners distinguish processable transactions, which can be included into a block, from future transactions, which can wait to be included. Therefore the transactions move between these two states over time as they are received and processed [13]. When a miner solves the PoW challenge to mine a block, the miner informs the adjacent nodes about that. As the adjacent nodes receive this piece of information about the newfound block, they will validate the received block and propagate the block data to peer nodes. In the case of mining nodes, they will remove all the transactions contained in the newfound block from their own transaction pool, checking that for each transaction the current balance is greater or equal to the money spent [32]. The miners have full control over their transaction pool and may adopt different policies to manage it. For instance, a miner could set up a minimum fee threshold, thus transactions with a *Gas price* lower than the threshold are immediately discarded from the transaction pool and only the new transactions with a price higher than the threshold are allowed to enter the transaction pool [29].

2.3. Gas Oracle

In the blockchain terminology, Oracle may have different meanings. An Oracle can be a program which provides the smart contracts with reliable data collected from outside the blockchain. Oracles are also software systems which analyse some data and make some prediction on that basis [16].

In this paper, the term Gas Oracle assumes a specific meaning related to the activity of forecasting Gas prices. The Ethereum wiki [2] reports the following definition: "a Gas Oracle is a helper function of the Geth client that tries to find an appropriate default *Gas price* when sending transactions and it can be parametrized". Thus an Gas Oracle analyses blockchain data to predict the best *Gas price* to pay for a transaction to be approved within a certain number of blocks. The Oracle's forecasts may be important for companies using the Ethereum blockchain because the time and the costs of performing transactions can affect their economical resources and clients' satisfaction [18]. It is thus crucial for them that Oracles forecasts are as reliable as possible. However, based on the analysis performed in this paper, it is not the case.

[2] https://eth.wiki/

Table 1: *Gas price* categories with the relative waiting time

Gas price category	Maximum waiting time to include the transaction into a block
'fastest'	at most in 30 seconds
'fast'	at most in 2 minutes
'average'	at most in 5 minutes
'safeLow'	at most in 30 minutes

We indeed analyzed the predictions of two Gas Oracles: EtherGasStation and Etherchain. Both Gas Oracles claim that all predicted values are estimations based on the current network conditions and should be used as a suggestion. However, the Gas Oracles only compute and update their predictions every 100 blocks (approximately 1,500 seconds or 25 minutes) [1]. Therefore, their estimations might not mirror the current status of the network.

2.4. Gas Price Categories

Gas Oracles, EtherGasStation and Etherchain, estimate the time interval required for a transaction to be included into the next blocks based on the Gas price attached to the transaction [1]. To estimate the waiting-time a transaction needs to be included into a block, many variables need to be considered, such as the number of transactions submitted by the users in a given period of time, the number of miners and their policy [22].

Gas Oracles have defined four categories based on the quantiles of the *Gas price* offered to the miners by the users which are accessible from the transactions data. The four percentile are the 50th, the 75th, the 95th and the 99th percentile [1]. The 50th percentile corresponds to the *'safeLow'* category, the 75th percentile to the *'average'* category, the 95th percentile corresponds to the *'fast'* category and finally the 99th percentile to the *'fastest'* category.

To make information more accessible to users, the Gas Oracles states that each category corresponds to a waiting time. In reality, it would be more effective for the users to know that the *Gas price* is related to the number of blocks to wait and not to the time because the median value to mine a block is 13 seconds but there can be strong oscillations ranging from a few seconds to over half an hour to mine a single block (see Section 5). Table 1 presents the categories defined by the Gas Oracles (Etherchain [3] and Etherscan [4]) and their waiting-times. The code which estimates the *Gas price* to pay to the Gas Oracles is publicly available under doi: 10.5281/zenodo.3758103.

[3] https://etherchain.org/tools/gasPriceOracle
[4] https://docs.ethgasstation.info/gas-price

3. Related Work

The blockchain can be disadvantageous from a user's perspective because of different kind of wasted resources. The paper focuses on the waste of Gas price or waste of time the users might experience to add a transaction to a block. Previous work highlighted other kinds of waste from a user-oriented perspective. Chen et al. [5] identified seven Gas costly patterns, *i.e.,* programming solutions that are not optimized by the Solidity compiler. A Gas costly pattern required more computational resources, thus reducing the number of transactions that can be included into a block. Therefore the users need to wait or pay more to have their transaction executed. The authors analyzed 4,240 smart contracts on three Gas costly patterns. They found that over 80% of the contracts suffer from this kind of costly patterns. The authors' work is therefore interesting because it highlights an existing waste of Gas units in the blockchain, which disadvantages the users' interests.

In a previous study, I. Weber et al. [33] measured the time for transactions to be committed in both Ethereum and Bitcoin blockchain. The authors performed a detailed analysis of issues that could negatively impact commit times in permissionless PoW blockchains such as Ethereum. Their study is very interesting for the purpose of this paper because it identifies the *Gas price* as a cause of delay in the commitment of transactions.

Sin Kuang Lo et al. [19] investigated the reliability of seven Oracles on different platforms such as Augur, Ms Bletchley, TownCrier and Corda. They discovered that the common causes of failure are the data sources used by the Oracle to make various kinds of predictions such as the weather forecast. These failures can have a serious impact on the economy because many smart contracts perform operations on the basis of these predictions. To meet the users needs, they provided a framework that can be used to assess the Oracles' reliability. The authors' work supports the interesting idea that, providing this framework together with the Gas Oracle, it is possible to help the users' decision making. Differently from their work, in this paper, we study the behavior of the Oracles that predict the Gas price in the Ethereum blockchain.

Ducasse et al. [9] pointed out that even more experienced users, as software developers of smart contracts, need to be helped to write smart contracts that are more effective by using fewer resources. This is the main reason why the authors proposed an open-source platform for blockchain analysis called SmartAnvil. Although SmartAnvil is independent from a specific blockchain platform and thus may be used to investigate any blockchain, their work focused on Ethereum blockchain and contracts written in Solidity. The authors provided a tool that can facilitate the identification of resources waste to solve a problem within the smart contracts. The authors' work is therefore interesting because it supports the idea that, providing this tool together with a Gas Oracle, it is possible to help users and companies to waste less time and money.

In a previous study [28], Singh and Hafid proposed a more fine-grained classification model when compared to the existing Gas Oracles' classification. The model split the inclusion time of transactions into eight classes: respectively

within 15 seconds, 30 seconds, 1 minute, 2 minutes, 5 minutes, 10 minutes, 15 minutes, and 30 minutes or longer. Interestingly, the authors proposed a classification that considers different possibilities for the users to set a Gas price that might meet their needs, while existing classification do not pay attention to the users' point of view. In this paper, we limit our research to the existing classification of the Gas Oracles, but we accept the idea that there may be some categories that better represent the users' needs and interests.

Generally, the users or the companies may have the following interests and needs: 1) they may sometimes be willing to pay a lot to have the transaction executed as soon as possible, 2) they may sometimes be willing to save money and wait a lot, as long as their transactions are eventually added to the blockchain. For instance, the users or companies may be willing to pay a lot during an initial coin offer (ICO), when only a limited supply of tokens is available and thus "the first to arrive is the first to be served". On the contrary, when the time is not constrained, they may wait to save money. For instance, when the smart contracts need to refund users having an assurance in case of delay of arrival, the users might want to wait a few hours before receiving the reimbursement [17].

Different academic works compared the performance of different machine learning regression models, such as Decision Tree, Logistic Regression and Random Forest, on the task of predicting the confirmation time for a transaction in both Ethereum [21, 28] and BitCoin blockchain [30, 14]. Interestingly, these works suggested that there are different, but not mutually exclusive, machine learning regression models that can reach high accuracy in predicting the transactions waiting time. Moreover, these studies confirm that the most important feature to predict the transaction waiting times is the Gas price attached to the transactions, which will also be used to train our model.

Another study [22] investigated other factors that might influence the Ethereum transaction fees and the possible resulting decision-making behaviour of Ethereum blockchain users, miners included. They observed that the past history of the Oracle Gas price prediction is useful to predict the number of waiting transactions, even though the converse is not true. The results of the Pearson correlation test showed that they are instead inversely correlated: when the Oracle price increases, the number of waiting transactions in the Ethereum network decreases. It stands to reason that when the Oracle suggests a high price to pay, the users that can wait, wait to submit a transaction, thus decreasing the overall number of pending transactions in their memory pools. This result pushes us to target our research towards a model oriented on the users and not on the mere data or miners.

In a previous study [24], a quantitative study was conducted to determine whether the *Gas price* prediction of the Oracle EtherGasStation is reliable. The study aimed to evaluate the correctness of the Gas price prediction the EtherGasStation made to have the transaction recorded in the blockchain. The study investigated the EtherGasStation's predictions and found that it brings about a higher margin of error than originally declared. EtherGasStation indeed claims to have a 2% margin of error, while the analysis of the predictions showed that the margin is at least twice as much. For instance, the *'fastest'* category

showed a 4% margin of error, while the *'fast'* category showed a 28% margin of error.

Moreover, the study argued that such a higher margin of error is due to the fact that EtherGasStation does not take into account changes in the Ethereum Network occurring in real-time. The study was anyway limited in various ways, as it considered just one Gas price Oracle in a short time framework, so that the results cannot be generalized and used to understand wider and general trends in Gas Oracles' prediction.

This study therefore provides a more comprehensive analysis of Gas price predictions, by performing: *a*) a quantitative analysis of the predictions of another Gas Oracle, Etherchain, to check whether they are reliable or whether also in this case they fail in suggesting the right Gas price as in the case of the EtherGasStation Oracle; *b*) a test for the hypothesis that the Gas price's margin error is reduced for each category, when reducing the time interval required for the estimation of the Gas price.

4. Research Methodology

The research methodology of the study includes the following phases: (a) the experimental hypotheses, (b) the Data Collection, (c) the Data Cleaning, (d) the Data Modelling, and (e) the Regression Analysis. The following sub-sections describe each phase.

4.1. Experimental Hypotheses

The study was designed to address the following Research Questions:

- RQ#1: Are the Oracles' predictions reliable as much as declared?
- RQ#2: Do the Gas price categories provided by the Oracles correspond to the Gas price categories the users set?
- RQ#3: How could the Oracles provide the users with more reliable predictions?

To answer the questions, we advanced the following hypotheses:

- H1: The Gas Oracles' predictions are not reliable. The Gas Oracles cannot indeed take into account all the changes in the Ethereum Network in real-time, especially because they compute the prediction every 30 minutes on average.
- H2: The Gas price categories proposed by the Oracles do not correspond to the categories set by the users and/or companies. Single users or companies may indeed set different requirements in terms of waiting time that is not provided by the default categories.
- H3: A reduction of the margin of error in the Gas price prediction can be achieved by calculating the *'recommended Gas price'* at smaller interval of time, thus considering the current changes of the network in real time.

Resource name	REST API service URI
EtherGasStation	https://ethgasstation.info/json/ethgasAPI.json
Etherchain	https://www.etherchain.org/api/gasPriceOracle
Block	https://api.blockcypher.com/v1/eth/main/blocks/0
Pending Transactions	https://api.blockcypher.com/v1/eth/main/txs

Table 2: RESTful Services list

4.2. Data Collection

In this research, we covered a 3-month analysis period and we made code publicly available to replicate the data collections of the transactions used in this research. The source code is available at the following online address `https://github.com/aphd/eset/tree/master/src`. The same code can be used to analyze the transactions' data in other time frames. We collected data by making requests to various REST API services at different times. The flow to collect data is:

- a request is sent to the server every 15 seconds;
- if the request is successful, the server responds to the client request sending a payload in JSON format;
- if the request is not successful, the client does not record any data for that time frame. During the data retrieving operation, an average of 1 request out of 20,160 requests was unsuccessful.

Table 2 shows the URI of the REST API services used to fetch the Gas Oracle data, the blocks data, and the pending transactions data, *i.e.,* the latest transactions that have not been included in any block. We choose to collect data from these Gas Oracles because they are very popular among the Ethereum community. Data were stored as files in JSON format in the file system of the server where the analyses were performed.

4.3. Data Cleaning

A control over the data quality was performed. The data retrieved were accepted when in compliance to the API documentation, or otherwise rejected. The 0.8% of data was rejected, distributed as follows: 770K out of 11M transactions (0.75%), 182 out of 345K blocks (0.05%) and 112 out of 345K Oracle's predictions of the Gas price (0.03%). Example of data, which were not in compliance to the API documentation, are:

- string value where a numeric value was instead expected;
- numeric value where a string value was instead expected;

- numeric value which is not in the expected range;
- date value which is not in the expected time frame;
- missing value;
- missing key/value pairs;
- numeric value with different units of measurement.

The data falling in one of the categories listed above were rejected, except for the latest category where the values were recalculated conforming with the expected measurement units. Just to give an example, a numeric value which is not in the expected range can be a negative or undefined block's height value, a date value in the future, or a negative value of the waiting time for transactions to be included into a block. Blocks with negative/undefined height were caused by a chain split. This might happen when the node that we were monitoring received from adjacent nodes a chain with more PoW (the longest chain). In this case, the last block(s) might become orphans/uncles with an invalid height.

According to Kanda and Shudo [15], a negative value of the waiting time variable may suggest a transaction propagation delay among different nodes. This means that different nodes in the blockchain could see the transaction at different instants of times. The transactions data-set contains indeed transactions with a negative waiting time, around the 1.16% of all transactions. This might have at least two reasons: 1) every node can have a different clock time, and 2) there is a propagation delay defined as the difference between the time when a node announced the discovery of a new block or a transaction and the time when this announcement was received by other nodes [15]. In the study, the transactions data were collected through an API that gives the transaction pool data of a single node. The API is available at the following online address `https://www.blockcypher.com/dev/`. As the Ethereum network is distributed, not all miners receive the same transactions at the same time, therefore some nodes might store more transactions than others at some time [27]. Furthermore, every node can be a miner with different hardware and software features and miners might have different RAM capacity to store pending transactions. As a result, each miner has its own representation of the pending transactions. The existence of such delay - which is not negligible - justifies the negative times, because blocks can be discovered while communication and validation is still in process. Decker and Wattenhofer [8], for the BitCoin blockchain, observed that the median time until a node receives a block was 6.5 seconds, the mean was 12.6 seconds and the 95th percentile of the distribution was around 40 seconds. Moreover, they showed that an exponential distribution provides a reasonable fit to the propagation delay distribution. It is reasonable to think that there is a similar effect in the Ethereum blockchain.

4.4. Modelling Data

In this subsection, we define the condition to assess the correctness of the Oracles' *Gas price* prediction. Both EtherGasStation and Etherchain make the

prediction based on the history of the mined blocks data, such as the lowest *Gas price* accepted by the miner to add the transaction to the block and the Gas offered by the users.

Suppose that during the time interval when the i-th block, B_i, is mined:

- *op* (Oracle Price) is the price predicted by the Gas Oracle to have the transaction included at most within j blocks;
- $B = \{B_{i+1}, \ldots, B_{i+1+j}\}$ is the set of j blocks mined in the blockchain following the $(i+1) - th$ block;
- $T = \{tx_1, tx_2, \ldots, tx_n\}$ is the set of pending transactions, *i.e.,* that have not been included in any blocks, with *Gas price* respectively of $tp_1, tp_2, \ldots, tp_n$, that there were in a transaction pool when the i-th block was mined.

In an ideal scenario where the Gas Oracles never fail, the transactors that set the Gas price equal or greater to the one suggested by the Gas oracle (*op*) should have the transaction confirmed in the blockchain after $n+1$ blocks where n depends on the category proposed by the Gas Oracle and chosen by the user. Figure 1 shows the ideal scenario where the users that set the Gas price following the Gas Oracle's suggestions have their transactions confirmed in the blockchain in the following n blocks. The condition is expressed by the following equation:

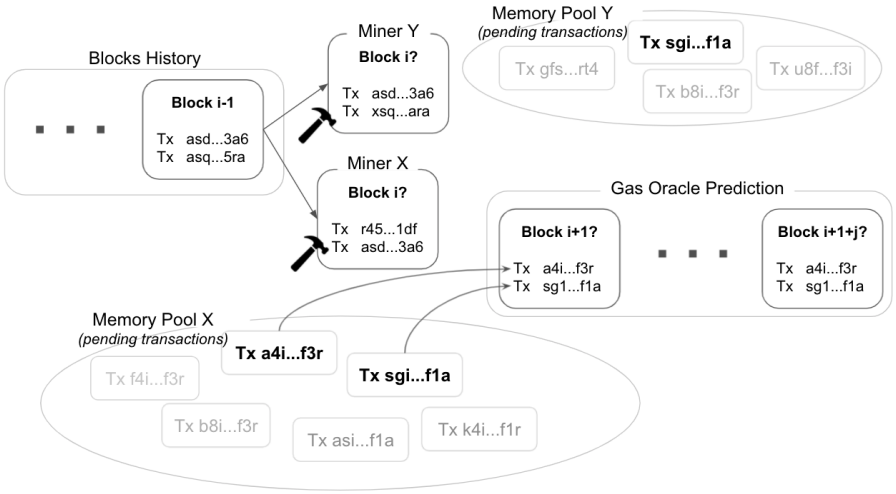


Figure 1: The transactions having a Gas price higher or equal to the one proposed by the Gas Oracle (*op*) are displayed in bold. B_i is the block that is mined in the interval of time in which the Gas oracle makes the prediction (*op*).

$$\forall tx_i \in T \wedge tp_i \geq op : tx_i \in \{B_{i+1}, ..., B_{i+1+j}\} \tag{1}$$

The equation 1 is used to verify the prediction of the Oracles' *Gas price*: Etherchain and EtherGasStation.

The existing Gas Oracles make the prediction every 100 blocks, performing a Poisson regression. Based on the *Gas price* distribution of the transactions mined in the last *200* blocks, the Gas Oracles estimate the *Gas price* to pay in relation to the number of blocks the users should wait to have their transaction added. The data are collected by querying the REST API Services mentioned in Table 2. To test the hypothesis that the prediction can be improved, we performed the same algorithm used by the Gas Oracles [1] at shorter time intervals (every 4 block for the *'fastest'*) instead of every 100 blocks as the Gas Oracles actually do. The results are collected in a table called realTimeOracle 2.

In this phase the data were collected and stored in a relational database, where each table represents the following items: blocks, transactions, Oracles, and OtherPrediction. Figure 2 shows the data contained in the database, the relationships between table fields and their types (*e.g.,* string, integer, boolean, enumerate). The table named *transaction* stores all the transaction information such as the received time detected in the transaction pool that was monitored for this research. It is noteworthy that the waiting time for a transaction is not stored in the database but calculated by the difference between its inclusion time and received time by considering just 1 confirmation block. The table named *block* stores all the block information such as the current block number in the blockchain (block_height), the number of transactions stored in a block (n_tx) and the lowest *Gas price* among all the transactions added in that block (lowest_gas_price). The tables *Etherchain* and *EtherGasStation* store the *'recommended Gas price'* to have the transaction included in the block for each category. Finally, table *realTimeOracle* stores all *'recommended Gas price'* to have the transaction included in the block for the *'fastest'* and *'safeLow'* category. The other two categories considered by the Gas Oracles have been excluded since this work, as well as previous works [24], shows how the *'fast'* and *'average'* categories do not reflect the requirements of companies and users.

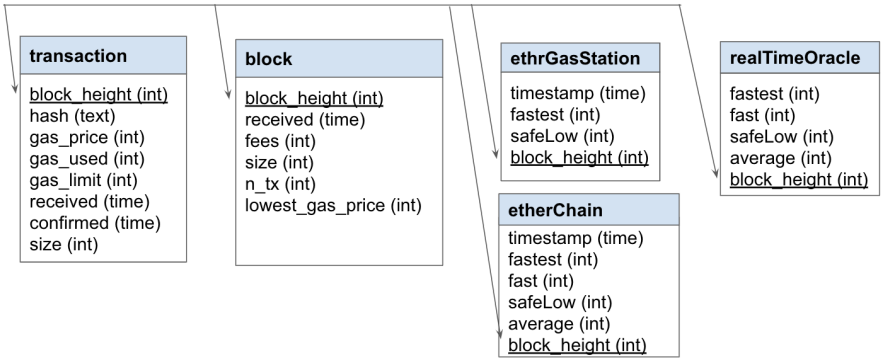


Figure 2: Database schema.

4.4.1. Poisson Regression Model

The Oracles adopt a Poisson regression model, as per source code available via Zenodo [1]. As anticipated in section 3, different models, such as the machine learning regression models, have been applied to improve the Gas Oracles prediction. Although some of these models can give better results compared with the Poisson Regression model, they have the drawback to be very expensive in terms of computing resources. The time required to make the prediction (around 20 minutes) is too long, as it is greater than the time taken by the blockchain network to mine a block. This is the main reason why we investigated how to improve the Poisson Regression model already used by the Gas Oracles.

In probability theory and statistics, the Poisson distribution is a discrete probability distribution of a given number of events occurring in a fixed interval of time or space [7]. We use the Poisson distribution to estimate the number of transactions added to the blockchain per block period, which median is 13 seconds. Let X represents the set of x transactions added to the blockchain in a one block period. For the sake of simplicity, let assume that all blockchain transactions are offering the same Gas price to the miners. Equation 2 shows the Probability Mass Function (PMF) of having x transactions added to the blockchain in one block period time.

$$P(X = x) = \frac{\lambda^x * e^{-\lambda}}{x!}, x \in [0, \infty) \tag{2}$$

In the Equation 2, λ is the mean number of transactions added to the blockchain in one block period of time and e is Euler's number.

To know the probability of having x transactions added in the n-th block, in the formula 2 the λ value is to be multiplied for the number of blocks. Equation 3 shows the probability of having x transaction added to the n-th block.

$$P(X = x) = \frac{n * \lambda^x * e^{-(n*\lambda)}}{x!}, x \in [0, \infty) \tag{3}$$

The equations 2, 3 are valid when the events are observed under certain conditions.

Conditions (C) for Poisson Distribution are:

- C1: An event can occur any number of times during a time period. In our case, the event is the transaction added to the blockchain and the time period is the block period.
- C2: Events occur independently from each others. In our case, if a transaction is included into a block, it should not affect the probability of another transaction to be included in the same block, i.e. in the same interval of time.
- C3: The average rate of events occurrences, i.e. the number of transactions added to the blockchain per block, should be constant, i.e. the rate should not change based on the block number added to the blockchain.

Of course, the Poisson distribution conditions are highly theoretical and do not fit the blockchain real situations (RS) for many reasons:

- RS1: As to the condition C1, our results 4 show that the number of transactions added to a block could be any integer number greater or equal to zero. Thus, the condition C1 is satisfied.

- RS2: As to the condition C2, a transaction included into a block does affect the probability to have another transaction added to the same block, simply because the number of transactions is finite. However, in most cases, the number of transactions is so high that they can be considered independent with good approximation.

- RS3: As to the condition C3, there are cases where it may not be satisfied. For example, if the policy of the miners suddenly changes and they decide to mine empty blocks, the average rate of occurrence will drastically change. Although the data in table 4 confirm that this can happen from time to time, this is not the normal situation, because it goes against the interests of the miners themselves. The network would indeed lose its usefulness and the value of the Ethereum cryptocurrency (Ether) would decrease when compared to other currencies (USD, EUR, etc.), and as a consequence, also the reward of the miners, who are paid in Ether. Moreover, a changing number of transactions submitted by the users to the blockchain network could change the average transactions number per block, based on the Gas offered to the miners. This can happen, for instance, during an ICO.

The points discussed above might explain why Gas Oracles' margin error is larger than expected. We suppose that, by recalculating the *lambda* factor in equation 3 at lower time block intervals compared to the Gas Oracles which recalculate the *lambda* factor every 100 blocks, the margin of error in the probability computation of having a transaction added to the blockchain in a certain number of blocks, can be lowered. This does not mean that the Poisson model, in which the lambda is recalculated every time a block is added to the blockchain, is the best way to model the blockchain. Of course, other models might be tested taking into account the time limit of 15 seconds, but up to now our model gives better results compared to the current Gas Oracles' model, as will be shown in section 5.6.

4.4.2. Regression Analysis

The purpose of this phase is to estimate the *Gas price* by running a Poisson regression analysis on the data stored in the block table as Gas Oracles currently do. The results of the Poisson Regression fill the table named "realTimeOracle". Unlike Gas Oracles, we perform the Poisson regression more frequently based on the four category *'fastest'*, *'fast'*, *'average'* and *'safeLow'*. This choice is justified by:

- the categories are very different from each other based on waiting time requirements, expressed by different constraints. For example, the category *'fastest'* has the constraint of having to guarantee 98% of transactions to be included into a maximum of two blocks. This means that the error on the lambda determination must be very small compared to the block interval. The category *'safeLow'* instead requires to have the transactions included into a larger time frame (120 blocks) and so the error on the λ can be greater compared to the category *'fastest'*.

- the time interval of 100 blocks to recalculate the λ could be very long compared to changes in the network. According to our observed data 100 blocks correspond to about 30 minutes with a sigma equal to 20 minutes. We suppose that during this interval of time the network condition can change and this change can affect the value of λ.

5. Results

This section presents the data-sets (blocks data-set, transactions data-set, Oracles' predictions data-set, user-oriented predictions data-set), as modeled in Section 4.4. The data-sets are stored in an SQLite database with five tables, one table for each data-set. The total size of the database is of 1.1 Giga-Byte and is publicly available via Zenodo [23]. The first table, named "transaction", contains more than 11 millions rows. The second table, named "block", contains around 345 thousand blocks. The blocks data-set consists of 103596 records starting from height 7590409 to height 7694005. At the date of the research the last block is 7764216, meaning that we analyzed the $103596/7764216 * 100 = 1\%$ of the Ethereum blockchain. The two tables, named respectively "EtherGasStation" and "Etherchain", contains 345 thousand rows of Oracles' predictions for the Gas price of each category: (*'fast'*, *'fastest'*, *'average'*, and *'safeLow'*). The Oracles' predictions data-set covers a period of three months starting from 15 March 2020 with 15 seconds temporal resolution. Finally the table "realTimeOracle" contains the data of the user-oriented model for Oracles' *Gas price* predictions. The data-sets refer to a three-months period of time, ranging from March 1, 2020 to May 28, 2020.

The following sections present the results of the study as some aggregated statistical metrics such as percentile, mean, standard deviation, mode of the numerical data series for transactions, blocks, Gas Oracles' predictions and User-oriented predictions. The sections 5.1 also show the distribution of different variables, such as Gas_prices and time a transaction needs to wait before being recorded in the blockchain. The error margins are summarized in Table 8, comparing the results of the user-orinted model to the existing data-centered model of the Gas Oracles.

5.1. Transactions Data Analysis

Table 3 shows the statistics of the transactions data-set. The mean, the standard deviation (SD), minimum (min), the 25th, 50th, and 75th percentiles

Table 3: Statistical description of transactions data

.

	mean	std	mode	min	25%	50%	75%	max
waiting_time (s)	44.02	82.65	25	0	25	29	38	1499
gas_price (GWei)	32.19	443.29	50	0	10	20	50	313734
gas_used	70124.2	320908	21K	0	21K	21969	49993	8e+06
gas_limit	303967	947926	21K	21K	42K	70000	150000	8e+06
size (Byte)	191.11	499.98	-	83	112	114	174	31791

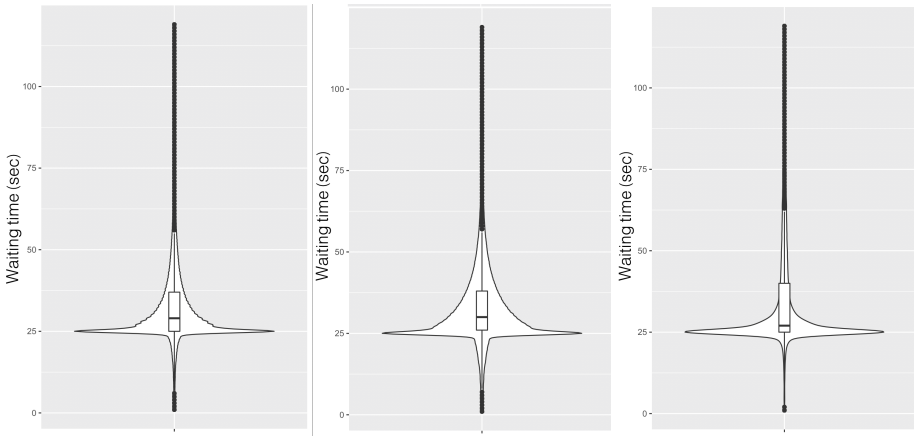


Figure 3: Violin Plot of the waiting time in seconds before a transaction is included into a block. (a) All transactions. (b) Transactions having a Gas price lower than 10 GWei. (c) Transactions having a Gas price higher than or equal to 10 GWei.

and maximum (max) are calculated for each variable shown in the table. The main variable is the "gas_price" value for each transaction included in a specific block.

Figure 3a shows the violin plot of the waiting time in seconds before a transaction is included into a block. The plot shows the presence of a peak at the value of 20 seconds with a tail that tends towards infinity. Figures 3b, 3c show the violin plots of the waiting time of the transactions for different Gas prices. Interestingly, the violin plots show that the Gas price attached to the transaction influences the interval of time the transaction needs to wait before being included into a block. The violin plots also present the same peek at the value of 20 seconds regardless of the Gas price.

5.2. Block Data Analysis

The blocks data-set gives information about each block, based on its height, *i.e.,* the index number that denotes its position in the blockchain. The data

Table 4: Statistical description of the Ethereum blocks data (from the 6 871 349-th block to the 7 694 005-th block)

.

	mean	std	min	25%	50%	75%	max
fees (GWei)	0.064	1.3	0	0.0245	0.046	0.07	381.1
size (Bytes)	18843.2	10580.7	524	9877	19045.5	27784.8	101294
n_tx	97.859	63.6377	0	45	92	144	381
lowest_gas_price (GWei)	4.46269	27.61	0	1	3	4	6215.03
block_time (s)	13.9453	12.9755	0	5	10	19	153

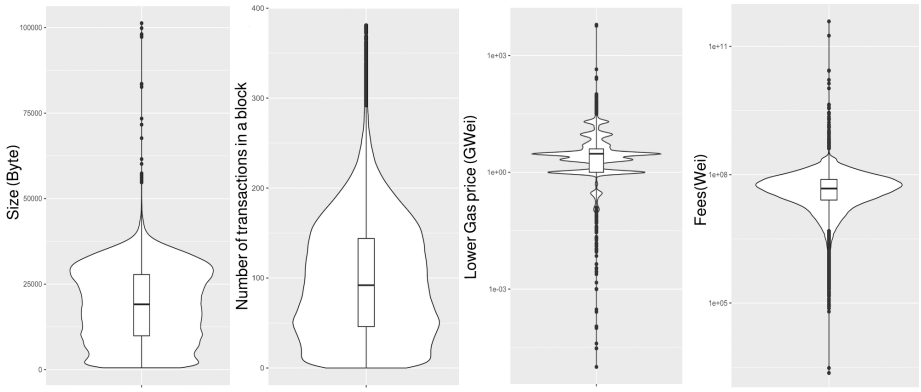


Figure 4: (a) Block size (including header and all transactions) in bytes. (b) Violin plot of the number of transactions included in each block. (c) Lowest Gas price attached to each transaction. (d) Violin Plot of the total fees (in Wei) collected by the miners in each block.

included in the blocks data-set are:

- the total number of fees in GWei, collected by miners in each block (fees);
- the size of the block (including the header and all the transactions) in Bytes (size);
- the number of transactions in each block (n_tx);
- the lowest Gas price attached to a transaction included in each block (lowest_gas_price).

Table 4 shows the statistics of the blocks data-set. The mean, the standard deviation (SD), minimum (min), the 25th, 50th, and 75th percentiles and maximum (max) are reported for each variable. Figures 4a, 4b, 4c, 4d show the probability density of each block variable at different values.

Table 5: Statistical description of Oracles categories

	mean	std	mode	min	25%	50%	75%	max
'fastest' (GWei)	15.33	6.60	20	3	10	20	20	61
'fast' (GWei)	4.58	2.42	3	3	3	3.6	5	60
'average' (GWei)	2.83	0.80	3	1	3	3	3	14.5
'safeLow' (GWei)	1.34	0.68	1	1	1	1	1.1	14.5

5.3. Oracles Data Analysis

We analyzed the predictions data of the Oracles, Etherchain and EtherGasStation. The Gas Oracles can diversely predict the *Gas price* values to attach to transactions to have the transaction included at the most within n blocks.

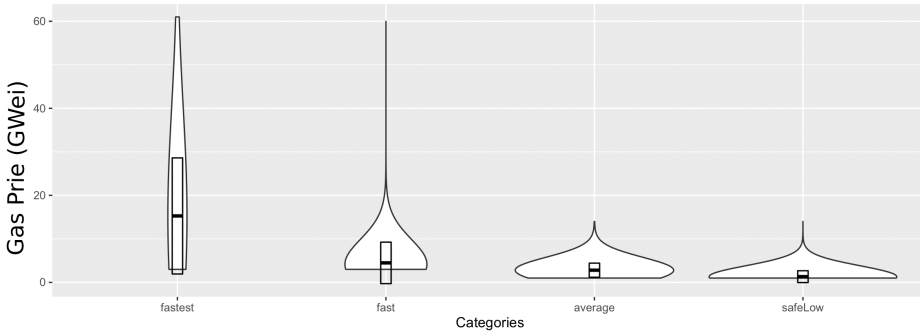
Figure 5a shows the violin plots of the EtherGasStation Oracle's *Gas price* predictions for each *Gas price* category: 1) *'fastest'*, 2) *'fast'*, 3) *'average'*, and 4) *'safeLow'*. The *'recommended Gas price'* range is highly variable and the variability depends on each category. For example, for category *'fast'*, the *'recommended Gas price'* ranges from a maximum of 61 GWei to a minimum of 1 GWei and the most frequent value is 20 GWei. On the other side, the category *'safeLow'* has a *'recommended Gas price'* which ranges from a maximum of 15 GWei to a minimum of 1 GWei and the most frequent value is 1 GWei.

Figure 5b shows the violin plots of the Etherchain Oracle's *Gas price* predictions for each *Gas price* category: 1) *'fastest'*, 2) *'fast'*, 3) standard, 4) *'safeLow'*. Likewise the EtherGasStation, the *'recommended Gas price'* range of the Etherchain is highly variable and the variability depends on each category. For instance, for the category *'fast'*, the *'recommended Gas price'* ranges from a maximum of 61 GWei to a minimum of 1 GWei and the most frequent value is 20 GWei, while the category *'safeLow'* has a *'recommended Gas price'* which ranges from a maximum of 15 GWei to a minimum of 1 GWei and the most frequent value is 1 GWei.

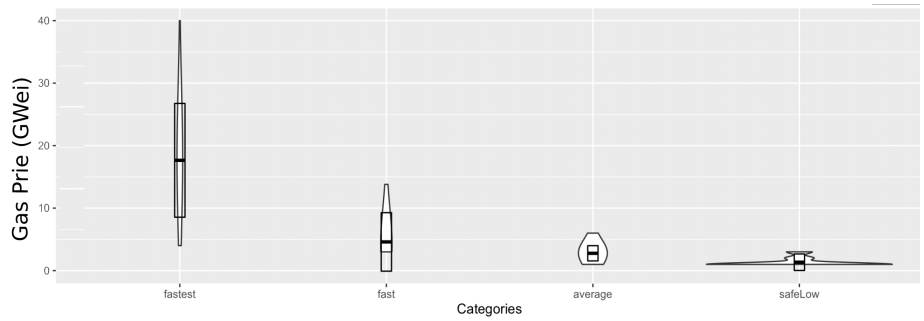
Table 5 reports the mean, the standard deviation (SD), the mode, the minimum (min), the first quartile (25%), the median (50%), the third quartile (75%) and maximum (max) of the Gas price recommendation for the transactions for each Gas price category.

Figure 6a shows the violin plot of the *Gas price* prediction according to the Etherchain Oracle (in blue) and EtherGasStation Oracle (in orange). The values refer to the Gas price to pay to have the transactions confirmed within 1-2 blocks.

Figure 6b represents the percentage of transactions having a Gas price equal to that suggested by the Gas oracle for each Gas price category. Figure 6b also shows that the transactions having a Gas price that exactly corresponds to the minimum Gas price are 7.6% for the category *'safeLow'* and 8.4% for the category *'fastest'*. The analysis revealed that 83% of transactions fall in the range provided by the Oracles' fee categories, because of their Gas price

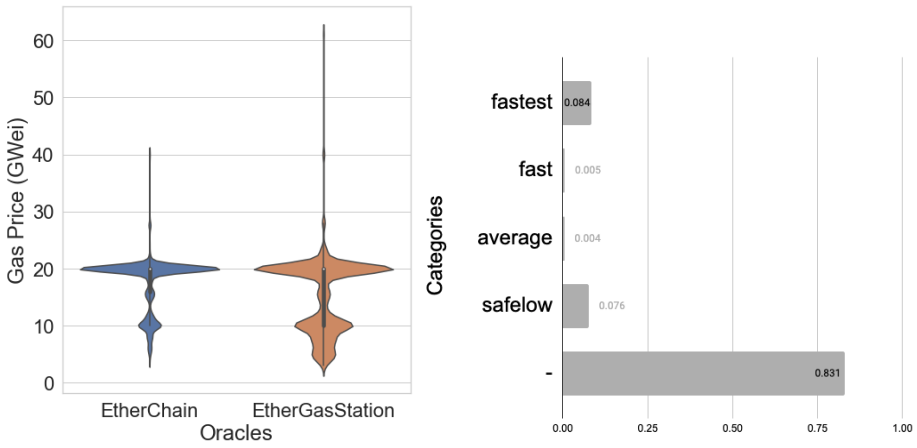


(a)



(b)

Figure 5: (a) Violin plot of the EtherGasStation Oracle's Gas price categories. (b) Violin plot of the Etherchain Oracle's Gas price categories.



(a) (b)

Figure 6: (a) Violin plot of the Oracles' Gas price prediction for the *'fastest'* category (b) Gas Oracles Categories corresponding to the *Gas price* set by the users or exchanges.

greater than the Gas price advertised by the Oracles. However, the study finally focused on the transactions having a Gas price exactly equal to the minimum price advertised by Oracle for each category, because the study aims to give the users an answer on the cheapest fee to attach to the transactions, in order to have the transactions confirmed in a certain number of blocks.

5.4. Evaluation of Oracles' Prediction

The Gas Oracles claim that at least 98% of transactions will be included at most into the next n blocks, if the Gas price of the transactions is equal to or greater than the Gas price they recommend. The Gas Oracles' predictions are four, one for each category. The waiting times are equal to 30 seconds, 2 minutes, 5 minutes and 30 minutes respectively for the categories *'fastest'*, *'fast'*, *'average'*, and *'safeLow'*.

To verify the Gas Oracles' predictions, we considered all the transactions which satisfied the requirements suggested by the Gas Oracles, i.e. transactions having a Gas price equal or greater to a certain value. For instance, for the category *'fast'*, we considered all the transactions having a Gas price attached equal or greater to the Gas price suggested by the Gas Oracle for that category.

Then we computed the percentage of transactions included in the blocks mined during the five minutes interval of time (the waiting time advertised by the Gas Oracle).

We performed the analysis for all the categories and the Gas Oracles considered in the paper. Table 6 summarized the results of the evaluation of the Gas Oracles' prediction, comparing included and confirmed blocks. The results suggest that the Gas Oracles' predictions might be wrong, as all the percentages are lower than 98%, i.e. the percentage declared by the Oracles.

Table 6: Gas Oracles' rate of success in predicting block inclusion and block confirmation

Gas Oracle	Category	Waiting Time	Included Blocks	Confirmed Blocks
Ether Gas Station	fastest	30 seconds	0.95	0.92
	fast	2 minutes	0.86	0.84
	average	5 minutes	0.92	0.91
	safelow	30 minutes	0.9	0.9
Ether Chain	fastest	30 seconds	0.91	0.89
	fast	2 minutes	0.84	0.82
	average	5 minutes	0.91	0.9
	safelow	30 minutes	0.9	0.9

However it might be claimed that the Gas Oracles are actually right, as the discrepancy is just due to statistical fluctuations occurring in the 100 blocks latency time the Oracles take to recalculate the lambda value. We therefore tested both the hypotheses: the hypothesis that the Oracles successfully predict the Gas price to pay to the miners (null hypothesis) and the hypothesis that their predictions are wrong (alternative hypothesis).

Equations 4, 5 respectively represent the null and the alternative hypotheses.

$$(H_0)\, \forall\, cat \in \{\textit{'fastest'}, \textit{'fast'}, \textit{'average'}, \textit{'safeLow'}\} : p >= 98\% \tag{4}$$

$$(H_a)\, \forall\, cat \in \{\textit{'fastest'}, \textit{'fast'}, \textit{'average'}, \textit{'safeLow'}\} : p < 98\% \tag{5}$$

Listing 1 represents the R code used to test the null hypothesis that the Oracles' prediction are right within a frame of 100 blocks time.

Listing 1: Test of Equal or Given Proportions

```
prop.test(x = transactions, p = 0.98, correct = FALSE,
          alternative = ''less'')
```

The variable x represents a two-dimensional table with 2 columns, which respectively provide the number of successful events (transactions included in the first n blocks) and failures (transactions included after the n-th block). The variable p represents the expected proportion of successful events (P_e), based on the Gas Oracles' predictions. The variable "alternative" specifies the proportion of successful events based on the alternative hypothesis.

Table 8 represents the results of the null hypothesis H_0 (Eq. 4) that the observed proportion (P_o) of transactions included in the blocks are equal or greater than the expected proportion ($P_e = 0.98$). The table is divided into four sections based on latency (2nd column). In particular, the first section represents the results based on 100 blocks latency, as claimed by the Oracles.

5.5. Evaluation of the Poisson Model

The Oracles assume that the observed data are distributed in accordance to the Poisson Model. Before checking whether the observed data actually

follow the Poisson distribution, we checked the null hypothesis (H_0) that the observed data are homogeneously distributed over time among all blocks. In other words, we tested the hypothesis that the observed data are distributed in accordance with the Equiprobable Model, which predicts that the transactions have the same probability of ending in any of the next n blocks. Equation 6 expresses the expected probability to have the transaction included in any block as predicted by the Equiprobable Model:

$$H_0 : p_1 = p_2 = \ldots = p_{200} = 0.005 \tag{6}$$

where p_i is the probability to have a transaction added to the i^{th} block and it goes from 1 to 200. Table 7 (1st section) presents the results of the comparison between the expected frequency and the observed frequency based on the Equiprobable Model.

Therefore, we tested the alternative hypothesis that the observed data follow a Poisson distribution with parameter $\lambda > 0$. Equation 7 expresses the expected probability to have the transaction included into a block i based on the Poisson Model.

$$H_0 : \forall i \in [0, 200), P(X = x) = \frac{i * \lambda^x * e^{-(i*\lambda)}}{x!}, x \in [0, \infty) \tag{7}$$

Listing 2 shows the R code used to calculate the expected frequency of transactions per block based on the Poisson Model.

Listing 2: R code to compute the expected counts of transactions

```
blocks = 1:200 #list of blocks
total = sum(observed)
expected =
    total * ((lambda^blocks)*exp(-lambda)) /
        factorial(blocks)
```

Table 7 (2nd section) presents the results of the comparison between the expected frequency and the observed frequency based on the Poisson Model. The results are divided into four categories, as per Oracles' definition.

5.6. Improving the Oracle Prediction

The Gas Oracles assume that the transactions' distributions have different lambda values of the PMF (Eq. 2), based on the Gas price. Figure 7 shows that the tail length of transactions' events is inversely proportional to the Gas price. Based on this assumption, the Oracles calculate the λ (and as a consequence the Gas price) every time 100 blocks are confirmed on the blockchain (100 blocks latency). We hypothesized that it is possible to improve the Oracles' performance by recalculating the λ of the PMF (Eq. 2) at intervals of time smaller than 100 blocks latency. Reducing the latency, we might indeed better take into account the possible network changes [10]. The network might have changed depending not only on the increasing vs. decreasing number of miners

Table 7: Null Hypothesis: the distribution of the transactions included in the blocks follows the Equiprobable vs Poisson Model

Model	Category	lambda	X-squared	df	p-value	Decision
Equiprobable	*'fastest'*	not applicable	864.1	199	$< .00001$	Rejected
	'fast'		910.9	199	$< .00001$	Rejected
	'average'		898.5	199	$< .00001$	Rejected
	'safeLow'		963.8	199	$< .00001$	Rejected
Poisson	*'fastest'*	1.1295	190.86	199	.648094	Accepted
	'fast'	1.3435	182.11	199	.799031	Accepted
	'average'	1.3437	175.82	199	.880322	Accepted
	'safeLow'	1.4973	182.12	199	.798881	Accepted

*$p < 0.05$ means that the hypothesis is rejected, as there is a statistically significant difference between the expected frequency and the observed frequency.

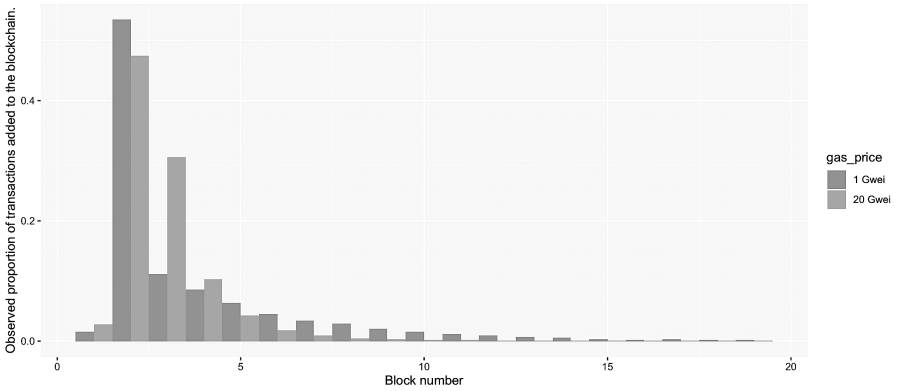


Figure 7: Histogram of observed data.

Table 8: Alternative Hypothesis: the observed proportion (P_o) of transactions included in the blocks at latencies < 100 blocks are equal or greater than the expected proportion (P_e) of the Gas Oracles.

Category	Latency	P_o	P_e	X-squared	df	p-value	Decision
'fastest'	100	0.92	0.98	1594.1	702	$<$ .00001	Rejected
'fast'	100	0.84	0.98	310.9	102	$<$ .00001	Rejected
'average'	100	0.91	0.98	698.5	389	$<$ .00001	Rejected
'safeLow'	100	0.90	0.98	1463.8	1071	$<$ .00001	Rejected
'fastest'	80	0.92	0.98	9212.8	601	$<$ .00001	Rejected
'fast'	80	0.89	0.98	763.9	89	$<$ .00001	Rejected
'average'	80	0.87	0.98	3295.2	301	$<$ .00001	Rejected
'safeLow'	80	0.99	0.98	1053.4	994	.093174	Accepted
'fastest'	60	0.96	0.98	523.2	402	.000042	Rejected
'fast'	60	0.99	0.98	101.3	81	.063046	Accepted
'average'	60	0.99	0.98	241.2	207	.051696	Accepted
'fastest'	4	0.99	0.98	230.1	201	0.077864	Accepted

*$p < 0.05$ means that the hypothesis is rejected, indicating that there is a statistically significant difference between the observed proportion (P_o) and the expected proportion (P_e).

and/or transactions in the network, but also and more importantly on the users' actual decisions on the Gas price to pay and time to wait. We therefore reduced the latency for each category, until the null hypothesis H_0 (Eq. 4) is accepted with a level of significance of $\alpha = 5\%$ ($p < 0.05$). Table 8 represents the results of the hypothesis H_0 (Eq. 4) that the observed proportion (P_o) of successful events are equal or greater than the expected proportion ($P_e = 0.98$) at latencies smaller than 100 blocks.

6. Discussion

The transactions data-set consists of over 10 million rows which covers a period of time of 3 months. This is a relatively small fraction compared with the total number of transactions in the same period, which should be around 77 million (`https://etherscan.io/chart/tx`). Of course, the blockchain networks can change for many reasons. Previous research [2] shows that the number of transactions moving through the Ethereum network can increase or decrease based on specific users-related events, for instance when a company looks to raise money to create a new coin, app, service, etc. or when it launches an ICO. Another scientific research [3] shows that there are several conditions under which mining infrastructures will be active or under which the miners will have no incentives to mine a given cryptocurrency due to the increase of the energy cost or unavailability of solar energy which can be used to make calculations at no cost. These or similar remarks are at the core of the idea that the Oracles' data-centered model might provide wrong predictions, because it does not take

into account the network changes depending on the users' actual behaviour or users-related events. This is the main reason why the research presented in this paper proposes a model shift, from a data-centered model to a user-oriented model for Oracles' *Gas price* predictions.

6.1. From a Data-Centered Perspective

The data-centered model is actually used by the Oracles, to provide the users with the predictions every 100 blocks confirmation. The model rely on data on the transactions history of the last 200 blocks confirmed on the blockchain. Our analysis showed that most transactions wait from one to two blocks before being included (see Table 3). The results of the analysis also showed that the Gas price influences the probability to have the transactions included into the next blocks. Figure 3a indeed shows that the shape of both the violin plots becomes larger at the decreasing of the Gas price. In particular, a transaction's Gas price higher than 10 GWei does not guarantee that the transaction is included within 1-2 blocks (30 seconds). The probability is anyway higher when compared to the transactions having a Gas price lower than 10 GWei.

We investigated the model actually followed by the Oracles to provide a *Gas price* prediction, i.e. the Poisson model. We defined the Poisson Model for the successful events of having a transaction included into a block. We pointed out that not all its conditions (defined in Section 4.4.1) are satisfied. We anyway checked whether the successful transactions were distributed in accordance with the Poisson Model instead of an equiprobable model. First, we tested the hypothesis that the transactions' distribution follow the Equiprobable Model. However, we found strong evidence that the Equiprobable Model does not fit the data, as per p-value = 2.2^{-16} inferior to 0.001. Second, although not all the conditions are met, we tested the alternative hypothesis that the successful transactions' distribution follow the Poisson Model. We found that the alternative hypothesis cannot be rejected with a confidence level of 95%. While in an Equiprobable Model it would make no sense to predict a *Gas price*, the Poisson Model does provide us with a meaningful insight to predict a *Gas price*, as it gives a lambda value which is inversely proportional to the transactions' Gas price. Indeed, as shown in Figure 7, the lower the transaction Gas price, the longer the queue of Poisson distribution is.

6.2. To a User-Oriented Perspective

By performing the *Poisson regression model* every 100 blocks, the Gas Oracles do not take into account all the changes in the Ethereum Network in real-time. As hypothesized in H1, the Gas Oracles' predictions based on the data-centered model are not reliable, especially because they compute the prediction every 30 minutes on average. We showed that the Oracles' predictions are not just accidentally wrong, due to possible statistical fluctuations in 100 blocks latency. We indeed found that the Oracles' prediction are actually wrong with a level of confidence of 98%. Moreover, we found that the margin of error is greater than declared (2%) and it is even 13% for the *'fastest'*. The greater the

latency, the more probable is that the Poisson model cannot take into account also user-related events or decisions occurring in the network.

Some special scenarios - our model can successfully manage - are those related to the network congestion which can happen when a company looks to raise money to create a new coin, app, or service [17]. These scenarios are special because they may imply a longer waiting time before the transaction is included and committed, because there are more transactors offering a higher Gas price when compared to a non-congested network state. One of the ways that may be used to reduce the waiting time is to make users aware that a higher fee is needed to have the transaction included into a block in a shorter time than usual. As our model for the fastest category is trained every 4 blocks, it can recommend Gas prices that reflect the current possibly congested situation instead of past situations when there was no congestion.

As to the research question RQ2, we suggested that there is a lack of correspondence between the Oracles' Gas price categories and the Gas price set by the users and exchanges. Of course, the users or the exchanges might have looked at the Gas Oracles' predictions, but they might also have acted independently, without looking at the Gas Oracles' predictions. However, it is reasonable to assume that both the users and exchanges, who set the Gas price equal to the Gas price suggested by the Oracle, might have either followed the Oracle's recommendation or have an interest in setting a Gas price corresponding to the category predicted by the Oracle. Indeed, even if they were not following the Oracle's recommendation in setting the same Gas price, it is likely that the users/exchanges might have agreed with the Gas price attached to the transaction and the waiting time. If the users/exchanges had disagreed with the Gas Price and the waiting time, they might have changed the Gas price to rely on the expected waiting time.

However, the analysis of the Gas price of the transactions in the transaction pool shows that just 16% of the transactions have a Gas price equal to the Gas price suggested by the Gas Oracle. The percentage of transactions having the Gas price equal to the Gas price suggested by the Oracle is distributed among the four categories as follows: 1) 7% *'safeLow'*, 2) 1% *'fast'* and *'average'*, 3) 8% *'fastest'*. Figure 6b presents the percentage of Gas price categories used in Ethereum blockchain. Table 5 shows how the mode for the *'fast'* and *'average'* categories are the same. This means that most times the Gas price is the same for both categories, in spite of being different categories in terms of execution time. The data analysis of the transactions waiting in the transaction pool to be included into a block also suggests that the categories *'fast'* and *'average'* are not set by the users probably because these categories do not correspond to their interests and/or needs. On the contrary, the categories *'fastest'* and *'safeLow'* are set the most. This means that, as hypothesized in H2, the users set a Gas price in relation to an interval time to include a transaction, which are not fully-fledged predicted by the default categories.

Both the Oracles present the same pattern of results: the distributions of the *'recommended Gas price'* are almost the same for Etherchain and EtherGasStation. The violin plots of both the Oracles also show the presence of two

different peaks of *'recommended Gas price'* at the same value. One of the peak corresponds to the third quartile value, *i.e.,* 20 GWei, while the other peak is below the median and it is equal to 10 GWei. Figure 6a shows the violin plot of the *Gas price* predictions.

We also showed, as hypothesized in H3, that a reduction of the margin of error in the Gas price prediction can be achieved by reducing the latency, thus considering the current changes of the network. The Oracles' prediction can indeed be improved, by reducing the latency of 100 blocks time. The results confirmed that the margin of error is reduced for all the categories, when the *Poisson regression model* is performed at shorter time intervals. Interestingly, the results also showed that the margin of error depends on the latency and it is different for each category. In particular, the category *'fast'* requires a shorter latency compared with the other categories. The *'fast'* category is indeed more demanding than the others in terms of waiting time, as the Gas Oracles estimate the Gas price to have the transaction included within two blocks at most. On the contrary, in the case of other categories, such as the *'safeLow'*, the Gas price does not need to be predicted so often.

6.3. Threats to Validity

We designed the model proposed in this study, by hypothesizing that a reduction of the margin of error in the Gas price prediction can be achieved by calculating the 'recommended Gas price' (dependent variable) based on a smaller interval of time (independent variable), when compared to the interval of time considered by the Gas Oracles. In this way, we could consider the current changes of the network in real time (such as a change in the number of transactions, the fees attached to these transactions, the number of miners and their policies). As to the internal validity of the study, it might be claimed that such changes are due to a variety of causes, ranging from an internal variation in the Blockchain network characteristics to other external causes, such as the varying of the currency pair's ETH/USD value, the changing behavior of companies looking to raise funds via an ICO, specific news about the cryptocurrency market that could lead investors to change their behavior. All these factors are not considered by the model proposed in the study, but they might be considered as variables that can in principle influence the recommended Gas price, as a result of a change in the Blockchain network status due to the users' behavior. By studying the changes of the Blockchain network, what we found is a correspondence between some Gas price categories proposed by the Oracles and the fees most selected by the users. The causes of the users' and/or brokers' behavior in the selection of the transaction fees depart from the aims of this study and need to be further investigated from other disciplinary perspectives.

As to the external validity of the study, the study focuses on two specific Oracles, but there are at least seven online Gas Oracles in the market and ongoing academic discussion on Gas Oracles, presenting different models that can be used to predict the Gas prices to attach to a transaction. We selected the most followed Oracles, namely Etherchain and EtherGasStation Oracles, based on GitHub metrics (number of forks and developer commits). However, there

are other Gas Oracles whose predictions may differ, as they are based on other models, in terms of value parameters or user categories. Table 9 reports the REST API service URI of the online Gas Oracles currently available on the market. Further research is therefore needed for a comprehensive evaluation of all the Gas Oracles reported in Table 9. However, it is reasonable to think that this study can be extended not only to other Oracles, but also to other cryptocurrency networks, such as the Bitcoin Blockchain and Litecoin Blockchain, which are based on the PoW consensus algorithm. Indeed, as for the Ethereum Blockchain, also for these Blockchains the model can provide a prediction for the Gas price (fee) to be attached to the transaction as an incentive for miners to solve the PoW puzzle.

Table 9: REST API service URI of the online Gas Oracles

Gas Oracle name	**REST API service URI**
EtherGasStation	https://ethgasstation.info/json/ethgasAPI.json
Upvest	https://fees.upvest.co/estimate_eth_fees
POA Network	https://gasprice.poa.network/
Etherchain	https://www.etherchain.org/api/gasPriceOracle
EtherScan	https://api.etherscan.io/api?module=gastracker&action=gasoracle&apikey=
GAS Now	https://www.gasnow.org/api/v3/gas/price?utm_source=:YourAPPName

7. Conclusions

The existing Gas Oracles are based on a data-centered model which relies on the analysis of the blocks data history to make the Gas price prediction, without considering any data on the categories set by the users or exchanges. To propose a user-oriented model of Gas Oracles' Gas price prediction, the paper explored both the overall validity of the Gas Oracles' predictions and the more specific validity of the Gas Oracles' Gas price categories, looking at the (lack of) correspondence with the categories set by the users.

The study first evaluated the validity of the Gas Oracles' predictions on the Gas price to pay to have the transaction recorded in the blockchain. It revealed that both Etherchain and EtherGasStation predict with a margin of error at least twice as much as the margin of error they declare. For instance, the *'fastest'* category showed a 4% margin of error, while the *'fast'* category showed a 28% margin of error. The user-oriented model proposed in the paper gives a prospective contribution to the improvement of the Gas Oracles' predictions to better indicate the categories that correspond to the users' requirements and the Equation that best provides them with a more effective Gas price to set.

The study shows that the four Gas price categories proposed by both the Oracles do not correspond with the categories set by the users and/or companies. As a result of the analysis, we found indeed that less than 1% of transactions set the Gas price suggested by the Gas Oracle in the categories *'average'* and *'fast'*. On the contrary, we found that it is worth predicting the Gas price for

the *'fastest'* and *'safeLow'* categories, as they make sense in terms of users' interests.

The paper contributes to the understanding of the Equation the Gas Oracles should use to provide the users' with a better Gas price prediction. The user-oriented model we propose recommends indeed to calculate the Gas price by reducing the latency of 100 blocks time to have a lower margin of error compared to the Oracles' actual one. The study shows that, by reducing the latency to perform the *Poisson regression model*:

- the error margin of the prediction for the *'fastest'* category is 2% compared to the 4% of the Gas Oracles' prediction. In this case, we performed the *Poisson regression model* every 4 blocks instead of 100 blocks.
- the error margin of the prediction for the *'average'* and *'fastest'* categories is 1% compared to the 14% and 13% of the Gas Oracles' prediction. In this case, we performed the *Poisson regression model* every 60 blocks instead of 100 blocks.
- the error margin of the prediction for the *'safeLow'* category is 1% compared to the 4% of the Gas Oracles' prediction. In this case, we performed the *Poisson regression model* every 80 blocks instead of 100 blocks.

The model can provide the users with a better estimation, because it can take into account the current changes of the blockchain and the users' network in real time.

The Gas Oracles suggest to the user and/or to the exchanges the Gas price to attach to a transaction to have the transaction included in a block in a certain amount of time. The information might be misleading for the users because the transaction inclusion in a block does not guarantee that the transaction will be also confirmed. Indeed, the exchange platforms consider a number of confirmation blocks before considering the transaction confirmed in the main chain. Furthermore, when the transaction involves a high value asset, the number of confirmation blocks to consider the transaction confirmed is likely to be even higher compared to a transaction with a low value asset. Therefore, further research should be carried out to tune up a user-centered model that suggests the users/exchanges the fee to pay to have the transactions included in a block within a certain number of confirmation blocks, depending on the value to be transferred in the transaction.

Acknowledgment

Acknowledgements without for the blind review.

References

[1] AA.VV. (2020). Oracle-gas-price-source-code, may 2020.

[2] Adhami, S., Giudici, G., and Martinazzi, S. (2018). Why do businesses go crypto? An empirical analysis of initial coin offerings. *Journal of Economics and Business*, 100(C):64–75.

[3] Altman, E., Menasché, D., Reiffers-Masson, A., Datar, M., Dhamal, S., Touati, C., and El-Azouzi, R. (2020). Blockchain competition between miners: A game theoretic perspective. *Frontiers in Blockchain*, 2:26.

[4] Buterin, V. (2014). A next generation smart contract & decentralized application platform. *Ethereum White Paper*, pages 1–36.

[5] Chen, T., Li, X., Luo, X., and Zhang, X. (2017). Under-optimized smart contracts devour your money. In *SANER(IEEE International Conferece on Software Analysis, Evolution, and Reengineering) 2017*.

[6] Chen, T., Li, Z., Zhu, Y., Chen, J., Luo, X., Lui, J. C.-S., Lin, X., and Zhang, X. (2020). Understanding ethereum via graph analysis. *ACM Trans. Internet Technol.*, 20(2).

[7] Coxe, S., West, S., and Aiken, L. (2009). The analysis of count data: A gentle introduction to poisson regression and its alternatives. *Journal of personality assessment*, 91:121–36.

[8] Decker, C. and Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10.

[9] Ducasse, S., Rocha, H., Bragagnolo, S., Denker, M., and Francomme, C. (2019). Smartanvil: Open-source tool suite for smart contract analysis. In *Blockchain and Web 3.0: Social, economic, and technological challenges*. Routledge.

[10] Dyson, S. F., Buchanan, W. J., and Bell, L. (2020). Scenario-based creation and digital investigation of ethereum erc20 tokens. *Forensic Science International: Digital Investigation*, 32:200894.

[11] Efanov, D. and Roschin, P. (2018). The all-pervasiveness of the blockchain technology. *Procedia Computer Science*, 123:116 – 121. 8th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2017 (Eighth Annual Meeting of the BICA Society), held August 1-6, 2017 in Moscow, Russia.

[12] Eklund, P. W. and Beck, R. (2019). Factors that impact blockchain scalability. In *Proceedings of the 11th International Conference on Management of Digital EcoSystems*, MEDES '19, page 126–133, New York, NY, USA. Association for Computing Machinery.

[13] Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., and Smaragdakis, Y. (2018). Madmax: Surviving out-of-gas conditions in ethereum smart contracts. *Proc. ACM Program. Lang.*, 2(OOPSLA).

[14] Gundlach, R., Gijsbers, M., Koops, D., and Resing, J. (2021). Predicting confirmation times of bitcoin transactions. *ACM SIGMETRICS Performance Evaluation Review*, 48(4):16–19.

[15] Kanda, R. and Shudo, K. (2019). Estimation of data propagation time on the bitcoin network. In *Proceedings of the Asian Internet Engineering Conference*, AINTEC '19, pages 47–52, New York, NY, USA. ACM.

[16] Kochovski, P., Gec, S., Stankovski, V., Bajec, M., and Drobintsev, P. D. (2019). Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Generation Computer Systems*, 101:747 – 759.

[17] Lee, J. Y. (2019). A decentralized token economy: How blockchain and cryptocurrency can revolutionize business. *Business Horizons*, 62(6):773 – 784. Digital Transformation and Disruption.

[18] Liu, X., Muhammad, K., Lloret, J., Chen, Y.-W., and Yuan, S.-M. (2019). Elastic and cost-effective data carrier architecture for smart contract in blockchain. *Future Generation Computer Systems*, 100:590 – 599.

[19] Lo, S. K., Xu, X., Staples, M., and Yao, L. (2020). Reliability analysis for blockchain oracles. *Computers and Electrical Engineering*, 83:106582.

[20] Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 254–269, New York, NY, USA. Association for Computing Machinery.

[21] Oliveira, V. C., Almeida Valadares, J., A. Sousa, J. E., Borges Vieira, A., Bernardino, H. S., Moraes Villela, S., and Dias Goncalves, G. (2021). Analyzing transaction confirmation in ethereum using machine learning techniques. *ACM SIGMETRICS Performance Evaluation Review*, 48(4):12–15.

[22] Pierro, G. and Rocha, H. (2019). The influence factors on ethereum transaction fees. In *2nd International Workshop on Emerging Trends in Software Engineering for Blockchain*, WETSEB '19, pages 24–31, Piscataway, NJ, USA. IEEE Press.

[23] Pierro, G. A. (2020). Oracles data-set.

[24] Pierro, G. A., Rocha, H., Tonelli, R., and Ducasse, S. (2020). Are the gas prices oracle reliable? a case study using the ethgasstation. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 1–8.

[25] Pinzón, C. and Rocha, C. (2016). Double-spend attack models with time advantange for bitcoin. *Electronic Notes in Theoretical Computer Science*, 329:79 – 103. CLEI 2016 - The Latin American Computing Conference.

[26] Salimitari, M., Chatterjee, M., and Fallah, Y. P. (2020). A survey on consensus methods in blockchain for resource-constrained iot networks. *Internet of Things*, page 100212.

[27] Silva, P., Vavricka, D., Barreto, J., and Matos, M. (2020). Impact of geo-distribution and mining pools on blockchains: A study of ethereum. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 245–252.

[28] Singh, H. J. and Hafid, A. S. (2019). Transaction confirmation time prediction in ethereum blockchain using machine learning. `https://arxiv.org/pdf/1911.11592`.

[29] Singh, H. J. and Hafid, A. S. (2020). Prediction of transaction confirmation time in ethereum blockchain using machine learning. In Prieto, J., Das, A. K., Ferretti, S., Pinto, A., and Corchado, J. M., editors, *Blockchain and Applications*, pages 126–133, Cham. Springer International Publishing.

[30] Stoepker, I., Gundlach, R., and Kapodistria, S. (2021). Robustness analysis of bitcoin confirmation times. *ACM SIGMETRICS Performance Evaluation Review*, 48(4):20–23.

[31] Sun, J., Tang, P., and Zeng, Y. (2020). Games of miners. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1323–1331, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[32] Vujivic, D., Jagodic, D., and Randjic, S. (2018). Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE.

[33] Weber, I., Gramoli, V., Ponomarev, A., Staples, M., Holz, R., Tran, A. B., and Rimba, P. (2017). On availability for blockchain-based systems. pages 64–73.

[34] Wood, G. (2019). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper. Byzantium version 7e819ec.*