Blockchain: Research and Applications 3 (2022) 100074

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Research Article

Transaction fees optimization in the Ethereum blockchain

Arnaud Laurent [a,*], Luce Brotcorne [a], Bernard Fortz [a,b]

[a] INRIA Lille Nord-Europe, 40 Avenue Halley, Lille, 59000, France
[b] Université libre de Bruxelles, Bruxelles, 1050, Belgium

ARTICLE INFO

Keywords:
Pricing
Blockchain
Ethereum
Optimization
Gasprice
Ether
Monte-Carlo

ABSTRACT

In blockchains, transaction fees are fixed by the users. The probability for a transaction to be processed quickly increases with the fee level. In this paper, we study the transaction fee optimization problem in the Ethereum blockchain. This problem consists of determining the minimum price a user should pay so that its transaction is processed with a given probability in a given amount of time. To reach this goal, we define a new solution method based on a Monte Carlo approach to predict the probability that a transaction will be mined within a given time limit. Numerical results on real data highlight the quality of the results.

1. Introduction

In 2008, Satoshi Nakamoto (an alias) introduced the first digital cash technology, which simultaneously has no backing or intrinsic value and no centralized issuer or controller: Bitcoin. This technology is the first implementation of cryptocurrency based on the blockchain principle. Five years later, Vitalik Buterin proposed a new technology based on blockchain: Ethereum [1]. The idea behind this technology is to enhance Bitcoin by developing a blockchain with a built-in fully fledged Turing-complete programming language. This blockchain is now used all over the world to support decentralized applications.

The blockchain is a transaction-based system. Every 15 s on average, in Ethereum, a new block is created and added to the main chain. These blocks contain the newly validated transactions impacting the system state. These blocks are created by miners, who are actors in the system. In most blockchains, the users set the price that they are willing to pay to miners as an incentive for selecting their transactions. Depending on the amount paid by the users, the number of blocks awaited before being mined can vary tremendously. The objective of this paper is to propose a method for Ethereum's users to predict the minimum fees (gasprice) such that their transactions will be mined in a reasonable time. In order to do that, we propose a prediction and optimization method to determine the minimum gasprice a user should pay for its transaction to be mined within a given time limit T, with a given probability α.

In Section 2, we first define the Ethereum blockchain operations: the block creation and the transaction acceptance in the system. Section 3 gives a brief literature review of works related to the cost of transactions for the users in the Ethereum blockchain. We next define in Section 4 the gasprice optimization problem and, in Section 5, a deterministic mathematical model for the computation of the mining probabilities, assuming that the transactions are known at time 0. In Section 6, we describe an algorithm based on the Monte-Carlo method and a binary search algorithm to solve the gas pricing problem. In Section 7, numerical results are provided for the prediction method on real data. Finally, we draw some conclusions on the performance of our methods.

2. The Ethereum blockchain

This section is devoted to the definition of the Ethereum blockchain. This blockchain has been proposed to be used as support for a cryptocurrency and to support decentralized applications. We refer to Ref. [2] for a taxonomy of all blockchains where the authors classify their architectures and protocols.

2.1. The Ethereum operation

The Ethereum blockchain is a decentralized ledger of transactions. No entity is controlling a blockchain, there are only actors participating in its growth and integrity. Based on the Proof of Work protocol, the system cannot be hacked, modified, or shut down by any of the users.

The transactions in a blockchain are the mechanism for external users to interact with the system. A currency, whose units are called tokens, is

* Corresponding author.
E-mail address: arnaud.laurent@inria.fr (A. Laurent).

https://doi.org/10.1016/j.bcra.2022.100074
Received 6 May 2021; Received in revised form 31 January 2022; Accepted 6 March 2022

associated with each blockchain. These tokens can be transferred through transactions, like bitcoins in the Bitcoin blockchain or ethers in Ethereum. A unit of ether is called a Wei, and one billion Wei is a GWei. A transaction is always sent from one account and interacts with other accounts. The classical accounts are called "Externally owned accounts" and are owned by one or a group of persons. They are characterized by a token balance, a private key, and a public key.

The private key is used to create the signature, which ensures that the transaction is sent by the account owner. Every transaction is associated with a nonce corresponding to the number of transactions previously sent by the same account.

As its name indicates, the blockchain is a chain of blocks. Each block contains the transactions validated by the actors impacting the system state called miners. A block can only contain transactions:

- Well structured with all the fields needed (sender address, receiver address, value sent, signature, …);
- Associated with a sender account having a token balance greater than the amount required by the transaction (the value sent plus the fee of the transaction);
- Such that the other transactions with a lower nonce sent by the same account are in a block (possibly the current one).

The transactions that cannot be processed for one of the last two reasons remain in the pending transactions list.

Each block contains the hashcode of the previous block in the chain (called its parent block), thus forming a chain of blocks that can't be broken. If someone tries to modify a block in the chain, the chain will be broken and it will be rejected by all the actors in the blockchain. Fig. 1 depicts a part of a blockchain.

At fixed time intervals, the miners create a block by selecting transactions to include in the blockchain. They can select any valid transaction based on their criteria within the size limit of the block, depending on the blockchain implementation.

A miner gets two rewards per block mined. The first one is fixed and does not depend on the transactions in the block. The second one is the fees of the transactions included in the block. The transaction fee is fixed by the sender. Only one miner is selected to create the next block linked to the chain of blocks.

The protocol to select the next miner is called "Proof of Work (PoW)". Each miner has to find a block with a hashcode lower than a given target. The target depends on the hashpower of the system in order to keep the average time to find a valid block between 10 and 19 s. This process is memory-less, which means that the probability of finding a valid block does not change with time. Consequently, the time distribution of blocks follows a Poisson law [3].

The first miner creating a valid block sends it to the other users, who verify the validity of the block and add it to their own chain.

2.2. Smart contract and the gas mechanism

The main specificity of the Ethereum blockchain relies on the built-in fully fledged almost Turing-complete programming language called Solidity. This feature allows us to create what are usually called "smart contracts". The term was first introduced by Nick Szabo [4] to "combine protocols with user interfaces to formalize and secure relationships over computer networks". The name smart contract is nowadays mostly used to describe automatic computations executed in a blockchain. For example, "send 10 tokens to someone at a given date if a condition is satisfied".

In Ethereum, smart contracts are stored within "contract accounts", which can only be initiated with a contract creation transaction containing the code of the contract. When such an account receives a transaction, possibly with some data, the code of the contract is executed.

The concept of smart contract introduces new issues for blockchain operation management:

- If an infinite loop runs in a smart contract, the miners will get stuck while running it.
- Transactions calling a contract consume more resources than other transactions, moreover, a miner does not know the transaction cost until they run it.

The gas mechanism has been introduced in Ethereum to prevent these issues. The action of sending a transaction costs a minimum of 21000 gas. Each operation in a contract has a predefined cost in gas (see Fig. 2).

To prevent the execution of infinite loops, each transaction has a limit on the amount of gas that can be consumed. This value is called the gaslimit and is set by the user sending the transaction. The miner executes the transaction up to the end of the contract or when they run out of gas. In the case when a contract runs out of gas, the effects of the transaction are cancelled but the fees are paid to the miner. The possibility for a program to stop before its natural end, classify solidity as an almost Turing-complete programming language. As it is not always possible to predict the gas consumption of a transaction, the users usually overestimate the gaslimit in order to ensure that the transaction will be performed.

The fee paid to the miner depends on the amount of gas consumed, and is linked to the resource consumption. The gasprice is the price in

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	Arithmetic operation
ADDMOD/MULMOD	8	Arithmetic operation
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/SGT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	Stack operation
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/20,000	Storage operation
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

Fig. 2. The gas cost of operations in Ethereum [5].

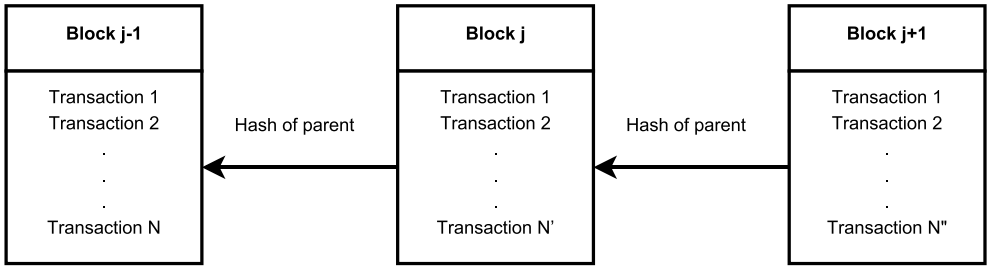

Fig. 1. The chain of blocks.

ether (the Ethereum token) determined by the user for every unit of gas used by their transaction. The fee of a transaction is given by:

$$\text{fee} = \text{gasprice} \times \text{gasConsumed} \quad (1)$$

To maximize their income per block, the miners usually add the transactions with the highest gasprice in their blocks. Note that the lower gasprice limit of a miner is the lowest gasprice value for which they accept transactions in their blocks.

The gaslimit of a block is the maximum amount of gas that can be consumed by all the transactions contained in this block. If the sum of gas consumed by transactions in a block is higher than its gaslimit, the block is not valid. The value of the gaslimit for a block can vary by a small percentage through its miner's vote.

3. Related work

In this section, we present a brief literature review on the paper dealing with the fees for users in Ethereum. The Ethereum yellow paper [5] is the specification of the Ethereum blockchain and its operation. The work of Antonopoulos and Wood [6] provided a guide to understand and use the Ethereum blockchain. Miller [7] presented the mechanisms that incentivize the miners, the clients, and the network to cooperate to make the Ethereum blockchain functional.

We first focus on the main papers related to the gasprice mechanism. Many papers deal with the Ethereum smart contracts; for example, Grech et al. [8] defined detection methods for the out-of-gas vulnerabilities in the Ethereum smart contracts. For a similar problem, Albert et al. [9] defined a method to avoid out-of-gas vulnerabilities of smart contracts by computing their gas consumption upper bound. Marescotti et al. [10] also proposed an algorithm to compute the exact gas consumption of a smart contract in the worst case. Atzei et al. [11] listed potential vulnerabilities of the Ethereum blockchain in smart contracts. The gas consumed by a contract depends on its implementation: Chen et al. [12] analysed the gas cost of smart contracts and identify patterns that generate over-consumption of gas. They also presented a tool to automatically detect these bad patterns in the bytecode of the contracts. Ethereum is a system in constant evolution, and one of the EIP (Ethereum Improvement Proposal) concerns the gasprice mechanism: the EIP-1559[1]. It has been proposed to reduce the volatility of transaction fees. The goal of this EIP is to replace the gasprice value with two distinct values in transactions (BASEFEE and PREMIUMFEE). At the present time, this proposition is still in progress and no date of deployment in the Ethereum blockchain has been selected.

Chen et al. [13] proposed a new dynamical way to determine the gas cost of operations in order to avoid the underpriced gas cost of operations leading to DoS attacks. Other studies focus on the gasprice mechanism. Weber et al. [14] analysed in Bitcoin and Ethereum the commitment of the transactions and the possible reasons why a transaction can be stuck in the system. They analysed statistically the impact of gasprice on the confirmation time for transactions. They showed that delays are shorter when the gasprice is higher but did not find any correlation between the confirmation time of a transaction and the gas limit value of transactions. They also experimented with cancelling transactions by sending another transaction with the same nonce and a higher gasprice. The chance of cancelling a transaction is higher when the initial gasprice is low.

Some papers studied the factors influencing the processing time of transactions, such as Pierro and Rocha [15]. They pointed out that the major factor of influence is the size of the pending list of transactions and that this value highly depends on the number of blocks mined per minute. This second correlation is only due to the variance of block creation following a Poisson law. de Azevedo Sousa et al. [16] provided an analysis showing that the correlation between fees and the pending time of transactions does not exist. This surprising result is surely related to the fact that the authors did not withdraw the transactions that cannot be processed before an event (precedence, lack of tokens, …). Donmez and Karaivanov [17] studied the correlation between different factors and the gas price median used in mined transactions recorded in the blockchain. They showed that block utilization is highly correlated to the median gas price in blocks.

Concerning the Bitcoin blockchain, Easley et al. [18] studied the influence of fees on miners, users, and the block size limit. They showed that the fees ensure the sustainability of the system. Ricci et al. [19] proposed a queuing model to characterize the delay for transactions to be added to the Bitcoin blockchain. Koops [20] proposed modelling the prediction problem of the validation time of Bitcoin transactions as a Cramér-Lundberg process.

Finally, some tools exist to provide predictions on the gasprice that users should use to broadcast their transactions, such as ETH Gas Station[2]. This oracle uses the last 200 blocks to predict the lowest gasprice value to set for a transaction to be mined in a given period of time. This statistical method is only based on the history of mined blocks. Thus, if the transactions waiting list gets congested, the prediction will not be influenced until the gasprice required rises in the coming blocks. The same thing happens if a user sends a huge number of transactions with a high gasprice value. Pierro et al. [21,22] studied the reliability of the ETH Gas Station and Ether Chain oracle. ETH Gas Station claims to have a 2% margin of error and their study shows that this margin is in reality between 4% and 28%, depending on the speed desired by the users. A machine learning approach has been proposed to predict the transaction confirmation time [23]. The prediction is a date statistically "the closest" to the future mining date of a transaction. With a similar objective, Liu et al. [24] proposed a machine learning regression-based gas price predicting approach aiming to find the lowest transaction gas price in the next blocks. The authors compared the Random Forest and the Multi-Layer Perception to the Ethereum Gas Station prediction and the Random Forest method gives the best predictions. Werner et al. [25] studied the daily seasonality of the optimal gasprice data. They proposed a deep learning price-forecasting parametrized by an urgency parameter, representing the user will. Carl and Ewerhart [26] also studied the seasonality of the gas price in Ethereum. They proposed a seasonal auto-regressive integrated moving average to predict the hourly median of the threshold gas price.

A few recent works have been published to predict a "good" gasprice to proceed with a transaction. Mars et al. [27] aimed to predict the lowest gasprice of a transaction to be added in a block in the next period of time. The gas price is determined on the basis of the Prophet model and deep learning models like Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Similarly, Chuang and Lee [28] addressed the same problem by relying on a different method. More precisely, they use a Bayesian approach to predict the lowest gasprice that will be added to the next "m" blocks. Finally, Lien and Su [29] provided a customizable forecasting framework without presenting a prediction method.

To our knowledge, no paper jointly addresses the three characteristics of the prediction, such as the price of the gas (see Table 1), the period of time desired, and the probability of being mined within the period of time. Most papers on gas price prediction aim at providing a low value for the transaction gas price to be mined in a reasonable time. Since the higher the gas price will be, the higher the probability of being mined will be, it is impossible to compare these methods to each other. In this paper, we aim at providing a configurable method that predicts the lowest gas price, ensuring that a transaction is mined in a given period of time with a given probability.

[1] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md.

[2] https://ethgasstation.info/.

Table 1
Comparison of the different literature oracles.

Methods	Consider pending list of transactions	Consider gas limit	Configurable limit of time	Configurable probability of success
ETH Gas Station	No	No	Speed indicator	No
Ref. [23]	Yes	No	No	No
Ref. [24]	No	Yes	No	No
Ref. [25]	No	No	Speed indicator	No
Ref. [27]	No	No	Yes	No
Our method	Yes	Yes	Yes	Yes

4. The gas pricing problem

The problem studied in this paper consists of determining the lowest gasprice gp_{tr} such that a transaction tr is added to a block within a time limit T with a probability of at least α. We name the event when the transaction tr is added to a block before time T:$V_{tr}(T)$. This event has a probability $P(V_{tr}(T))$ to happen. Our goal is to determine the lowest gasprice such that this probability is higher than α. To be mined, a transaction μ must be valid, have its predecessor $prec_\mu$ already added to a block, and be selected by the miner $m \in M$ that finds the next block. The transaction $prec_\mu$ is the last transaction sent before μ by the same account. For the consistency of the system, the transactions have to be processed in the same order as they were sent (see Section 2.1).

The miner selects a transaction μ depending on several criteria, such as its gasprice gp_μ, its gaslimit gl_μ, or its nonce. The most commonly used strategy is to rank by gasprice while respecting the precedence constraints on nonces.

To avoid the underpayment of transactions, miners set a lower limit on the gasprice that they are willing to accept (see Fig. 3). We define $LGP(gp)$ as the probability that the miner of the next block accepts a transaction with a gasprice greater than or equal to gp.

Each block has a limit BL of gas that can be consumed by all the transactions that it contains. The creation of the blocks follows a Poisson law of parameter λ [3].

At time 0, the n transactions previously broadcast and not yet added to a block are known, with their gasprice gp_μ and their gaslimit gl_μ, $\mu \in \{1, \ldots, n\}$. The gas consumption value gc_μ of a transaction μ is not known until μ is added to a block and processed. This is why, before adding transaction μ to a block, a miner considers the maximum possible consumption, given by the gaslimit gl_μ.

During a period of duration T, several transactions $\mu \in \{n+1, \ldots, N\}$ are broadcast. These future transactions are not known but can be predicted based on historical data. In order to do that, the set of validated blocks V, the set of transactions U in these blocks, and the set of pending transactions W are known.

The objective function is to minimize the gasprice gp_{tr} for a new transaction tr under the constraint that the probability $P(V_{tr}(T))$ of event $V_{tr}(T)$ is greater than or equal to α, leading to the following problem:

$$\min \quad gp_{tr} \tag{2}$$

$$P(V_{tr}(T)) \geq \alpha \tag{3}$$

The difficulty of this problem lies in the computation of the probability $P(V_{tr}(T))$. If we can compute $P(V_{tr}(T))$ for a fixed value of the single variable gp_{tr}, finding the optimal gasprice value is straightforward with a binary search algorithm, as detailed in Section 6.3. In the following sections, we present two approaches to computing $P(V_{tr}(T))$ for a fixed value of gp_{tr}.

The first one assumes that the processing order of transactions is deterministic. This means that all miners add transactions in the same order. It implies that:

- All the miners follow the same strategy.
- No transactions are received by a miner between time 0 and time T.
- A transaction can be added to a block only if all the transactions with a better rank in the strategy applied by the miners are already in a block.

For the second approach, we introduce the uncertainty of the problem. To model the unknown future broadcast of transactions, we consider a set of SC scenarios. Each scenario sc describes a potential future set of transactions. In this model, the order of processing of the transactions depends on the space left in the block and the date of mining of the blocks.

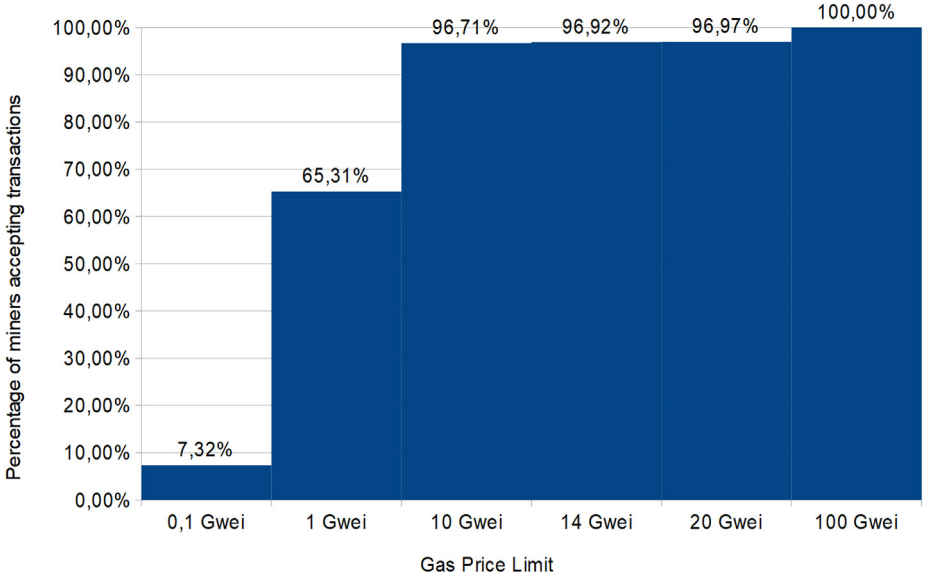

Fig. 3. Example of distribution of the lower gasprice limit of miners collected, in GWei.

Table 2
Frequent notations used throughout the paper.

μ	A transaction
tr	The transaction that needs to be priced
T	Period of time considered for the prediction
α	The probability required by the user that the transaction tr will be mined before time T
$V_{tr}(T)$	The event "tr is added to a block during the period T"
$P(V_{tr}(T))$	The probability that the transaction tr is mined in the period T
gl_μ	Gas limit of transaction μ
gc_μ	Gas consumed by transaction μ
gp_μ	Gasprice of transaction μ
BL	The limit of gas that can be consumed in a block
$prec_\mu$	The last transaction broadcast before μ by the same user
λ	Expected number of blocks mined during a period of T units of time
V	The set of validated blocks
U	The set of transactions in validated blocks
W	The set of pending transactions
n	The number of pending transactions
N	The number of pending transactions and transactions broadcast during the period T
T'	The past period of time used to extract the set U, V, and W
W'	The set of transactions broadcast during the period T, added in the waiting list
σ	A sequence of transactions
$FB(gc)$	The function is equal to 1 if $gc \leq BL$, 0 otherwise
$LGP(gp)$	The probability that the next miner mining the block accept transactions with a gasprice gp
$W_{\mu,b}$	The event "the transaction μ is in the waiting list when the block b is mined"
$A_{\mu,b}$	The event "the transaction μ is added in the block b"
sc	Potential future scenario for the broadcasting of transaction during T
SC	The number of scenarios considered
X_{sc}	Equals 1 if the transaction tr is added to a block during the period T in scenario sc, 0 otherwise
$B_{\mu,sc}$	The date of broadcasting of the transaction μ in scenario sc
d^α	The date from which the probability that the transaction tr is mined is greater than α
rdm_{tr}	The real date when the transaction tr is added to a block
Y_{tr}	Equals 1 if the predicted date is greater than the real date of mining for the transaction tr, 0 otherwise
rsf_α	The ratio of transactions mined before the predicted date d^α

For ease of reading, we introduce the notations used in this paper in Table 2.

5. The deterministic case

In this section, we present the computation of the probability $P(V_{tr}(T))$ that a transaction tr is mined in a given period of T units of time, given its gasprice gp_{tr} and an ordered list of transaction σ with $tr \in \sigma$.

5.1. Probabilities computation

We consider that each miner follows the same ordering strategy as the transactions by their gasprice, and all transactions are known at time 0. The data are composed of a sequence of N transactions $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ where σ_j, $j \in \{1, \ldots, N\}$, is the next transaction to add when all transactions $\sigma_{j'}$ with $j' < j$ have already been added to a block.

For each transaction μ, the following data are known:

gl_μ Gas limit of transaction μ

gc_μ Gas consumed by transaction μ

gp_μ Gasprice of transaction μ.

Our objective is to compute the probability $P(V_{tr}(T))$ that transaction tr is mined in a given period of T units of time, given its gasprice gp_{tr}.

The following additional parameters are needed:

λ Expected number of blocks mined during a period of T units of time.

$$FB(gc) = \begin{cases} 1 & \text{if } gc \leq BL, \\ 0 & \text{otherwise.} \end{cases}.$$

$LGP(gp)$ The probability distribution function of the lower gasprice limit of the miners.

We also introduce the following variables:

$P(W_{\mu,b})$ Probability for transaction μ=1, .., N to be in the waiting list for being added to block b.

$P(A_{\mu,b})$ Probability of adding the transaction μ=1, .., N in a block $b \in \mathbb{N}$

The probability that the first transaction σ_1 is added to the first block is given by:

$$P(A_{\sigma_1,1}) = FB(gl_{\sigma_1}) \times LGP(gp_{\sigma_1}) \tag{4}$$

The probabilities FB and LGP are assumed to be independent, so the product of these probabilities is the probability that the first task is added in the first block. The gas limit of transaction 1 is considered because the gas consumption of this transaction is not yet known at this point by the miner.

For all the transactions σ_j, the probability of being mined in block 1 depends on the $j-1$ first transaction of the σ order. The filling block function must also consider the gas consumption of the transaction already added to block 1. This probability is given by:

$$P(A_{\sigma_j,1}) = FB\left(\sum_{i=1}^{j-1} gc_{\sigma_i} + gl_{\sigma_j}\right) \times LGP(gp_{\sigma_j}) \tag{5}$$

The probability $P(W_{\mu,b})$ that the transaction μ is still in the waiting list for a block $b \geq 2$ is given by:

$$P(W_{\sigma_j,1}) = 1 \tag{6}$$

$$P(W_{\sigma_j,b}) = P(W_{\sigma_j,b-1}) - P(A_{\sigma_j,b-1});\ b \geq 2 \tag{7}$$

For a block $b > 2$, the previously mined blocks have to be taken into account to compute the probability for the transaction σ_j to stay in the waiting list and be added to block b. To do that we consider the $|j|$ possibilities for the first transaction in σ that is still in the waiting list when block b is mined. The probability that σ_k is the first transaction in the waiting list, is equal to the probability that σ_k is still in the waiting list and that σ_{k-1} is not. If σ_k is the first transaction still in the waiting list, the filling of the block is equal to the gas consumed by transactions between positions k and j.

$$\begin{aligned} P(A_{\sigma_j,b}) = &\textstyle\sum_{k=2}^{j}(P(W_{\sigma_k,b}) - P(W_{\sigma_{k-1},b})) \\ &\times FB\left(\textstyle\sum_{i=k}^{j-1} gc_{\sigma_i} + gl_{\sigma_j}\right) \times LGP(gp_{\sigma_j}), \end{aligned} \tag{8}$$

$$\begin{aligned} P(W_{\sigma_j,b} \cap \overline{W_{\sigma_{j-1},b}}) &= P(W_{\sigma_j,b} \cup W_{\sigma_{j-1},b}) - P(W_{\sigma_{j-1},b}) \\ &= P(W_{\sigma_j,b}) - P(W_{\sigma_{j-1},b}). \end{aligned} \tag{9}$$

The computation of $P(W_{\sigma_j,b})$ requires $\frac{j(j+1)}{2}b$ operations. Using these values, we can compute the probability $P(V_{tr}(T))$ that the transaction tr with gasprice gp_{tr} is added to the blockchain before time T as

$$P(V_{tr}(T)) = \sum_{k=1}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} P(W_{tr,k+1}) \tag{10}$$

The probability in Eq. (10) is computed using the Poisson law distribution, as the number of blocks created during a period of time follows a Poisson law [3].

6. The stochastic case

The major drawback of the deterministic case presented above is the assumption that all transactions and their order of processing are known in advance. In this section, we lift these restrictions by modelling the arrival of future transactions through a set of equally probable scenarios. This requires taking more elements into account as follows:

- The arrival of transactions between time 0 and T,
- The precedence constraints between transactions from the same sender,
- The possibility to skip transactions that cannot be added to a block.

6.1. Assumptions and notation

To model the stochastic arrival of future transactions, we use a set of SC different scenarios, indexed by $sc \in \{1, \ldots, SC\}$. Each scenario sc is composed of N_{sc} transactions. The first n transactions are already broadcast at time 0, while each future transaction $\mu \in \{n+1, \ldots, N_{sc}\}$ has a date of broadcasting $B_{\mu,sc} \in [0, T]$, a gasprice $gp_{\mu,sc}$, a gaslimit $gl_{\mu,sc}$ and an amount of gas consumed $gc_{\mu,sc}$. For the other transactions $\mu \in \{1, \ldots, n\}$ already broadcast, the gas limit and gas price are the same for all scenarios, but the gas consumed might differ.

$prec_\mu$ Transaction with the same sender account and nonce -1 that must be added in a block before transaction μ.

Each scenario $sc \in \{1, \ldots, SC\}$ is characterized by a sequence of transactions σ^{sc} such that:

- $\sigma^{sc} = (\sigma_1^{sc}, \sigma_2^{sc}, \ldots, \sigma_{N^{sc}}^{sc})$ is a sequence of N^{sc} transactions where σ_j^{sc} will be the next transaction to add to a block b if all transactions $\sigma_{j'}^{sc}$ with $j' < j$ are already added to a block or cannot be added to the current block.

A transaction $\mu \in \sigma^{sc}$ cannot be added to a block b if:

- Its gasprice gp_μ is lower than the lower gasprice limit of the miner of block b.
- The gas consumption of the transactions already added to the block plus the gaslimit of transaction μ is greater than the block gaslimit BL. The gas limit of the transaction μ is considered instead of the gas consumption because when the miner selects a transaction, they only know its gas limit. Once added to the block, the miner knows the gas consumption of the transaction and thus the exact possible gas consumption remaining.
- The transaction $prec_\mu$ is not in a block yet.
- Transaction μ has not been received yet when block b is created. The date of broadcasting $B_{\mu,sc}$ must be greater than the date of mining of the previous block $b-1$. We consider the mining date of the previous block as a limit because the miner will create a new block, fill it quickly, and then try to find a valid block for the proof of work. Thus the transaction must be received before filling the new block b, which starts when the previous block $b-1$ is received.

If a transaction σ_j^{sc} does not satisfy these requirements, the following transaction σ_{j+1}^{sc} is the new candidate to be added to the current block. This causes the situation in which a transaction σ_j^{sc} following another transaction $\sigma_{j'}^{sc}$ in the σ^{sc} order can be added to a block earlier.

Because of these situations, the computation of the probability $P(A_{\sigma_j^{sc},s,b,k})$ that transaction σ_j^{sc} is mined in block b in scenario sc, given that k blocks have been mined during the period of length T, depends on every 2^{j-1} possible state of the waiting list defined by the presence or not of the transactions preceding σ_j^{sc}.

Due to a lack of space, we do not present the computation of these probabilities as they are of little practical use due to the exponential time complexity.

6.2. Monte-Carlo algorithm

The computation of probabilities $P(V_\mu(T))$ with the stochastic model takes an exponential time. To overcome this difficulty, we approximate the probabilities using a Monte-Carlo algorithm. The algorithm generates SC scenarios based on the set of validated blocks V, the set of transactions U in these blocks, and the set of pending transactions W. Each scenario is defined by data drawn randomly according to the probability distributions presented in Section 7.

To approximate $P(V_\mu(T))$, the algorithm computes for each scenario sc

$$X_{sc} = \begin{cases} 1 & \text{if } \mu \text{ is added to a block before time } T, \\ & \text{in scenario } sc, \\ 0 & \text{otherwise.} \end{cases}$$

and then returns

$$\overline{X} = \frac{\sum_{sc=1}^{SC} X_{sc}}{SC} \tag{11}$$

as an approximation of $P(V_\mu(T))$. This procedure is summarized in Algorithm 1.

Algorithm 1 Monte Carlo algorithm to approximate the probability of mining of transaction μ

Require: μ, SC, U, V, W
  Generate SC scenarios with the distribution obtained from the set U, V, W
  $X \leftarrow 0$
  for $sc = 1$ to SC **do**
    $Mined \leftarrow \emptyset$
    for $b = 1$ *to* Number of blocks mined in sc **do**
      $spaceLeft \leftarrow BL$
      for $j = 1$ to N^{sc} **do**
        if $(M_{b-1} \geq B_{\sigma_j^{sc}}) - delay \wedge (spaceLeft \geq gl_{\sigma_j^{sc}}) \wedge (lgp_b \leq gp_{\sigma_j^{sc}}) \wedge$ ($prec_{\sigma_j^{sc}}$ has already been added to a block) $\wedge (\sigma_j^{sc} \notin Mined)$ **then**
          σ_j^{sc} is added to $Mined$
          $spaceLeft \leftarrow spaceLeft - gc_{\sigma_j^{sc}}$
          if $\sigma_j^{sc} = \mu$ **then**
            $X \leftarrow X + 1$
          end if
        end if
      end for
    end for
  end for
  return $\frac{X}{SC}$

In this algorithm, we introduce a delay between the date of mining of the previous block and the date of reception of a transaction, to offset two phenomena:

- The reception delay of the transaction: The date of broadcasting considered for a transaction is its reception date and can be slightly different from the broadcasting date.
- The creation time of a block: When a miner creates a block, it takes time to add all the transactions, and if a transaction arrives during this period, it might be added to this block.

The precision of the method will be influenced by SC, the number of scenarios generated. To determine a suitable value of SC, we can estimate the variance of $\overline{X}$ as

$$Var(\overline{X}) = Var\left(\frac{\sum_{sc=1}^{SC} X_{sc}}{SC}\right) = \frac{\sum_{sc=1}^{SC} Var(X_{sc})}{SC^2} = \frac{\sum_{sc=1}^{SC} p(1-p)}{SC^2} = \frac{p(1-p)}{SC} \tag{12}$$

where, for each transaction sc, X_{sc} follows a Bernoulli distribution with parameter $p = P(V_\mu(T))$.

Hence, the standard deviation $\delta_{\overline{X}}$ of $\overline{X}$ satisfies

$$\delta_{\overline{X}} = \sqrt{Var(\overline{X})} = \sqrt{\frac{p(1-p)}{SC}} \leq \frac{0.5}{\sqrt{SC}} \tag{13}$$

as $p(1-p) \leq 0.25$. So to obtain a standard deviation of $\delta_X \leq 0.01$, SC must be greater than or equal to 2500.

6.3. Binary search algorithm

Assuming the fact that increasing the gasprice value gp_{tr} of a transaction tr can only increase the probability $P(V_{tr}(T))$, determining the optimal gasprice value for a transaction tr remains the same as determining the optimal position of the transaction tr in σ. Increasing the gasprice will decrease the position of tr in σ and thus increase the probability of being mined in time T.

Therefore, the optimal value of gp_{tr} can be determined by performing a binary search on the position of tr in σ. This procedure is presented in Algorithm 2.

Algorithm 2 Binary search algorithm to determine the optimal price for transaction tr

Require: U, V, W, tr
  $low \leftarrow 0$
  $high \leftarrow n$
  Create σ from W
  while $low \neq high - 1$ **do**
    $candidate \leftarrow \lfloor low + (high - low)/2 \rfloor$
    $gp_{tr} \leftarrow gp_{\sigma_{candidate}} + 1$
    Insert transaction tr in σ in position $candidate$
    $prob \leftarrow MonteCarlo(\sigma, U, V, W)$
    if $prob \leq \alpha$ **then**
      $high \leftarrow candidate$
    else
      $low \leftarrow candidate$
    end if
    Remove tr from σ
  end while
  return $gp_{\sigma_{low}} + 1$

7. The data

The Ethereum blockchain keeps the history of all the transactions that have happened since the first block. However, some important data needed by the model are not recorded in the blockchain, such as the broadcasting time of the transactions in the network. The blockchain contains the set of validated blocks V and the set of transactions U in these blocks.

To extract the other important information, we use a tool that listens to the network continuously. The tool simulates a node in the network that records in real time the set W of pending transactions sent by the other nodes. With this tool, we can recover all the data needed for the models previously presented.

The data collected by the tool consist of:

- The transaction broadcasting date. We consider for this value the date of reception of the transaction by our tool. We consider the broadcasting delay insignificant.
- The mining date of blocks. The blocks contain their theoretical date of mining, but these date are written by the miners themselves. Based on our observations, these date are not representative of the real date of mining a block. The miners probably don't update these date every time they try to find a solution to the proof of work process. To bypass this problem, we consider the date of reception in our tool as the mining date of the block. The broadcasting delay is assumed to be negligible.

7.1. Data clean up

The tool provides raw data that need to be cleaned up. Some transactions are not received before they are added to a block. There are two possible causes for this situation:

- The transaction has been broadcast, then the miner that created the next block added it to their block and broadcast it. The node used to gather the data received the block first due to a delay in the network, and the other node discarded the transaction because it was already mined.
- The transaction has not been broadcast at all. The sending address of the transaction is the miner of the block (or a "friend's account"), and they kept it until they created a block in which they put their own transactions to avoid paying fees to other miners.

Moreover, some transactions received cannot be mined. There are two cases where transaction μ is stuck in the waiting list:

- Transaction μ consumed more ether than the sender has. If the amount of ether consumed by a transaction $gasPrice \times gasConsumed + valueSend$ is greater than the balance of the sender, the transaction cannot be added to a block. The transaction will stay in the waiting list until the sender account receives an amount of ether allowing the addition of transaction μ to a block.
- Transaction $prec_\mu$ sent by the same account does not belong to a block and cannot be added to a block yet. A transaction $prec_\mu$ can block a transaction μ for several reasons:
  - The balance of the sender is lower than the amount of ether spent by transaction $prec_\mu$, as in the previous case.
  - Transaction $prec_\mu$ has not been broadcast.

To clean up the data, we remove from set U the transactions that have not been received by our tool. We also remove from the set W transactions that cannot be mined at time t=0.

7.2. Distribution of the random variables

To infer the random distribution of the variables, the sets V of blocks, U of mined transactions, and W of pending transactions are collected from the K last periods of duration T, for a total duration $T'=KT$. From these data, we infer a random distribution of the variables.

7.2.1. The average number λ of blocks mined during a period of length T

We approximate λ by the average number of blocks mined during the last T' units of time:

$$\lambda = \frac{|V|T}{T'} \tag{14}$$

7.2.2. The block limit BL

In Ethereum, miners can increase/decrease the gas limit BL for the current block by 1/1024 of the gas limit of the last block. We approximate BL using the median value of the gas limit of blocks in V.

7.2.3. The number of future transactions $|W'|$

To estimate the number of future transactions, we collect from our tool the reception date of previous transactions in $U \cup W$. We approximate the number of future transactions using a normal distribution:

$$|W'| \sim N\left(\frac{(|U| + |W|)T}{T'}, v\right) \tag{15}$$

where v is the variance in the number of transactions over the K periods during which data was collected. This variance v is computed as

$$v = \frac{\sum_{k=1}^{K} \left(\left|U_k \cup W_k\right| - \frac{(|U|+|W|)T}{T'}\right)^2}{K} \tag{16}$$

7.2.4. The gaslimit gl_μ and gas consumption gc_μ of a future transaction $\mu \in W'$

For a future transactions $\mu \in W'$, we set $gl_\mu = gl_{\mu'}$ where μ' is a randomly selected transaction in $U \cup W$.

The minimum gas consumption for a transaction is 21 000 gas if the transaction does not imply the execution of a smart contract in the Ethereum blockchain. We compute the ratio rlc_μ of the gas consumption compared to the gas limit of a transaction $\mu \in \{U \cup W\}$ as:

$$rlc_\mu = \frac{gc_\mu}{gl_\mu} \tag{17}$$

Then, we set the gas consumption gc_μ of a transaction $\mu \in W'$, by choosing a random transaction $\mu' \in U$ and setting

$$gc_\mu = \max(21000, rlc_{\mu'} gl_\mu) \tag{18}$$

7.2.5. The gasprice of a future transaction $\mu \in W'$

As the gasprices of future transactions are unknown, we make the following assumptions:

- The gasprice of a future transaction will be similar to the gasprice of the transactions received during the last T' units of time.
- The probability that a transaction with the same gasprice as a previous transaction will be broadcast in the next T units of time decreases with the age of the previous transaction.

Therefore, we set the gasprice of a transaction $\mu \in W'$ to $gasprice_\mu = gasprice_{\mu'}$ with μ' the E^{th} most recently received transaction $\in U \cup W$, where E is drawn according to an exponential distribution with parameter λ'. If E is greater than $|U|+|W|$, another value for E is chosen following the same random distribution. In this paper we use $\lambda' = 0.01$.

7.2.6. The lower gas limit of miners LGP

The miners have a lower limit on the transaction gasprice LGP under which they do not include a transaction in their blocks. This lower limit is not known and can change over time. To approximate the distribution of LGP, we compute the most probable value for LGP_m for every miner $m \in MN$ that mined blocks $V_m \subseteq V$:

$$LGP_m = \min\{gasprice_\mu | \mu \in (V_m \cap (U \cup W))\} \tag{19}$$

With these values, we compute the discrete distribution of LGP as:

$$P(LGP \leq x) = \frac{\sum_{m \in MN | LGP_m \leq x} hp_m}{\sum_{m \in MN} hp_m} \tag{20}$$

where hp_m is the hashpower of the miner, corresponding to the probability that this miner will create the next block.

7.2.7. The dates of broadcasting of future transactions W' and the dates of mining of future blocks V'

The number of blocks mined during a period of T units of time follows a Poisson distribution with parameter λ. The date of broadcasting B_μ of a transaction $\mu \in W'$ and the date of mining M_b of a future block b both follow a uniform distribution between times t=0 and t=T:

$$B_\mu \sim U(0, T) \tag{21}$$

$$M_b \sim U(0, T) \tag{22}$$

8. Experimental results

In this section, we present experimental results on real-life data. The tests are run from data collected between Wed Nov. 06, 2019, 16:43:33 GMT+0100 and Wed Nov. 13, 2019, 15:23:33 GMT+0100 on the Ethereum Constantinople version. During this period, we collected nearly 5 million transactions. In these 5 million transactions, only 0.4% have not been received by our tool, and thus have been collected directly in the blocks. As explained before, these transactions are just ignored and considered lost space in blocks.

The computation times in seconds to find the optimal gasprice value with Algorithm 2 are presented in Fig. 4.

In order to demonstrate the reliability of our method to return the lowest gasprice for a transaction to be mined in a given time and a given probability, we need to demonstrate the accuracy of our prediction method. Indeed, if the Monte-Carlo method is able to predict the probability for a transaction to be mined before time T, it is trivial that the binary search algorithm (Algorithm 2) will always return the lowest gasprice insuring the probability α.

We next present results obtained with the Monte-Carlo method to predict the probability for a transaction to be mined within T units of time. To validate the prediction, we execute 2500 runs of the Monte-Carlo method that give 2500 potential dates of mining d_r, r=1, ..., 2500, for a transaction tr. To do that, we just return the date of the transaction tr mining in Algorithm 1 instead of 1 or zero, with a time T large enough. If the transaction is not mined before the time limit T, we return $+\infty$. We then compute the date d^α that satisfies

$$\min\{d^\alpha | \{|d_{r=1,..,2500} \leq d^\alpha| \geq 2500^*\alpha\}\} \tag{23}$$

For example, if α=0.7, we define date $d^{0.7}$ as the theoretical date given by the method from which the transaction has a probability of 70%

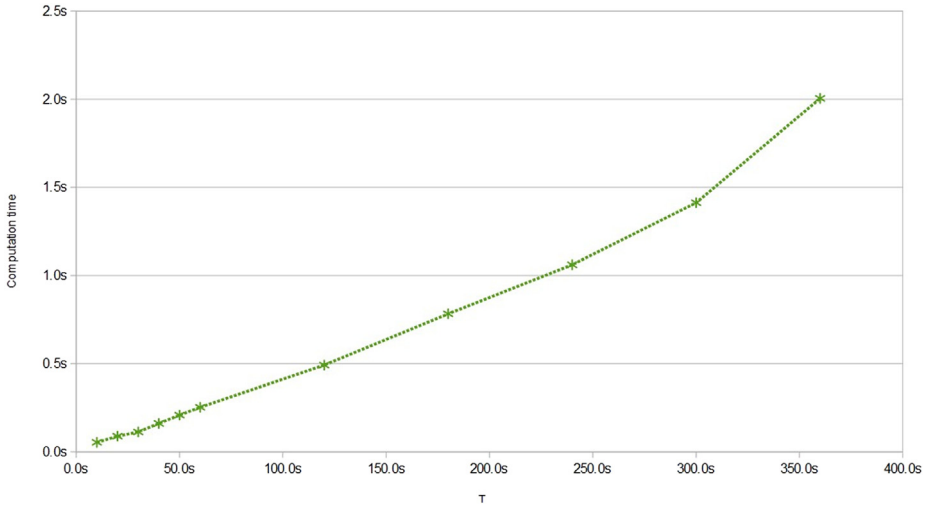

Fig. 4. Computation time in seconds to find the optimal gasprice value depending on T value.

to be mined. Thus, $d^{0.7}$ will be the date such that 1750 transactions over 2500 are predicted by Algorithm 1 to be mined.

We then compare this date to the real date of mining rdm_{tr} of the transactions tr=1, .., 1000. If the real date is larger, the experiment is a success; otherwise, it is a failure. We note Y_{tr} such that:

$$Y_{tr} = \begin{cases} 1 & \text{if } d^{\alpha}_{tr} \geq rdm_{tr} \\ 0 & \text{otherwise.} \end{cases}.$$

We then compute the ratio of success to failure rsf_{α} as described in Eq. (24).

$$rsf_{\alpha} = \frac{\sum_{tr=1}^{1000} Y_{tr}}{1000} \tag{24}$$

This ratio represents the precision of the prediction given by the Monte-Carlo algorithm. The closer the ratio rsf_{α} is to α, the more accurate the prediction is.

The results of this evaluation are presented in Figs. 5 and 6, as well as Tables 3 and 4. Figs. 5 and 6 present in abscissa the different values for α and in ordinate the value rsf_{α} obtained over 10 series (No.1, ..., No.10) of 1000 transactions from different periods of time. The min, max, and average values obtained are reported in Tables 3 and 4.

The results obtained with delay=0 ms (Fig. 5, Table 3) show that the prediction is a bit pessimistic and underestimates the probability of a transaction being mined within T units of time. This implies that the optimal value returned by the binary search algorithm ensures a higher probability of mining than requested by the user. The results obtained with the introduction of an artificial delay of 2500 ms, as presented in

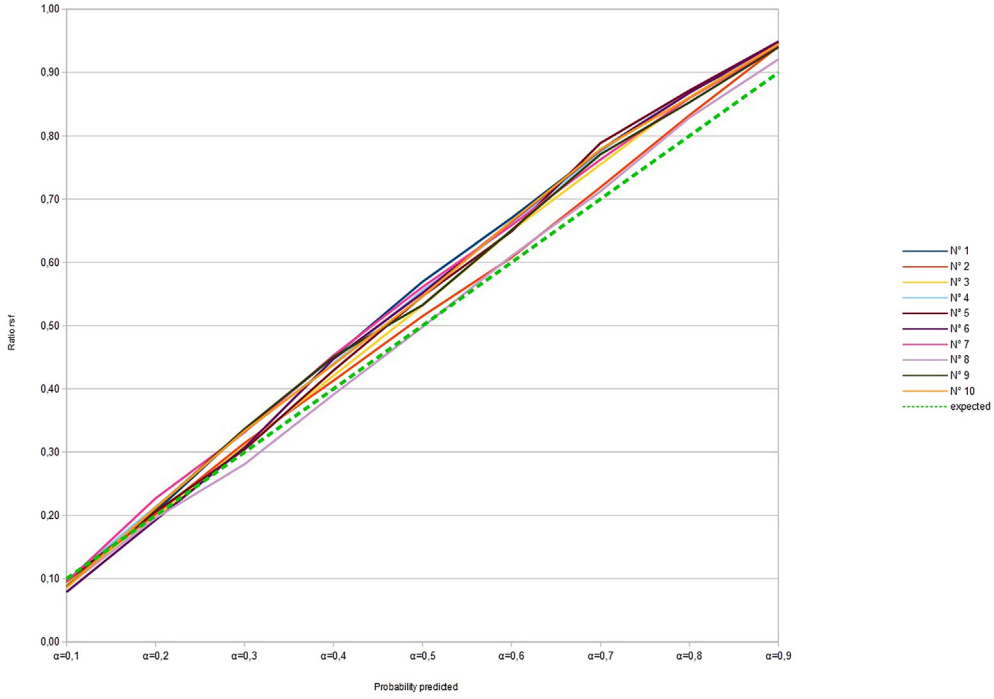

Fig. 5. Results obtained for the mining date prediction with delay=0 ms

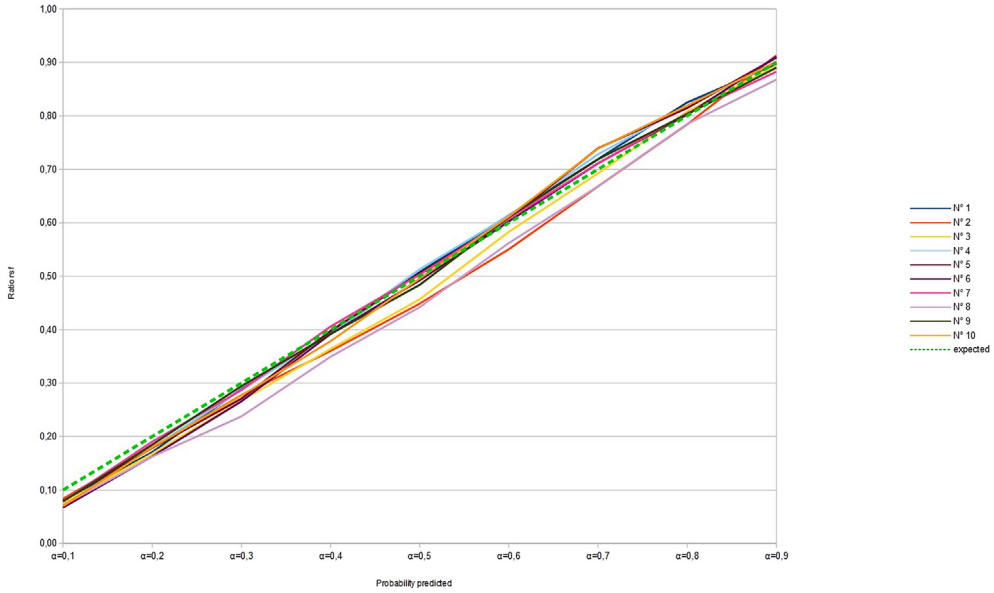

Fig. 6. Results obtained for the mining date prediction with delay=2500 ms

Table 3
Prediction precision obtained with the proposed Monte-Carlo method and delay=0 ms

α	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
min	0.08	0.19	0.28	0.39	0.50	0.61	0.71	0.83	0.92
max	0.10	0.23	0.34	0.45	0.57	0.67	0.79	0.87	0.95
average	0.09	0.21	0.32	0.43	0.54	0.65	0.76	0.86	0.94

Table 4
Prediction precision obtained with the proposed Monte-Carlo method and delay=2500 ms

α	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
min	0.07	0.16	0.24	0.40	0.44	0.55	0.67	0.78	0.87
max	0.08	0.19	0.29	0.35	0.51	0.61	0.74	0.83	0.91
average	0.08	0.18	0.28	0.38	0.48	0.60	0.71	0.81	0.90

Table 5
Characteristics and performances of the prediction methods in the literature.

Methods	Prediction considered	Computation time	Precision	Configurability
ETH Gas Station	The lowest gasprice in future blocks	–	Overestimate the real value up to 28% of the time [22]	Urgency parameter
Ref. [23]	The closest date for a transaction to be mined	–	The best method returns a date with a marginal absolute error of 13% to the real mining date	No
Ref. [24]	The lowest gasprice in future blocks	0.04 s to complete prediction	The best method provides a gasprice with a marginal absolute error of 154% to the real lowest gasprice	No
Ref. [25]	Provide a "relatively" good value for the gasprice	–	No precision provided	Urgency parameter
Ref. [27]	The lowest gasprice in future blocks	–	The best method provides a gasprice with a marginal absolute error of 6.3% to the real lowest gasprice	No
Our method	The lowest gasprice insuring a given transaction to be mined before a given date with a given probability	Up to 2 s for a forecast for the next 24 blocks	The gasprice obtained insures a probability of being mined in the given time with a deviation of $\leq$ 2% from the given probability α (see Table 4)	Configure the probability of being mined and the limit date

Algorithm 1, show that the prediction is closer to reality (Fig. 6, Table 4). With the values obtained in Table 4, we can adjust the α probability to fit the will of the users.

A direct comparison of the results presented in this paper to those existing in the literature is not possible since most oracles don't pursue the same goal as our model. Indeed, we aim to predict the lowest gasprice for a transaction to be mined in a given time, with a given probability, although most of the existing oracles try to predict the lowest gasprice that will be present in the next future blocks or a gasprice insuring a "relatively good" probability of being mined. We next assess that the lowest gasprice in a future block is not always a good indicator of the probability of a transaction being mined. For example, a transaction with a gaslimit equal to the maximum size of a block BL will not be mined, except if this transaction has the largest gasprice in the pool. We provide in Table 5 a comparison between the results of our method and the different oracles for which data on their performances are available. The value "–" for the column "Computation time" refers to a negligible computation time or a value not reported in the original paper.

As illustrated in Table 5 the computation time of the method defined in this paper is quite large to give a prediction compared to other approaches. This is due to the intrinsic nature of our approach. Machine learning methods spend a larger amount of time training a model, but after that, the prediction is almost instantaneous. Our method takes more time to simulate the system and obtain the probability for a transaction to be mined in a given time. Concerning the precision of the predictions, no conclusion can be drawn from these results, given that the methods do not predict the same indicators. As stated in Table 5, each machine learning method aims at predicting an indicator, such as the lowest gasprice in future blocks. Our method is the only one aiming to predict the probability of a transaction being mined. We are also the only one to provide a validation of the method based on the realization of the prediction instead of a measure of the deviation from an indicator.

9. Conclusion

The minimum fees needed to process a transaction are highly time-dependent, according to various phenomena like system congestion. The users choose between struggling to manually set a low gasprice value corresponding to the actual system state, or they fix a high value that will ensure the transactions to be mined but that will spend an unnecessary amount of ether. Some prediction methods provide value for the gasprice but most of these methods are not reliable and not configurable.

In this paper, we define an efficient solution approach to optimize the expenses of Ethereum's users in fees while ensuring a good probability of transactions being processed. In order to approximate the probability to be mined, we propose a deterministic model and a Monte-Carlo method to approximate a more realistic stochastic model. The Monte-Carlo method provides a good approximation of the probability of being mined. To our knowledge, the proposed method is the first one to take into account the pending list state and the gaslimit to provide an optimal gasprice value for transactions to be processed. With this method, a user can configure a time limit and a probability to obtain the lowest gasprice for their transaction.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Arnaud Laurent reports that financial support was provided by Utocat.

Acknowledgment

This research has been funded by Utocat™. The authors would like to thank them for providing the tool to extract the data needed in order to test the model.

References

[1] V. Buterin, Ethereum White Paper, GitHub, 2014. Available online: https://github.com/ethereum/wiki/wiki/White-Paper. (Accessed 24 April 2021).

[2] X. Xu, I. Weber, M. Staples, et al., A taxonomy of blockchain-based systems for architecture design, 2017 IEEE International Conference on Software Architecture (ICSA); 3–7 Apr 2017; Gothenburg, Sweden, IEEE, Piscataway, NJ, USA, 2017, pp. 243–252.

[3] N. Houy, The bitcoin mining game, SSRN, 2014 preprint.

[4] N. Szabo, Formalizing and securing relationships on public networks, First Monday 2 (9) (1997), https://doi.org/10.5210/fm.v2i9.548.

[5] G. Wood, Ethereum: a secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper, 2014. Available online: https://ethereum.github.io/yellowpaper/paper.pdf. (Accessed 24 July 2020).

[6] A.M. Antonopoulos, G. Wood, Mastering Ethereum: Building Smart Contracts and Dapps, O'Reilly Media, Sebastopol, CA, USA, 2018.

[7] A. Miller, Gas economics. https://github.com/LeastAuthority/ethereum-analyses/blob/master/GasEcon.md, 2015. (Accessed 1 April 2019).

[8] N. Grech, M. Kong, A. Jurisevic, et al., Madmax: surviving out-of-gas conditions in ethereum smart contracts, In: Proceedings of the ACM on Programming Languages, ACM, New York, NY, USA, 2018, pp. 1–27. OOPSLA.

[9] E. Albert, P. Gordillo, A. Rubio, I. Sergey, GASTAP: a gas analyzer for smart contracts, arXiv, 2018.

[10] M. Marescotti, M. Blicha, A.E. Hyvärinen, S. Asadi, N. Sharygina, Computing exact worst-case gas consumption for smart contracts, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation, Springer, Cham, Switzerland, 2018, pp. 450–465.

[11] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on ethereum smart contracts (sok), in: M. Maffei, M. Ryan (Eds.), Principles of Security and Trust, Springer, Heidelberg, Berlin, Germany, 2017, pp. 164–186.

[12] T. Chen, X. Li, X. Luo, X. Zhang, Under-optimized smart contracts devour your money, 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER); 20–24 Feb 2017; Klagenfurt, Austria, IEEE, Piscataway, NJ, USA, 2017, pp. 442–446.

[13] T. Chen, X. Li, Y. Wang, et al., An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks, in: J. Liu, P. Samarati (Eds.), Information Security Practice and Experience. ISPEC 2017, Springer, Cham, Switzerland, 2017, pp. 3–24.

[14] I. Weber, V. Gramoli, A. Ponomarev, et al., On availability for blockchain-based systems, 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS); 26–29 Sept 2017; Hong Kong, China, IEEE, Piscataway, NJ, USA, 2017, pp. 64–73.

[15] G.A. Pierro, H. Rocha, The influence factors on ethereum transaction fees, 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB); 27 May 2019; Montreal, QC, Canada, IEEE, Piscataway, NJ, USA, 2019, pp. 24–31.

[16] J.E.d.A. Sousa, V. Oliveira, J. Valadares, et al., An analysis of the fees and pending time correlation in ethereum, Int. J. Netw. Manag. 31 (3) (2021), e2113.

[17] A. Karaivanov, A. Donmez, Transaction Fee Economics in the Ethereum Blockchain, IDEAS, 2021. Available online: https://ideas.repec.org/p/sfu/sfudps/dp21-02.html. (Accessed 24 July 2020).

[18] D. Easley, M. O'Hara, S. Basu, From mining to markets: the evolution of bitcoin transaction fees, J. Financ. Econ. 134 (1) (2019) 91–109.

[19] S. Ricci, E. Ferreira, D.S. Menasché, A. Ziviani, J.E. Souza, A.B. Vieira, Learning blockchain delays: a queueing theory approach, Perform. Eval. Rev. 46 (3) (2019) 122–125.

[20] D. Koops, Predicting the Confirmation Time of Bitcoin Transactions, 2018 arXiv preprint arXiv:1809.10596.

[21] G.A. Pierro, H. Rocha, S. Ducasse, M. Marchesi, R. Tonelli, A user-oriented model for oracles' gas price prediction, Future Generat. Comput. Syst. 128 (2022) 142–157.

[22] G.A. Pierro, H. Rocha, R. Tonelli, et al., Are the gas prices oracle reliable? a case study using the ethgasstation, 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE); 18 Feb 2020; London, ON, Canada, IEEE, Piscataway, NJ, USA, 2020, pp. 1–8.

[23] H.J. Singh, A.S. Hafid, Transaction confirmation time prediction in ethereum blockchain using machine learning, arXiv, 2019.

[24] F. Liu, X. Wang, Z. Li, J. Xu, Y. Gao, Effective gasprice prediction for carrying out economical ethereum transaction, 2019 6th International Conference on Dependable Systems and Their Applications (DSA); 3–6 Jan 2020; Harbin, China, IEEE, Piscataway, NJ, USA, 2020, pp. 329–334.

[25] S.M. Werner, P.J. Pritz, D. Perez, Step on the gas? A better approach for recommending the ethereum gas price, in: P. Pardalos, I. Kotsireas (Eds.), Mathematical Research for Blockchain Economy, Springer, Cham, Switzerland, 2020, pp. 161–177.

[26] D. Carl, C. Ewerhart, Ethereum gas price statistics, SSRN, 2020. University of Zurich, Working Paper No. 373.

[27] R. Mars, A. Abid, S. Cheikhrouhou, S. Kallel, A machine learning approach for gas price prediction in ethereum blockchain, 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC); 12–16 Jul 2021; Madrid, Spain, IEEE, Piscataway, NJ, USA, 2021, pp. 156–165.

[28] C. Chuang, T. Lee, A practical and economical bayesian approach to gas price prediction, in: I. Awan, S. Benbernou, M. Younas (Eds.), The International Conference on Deep Learning, Big Data and Blockchain, Springer, Cham, Switzerland, 2021, pp. 160–174.

[29] C.-C. Lien, W.-T. Su, A qos-driven customizable forecasting framework for blockchain transaction fee recommendation, 2021 IEEE International Conference on Smart Internet of Things (SmartIoT); 13–15 Aug 2021; Jeju, Republic of Korea, IEEE, Piscataway, NJ, USA, 2021, pp. 348–349.