mathematics

MDPI

Article

Auction and Classification of Smart Contracts

Damián-Emilio Gibaja-Romero [1,*,†] and Rosa-María Cantón-Croda [2,†]

[1] Department of Mathematics, UPAEP-University, C. 17 Sur 901, Barrio de Santiago, Puebla 72410, Mexico
[2] Deanship of Engineering, UPAEP-University, C. 17 Sur 901, Barrio de Santiago, Puebla 72410, Mexico; rosamaria.canton@upaep.mx
* Correspondence: damianemilio.gibaja@upaep.mx
† These authors contributed equally to this work.

Abstract: The execution of smart contracts (SCs) relies on consensus algorithms that validate the miner who executes the contract and gets a fee to cover her expenditure. In this sense, miners are strategic agents who may focus on executing those contracts with the largest fee, to the detriment of other SCs' execution times, which also harms the blockchain's reputation. This paper analyzes the impact of miners' competition on SCs' execution times in a public blockchain. First, we explain that the Proof-of-Work mechanism casts similarities with a time auction, where the one who first adds blocks is the one who executes the contract and gets the fee. At equilibrium, costs negatively affect execution times, while the opposite holds concerning fees. However, this result does not capture the competition for other contracts; hence, we apply the Naïve Bayes method to classify SCs by considering a simulated database that comprises miners' competition for several contracts. We observe that simultaneous competition generates patterns that differ from the ones expected by the auction solution. For example, miners' valuation does not accelerate contracts' execution, and high-cost smart contracts do not necessarily execute at last places.

Keywords: Naïve Bayes classification; smart contracts; execution times; sealed bid auction

MSC: 91B26

Citation: Gibaja-Romero, D.-E.; Cantón-Croda, R.-M. Auction and Classification of Smart Contracts. *Mathematics* **2022**, *10*, 1033. https://doi.org/10.3390/math10071033

Academic Editors: Mahendra Piraveenan and Samit Bhattacharrya

Received: 2 February 2022
Accepted: 11 March 2022
Published: 24 March 2022

1. Introduction

Blockchain networks have boosted the implementation of smart contracts (SCs) since they provide a decentralized process that self-activates such agreements when certain conditions are satisfied. During this process, the network's nodes (miners) create blocks by solving mathematical puzzles, which is an activity that requires resources like hash power in Ethereum [1]. Later, a 'consensus algorithm' determines the miner who adds the block that the SC's execution needs and this miner gets a fee as a reward [2]. So, consensus mechanisms promote miners' competition because miners only get fees by adding the blocks that smart contracts need [3,4].

Consequently, miners follow an economic and strategic behavior since they choose those SCs that provide them the largest benefit [5] and join pools to share resources, which allows them to compete for a larger number of smart contracts [6]. So, public blockchain platforms deal with the issues associated with peer-to-peer networks, such as lack of security and coordination [7]. Nevertheless, SCs' execution times also rely on miners' decision-making. Specifically, miners may slow the activation of smart contracts because puzzles are costly to solve [8] or there are simultaneous requisitions to execute contracts due to an increasing number of SCs' applications [9,10]. In general, it is not clear the impact of miners' interactions, mining costs, and contracts' fees on the time that independent smart contracts last to be executed [11]. In this sense, by considering that miners simultaneously compete to add blocks in other contracts, this paper addresses the following research question: How do costs, fees, and miners' valuation impact SCs' execution times? To

answer this question, we first propose a game-theoretical model to determine the execution time that miners last in executing a contract characterized by its cost and fee. Later, we use the previous result to simulate the simultaneous execution of several smart contracts. Finally, we use machine learning techniques to classify execution times by considering costs, fees, and miners' features.

It is worth recalling that consensus algorithms play a major role in activating smart contracts since they determine the miner who adds a contract to the blockchain. To build our game-theoretical model, we assume the blockchain uses the proof-of-work (PoW) algorithm because it assigns a contract to the miner who first solves the contract's puzzle after spending her resources. Furthermore, we simplify the analysis by considering that contracts are independent among, which means that contracts do not trigger other contracts' variables [12].

By the previous discussion, miners' competition resembles a sealed bid time auction because the miner who sets the minimum time to create a node is the one who executes the contract's code. So, we model the miners' competition as a simultaneous game of incomplete information where miners observe each contract's features but do not know other miners' valuations. The equilibrium analysis provides a closed-form solution concerning the time each miner bids, which is described in terms of the miners' and contracts' features. At equilibrium, the comparative statics point out that execution times diminish as the number of miners and the fee increase, while the opposite happens when cost increases. Nevertheless, this result ignores the fact that miners can compete to get other contracts [9]. Although we can understand the previous interaction as a simultaneous auction problem [13], this framework is not necessary since SCs' are not interrelated, miners independently value each smart contract, and the consensus algorithms serially allocate contracts among miners. Thus, we propose a Naïve Bayes classification to determine how competing for other contracts impacts execution times. The Naïve Bayes technique groups SCs by considering the execution time as the independent variable. In contrast, miners' and contracts' features are the independent variables. Then, we simulate the competition between 1000 miners and 10,000 contracts by considering the auction solution.

Surprisingly, our classification's results differ from those we expected, given the comparative statics analysis. First, miners' valuation does not increase or decrease execution times; specifically, we get that miners' valuation induces an almost uniform conditional probability distribution. Second, fees and costs modify how fast a smart contract is executed, observing counter-intuitive results. For example, low cost and high fees slow the execution of some contracts.

Our paper is closely related to the literature that analyzes how miners impact the execution of smart contracts [14]. For example, miners pool resources as the contracts' complexity increases [15] to maximize their benefits [16]. Moreover, consensus algorithms are coordination mechanisms that transparently determine who has the right to execute a smart contract without interfering in how miners pick contracts, which also may cause dishonest behavior from them [17]. So, miners can manipulate the consensus mechanism by centralizing resources in mining pools [18], which tends to happen in those platforms that use algorithms that do not require the explicit use of resources for mining blocks, such as the Proof-of-Stake (PoS) and the Proof-of-Authority (PoA) algorithms [19,20]. In this sense, our contributions unfold in two streams. First, we describe the PoW mechanism as a time auction that allows us to find a closed-form solution that echoes empirical findings: the execution time of a contract increases as the contracts need more resources to be executed (costs) [21], and large fees reduce execution times [22]. This shows that game-theoretical models serve for designing consensus algorithms, which is of growing interest to avoid manipulation and safeguard users' welfare [23,24]. Second, we extend the applications of the Naïves Bayes technique to classify smart contracts based on the features that characterize their time of execution.

Typically, classification techniques have widely used to determine if a SC is not legal or duplicates a transaction [25,26] because not all smart contracts are secure [27]

or well-defined [28]. Furthermore, we can classify miners to identify pools that try to prioritize the execution of certain contracts over others [14]. Our classification shows that the probability of being executed with a certain priority is independent of miners' valuation, while economic variables (fee and costs) are the ones that change this probability. This result supports the necessity of designing consensus mechanisms based on auctions to prevent miners' collusion [29] and reduce the use of resources [30].

The paper is organized as follows. The next section presents the auction game in which miners interact to set execution times. Later, in Section 3, we describe the game´s solution, which we use to build the dataset. The theoretical model allows us to create a Naïve Bayes classifier, in Section 4, such that dependent variables are the contracts' features. Section 5 discusses the main results concerning Naïve Bayes evaluation and classification. The last section presents the paper's conclusions.

2. Model

In a public blockchain platform, miners compete to execute a SC. Specifically, we focus on a public blockchain that uses the proof-of-work consensus mechanism. To model miners' competition in such a structure, we first consider a market where agents a and p exogenously establish a smart contract x with a fee f_x. To simplify the notation, we refer to the previous smart contract as $x(f_x)$. The fee is a reward expressed as a positive amount of the digital coin used in the public blockchain $\mathcal{P}$ where the SC stands. Miners can use f_x to get resources exchanged in $\mathcal{P}$.

We assume the existence of a trustworthy oracle that points out the necessity to execute $x(f_x)$. For instance, we say that the oracle is not strategic. Moreover, all miners observe the need of adding nodes to $\mathcal{P}$.

2.1. Basic Elements

Let $M = \{1, 2, \ldots, m\}$ be the set of all miners in the digital platform; we use i to denote a generic miner in M. The time that miner i lasts to pick $x(f_x)$ is denoted by t_i, where $t_i \in T_i = [0, \infty)$. Given that smart contracts execution requires miners' resources as time passes, miner i faces a cost function $c_i(t_i)$ for picking the contract in a time t_i. We consider that c_i is a function from T_i to $\mathbb{R}$ such that $dc_i/dt_i < 0$, which captures the fact that miners can get better resources to solve puzzles with more time [31]. We characterize miners by a valuation v_i about picking the contract $x(f_x)$ to get f_x coins. Let V_i be the set of all valuations of miner i.

The **state of the market** is a vector $v = (v_1, v_2, \ldots, v_m)$ that is an element of $V = V_1 \times \cdots \times V_m \subset \mathbb{R}^m$. We consider that miners' valuation is private information, i.e., each miner i knows their valuation v_i but does not know other miners' valuations. Furthermore, agents p and a do not observe v. So, the market state v is the realization of a random vector $V = (V_1, V_2, \ldots, V_m)$ that is drawn from a probability density function $f : V \to \mathbb{R}$, which we assume common knowledge.

2.2. The Game

Regardless of digital platforms' architecture, they have similar rules concerning their application on smart contracts. Specifically, digital platforms establish that miners should compete among them to execute the smart contract, i.e., the miner who solves the puzzle as soon as possible is the one who adds the block to the blockchain and gets the associated fee. Thus, we model the competition between miners as a time auction where the winner is the one that sets the minimum time to pick $x(f_x)$.

The interaction between miners starts when all of them observe the request to execute $x(f_x)$. Hence, the game proceeds in the following two stages:

Stage 1. Nature draws the market state $v = (v_1, v_2, \ldots, v_m)$ by following the probability density function f.

Stage 2. Each miner i observes their valuation v_i, but no other miners' valuations. Simultaneously, each miner sets the time t_i to pick $x(f_x)$.

All nodes in the platform $\mathcal{P}$ observe the profile of picking times $t = (t_1, t_2, \ldots, t_m)$. Since picking times represent how fast miners mine a block, the miner who sets the lowest picking time is the one who executes $x(f_x)$. If two or more miners set the lowest picking time, the tie is randomly broken, i.e., the probability of executing the code is the same for all these miners. The game finishes with the execution of the smart contract, and the miner who runs it gets f_x.

Let u_i be the payoff function of i that depends on σ, and the amount of (digital) money μ_i that miner i gets when the game finishes.

2.3. The Solution Concept

The set of all possible actions of miner i is T_i. Recalling that miners' valuation are private information, a pure strategy is a decision rule σ_i that maps i's valuation into a picking time t_i, i.e., σ_i is a function from $\mathcal{V}_i$ to T_i. So, $\Sigma_i = \{\sigma_i | \sigma_i : \mathcal{V}_i \to T_i\}$ is the set of all possible decision rules of miner i.

A profile of strategies is a vector of decision rules $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m) \in \Sigma$, where $\Sigma = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_m$. As usual, σ_{-i} is the decision rules' profile of all miners that are different to i.

Remembering that miners' valuation is private information, miners have no certainty concerning the payoff that they get when the game finishes with a profile of decision rules $\sigma(v) = (\sigma_1(v_1), \sigma_2(v_2)..., \sigma_m(v_m))$. We denote by $E[u_i(\sigma)]$ the expected payoff of miner i at the end of the game; and we consider the **Bayesian Nash equilibrium** as the solution concept for the game described in Section 2.2.

Definition 1. *We say that* $\sigma^* = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ *is a Bayesian Nash equilibrium if*

$$E[u_i(\sigma_i^*, \sigma_{-i}^*)] \geq E[u_i(\sigma_i, \sigma_{-i}^*)]$$

for all $i \in \mathcal{M}$ *and* $\sigma_i \in \Sigma_i$.

In other words, a Bayesian Nash equilibrium is a profile of decision rules under which all miners do not have incentives to deviate from the action that they choose concerning the state of the market v.

3. Picking Times at Equilibrium

This section searches for the Bayesian Nash equilibria of the game described in the previous section. To simplify the analysis, we assume that miners follow symmetric behavior, and later we perform some comparative statics by considering that random variables V_i are independent and uniformly distributed.

3.1. Solving the Game

The game described in Section 2.2 establishes that miners simultaneously set the time to pick $x(f_x)$ whereas each miner's valuation is private information. Furthermore, each miner i knows that other miners picking times depend on valuations v_j, for all $j \in \mathcal{M} - \{j\}$, i.e., i knows that others decision rules profile $\sigma_{-i}(v_{-i}) = (t_j(v_j))_{j \neq i}$ depends on the market state. Then, at the end of the game, i faces three possible scenarios: (a) i executes the smart contract because he sets the lowest picking time, (b) i ties with other miners in setting the lowest time to pick the contract, or (c) i does not set the lowest picking time which means that i does not execute the code.

If i gets the right to execute the contract's code, i gets the fee associated with $x(f_x)$, and he also spends resources to mine blocks that guarantee $x(f_x)$ execution. For example, in Ethereum, these resources are gas and ether. In the opposite case, miners do not get f_x and do not spend resources to execute the code. Hence, the payoff function of miner i is

$$u_i(t_i, t_{-i}; v_i) = \begin{cases} f_x + v_i - c_i(t_i) & \text{if } t_i < \min_{j \neq i}\{t_j\}, \\ \frac{f_x + v_i - c_i(t_i)}{k} & \text{if } |\{j \in M | t_j = t_i = \min\{t_k | k \in \mathcal{M}\}\}| = k, \\ 0 & \text{if } t_i > \min_{j \neq i}\{t_j\}. \end{cases}$$

Thus, the expected payoff function of miner i is

$$E[u_i(\sigma_i, \sigma_{-i}); v] = Pr[W](f_x + v_i - c_i(t_i)) + Pr[T]\left(\frac{f_x + v_i - c_i(t_i)}{k}\right) + Pr[L]0,$$

where W, T, and L are the probability events associated with scenarios (a), (b) and (c), respectively.

Note that $\sigma(v)_{-i} = (t_j(v_j))_{j \neq i}$ is the realization of the random vector $\sigma(V)_{-i} = (t_j(V_j))_{j \neq i}$, i.e., each decision rule t_j is a transformation of the random variable V_j for all $j \in \mathcal{M} - \{j\}$. By assuming that V_j is a continuous random variable for all $j \in \mathcal{M}$, we have that

$$\begin{aligned} Pr[T] &= Pr[|\{j \in M | t_j = t_i = \min\{t_k | k \in \mathcal{M}\}\}| = k] \\ &= Pr[|\{j \in M | t_j(v_j) = t_i = \min\{t_k(v_k) | k \in \mathcal{M}\}\}| = k] = 0. \end{aligned}$$

Consequently, we can rewrite the expected payoff function of i as follows

$$E[u_i(\sigma_i, \sigma_{-i}); v] = (f_x + v_i - k)Pr[W], \tag{1}$$

where

$$Pr[W] = Pr\left[t_i < \min_{j \neq i}\{t_j(v_j)\}\right].$$

To search for the best decision rules of miner i to $\sigma(v)_{-i}$, we maximize their expected payoff. By expression (1), we first compute the probability $Pr[W]$of winning the right to execute $x(f_x)$. Note that $Pr[W]$ depends on the profile $\sigma_{-i}(v_{-i})$; as in Auction theory, we assume that all miners behave symmetrically picking $x(f_x)$ in a time that is inversely proportional to their valuation, i.e., all miners $j \neq i$ choose the decision rule

$$t_j = \frac{\alpha}{v_j}, \tag{2}$$

where α is a positive constant.

Intuitively, expression (2) captures the fact that miners pick the contracts as soon as their valuation on $x(f_x)$ increases. For example, the previous case arises when miners need money to acquire resources in $\mathcal{P}$.

Now, given that $t_j(v_j) = \alpha / v_j$ is the realization of the random variable α / V_j, the probability of winning the right to execute $x(f_x)$ is

$$Pr\left[t_i < \min_{j \neq i}\left\{\frac{\alpha}{v_j}\right\}\right].$$

By assuming that $V = (V_1, \ldots, V_m)$ is a random vector of independent and identically distributed random variables, the definition of the minimum order statistic implies that

$$Pr\left[t_i < \min_{j \neq i}\left\{\frac{\alpha}{v_j}\right\}\right] = \left(Pr\left[t_i < \frac{\alpha}{v_j}\right]\right)^{m-1}.$$

Finally, we consider that V_j follows a uniform distribution on the interval $[0,1]$, which means that

$$\begin{aligned} Pr[W] &= \left(Pr\left[t_i < \frac{\alpha}{v_j}\right]\right)^{m-1} \\ &= \left(Pr\left[v_j < \frac{\alpha}{t_i}\right]\right)^{m-1} \\ &= \left(\frac{\alpha}{t_i}\right)^{m-1}. \end{aligned} \tag{3}$$

To simplify the calculation of miners' best decision rules, we consider that $c_i(t_i) = 1/t_i + \kappa$, where κ is a fixed cost to mine $x(f_x)$, and $1/t_i$ are the variables costs, in accordance with the assumption that $\partial c_i/\partial t_i < 0$. The previous cost function, together with expression (3), implies that the expected payoff function in (1) can be rewritten as follows

$$E[u_i] = \frac{\alpha^{m-1}}{t_i^{m-1}}\left(f_x + v_i - \frac{1}{t_i} - \kappa\right). \tag{4}$$

Then, we compute the best response of i to σ_{-i} by considering that other miners decision rules profile is $\sigma_{-i} = (\alpha/v_j)_{j\in\mathcal{M}-\{i\}}$.

We use the criterion of the first derivative to search for the best response of i; that is to say, we need to solve the equation $\partial E[u_i]/\partial t_i = 0$ where

$$\frac{E[u_i]}{\partial t_i} = -\frac{(m-1)\alpha^{m-1}(v_i + f_x - \kappa)}{t_i^m} + \frac{m\alpha^{m-1}}{t_i^{m+1}}.$$

So, the critical points of the expected payoff function of i are the solutions of

$$-\frac{(m-1)\alpha^{m-1}(v_i + f_x - \kappa)}{t_i^m} + \frac{m\alpha^{m-1}}{t_i^{m+1}} = 0.$$

Note that the previous expression is equivalent to

$$\begin{aligned} \frac{(m-1)\alpha^{m-1}(v_i + f_x - \kappa)}{t_i^m} &= \frac{m\alpha^{m-1}}{t_i^{m+1}} \\ \frac{(m-1)(v_i + f_x - \kappa)}{t_i^m} &= \frac{m}{t_i^{m+1}} \\ t_i(v_i + f_x - \kappa)(m-1) &= m, \end{aligned}$$

which is a linear equation on t_i. Consequently, the unique solution of the first order condition $\partial E[u_i]/\partial t_i = 0$ is

$$t_i = \frac{m}{(m-1)(v_i + f_x - \kappa)}. \tag{5}$$

In other words, $E[u_i]$ has a unique critical point t_i that depends on v_i. The next theorem demonstrates that $t_i(v_i)$, as in expression (5), maximizes the expected payoff function of i when other miners set a picking time that is inversely proportional to their valuation.

Theorem 1. *Consider that $|\mathcal{M}| > 2$ and $v_i + f_x > \kappa$. The best response of miner i to $\sigma_{-i} = (\alpha/v_j)_{j\neq i}$ is*

$$\sigma_i^*(v_i) = t_i(v_i) = \frac{m}{(m-1)(v_i + f_x - \kappa)}.$$

Proof. By expression (5), $t_i = m/[(m-1)(v_i + f_x - \kappa)]$ is a critical point of $E[u_i]$; to guarantee that such decision rule is best response of i to σ_{-i}, we need to verify that such

rule induces the maximum expected payoff for i. For this end, we follow the second order condition. Thus, we first compute the second derivative of $E[u_i]$

$$\frac{\partial^2 E[u_i]}{\partial t_i^2} = \frac{m(m-1)\alpha^{m-1}(v_i + f_x - \kappa)}{t_i^{m+1}} - \frac{m(m+1)\alpha^{m-1}}{t_i^{m+2}}.$$

By performing some algebraic simplifications on the previous expression, we have that

$$\frac{\partial^2 E[u_i]}{\partial t_i^2} = \frac{m\alpha^{m-1}}{t_i^{m+2}}[(m-1)(v_i + f_x - \kappa)t_i - (m+1)]. \tag{6}$$

Notice that $t_i(v_i)^{m+2} > 0$ because $v_i + f_x > \kappa$. Furthermore, $m\alpha^{m-1}$ is a positive constant. So, to verify if the expected payoff function reaches a maximum, or a minimum, on t_i, we focus on identifying the sign of

$$\Delta(t_i) = (m-1)(v_i + f_x - \kappa)t_i - (m+1)$$

when $\Delta(t_i)$ is evaluated on $t_i(v_i)$.

$$\Delta(t_i(v_i)) = (m-1)(v_i + f_x - \kappa)\frac{m}{(m-1)(v_i + f_x - \kappa)} - (m+1),$$

where $m-1$ and $v_i + f_x - \kappa$ are greater than zero. So

$$\Delta(t_i(v_i)) = m - (m+1) = -1.$$

Consequently, we have that

$$\left.\frac{\partial^2 E[u_i]}{\partial t_i^2}\right|_{t_i = t_i(v_i)} = \frac{m\alpha^{m-1}}{t_i(v_i)^{m+2}}(-1) < 0.$$

In other words, miner i maximizes their expected utility function by picking $x(f_x)$ in a time

$$t_i(v_i) = \frac{m}{(m-1)(v_i + f_x - \kappa)}.$$

□

Corollary 1. *If $v_i + f_x - \kappa > 0$, the game described in Section 2.1 has a unique symmetric Bayesian Nash equilibrium $\sigma^*(v) = (\sigma_1^*(v_1), \ldots, \sigma_m^*)$ where*

$$\sigma_i^* = \frac{m}{(m-1)(v_i + f_x - \kappa)}$$

for all $i \in \mathcal{M}$.

Proof. It follows from Theorem 1. □

Expression (5) establishes a closed-form solution for miners' decision rule at the symmetric Bayesian Nash equilibrium (SBNE). Aside from the fact that such an equilibrium decision rule is unique and single-value, we observe that picking times are inversely proportional to the net revenue that each miner gets, after covering fixed costs, when they get the right to execute the smart contract. The following corollary shows that picking times tend to be such net revenue as the number of miners increases to infinity.

Corollary 2. *As $m \to \infty$, the equilibrium decision rule of i tends to $\frac{1}{v_i + f_x - \kappa}$.*

Proof. From Theorem 1, the equilibrium decision rule of miner i is $\sigma_i^*(v_i) = m/(m-1)(v_i + f_x - \kappa)$. Then, we have that

$$\begin{aligned}\lim_{m\to\infty} \sigma_i^*(v_i) &= \lim \frac{m}{(m-1)(v_i + f_x - \kappa)} \\ &= \lim_{m\to\infty} \frac{m}{m-1} \frac{1}{v_i + f_x - \kappa} \\ &= \left(\frac{1}{v_i + f_x - \kappa}\right) \lim_{m\to\infty} \frac{m}{m-1}.\end{aligned}$$

Since $\lim_{m\to\infty} m/(m-1) = 1$, we conclude that

$$\lim_{m\to\infty} \sigma_i(v_i)^* = \frac{1}{v_i + f_x - \kappa}.$$

□

3.2. Comparative Statics

As we mentioned before, one of the main advantages of establishing smart contracts relies on the fact that agents p and a do not need the intervention of a third party agent to guarantee the fulfillment of all agreements in $x(f_x)$. By an almost immediate execution, smart contracts provide security and transparency to agents p and a. However, there exogenous factors that may slow or accelerate the execution of smart contracts.

Specifically, expression (5) points out that picking times at equilibrium depend on the number of miners m, the fee f_x provided by $x(f_x)$, and the fixed cost κ that miners face by belonging to $\mathcal{P}$. Moreover, Theorem 1 and Corollary 1 imply that the equilibrium decision rule σ_i^* is single-value, which allow us to perform some comparative statics with respect to the exogenous variables m, f_x and κ. The following propositions summarizes the relationship between equilibrium picking times and exogenous variables.

First, picking times reduce as the number of miners increases. Intuitively, a larger number of miners promote competition among them since the consensus algorithm drive a strategic behavior.

Proposition 1. *At equilibrium, if $v_i + f_x - \kappa > 0$, then picking times diminish as the number of miners increases.*

Proof. To demonstrate this proposition, we take the partial derivative of expression (5) concerning m. So, we have that

$$\begin{aligned}\frac{\partial \sigma_i^*}{\partial m} &= \left(\frac{1}{v_i + f_x - k}\right) \frac{m - 1 - m}{(m-1)^2} \\ &= \left(\frac{1}{v_i + f_x - k}\right) \frac{-1}{(m-1)^2}.\end{aligned}$$

Since $v_i + f_x - \kappa > 0$, we have that

$$\frac{\partial \sigma_i^*}{\partial m} < 0.$$

In words, there is negative relationship between the time that i lasts to pick $x(f_x)$ and the number of miners in $\mathcal{P}$. □

As a consequence of Corollary 2 and Proposition 1, we have that picking times monotonically decreases to $1/(v_i + f_x - \kappa)$ as the number of miners m increases to infinity. In other words, after covering fixed costs, the inverse of the net revenue represents a lower bound for the time that miners last to pick the smart contract $x(f_x)$. Consequently, smart

contracts do not immediately execute. Thus, a large number of miners does not guarantee the execution of a smart contract despite that picking times diminish as there are more miners in the public blockchain.

The existence of a lower bound for picking a contract, different from zero, implies that the need for resources plays a major role in the execution of a smart contract $x(f_x)$. The following proposition shows the relationship between the equilibrium decision rule and the economic variables that characterize the contract.

Proposition 2. *Consider σ_i^*, the equilibrium decision rule of miner i. The relationship between σ_i^* and f_x is negative, and the relationship with respect to κ is positive.*

Proof. First, we compute the partial derivative of σ_i^* with respect to κ. So, we have that

$$\begin{aligned}\frac{\partial \sigma_i^*}{\partial \kappa} &= \frac{m}{m-1}\frac{-1(-1)}{(v_i+f_k-\kappa)^2} \\ &= \frac{m}{m-1}\frac{1}{(v_i+f_k-\kappa)^2}.\end{aligned}$$

Since $v_i + f_k - \kappa > 0$, we conclude that $\partial \sigma_i^* / \partial \kappa > 0$.
Now, the partial derivative of σ_i^* with respect to f_k is

$$\begin{aligned}\frac{\partial \sigma_i^*}{\partial f_x} &= \frac{m}{m-1}\frac{-1}{(v_i+f_k-\kappa)^2} \\ &= -\frac{m}{m-1}\frac{1}{(v_i+f_k-\kappa)^2}.\end{aligned}$$

Given that $v_i + f_k - \kappa > 0$, the relationship between σ_i^* and f_x is negative. □

The previous proposition establishes that higher costs of belonging to $\mathcal{P}$ may discourage miners from picking a contract in case of not having enough resources. So, picking times increase as the fixed cost increases. Moreover, the relationship between the equilibrium picking time and the fee is negative, which means execution times reduce as p and a provide a larger reward for executing $x(f_x)$.

4. Contracts Classification

Blockchain platforms provide a decentralized environment for executing smart contracts. The decision rule in Corollary 1 shows the equilibrium picking time that each miner spends to execute contract $x(f_x)$; hence, the minimum picking time is the execution time of $x(f_x)$. So, Corollary 1 allows us to analyze the relationship between the execution time and the exogenous variables v_i, f_x and k. However, the equilibrium decision rule σ_i^* does not indicate how the simultaneous competition to execute other contracts $y(f_y) \neq x(f_x)$ impact the execution time. This section analyzes the implications of simultaneous competition on the SCs' execution times.

It is worth recalling that we consider independent smart contracts, which means that $x(f_x)$'s features and execution does not depend on other contract $y(f_y)$ features and execution. So, if miner i gets a set of contracts $\{x_1(f_{x_1}), x_2(f_{x_2}), \ldots, x_K(f_{x_K})\}$, her benefit is

$$E[u_i(x_1(f_{x_1}), x_2(f_{x_2}), \ldots, x_K(f_{x_K}))] = \sum_{k=1}^{K} E[u_i(x_k(f_k))].$$

Consequently, we do not follow a simultaneous auction approach to analyze how the competition for getting other contracts impacts execution times since such an interaction is different from classical simultaneous auctions [13]. Hence, we use machine learning techniques to analyze execution times when miners simultaneously compete to get other contracts.

First, we simulate the game in Section 2.2. The previous simulation provides a database that comprises execution times, fee, cost, and miner's valuations. Then, we classify execution times in terms of the contracts and miners' features.

4.1. Building the Data

At equilibrium, miners pick smart contracts by considering the decision rule in Corollary 1. Such a closed-form solution is appealing for classifying smart contracts since it summarizes the SCs' features and the equilibrium behavior of miners. By using the software *R*, we build a database that comprises the interaction of 1000 miners to execute 10,000 smart contracts. The simulation considers as basis values maximum and minimum costs and fees from the Ethereum platform [32]. Below, we present the code we use to create such a database; it is divided into sections describing the platform, miners, and contracts' features.

```
#PLATFORM DESCRIPTION
#General features of the platform where the SCs stand.
#Minimum payoff
feeMin= 0.95
#Maximum payoff
feeMax= 7.56

#PLAYERS DESCRIPTION
#Number of miners
miners = 1000
#Miner's valuation
vs. = c(runif(miners, min=0, max=2*feeMax))

#SMART CONTRACTS DESCRIPTION
#Total of miners' interactions
#Number of contracts
sc =10000
#The fixed cost of running each SC
cost = runif(sc, min=feeMin/3, max=3*feeMax)
#The fee that offers each contract
fee = c(runif(sc, min=feeMin, max=feeMax))

#MINERS INTERACTION
#Each miner sets the time to pick a contract
#Matrix of execution times initialization
Ex_times = matrix(0,nrow=10000, ncol=miners)
#Each entry Ex_times[i, j] should represent the time
#that miner i spends for executing the SC j (see Corollary 1)
for (j in 1:10000) {
for (i in 1:miners){
Ex_times[i,j]
= miners/((miners-1)*(v[i] + fee[j] - cost[j]))
}
}

#THE SCs EXECUTION
#The minimum picking time
winner_time = apply(Ex_times, 1, FUN=min)

#WINNERS
#Those miners that execute a contract
```

```
#Initialization of the vector of miners who get a SC
winner = matrix(0, nrow = sc, ncol=1)
#We search for the miner who executes the contract in the minimum time
for (i in 1:10000){
winner[i] = which(Ex_times[i,]
== min(Ex_times[i,]), arr.ind=TRUE)
}

#SMART CONTRACTS DATABASE
#We summarize smart contracts features and miners interaction
#contracts' fee, execution cost, miners' valuation, winning time
Contracts_ex = data.frame(cbind(fee,cost,v,winner_time))
```

Note that all variables in the database **Contracts_ex** are continuous. By applying the classification technique, we get back the mean and standard deviation of the random numbers we used to create the database. So, we need to transform fee, cost, valuation, and picking time into categorical variables to classify contracts.

The transformation of variables is based on quantiles of five levels, and each of them represents a qualitative feature. In the case of fee and cost, each level refers to magnitude, that is to say, how large fees and costs are. Specifically, the fee or cost of a smart contract can be considered as Low, Medium-Low, Medium, Medium-High, and High. Concerning valuation, each level represents preference. Then, we can say that the SC x is the Top, Second-best, Third-best, Fourth-best, or Bottom for a miner i depending on her valuation v_i. Finally, the picking time categories refer to how fast a smart contract is executed; in this sense, a smart contract can be executed in First, Second, Third, Fourth, and Fifth place. Table 1 shows the labels of each quantile with respect to variables f_x, c_x, v and t_x.

Table 1. Categorical values of the smart contracts execution main features.

Quantiles	f_x	c_x	v	t_x
1	Low	Low	Bottom	First
2	Medium-Low	Medium-Low	Fourth-best	Second
3	Medium	Medium	Third-best	Third
4	Medium-High	Medium-High	Second-best	Fourth
5	High	High	Top	Fifth

Below, we explain the code that we use to perform the previous transformation in *R*.

```
#FEE TRANSFORMATION
#Fee is divided into quartiles
q_fee = as.numeric(quantile (fee, prob = c(0.2, 0.4, 0.6, 0.8)))
#Naming fee's quartiles as low, medium-low, medium-high, high
c_fee = cut(fee, breaks = c(min(fee),q_fee,max(fee)),
labels = c("Low","Medium-Low","Medium","Medium-High", "High"))

#COST TRANSFORMATION
#Cost is divided into quartiles
q_cost = as.numeric(quantile (cost, prob = c(0.2, 0.4, 0.6, 0.8)))
#Naming cost's quartiles as low, medium-los, medium-high, high
c_cost = cut(cost, breaks = c(min(cost),q_cost,max(cost)), labels =
c("Low","Medium-Low","Medium","Medium-High", "High"))

#VALUATION TRANSFORMATION
#Miners' valuation is divided into quintiles
q_v = as.numeric(quantile (v, prob = c(0.2, 0.4, 0.6, 0.8)))
```

```
#Naming miners' valuations as Top, Second-best, Third-best,
#Fourth-best, Bottom
c_v = cut(v, breaks = c(min(v),q_v,max(v)),
labels = c("Top", "Second-best", "Third-best",
"Fourth-best", "Bottom"))

#TIME TRANSFORMATION
#Execution times are divided into quartiles
q_winner_time = as.numeric(quantile (winner_time,
prob = c(0.2, 0.4, 0.6, 0.8)))
#Naming execution times as first, second, third, fourth, fifth
c_winner_time = cut(winner_time,
breaks =c(min(winner_time),q_winner_time,max(winner_time)),
labels = c("First", "Second", "Third", "Fourth", "Fifth"))

#QUALITATIVE DATABASE
Contracts_ex =
data.frame(lapply(Contracts_ex_b, as.character),
stringsAsFactors=FALSE)
```

4.2. The Näive Bayes Classifier

To analyze the impact of cost, fee and valuation on the simultaneous execution of smart contracts, we classify them through the Naïve Bayes (NB) technique. We also consider NB's variations to check the classification's accuracy since the database **Contracts_ex** is the result of a qualitative transformation of continuous variables. Table 1 summarizes the categorical values that each variable can take. We consider that execution time (t_x) is the independent variable, while fee (f_x), fixed cost (k) and valuation (v) are the dependent variables. In other words, we classify execution times through the main features of contracts and miners.

Note that our database satisfies the **naïve assumption** since its variables are independent by construction. Then, the Naïve Bayes method classifies smart contracts through the Bayes' theorem. Mathematically, this technique computes

$$Pr(t_x = t|f_x, c_x, v) = \frac{Pr(f_x, c_x, vs.|t_x = t)Pr(t_x = t)}{Pr(f_x, c_x)},$$

which refers to the conditional probability of being executed at place $t \in \{First, Second, Third, Fourth, Fifth\}$, given the features that characterize the contract (fee, cost) and miners' valuation.

4.3. Naïve Bayes Implementation

Remember, the **Contracts_ex** summarizes a simulated interaction between 1000 miners that compete for executing 10,000 contracts. Thus, contracts' classification identifies those features that accelerate or slow the execution of SCHence, we use the **Contracts_ex** database that comprises categorical values related to miners' preferences, magnitude, and velocity (see Table 1).

We build the classification model by splitting the database into two databases for training and testing. We use the **NaïveBayes** and **e1072** packages of *R* to verify the robustness of our results. Furthermore, we apply the Kernel and Laplace correction methods.

First, note that execution times classes (First, Second, Third, Fourth, and Fifth) are (almost) perfectly balanced. In other words, each class has an a priori probability of around 20% (see Table 2).

Table 2. A priori probabilities.

Class	First	Second	Third	Fourth	Fifth
A priori probability	0.195	0.204	0.20	0.201	0.197

Second, Table 3 illustrates the accuracy, for the training and the testing database, concerning the different packages and the correction methods that we use. We observe that accuracy is the same regardless of the package or the correction method. Although accuracy is lower than 50%, this is not surprising since the five classes are almost perfectly balanced. In other words, random guessing provides an accuracy of around 50% for each execution time, which is relatively high by considering that the smart contracts variables take five values.

Table 3. Techniques' accuracy.

	Accuracy (Train)	Accuracy (Test)
Naïve Bayes	43.21%	42.58%
Naïve Bayes (e1072)	43.21%	42.58%
Naïve Bayes Laplace	43.21%	42.58%
Naïve Bayes	43.21%	42.58%

Finally, in Table 4, we present the statistics concerning the execution time classes. Contracts executed at fifth place, the slowest category, have the highest *sensitivity*. Then, the classifier correctly predicts those contracts that have a slow execution. In other words, the simultaneous competition for getting SCs makes it difficult to predict which contracts are executed as far as possible. In contrast, the correct identification of false cases concerning how fast an SC is executed is around 80% or more for each execution time level. Thus, the classifier correctly identifies those contracts that do not cope with the features to be executed almost immediately.

Table 4. Sensitivity, Specificity, PPV, and NPV concerning SCs' execution times classification.

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value
First	0.275	0.845	0.303	0.827
Second	0.348	0.798	0.308	0.826
Third	0.248	0.825	0.265	0.813
Fourth	0.557	0.887	0.555	0.88
Fifth	0.732	0.932	0.725	0.934

Even though high sensitivity is desirable, the trust and credibility of Blockchain networks rely on preventing execution times manipulation, which holds when we look out for the Positive Predicted Value (PPV) and the Negative Predicted Value (NPV). So, the NPV is larger than the PPV for all execution time categories, which means that miners' interaction is trustworthy. Even more, the auction mechanism does not execute those contracts incorrectly identified as first class. This is important since most results are false positive according to the sensitivity measure. In other words, the Naïve Bayes classifier identifies those contracts that do not cope with the features for being executed in first, second, third, or fourth place. Hence, a large specificity contributes to identifying miners who try to cheat the blockchain. In this sense, the classification may prevent the blockchain platform's loss of trust and credibility.

4.4. Conditional Probabilities

The Naïve Bayes classifier computes the conditional probabilities of how fast a SC is executed, given the features of a smart contract. Table 5 shows the features of some contracts and the execution time that the NB classifier predicts.

Table 5. Predicted execution time given the features of some contracts.

Fee	Cost	Valuation	Execution Place
Low	Medium	Second-best	Second
Medium-Low	Medium-High	Bottom	First
Medium	Medium-High	Third-best	First
High	Medium-High	Bottom	First

Table 6 presents the conditional probabilities concerning smart contracts features. This table provides some interesting insights concerning how fast SCs are executed during the simultaneous competition to execute several contracts. First, we observe that miners' valuation induce a uniform distribution, reflecting the simulation's features (see Figure 1). Specifically, results are balanced around a probability of 0.20, which means that the miners' valuation balances the conditional probability. In other words, such a variable does not accelerate or slow the miners' intervention to execute a contract.

Table 6. Conditional probabilities of time execution given the smart contracts features and miners' valuation.

CF	Label	First	Second	Third	Fourth	Fifth
v	Bottom	0.196	0.199	0.199	0.205	0.194
	Fourth-best	0.212	0.189	0.194	0.191	0.195
	Third-best	0.207	0.205	0.198	0.21	0.207
	Second-best	0.185	0.202	0.213	0.202	0.211
	Top	0.198	0.202	0.194	0.19	0.19
f_x	High	0.21	0.196	0.209	0.107	0.275
	Medium-High	0.206	0.205	0.182	0.161	0.227
	Medium	0.211	0.196	0.182	0.229	0.187
	Medium-Low	0.196	0.191	0.208	0.241	0.152
	Low	0.175	0.21	0.217	0.258	0.158
c_x	High	0.111	0.099	0.097	0.557	0.14
	Medium-High	0.283	0.265	0.265	0.182	0
	Medium	0.274	0.315	0.285	0.112	0
	Medium-Low	0.252	0.247	0.27	0.105	0.126
	Low	0.076	0.072	0.08	0.042	0.732

Given the economic variables (cost and fee) that characterize smart contracts, conditional probabilities are not balanced as it happens with the miners valuation. In both cases, we observe different patterns that contrast with the relationships in Proposition 2.

First, we discuss the probability of execution times given a fee (see Figure 2). Table 6 shows that the probability of being executed in first or second place weakly increases as the SCs' fee increases, in line with the positive relationship between picking times and fee that we find in Proposition 2. Furthermore, contracts with a low fee have a higher probability of being executed in fourth place. However, the probability of being executed at fifth place calls our attention since such a probability increases as the fee also increases; in other words, simultaneous competition. Even more, this probability is larger (0.275) in comparison with the execution of contracts at first (0.21) or second places (0.19). Finally, those contracts in the third category present a balanced probability distribution (0.20).

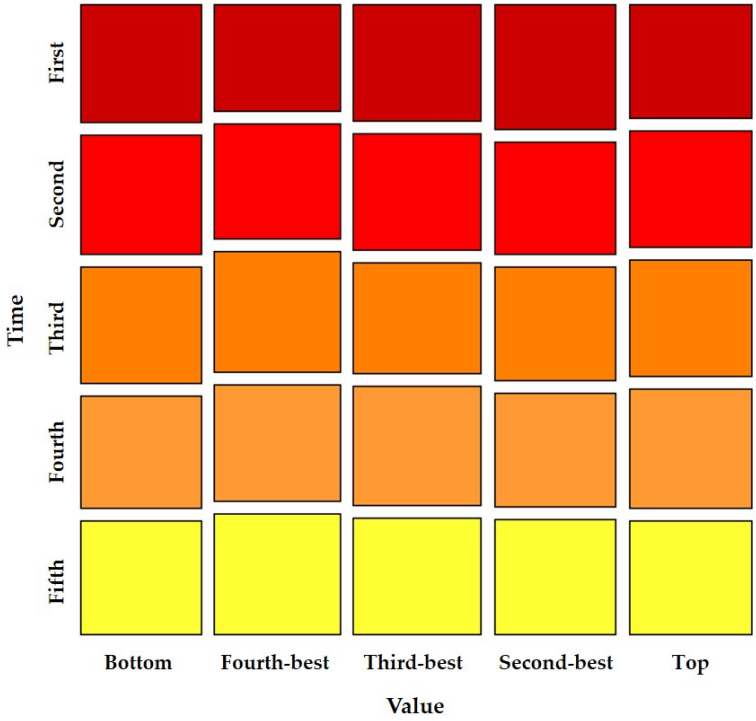

Figure 1. Graphical representation of the probability of executing a SC given the preference of a miner.

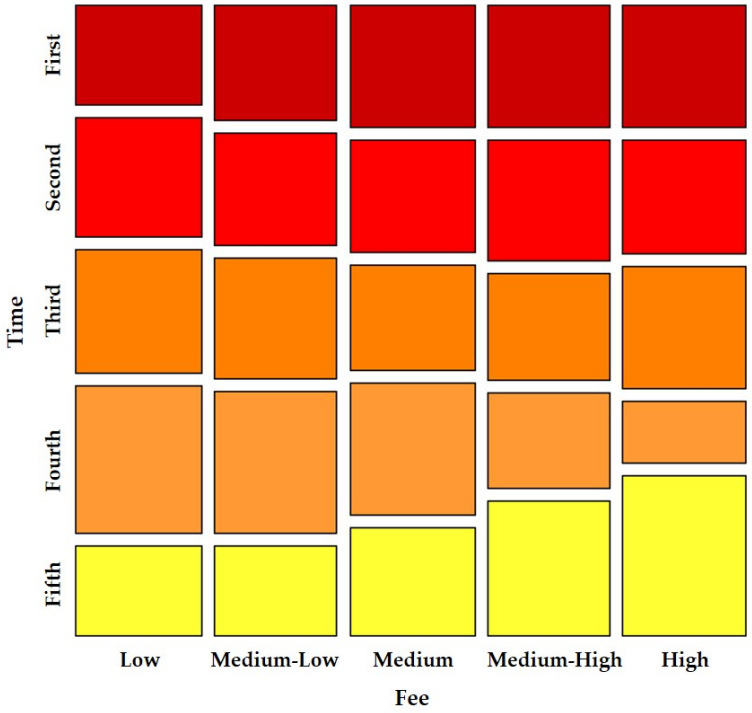

Figure 2. Graphical representation of the probability of executing a SC given the SCs fees.

Finally, conditional probabilities related to costs present the most drastic results (see Figure 3). For example, given a low cost, the probability of being executed at fifth place is greater than 0.7, which is surprising since we expect the opposite; that is to say, the prioritization of cheap contracts. Furthermore, we observe that low cost (cheap contracts) do not guarantee the immediate execution of smart contracts. A medium cost induces the highest probability of being executed at second (0.315) and third place (0.285), while smart contracts with medium-high costs have priority one to be executed; a probability

equal to 0.283. Nevertheless, we get zero probability of being executed in fifth place, given medium and medium-high costs; consequently, no smart contracts are classified in fifth place with such costs. We attribute the previous result to the low accuracy since variables take five values.

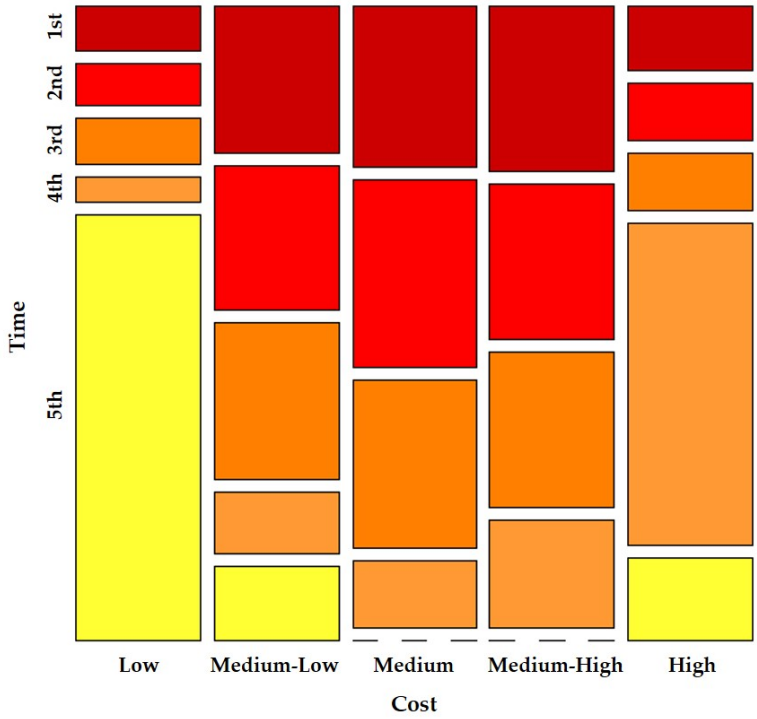

Figure 3. Graphical representation of the probability of executing a SC given the SCs' cost.

By the above discussion, the simultaneous competition to get several contracts modifies how fast a smart contract is executed. Particularly, given some combinations of qualitative features, we observe the opposite relationship that we expect from Proposition 2.

5. Conclusions

In this paper, we study the execution times of smart contracts by considering that miners simultaneously compete to get the right to execute more than one contract. Our analysis focuses on independent contracts that stand in a public blockchain that uses the Proof-of-Work mechanism since this consensus mechanism is secure and guarantees coordination among the blockchain's members [16]. Given that PoW induces a direct use of resources, we model miners' competition for a single contract as a time auction. In other words, the miner who first solves the contract's puzzle is the one who gets the associated fee. The analysis of this game-theoretical framework provides a closed-form solution concerning each miner's time to pick a single contract, which also echoes empirical findings. For example, the comparative statistics point out that picking times reduce as fee increases, while mining costs induce the opposite relationship [21,22].

The previous solution does not capture the fact that miners can compete to pick other contracts at the same time, an interaction that we analyze by classifying contracts. Given that contracts are independent, a simultaneous auction does not interrelate the miners' interaction in multiple executions, while the Naïve Bayes classifier allows us to predict execution times based on the smart contracts and miners' features.

Based on equilibrium picking times, we simulate the interaction of 1000 miners that compete for 10,000 contracts. Unsurprisingly, accuracy and sensitivity are below the standards since the independent variables are not dichotomous (we transform continuous variables into qualitative variables with five levels). However, the classification results show some interesting patterns that do not correspond with those expected from the comparative

statics analysis. For example, miners' valuation does not accelerate execution times, while costs induce counter-intuitive patterns (the probability of being executed in the last place increases as cost increases). Concerning the variable fee, there is no clear patron of their impact on execution times.

It is worth recalling that the Naïve Bayes classifier provides interesting insights concerning the possibility that miners manipulate the consensus mechanism when multiple contracts are executed. Although the classification results are not good for identifying those contracts that should be executed in a certain order (low sensitivity and positive predicted value), the opposite happens with the identification of contracts that should not be executed by considering a certain priority (high specificity and negative predicted value). We pretend to verify the previous results in future works by considering real data from public blockchains such as Ethereum.

Finally, our main contribution relies on providing a theoretical framework to analyze concurrent operations, whose efficient performance in public blockchains is of interest given the increasing popularity of smart contracts [9,33]. Particularly, the auction model points out the possibility of designing transparent consensus mechanisms that allow the monitoring of dishonest behavior of miners. Such a game-theoretical framework is suitable for including regulatory agents (such as time oracles) to take care of SC users' welfare, which is also a research topic of growing interest [23]. Moreover, smart contracts classification may predict the features that drive miners to follow dishonest behavior, such as manipulating the consensus algorithm. Avoiding dishonest behavior from miners is crucial for blockchains since it compromises their reputation. In this sense, we explain how the Naïve Bayes method is able to identify if miners' interactions modify the execution order of a smart contract and consequently prevent miners' misbehavior.

Author Contributions: Conceptualization, D.-E.G.-R. and R.-M.C.-C.; methodology, D.-E.G.-R.; software, R.-M.C.-C.; formal analysis, D.-E.G.-R.; investigation, D.-E.G.-R. and R.-M.C.-C.; data curation, R.-M.C.-C.; writing—original draft preparation, D.-E.G.-R.; writing—review and editing, D.-E.G.-R. and R.-M.C.-C.; visualization, R.-M.C.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Massobrio, R.; Nesmachnow, S.; Tchernykh, A.; Avetisyan, A.; Radchenko, G. Towards a cloud computing paradigm for big data analysis in smart cities. *Program. Comput. Softw.* **2018**, *44*, 181–189. [CrossRef]
2. Varnovskiy, N.; Martishin, S.; Khrapchenko, M.; Shokurov, A.V. Secure cloud computing based on threshold homomorphic encryption. *Program. Comput. Softw.* **2015**, *41*, 215–218. [CrossRef]
3. Chang, C.H.; Molahosseini, A.S.; Zarandi, A.A.E.; Tay, T.F. Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications. *IEEE Circuits Syst. Mag.* **2015**, *15*, 26–44. [CrossRef]
4. Sousa, L.; Antao, S.; Martins, P. Combining residue arithmetic to design efficient cryptographic circuits and systems. *IEEE Circuits Syst. Mag.* **2016**, *16*, 6–32. [CrossRef]
5. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G. Digital filtering of images in a residue number system using finite-field wavelets. *Autom. Control Comput. Sci.* **2014**, *48*, 180–189. [CrossRef]
6. Tchernykh, A.; Schwiegelsohn, U.; Talbi, E.g.; Babenko, M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. *J. Comput. Sci.* **2019**, *36*, 100581. [CrossRef]
7. Ye, R.; Boukerche, A.; Wang, H.; Zhou, X.; Yan, B. RESIDENT: A reliable residue number system-based data transmission mechanism for wireless sensor networks. *Wirel. Networks* **2018**, *24*, 597–610. [CrossRef]
8. Chervyakov, N.; Babenko, M.; Tchernykh, A.; Kucherov, N.; Miranda-López, V.; Cortés-Mendoza, J.M. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. *Future Gener. Comput. Syst.* **2019**, *92*, 1080–1092. [CrossRef]

9. Dickerson, T.; Gazzillo, P.; Herlihy, M.; Koskinen, E. Adding concurrency to smart contracts. *Distrib. Comput.* **2020**, *33*, 209–225. [CrossRef]
10. Singh, H.J.; Hafid, A.S. Prediction of transaction confirmation time in Ethereum blockchain using machine learning. arXiv **2004**, arXiv:1911.11592
11. Miranda-López, V.; Tchernykh, A.; Cortés-Mendoza, J.M.; Babenko, M.; Radchenko, G.; Nesmachnow, S.; Du, Z. Experimental analysis of secret sharing schemes for cloud storage based on rns. In Proceedings of the Latin American High Performance Computing Conference, Buenos Aires, Argentina, 20–22 September 2017; Springer: Berlin, Germany, 2017; pp. 370–383.
12. Vukolić, M. Rethinking permissioned blockchains. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, Abu Dhabi, United Arab Emirates, 2 April 2017; pp. 3–7.
13. Milgrom, P. Putting auction theory to work: The simultaneous ascending auction. *J. Political Econ.* **2000**, *108*, 245–272. [CrossRef]
14. Babenko, M.; Chervyakov, N.; Tchernykh, A.; Kucherov, N.; Shabalina, M.; Vashchenko, I.; Radchenko, G.; Murga, D. Unfairness correction in P2P grids based on residue number system of a special form. In Proceedings of the 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), Lyon, France, 28–31 August 2017; pp. 147–151.
15. Dana Troutman, N.; Laszka, A. PoolParty: Efficient Blockchain-Agnostic Decentralized Mining Pool. In Proceedings of the 2021 The 3rd International Conference on Blockchain Technology, Shangai, China, 26–28 March 2021; pp. 20–27.
16. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
17. Motaqy, Z.; Almashaqbeh, G.; Bahrak, B.; Yazdani, N. Bet and Attack: Incentive Compatible Collaborative Attacks Using Smart Contracts. In Proceedings of the International Conference on Decision and Game Theory for Security, Prague, Czech Republic, 25–27 October 2021; Springer: Berlin, Germany, 2021; pp. 293–313.
18. Dong, C.; Wang, Y.; Aldweesh, A.; McCorry, P.; van Moorsel, A. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 211–227.
19. Lee, S.; Kim, S. Shorting attack: Predatory, destructive short selling on Proof-of-Stake cryptocurrencies. *Concurr. Comput. Pract. Exp.* **2021**, *9*, e6585. [CrossRef]
20. Joshi, S. Feasibility of Proof of Authority as a Consensus Protocol Model. *arXiv* **2021**, arXiv:2109.02480.
21. Wüst, K.; Matetic, S.; Egli, S.; Kostiainen, K.; Capkun, S. ACE: Asynchronous and concurrent execution of complex smart contracts. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, 9–13 November 2020; pp. 587–600.
22. Aldweesh, A.; Alharby, M.; Solaiman, E.; van Moorsel, A. Performance benchmarking of smart contracts to assess miner incentives in Ethereum. In Proceedings of the 2018 14th European Dependable Computing Conference (EDCC), Iasi, Romania, 10–14 September 2018; pp. 144–149.
23. Zhou, H.; Ouyang, X.; Su, J.; de Laat, C.; Zhao, Z. Enforcing trustworthy cloud sla with witnesses: A game theory–based model using smart contracts. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5511. [CrossRef]
24. Patel, N.S.; Bhattacharya, P.; Patel, S.B.; Tanwar, S.; Kumar, N.; Song, H. Blockchain-envisioned trusted random oracles for IoT-enabled Probabilistic Smart Contracts. *IEEE Internet Things J.* **2021**, *8*, 14797–14809. [CrossRef]
25. Hu, T.; Liu, X.; Chen, T.; Zhang, X.; Huang, X.; Niu, W.; Lu, J.; Zhou, K.; Liu, Y. Transaction-based classification and detection approach for Ethereum smart contract. *Inf. Process. Manag.* **2021**, *58*, 102462. [CrossRef]
26. Tian, G.; Wang, Q.; Zhao, Y.; Guo, L.; Sun, Z.; Lv, L. Smart contract classification with a bi-LSTM based approach. *IEEE Access* **2020**, *8*, 43806–43816. [CrossRef]
27. Chen, W.; Zheng, Z.; Cui, J.; Ngai, E.; Zheng, P.; Zhou, Y. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1409–1418.
28. Norvill, R.; Pontiveros, B.B.F.; State, R.; Awan, I.; Cullen, A. Automated labeling of unknown contracts in Ethereum. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–6.
29. Wu, S.; Chen, Y.; Wang, Q.; Li, M.; Wang, C.; Luo, X. CReam: A smart contract enabled collusion-resistant e-auction. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1687–1701. [CrossRef]
30. Braghin, C.; Cimato, S.; Damiani, E.; Baronchelli, M. Designing smart-contract based auctions. In Proceedings of the International Conference on Security with Intelligent Computing and Big-Data Services, Guilin, China, 14–16 December 2018; Springer: Berlin, Germany, 2018; pp. 54–64.
31. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access* **2019**, *7*, 22328–22370. [CrossRef]
32. Corwintines. Gas and Fees. *Ethereum* **2022**. Available online: https://ethereum.org/en/developers/docs/gas/ (accessed on 14 July 2021).
33. Muchhala, Y.; Singhania, H.; Sheth, S.; Devadkar, K. Enabling MapReduce based parallel computation in smart contracts. In Proceedings of the 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 20–22 January 2021; pp. 537–543.