Blockchain: A novel approach for the consensus algorithm using Condorcet Voting procedure

David Vangulick
University of Liège,
Liège, Belgium
david.vangulick@skynet.be

Bertrand Cornélusse, Damien Ernst
University of Liège,
Liège, Belgium
{bertrand.cornelusse, dernst}@uliege.be

***Abstract*—The blockchain technology allows interested parties to access a common register, the update, and integrity of which are collectively managed in a decentralized manner by a network of actors. It is the consensus protocol that ensures a common and unambiguous update of transactions by creating blocks of transactions for which integrity, veracity, and consistency are guaranteed through geographically distributed nodes. Bitcoin, the first popular blockchain concept, introduced the Proof of Work consensus based on work validation, and has been extended to thousands of participants. Despite its success and its large use in other crypto-currencies, Proof of Work's disadvantages are a high latency, a low transaction rate, and a high energy expenditure, making it a less-than-perfect choice for many applications. In addition the validation of transactions is not carried out with a definite temporality. For Bitcoin, it takes on average 10 minutes to create a block. However, for certain use cases such as auctions or the exchange of energy, there is a need for this temporality. The purpose of this article is to propose a new type of consensus that is faster, less energy-consuming and that can be synchronized with a time reference. The core of the reflection is the use of the Condorcet voting mechanism to determine the miner.**

***Index Terms*—Blockchain, consensus protocol, Condorcet voting, Independence of Irrelevant Alternatives**

I. Introduction

Blockchains can be regarded as decentralized and distributed ledgers that keep track of any type of transaction. A blockchain is maintained and developed by a set of actors called *nodes*. Since the arrival of Bitcoin [1] and its success as a cryptocurrency, the blockchain has emerged as a disruptive factor in many areas, starting with banking transactions.

Blockchain systems can be either open, i.e. without permission, or private. Open systems such as Bitcoin and Ethereum allow anyone to access the chain, any node can perform transactions and participate in the consensus process to build up the blockchain. Authorized platforms such as Hyperledger are intended for consortia with no open participation. While clients are allowed to submit transactions, the blockchain's progress is limited to a fixed set of peering nodes executed by consortium members. In a permission-less configuration, the number of nodes should be large, these nodes are anonymous and can be untrustworthy. The consensus mechanism for such a configuration must be robust to malicious nodes.

This paper covers the scope of open blockchains. It is structured as follows. Section II recalls some important concepts regarding blockchain and the Proof of Work (PoW) consensus method. Then, Section III states the problems addressed in this paper. Section IV presents the new consensus model using a Condorcet voting procedure. Section V analyzes how the proposed consensus method fulfils the requirements described in Section II and presents some simulation results. Section VI concludes.

II. Consensus using Proof of Work

Table I gives a general view of the blockchain consensus algorithm. This algorithm aims to define the correct transactions and to seal those digitally into a a *reference* or *canonical* block of information. Some nodes are candidates to create this block. In order to add blocks to a blockchain using "Proof of Work", at step 3, each candidate node must demonstrate that it has done some work. It is useful to introduce the concept of the hash function to describe the Proof of Work algorithm.

A. Hash Functions

Hash functions are largely used in communication protocols to ensure that a transmitted message is still exactly the same as the original one. They are also widely used in cryptography to verify that the message has not been tampered with. A hash function $Hash(i) = o$ maps every type of data to a fixed size of other data, called hash value, hash code, or simply hash.

The standard Hash function in blockchain applications is the SHA 256. To be used in blockchain, a hash function must be *deterministic*, for the same i, the output o is the same; *uniform*, all possible values of o are generated with the same probability; *non-invertible* it is impossible to trace back i from o; *collision-free*, for two different inputs i and j the probability to find the same output o is almost zero.

TABLE I
Blockchain consensus algorithm.

1. New transactions are broadcast to all nodes (there are put in a so called *Mem Pool*);
2. Each candidate node creates a block with the valid new transactions;
3. A candidate node is randomly selected and broadcasts its block;
4. Other nodes check the validity of the block and, if they agree, increment their chain;
5. If the majority of nodes agree, the block is definitively approved.

B. The Proof of work algorithm

In Bitcoin and other cryptocurrencies from the same root (Monero, Dash, etc.), the candidate node must find a hash value less than a reference value, known as the "difficulty level". This process is called "mining" and therefore, the candidates are called "miners". The difficulty level is dynamically set by the consensus protocol, which aims at the production of a block every ten minutes on average for Bitcoin. The computed hash value could be the result of different elements: the block reference (blockheader) of the previous block; all the transactions that are in the proposed block; a timestamp of the approximate time of the block creation; a nonce, i.e. a counter for PoW. Given the uniformity characteristic of the hash function, the only way for a miner to find a hash required by the difficulty level is to try all the possible nonces. The *hash rate* indicates the number of nonces a miner can test every second. Current standard hash rates for a candidate are approximatively 18 TH/s (e.g. Antminer S9). The first candidate node that finds a winning hash must add its proposed block to the blockchain and claim the mining reward (12.5 tokens for the Bitcoin). To do so it broadcasts its block to the other nodes.

C. Illustrative example

The following illustrative use case provides a common thread for the remainder of this article:

- Alice and Bob want to make a bet about a competition that is running right now (e.g. American Football 2018-2019). Alice thinks that *San Francisco 49ers* will win. Bob thinks the *Pittsburgh Steelers* will win.
- They place their bets with a well-known bookmaker using a BlockChain token (BCT). To obtain their tokens, both have exchanged US$ on a specific platform. These tokens are available in their digital wallet.
- From now on, they have only 30 minutes to place their bets. During this time, watching the match on TV, they can bet any amount of BCT from their digital wallet.

In blockchain terms, Alice and Bob first need to create their public address and to protect it with a private key. The combination of these two elements can be seen as the digital wallet, for the moment these are empty. When Alice and Bob exchange their US$ to BCTs, the platform, that has a stock of BCT, credits the equivalent value to their individual wallets. To do so, the platform performs two transactions *send x BCT from address 0 to address 1* and *send x BCT from address 0 to address 2* (where address 0 is the address of the platform, address 1 is the public address of Alice and address 2 is Bob's public address). We now consider that that Alice has 100 BCT and Bob 150 BCT. During the 30 minutes prior to the closure of bets, Alice and Bob perform the transactions as shown in Table II with the bookmaker. These operations are equivalent to *send x BCT from address 1 to address 3* and *send x BCT from address 2 to address 3*, where address 3 is the public address of the bookmaker. Transactions also contain additional data such as the competition and the winning team. These transactions are broadcast to all the blockchain participants. Assume we have a pool of potential *verifiers*: George, Henry, Edward, John, and Richard. Their role is to check if all transactions are correct, i.e. according to the generally used terminology, they are *miners*. All of them reject the last transaction of Alice (cf. Table II) because it exceeds the amount of BCTs in her wallet. The valid transactions are sealed in a block that is added to the block chain. A block can be assimilated to a page of an accountant's register where each line corresponds to a transaction, and the blockchain is the ledger of the accountant. Among the miners, according to the algorithm defined in Table II, George is chosen to be the holder of the reference block (which we will call the *canonical block*). In this illustrative example, when George broadcasts his block, the other miners and the other participants (amongst them Alice and Bob) consider that the block thus mined is correct, and add it to their own chain. The resolution/settlement of the bet itself is not in the scope of this paper.

TABLE II
ALICE AND BOB BETS TIMELINE.

Time	Alice's bet	Bob's bet
1	20	20
5		20
12	30	10
15	50	30
23		40
25		4
28	10	5

III. PROBLEMS STATEMENT

Blockchain can be used in various ways. For instance, it can be used to register an Internet auction or a sports bet on a permanent and immutable way. It can also be used to register energy exchange in peer to peer local energy communities [2], [3]. This section describes the desired features of a blockchain implementation for such applications.

A. Synchronicity

For the applications mentioned in the introduction of this section, a strong guarantee on the time when a block is created (that we will call *market period*) is required. PoW consensus algorithm cannot offer this guarantee, since it creates a block when a nonce is found, but there is only a probabilistic average time to create a block, which is set by the difficulty level.

In the illustrative example, the *market period* is the 30 minutes period to place the bets. The proposed consensus will then make it possible.

B. Failure to reach consensus

The ability to reach consensus in every situation is not obvious for every consensus algorithm. For instance, it could be difficult to select a candidate between different competing nodes with the same score. In PoW, the score is the time stamp that proves that a miner is the first one to find the good nonce. We will call this situation a *tie*. PoW proves to be good

at reaching a consensus. Any proposed method must also have this capability.

In our illustrative example, whatever the possible ties that could be between miners, George is the only one to have been selected.

C. Sybil Attack

The open blockchain can be considered as a peer-to-peer system for which the main threat is the Sybil attack [4]. This is an attack wherein the system is subverted by forging false identities. To succeed with their attack, the malicious user creates a huge number of false independent nodes and selects some of them in order to force or to disrupt the consensus model (e.g. accepting double-spend transaction, or creating a false block). The PoW consensus model solves this issue owing to the significant amount of energy needed to create a block. It is simply too costly to run multiple false identities when compared to the chances of success of such an attack. Other consensus methods like Proof of Stake (PoS) could suffer from this attack because there is insufficient barrier to prevent the launch of multiple identities [5]. This problem is known as a "Nothing at Stake" issue. The newly proposed consensus has to provide the same level of resistance as PoW to such an attack.

In our illustrative example, if John wants to cheat by creating clones of himself so that one of these clones is chosen instead of George, it must cost him a sufficient number of money/tokens or amount of energy to discourage him, and/or if George is the best candidate (according to criteria to be specified), he remains the one who will be designated.

D. Dominance: Concentration of decision-making power

The core concept of blockchain technology is decentralization. The main idea is that a group of interested parties are less vulnerable than a single trusted party. In blockchain, there is no "chief of the staff", but the consensus protocol ensures a good balance between different parties with different interests such as miners, transaction makers, smart contract promoters, etc. With the recent development of the "oldest" blockchain (Bitcoin and Ethereum), it should be noted that this balance could be endangered by the concentration of decision-making power due to the concentration of the miners. For Bitcoin, as of August 19th 2018, only four miners pools share 54.7% of the hash-rate computation power (BTC.com, AntPool, BTC.TOP, and SlushPool). If they make an agreement, they can force the evolution of the consensus algorithm, can approve (or not approve) certain kind of transactions, etc. For private blockchain, the consensus methods chosen are mainly "Proof of Authorities" or "Proof of Cooperation" where only trusted and well-selected nodes could stand as a candidate. By definition, it gives the decision-making power to a relatively small group of agents. For such a blockchain with relativly few nodes, practical Byzantine Fault Tolerance (pBST) [6] is probably the most suitable method to ensure the block consensus. The proposed new consensus has to ensure that the risk of such dominance is low. Therefore, we pose also as a requirement for *openness* that the candidate miner must not have to be authorized to stand for the role of the miner by other parties (aka permission-less blockchain). Considering this openness requirement and the possible large number of nodes, we decide to compare our proposed algorithm with PoW (and not, for instance, with pBST). In our illustrative example, if George has been selected for this period, ideally each of the other candidates must see their chances of being selected for the next round increase. This tends to maintain the number of verifiers and therefore the quality of the verifications.

E. Sustainability: energy consumption

The energy consumption is one of the most cited flaws of the PoW method, because it requires a very high hash-rate. The AntMiner S9 consumes 1273 W and must be powered continuously to expect to become a winning node. In 2014, the total energy consumption of Bitcoin per year was comparable to the total electricity consumption of Ireland [7]. Other sources estimate that electricity consumption in Bitcoin is 826 kWh/transaction [8], while banking systems such as Visa use between 1,500 and 2,000 kWh per million transactions. The proposed consensus must be of the same order of consumption as the traditional banking system.

IV. Consensus design using Condorcet

A. Previous work

In [2], a blockchain using an evolution of the Proof of Stake (PoS) algorithm that can be synchronized with market periods was proposed. Candidate miners k place their service offers in the form of voting tokens sent to the previously selected miner (this miner cannot be a candidate for this block anymore). As shown in Figure 1, they may do this for a period of time between two instances called "candidates gate opening" for the launch of the selection and "candidates gate closure" for the end. It is a sort of auction. The period of time called "Selection" is used to select the winner and to broadcast the result. From a transactional perspective, it is important to note that, in order to guarantee that the future canonical block contains all the transactions for one market period, every candidate miners progressively creates its own block during the market period. With this procedure, the selected miner can be revealed shortly after the market period without endangering the completeness of the information. For the selection of the miner who will generate the canonical block, the *stake* used in PoS is replaced by a wealth indicator $W_{t,k}$ computed from *voting tokens* $E^k_{T_{i-1}} \in \mathbb{N}$ sent by each candidate k, the *age of the last block mined by the candidate* $A^k_{T_{i-1}} \in \mathbb{N}$, and the *reputation of the candidate miner* $R^k_{T_{i-1}} \in \mathbb{N}$. The notation T_{i-1} denotes that these values are set prior to the creation of the block at T_i. A random number $U_k \in]0, 1]$ is also used. Considering the risk of decision-making power concentration, we proposed that each wealth parameter would have a specific relative weight namely $\beta A^k_{T_i-1} > \alpha E^k_{T_i-1} > \gamma R^k_{T_i-1}$ (all these weights are in $\mathbb{R}^+$). The general consensus algorithm and the miner selection algorithm of [2] is described in Table III.

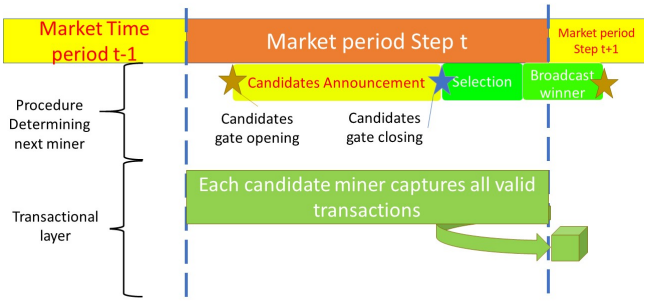

Figure 1. Relation between market period, transaction and miner selection.

TABLE III
MINER SELECTION ALGORITHM PROPOSED IN [2].

Let $\mathcal{K}$ be the set of nodes willing to support the chain at time T_i.
1. Determine the wealth of each candidate miner. We choose the following wealth criteria to define the wealth of a node k, for a given $\mathcal{K}$ ($k/in\mathcal{K}$) and for a time step T_i, as

$$W^k_{T_i} = \alpha E^k_{T_{i-1}} + \beta A^k_{T_{i-1}} + \gamma R^k_{T_{i-1}} \quad (1)$$

where we define

- $E^k_{T_{i-1}}$ as the voting token set by candidate k
- $A^k_{T_{i-1}}$ is a measure of the age. It corresponds to the difference between the number of the last block of the (current) chain also called *height of the chain* and the number of the last canonical block mined by this candidate. If the candidate has never been chosen, the corresponding age will be equal to the *height of the chain*.
- $R^k_{T_{i-1}}$ is a reputation measure: At most a candidate k has been selected for the creation of canonical blocks, the more he is trustworthy, the greater his chances of being selected again. It is computed as the sum of canonical blocks that candidate k has mined.

$A^k_{T_{i-1}}$ and $R^k_{T_{i-1}}$ are computed from the data directly contained in the chain, so these can be controlled by every participant. $E^k_{T_{i-1}}$ are put in a special transaction so all the participants can trace it.
2. Randomize. Generate a random number U_k for every candidate k with a uniform distribution in $]0, 1]$.
3. Output. The selected node has the maximum ratio W_k/U_k:

$$k^s_{T_i} = \arg\max_{k \in \mathcal{K}} \frac{W_k}{U_k} \quad (2)$$

There are several possible ways to create a voting token. A token is a digital asset with a value $\in \mathbb{N}$ (a sort of digital *"coin"*) that can be transferred without duplication between two players on the Internet, without the need for a third-party agreement. Tokens are largely used in the Initial Coin Offering (ICO) mechanism. During this offering, a number of tokens are issued and sold to investors to finance the launch and development of a blockchain project. When the blockchain is up and running, investors can use or exchange these tokens. The tokens can also be created and used to reward a node for its supporting activities. In Bitcoin, 12.5 tokens are created when a block is mined as a fee for the miner. It is also possible to issue some token when transactions are made. All these ways to manage the creation and the evolution of the number of tokens can be combined. Generally, the total amount of tokens that will be issued is known in advance and cannot be changed, and tokens are distributed progressively. It is out of the scope of this paper to describe the advantages or inconvenients of each token creation. We make the assumption that tokens are available to remunerate the miners and that the number of tokens distributed increase linearly.

The formulation of Equations 1 and 2 can be improved in three directions.

1) The number of tokens sent by one candidate can be much higher than the *age of the last block* or *reputation* of a candidate. If the volume of token sends by a candidate is an absolute value, the *age of the last block* and *reputation* have to be seen relative to other candidates. This introduces an implementation difficulty.
2) As time passes, the number of voting token increases constantly. As a consequence, in order to maintain the initial balance, α, β and γ have to constantly be adapted. In a blockchain environment, such an adaptation has to be consensually agreed by the miners'pool. This is also an implementation issue.
3) More importantly, to find a good weighting factor between the parameters, we simulated different possible configurations among which the most important were:
   - Calculate E_{bal} the total volume of voting tokens sent by each candidate, and use $E^k_{T_i-1}/E_{bal}$ instead of $E^k_{T_i-1}$
   - Calculate E_{tot} the total volume of voting tokens in the full ledger, and use $E^k_{T_i-1}/E_{tot}$ instead of $E^k_{T_i-1}$
   - Limit the number of voting tokens that a candidate may send

   To analyze these configurations, we recorded which parameter determines the winning candidate. It could be $E^k_{T_i-1}$ (i.e. the winning candidate is the one that submitted the most voting tokens), $A^k_{T_i-1}$ (i.e. the winner is the one with the oldest mined block among the candidates), $R^k_{T_i-1}$ (i.e. among the candidates, the winner is the one who was most frequently a miner) but also U_k (i.e. ignoring the other criteria, the winner is chosen only as a consequence of the value of random U_k). As this factor intervenes in the denominator of Formula (2) and can take any value between $]0 : 1]$, it appears that whatever the chosen configuration, in at least 80% of cases, this random number determines the winner. This consideration raises the following question: for a rational candidate, what would be the number of voting tokens they should send to become an official block miner? In game theory, as explained in [9], we can consider that this problem is equivalent to the imperfect information Bayesian game and this question can be analysed in that way. The conclusion of this analysis is that rational candidates (Nash equilibrium) should send as few tokens as possible, which is incompatible with the

very principle of solving the *Nothing at Stake* problem. Therefore, a new consensus algorithm is needed.

B. New consensus algorithm

The selection of the next miner based on the computation of a wealth index is replaced by a vote among candidates inspired by the theory of social choices and voting procedures[1]. Each candidate challenges the other candidates to be selected as winner for the canonical block by a ballot of voters. We use the term *duel-pair* when we describe a confrontation between two candidates (e.g. a duel-pair is $George$ against $Richard$). As it was foreseen in Section IV-A, the previously designed miner will manage and coordinate the selection. We define the following elements:

- I_m is a committee of voters, as a set of size $m \geq 1$.
- $\mathcal{K} = \{1, .., g, h, ..., k, ..., n\}$ is a set of $N = |\mathcal{K}|$ candidates. In our illustrative example, the set of candidates is composed of the set of our five verifiers: George, Henry, Edward, John and Richard.
- R_i is a weak ordering function on $\mathcal{K}$ called *preference ordering*, and has the following characteristics:
  - $R_i :\succ\rightarrow x \succ_i y$ means that voter i prefers x over y
  - $R_i :\equiv\rightarrow x \equiv_i y$ voter i is indifferent between x and y
  - Reflexive: $\forall x \in \mathcal{K}, xR_ix$ is true
  - Transitive: if $x \succ y$ and $y \succ z$ then $x \succ z$
  - Complete: $\forall x, y \in \mathcal{K}, x \neq y \rightarrow xR_iy \vee yR_ix$
- π_m is the set of all possible ordering with respect to $\mathcal{K}$
- For each voter $i \in I_m$, there is a weak ordering $B_i \in \pi_m$ on $\mathcal{K}$ called *ballot*
- $B = B_1, ..., B_m$ is the set of ballots
- The *voting procedure* is a function $v^{mn} \rightarrow v^{mn}(B) = \{k\}$, i.e. v^{mn} selects in each ballot of B the elected candidate $k \in \mathcal{K}$.

There is no indisputable choice process that allows a coherent hierarchy of preferences [11], [12]. We propose that v^{mn} needs to fulfil the following requirements.

a) Condorcet winner: any alternative which is preferred to any other by the majority of voters should be selected [13]

b) Monotonicity: If $v^{mn}(B) = \{k\}$, k is still the winner when k has been up-ranked but $h \neq k$ will not become the winner if h is down-ranked [14].

c) Independence of Irrelevant Alternatives (IIA): We will use the formulation of Independence of Smith-dominated alternative which is less demanding than the full "IIA" requirement [15]: v^{mn} is IIA when $v^{mn}(B) = \{k\}$ if $k \in S$ is a subset of B, $v^{mn}(S) = \{k\}$

d) Independence of Clones (IC): In the context of blockchain, this requirement is very important to resist to a Sybil attack. [16] defined that there are clones if $\forall c \in \mathcal{C}, \forall n \in \mathcal{J} \rightarrow c \succ_i n$ or $c \succ_i n$ s.t.

- a subset $\mathcal{C} \subset \mathcal{K}$, called "*subset of clones*", containing two or more candidates,
- $\mathcal{J} \subset \mathcal{K}$, subset of non-clone candidate
- $\mathcal{J} \cup \mathcal{C} = \mathcal{K}$ and $\mathcal{J} \cap \mathcal{C} = \emptyset$

In other words, if there are clones, $c \equiv n$ is not possible. v^{mn} is IC if the winner does not change with the addition of a non-winning candidate who is similar to a candidate already present in the ballot. One of the two conditions required to fulfil this criterion (as described in [16]) is very useful in our context:

- The winner is not a clone if it also wins when the set of candidates is reduced to the subset of non-clone. $v^{mn}(B) = n = v^{mn}(B^*)$ s.t. $\notin \mathcal{C}$) and B^* is a set of ballots on $\mathcal{J}$.

e) Resolvability: if the first round of voting rules generates a tie among candidates, a second round generates a unique winner amongst the tied candidates [16].

f) Voting Procedure Efficiency: For implementation reasons, if a Condorcet winner exists, it has to be selected by a single round of votes.

Owing to these requirements, and in particular IIA, classical positional-scoring-rule voting mechanisms like majority, plurality or Borda count ([17]) are not valid. There are only two voting mechanisms that fulfil all of our requirements: the Condorcet with Decreasing Ranked (Duel-)Pairs is also called the Tideman Method [16] and the Schulze Method [18] (that is also a Condorcet voting mechanism). [19] demonstrated that the Schulze method is more vulnerable to *coalition destructive manipulation* (defined as a coalition of voters casting false votes to make a potential winner less favourable). For the Tideman Method, this paper also describes that this sort of attack can be classified as *NP-hard* using the computational complexity theory. For this reason, we decided to implement the Tideman Method.

In our design, the voters are not human being but the criteria *voting token*, *age of the last block*, *reputation*, and *random* instead. The question about the weight of α, β, and γ presented in [2] and described at Section IV-A is now transferred to the number of votes that corresponds to each parameter including a new factor $\eta \in \mathbb{R}^+$).

In other words, the committee I_m comprises:

- Voter $\mathcal{E}$ with α votes for the criterion *voting token*
- Voter $\mathcal{A}$ with β votes for the criterion *age of the last block*
- Voter $\mathcal{R}$ with γ votes for the criterion *reputation* and
- voter $\mathcal{U}$ with η votes for the criterion *random*

One should also note that the generation of the $\mathcal{U}$ has to be transparent in order that all nodes can redo the computation and come to the same conclusion and thus avoid manipulation. To achieve this, as an example, the random number for each candidate can be the ratio between the hash value of their public address divided by the sum of the hashes of all the candidates.

As a procedure to elect the canonical miner, we thus propose to adapt the procedure presented in [16] as described in the next subsection.

[1]The reader can find some theoretical notions in [10], Chapter 4.

TABLE IV
ILLUSTRATION DUEL-PAIRS MATRIX MTOT WITH CYCLE

	George	Henry	Edward
George	0	7	-5
Henry	-7	0	6
Edward	5	-6	0

C. Election procedure

1) Tally: For g and h, two distinct elements of $\mathcal{K}$, let us define $M^i_{gh} = +1$ if the candidate $g \succ_i h$, $M^i_{gh} = -1$ if $h \succ_i g$, $M^i_{gh} = 0$ if $g \equiv_i h$. To give an illustration of this rule, if candidate g has submitted E_g voting tokens and candidate h has submitted E_h tokens, then $M^{\mathcal{E}}_{gh} = 1$ if $E_g > E_h$, $M^{\mathcal{E}}_{gh} = -1$ if $E_g < E_h$ and $M^{\mathcal{E}}_{gh} = 0$ if $E_g = E_h$.

We can create a duel-pairs matrix $\mathcal{M}^i$, of dimensions NxN, with all the combinations for voter i:

$$\begin{bmatrix} 0 & M^i_{12} & \dots & M^i_{1g} & M^i_{1h} & \dots & M^i_{1k} & \dots & M^i_{1n} \\ M^i_{21} & 0 & \dots & M^i_{2g} & M^i_{2h} & \dots & M^i_{2k} & \dots & M^i_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M^i_{g1} & M^i_{g2} & \dots & 0 & M^i_{gh} & \dots & M^i_{gk} & \dots & M^i_{gn} \\ M^i_{h1} & M^i_{h2} & \dots & M^i_{hg} & 0 & \dots & M^i_{hk} & \dots & M^i_{hn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M^i_{n1} & M^i_{n2} & \dots & M^i_{ng} & M^i_{nh} & \dots & M^i_{nk} & \dots & 0 \end{bmatrix}$$

It should be noted that $M^i_{gh} = -M^i_{hg}$. We make a new duel-pairs matrix called *total vote matrix*:

$$\mathcal{M}^{tot} = \alpha\mathcal{M}^{\mathcal{E}} + \beta\mathcal{M}^{\mathcal{A}} + \gamma\mathcal{M}^{\mathcal{R}} + \eta\mathcal{M}^{\mathcal{U}}$$

It is important to stress that all the elements of the $\mathcal{M}^{tot}$ are public and transparent, hence the canonical block is checked and could be approved by a majority of nodes if the whole procedure has been executed correctly:

- There is a transaction of voting tokens from each candidate to the previous miner. It is broadcast through the *Mem Pool*, and thus visible to all.
- The *age of the last block* mined by a candidate and its *reputation* are computed from information contained in the chain.
- $\mathcal{U}$ is defined to be transparent (cf. Section IV-B).
- α, β, γ and η are also known as there are an important part of the chain setting.

2) Sort and lock: From the duel-pairs matrix $\mathcal{M}^{tot}$, it is possible to rank in descending order each duel-pair by creating a *sorted table*. Taking into account the preference to keep duel-pairs with a higher value M^i_{gh}, the duel-pairs that would create a cycle are skipped.

For instance, let's consider the following $\mathcal{M}^{tot}$ (see Table IV) with three of candidates serving as an illustration. The ranking duel-pairs matrix is shown in Table V. As the ordering is transitive, the relationship $George > Henry$, $Henry > Edward$ and $Edward > George$ is impossible as it forms a cycle. Considering the fact that $Edward > George$ is ranked lower than the other duel-pairs, this relationship is skipped. The sorted table with no cycle is referred to as being *locked*. From this locked table, it is possible to order all the candidates. In our example, the final order of the candidates is then $George - Henry - Edward$ and $George$ is the winner.

TABLE V
ILLUSTRATION MTOT WITH CYCLE: RANKING OF DUEL PAIRS.

Duel	$\mathcal{M}$
George-Henry	7
Henry-Edward	6
Edward-George	5

TABLE VI
ILLUSTRATION MTOT WITH TIES

	George	Henry	Edward	John
George	0	-6	-6	-6
Henry	6	0	7	7
Edward	6	-7	0	8
John	6	-7	-8	0

3) Resolving ties: There are two possible types of tie: the first one is a tie to define rank in the ranked table that could endanger the resolution of cycles. The second one is a tie in the election of the miner.

To resolve the first tie issue, we use the following method:

- Case 1: only two tied relations and no common *loser*: The direct duels of the loser are used to separate them. For instance, if relation $a > b$ is tied with relation $c > d$, look at the relation between b and d. If $b > d$, take the convention that the relation $c > d$ is ranked higher than $a > b$ (and the other way if $d > b$).
- Case 2: only two tied relations and a common *loser*: We will use the direct confrontations of the *winners*. For instance, if relation $a > b$ is tied with $c > b$, and if $a > c$, we will consider that $c > b$ will be ranked before $a > c$.
- Case 3: more than two tied relations and no common *loser*: The different direct confrontations between losers are analysed to create a relative order between them. To do that, the previous relations (highest ranked) are used first. The rank is set from the weakest to the strongest.
- Case 4: more than two tied relations and common *losers*: Using the same logic as before, the direct confrontations of the *winners* creates a relative order. The rank is also set from the weakest to strongest.

To illustrate the first type of tie, we will use $\mathcal{M}^{tot}$ as presented in Table VI where only four candidates are standing for the canonical block: If it is obvious that the first duel-pair in the ranking table will be $Edward > John$ (score = 8), there is a first tie between two relationships: $Henry > Edward$ and $Henry > John$ with $\mathcal{M}^{tot} = 7$. Observing that candidate $John$ *loses* against $Edward$, we will consider, by convention, that the duel $Henry > John$ is stronger (higher ranked) than $Henry > Edward$. For the next tie $Henry > George$, $Edward > George$ and $John > George$. From the previous duels, it is possible to determine the relative order between these candidates, namely $Henry > Edward > John$. The tied duels can now be ranked (from the duel with the weakest candidate to the duel with strongest): $John > George$,

TABLE VII
ILLUSTRATION OF A TIE BETWEEN POSITIVE ROWS

Value for:	George	Henry	Edward	John	Richard
Age	2	4	3	5	5
Reputation	3	1	1	5	4
Voting tokens	4	1	1	5	5

TABLE VIII
ILLUSTRATION OF $\mathcal{M}^{tot}$ TIE BETWEEN ROWS

	George	Henry	Edward	John	Richard
George	0	0	0	-4	-4
Henry	0	0	2	-2	-2
Edward	0	-2	0	-2	-4
John	4	2	2	0	0
Richard	4	2	4	0	0

$Edward > George$, and $Henry > George$. Note that in this example, there is no cycle ($Henry > Edward > John > George$) and thus the sorted table is equal to the locked table and $Henry$ becomes the canonical miner (as pure Condorcet winner). To avoid these complications, the proposed algorithm must check if there is a pure Condorcet winner prior to launching this procedure. $Henry$ is the only candidate with non-negative values for his row (see the corresponding row in Table VI).

For the second type of tie, there is more than one row with no negative value. Therefore choosing from tied candidates with equal probability (as a purely random) would reward the creation of clones and thus not fulfil the IC requirement. Ties can be resolved in a way that does not reward or penalize clones by choosing a voter who selects the only candidate among those who are tied. This voter can be considered *dictatorial* as defined by [11]. Considering that the choice of the miner is mostly a question of trust, we decided to select $\mathcal{R}$ reputation as this voter to solve the issue. In practice, if there are candidates who are tied, the best-placed candidate regarding reputation among the tied candidates is selected. If, in this selection, there are still candidates with the exact same reputation, then the selection is made among them by the voter $\mathcal{A}$ age of the last mined block. If there is still a tie, $\mathcal{E}$ can be used, and finally $\mathcal{U}$. If there is still a tie, a Sybil attack is probably occurring. In this case, the winner will be the Condorcet winner after elimination of all the tied candidates from ballots.

As an example, for this last rule, Table VII presents different criteria of each of the five miners. The result of the random number $\mathcal{U}$ is : $George$= 0.11, $Henry$ =0.61, $Edward$ = 0.37, $John$=0.24, and $Richard$=0.51. With α=β=γ=η=1, Table VIII shows the corresponding $\mathcal{M}^{tot}$.

Upon examination of $\mathcal{M}^{tot}$, as two lines have only positive values, that means that two candidates are possibly Condorcet winner, $John$, and $Richard$. As $John$ has the highest reputation, he is declared the winner.

Table IX summarizes the proposed method.

TABLE IX
PROPOSED CONDORCET-TIDEMAN RESOLUTION

1. Tally: Create the different matrix.

- From the data contained in the blockchain, for every candidate that has submitted a voting token before gate closure, determine the different ballots: Age, Reputation and Voting token
- Create the $\mathcal{U}$ matrix: attribute a random number to each candidate
- Compute $\mathcal{M}^{tot} = \alpha\mathcal{M}^{\mathcal{E}} + \beta\mathcal{M}^{\mathcal{A}} + \gamma\mathcal{M}^{\mathcal{R}} + \eta\mathcal{M}^{\mathcal{U}}$

2. Check if there is an immediate Condorcet Winner.

If there is one and only one row with non-negative values in the $\mathcal{M}^{tot}$ matrix, this row corresponds to the pure Condorcet winner.

3. Check if there are ties between potential Condorcet winners.

In the $\mathcal{M}^{tot}$ matrix, if there is more than one row with only positive or equal-to-zero values, these rows correspond to the different potential Condorcet winner. Thus, the dictatorial rule has to be applied.

- Create a new matrix with only the tie candidates with their respective score in $\mathcal{R}$, $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{U}$
- Check if one candidate has the highest score in $\mathcal{R}$, if so, this candidate is the winner.
- If there is still a tie, check the same for $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{U}$ respectively. If there is only one candidate that has a higher score, there are declared the winner.
- If there is still a tie, delete the tied candidates from the ballots and restart the procedure

4. Apply the full Tideman procedure.

All the rows in $\mathcal{M}^{tot}$ contain at least one value which is strictly negative.

- *Sort:* Create a ranked table with the duel-pairs. Stop when you reach a strictly negative value.
- *Lock:* Resolve the tie to define rank in the sorted duel pairs table (see the previous chapter).
- *Sub-routine resolving cycles*.
  - Gradually create a direct graph starting with the first candidate of the first-ranked duel-pairs.
  - Detect if there is a cycle. If a cycle occurs, go to the next ranked duel pairs.
- When the bottom of the ranked duel-pairs'list is reached and if there are no more cycles, the winner is the first-ranked candidate.

V. MAPPING REQUIREMENTS TO PROPOSED APPROACH

In this section we argue and illustrate how the proposed method solves the issues defined in Section III.

A. Synchronicity

The synchronicity issue is solved by the design of the voting process itself, namely by the open-closed candidates'gates as presented in Figure 1. To correctly define these two important events, a time reference needs to be used. In Bitcoin protocol, this reference is the UNIX Epoch Time that is used when a transaction and a block is created. The reason for its use is to facilitate the ordering of the block (block number) and to avoid manipulation of this ordering. This reference can also be used in the context of this paper. UNIX Epoch time corresponds to the number of seconds that have elapsed since 00:00:00 UTC, January 1st, 1970. The accuracy of this time reference is to the second which is also impacted by the *leap second* which is the adjustment occasionally applied to UTC with a probability of $3.2\text{x}10^{-8}$. If the market period is quite large, while also considering that it is the time elapsed during the different moments (gate opening-closing), it is not an issue to use this time reference. For instance, if the time elapsed between two

gate openings is 100s, the accuracy is 1%. Candidates and other nodes need to take that inaccuracy into account. Network Time Protocol (NTP) can be used as well. However, NTP has encountered severe attacks (Denial of service) and a patch was issued in August 2018. This is the reason why it is not recommended here.

B. Failure to reach consensus

This requirement is fulfiled thanks to three different defenses. First, the proposed algorithm is based on the Tideman method that has the characteristics described in section IV-B and ensures a proper selection of the Condorcet winner if there is one. Second, if there is a tie at the first rank of the list of ranked duels, we introduce a dictatorial selection to separate candidates and determine a winner. Finally, if there is still a tie, we make a subset of all the candidates that are not tied and find the Condorcet winner from this subset.

We ran several simulations of the proposed algorithm by combining the following factors:

- Change the relation between α, β, γ and : eta: e.g. a strict equality (i.e. $\beta = \alpha = \gamma = \eta$), than $\beta = 2\alpha = 3\gamma$ to a ratio of five between them (i.e. $\beta = 5\alpha = 25\gamma$)
- Change the numbers of candidates: from 2 to 10, afterward from 100 to 1000 (by increments of 100)
- The age, reputation and voting token ballots were chosen randomly in a large possibility space.

For every round of the simulations, for which certain results are shown in Table XII, we recorded how the winner was chosen: immediate Condorcet (*IC*), after dictatorial selection (and in this case which one (Reputation *DR*, Age *DA*, Voting token *DE*, or random number *DU*)), or after a full Tideman procedure (*FT*). The results in the aforementioned table are expressed as a percentage of the cases.

It is interesting to see that, when some ordering between voters (e.g. $\beta > \alpha > \gamma > \eta$) is implemented, the risk of a tie and therefore the need to apply the entire Tideman procedure is largely reduced. This particularity leads to a more efficient algorithm in these cases. As a further assumption for all the next simulations, we fix the values of voters to $\beta = 2\alpha = 2\gamma = 2\eta$. We will explain this choice later on.

Obviously, as we may expect, when the number of candidates increases, so does the risk of a tie. In all these simulations, the algorithm always determines a winner.

C. Sybil Attack

We made the choice of the Tideman procedure because of its characters' immunisation against clones. Nevertheless, there is still a small risk of not having a consensus when all the candidates are clones of each other. In order to explain this, we go back to our illustrative case.

Imagine that $John$ agrees with $Alice$ to accept the faulty last transaction (as shown in Table II). Even if $John$ knows the other candidates and their relative position for A and R compared to him, he still doesn't know the random factor and how many voting tokens the other candidates will submit.

TABLE X
TYPICAL EXAMPLE $\mathcal{A}$ MATRIX WITH TWO NEW IDENTITIES

	George	**...**	**Richard**	**New 1**	**New 2**
George	0	...	...	-1	-1
...	...	...	...	-1	-1
Richard	...	...	0	-1	-1
New 1	1	1	1	0	0
New 2	1	1	1	0	0

TABLE XI
TYPICAL EXAMPLE $\mathcal{R}$ MATRIX WITH TWO NEW IDENTITIES

	George	**...**	**Richard**	**New 1**	**New 2**
George	0	...	...	1	1
...	...	...	...	1	1
Richard	...	...	0	1	1
New 1	-1	-1	-1	0	0
New 2	-1	-1	-1	0	0

$John$ has two options: he could make clones of himself or he could create new false identities.

1) Clones with new false identities: We first define the consequence of new identities.

New identities are characterized by an age equal to the height of the chain i.e. they have the highest age score in relation to other candidates. The reputation is equal to zero as well (they did not yet create any block). Table X is an example of a typical $\mathcal{A}$ matrix and Table XI is an illustration of an $\mathcal{R}$ matrix for these new identities.

Off course, if the number of new identities is higher than the number of real candidates (let's say, for instance, ten new identities in our case), these drastically increase their chance to win as the random $\mathcal{U}$ is equally distributed. So, it is important that there is a cost associated with the use of or creation of these identities. There are several ways to implement such a cost e.g. one or a combination of the following :

- Impose a minimum value for voting tokens that a candidate must send to be approved as a valid candidate
- Fix the maximum number of candidates that can stand for a specific block (e.g the first 100 to submit a voting token between the opening/closing of the candidates' gate)
- Fix the absolute number of candidates (e.g. 10 000 but probably as a function of the chain height)
- etc.

The relationship between α, β, γ and η has also an impact on the success of such an attack.

2) John's clones: The Tideman method ensures that if $John$ is not a Condorcet winner, none of his clones will be. Let's have a look at our illustration. $John$'s clones will have exactly the same value as the "original" $John$ for the criteria Age and Reputation. If the random method to create $\mathcal{U}$ is the one taken as an example in section IV-B, $John$'s clones will have the same $\mathcal{U}$ as well. For example, considering the following ballots (with $\beta = 2\alpha = 2\gamma = 2\eta$)

- $\mathcal{A}$: $George$: 3, $Henry$: 5, $John$: 0, $John1$: 0, $John2$: 0, $Richard$: 1, and $Edward$: 2
- $\mathcal{R}$: $George$: 1, $Henry$: 1, $John$: 1, $John1$: 1, $John2$: 1, $Richard$: 2, and $Edward$: 1

- $\mathcal{E}$: $George$: 16, $Henry$: 5, $John$: 5, $John1$: 998, $John2$: 16, $Richard$: 997, and $Edward$: 47
- $\mathcal{U}$: $George$: 0.78, $Henry$: 0.55, $John$: 0.82, $John1$: 0.82, $John2$: 0.82, $Richard$: 0.67, and $Edward$: 0.05

where $John1$ and $John2$ are clones of $John$. It can be proven that $George$ is declared as the winner for the canonical bock whatever the number of $John$'s clones (here two) and even if $John$ places the biggest bet.

D. Dominance issue

Firstly, regarding the *openness* requirement, the only condition for a node to stand as a candidate miner is to have some voting tokens. The whole miner selection procedure is done by the previous miner. To tackle this issue, it is interesting to observe the conditions where dominance could occur. Dominance, as a concentration of decision-power, appears when candidates with a high stock of voting tokens and high reputation have the maximum probability of winning the canonical block designation procedure. To counteract this, we propose to give a greater probability of winning when the candidate has not won for a while or is a new candidate (taking into account the risk of a Sybil attack). It is the age criteria. Upon that the random number makes the result more uncertain. The good balance between the four criteria is important to reach the desired result. As already proposed in [2], β has to be the highest criteria.

To illustrate this and using our illustrative use case, we carry out some simulations with our five miners/candidates. In these simulations,we fixed some specific rules. Each of them starts with the same fixed number of voting tokens (1000) in their respective wallet, and we increment the chain, letting the proposed algorithm play its role. The candidates may not refill their wallets with new tokens and there is no minimum number of voting tokens imposed to declare oneself as a candidate. We do these simulations with different relationships between α, β, γ, and η. Two results are shown in Figure 2 and 3. It can be seen that with no difference between the criteria/voters (Figure 2), dominance occurs very quickly. However, with a proper choice of these parameters (e.g. $\alpha = 1, \beta = 2, \gamma = 1$ and $\eta = 1$, Figure 3), a balance between the candidates takes place. Using the same simulation, we let the chain expand to 1000 with this criteria relationship (thus with a little benefit for age). The repartition for the winner between candidates is quite well balanced: $Edward$: 28%, $Richard$: 23%, $George$: 19%, $Henry$: 16% and $John$: 14%

So we can declare that our proposal fulfils this requirement.

E. Decision map

A consensus algorithm that effectively fulfils all these requirements passes through, in our case, a good determination of α, β, γ, and η. We can summarize the decision that we would like to implement as:

- If a candidate wins against all other candidates regarding all four- or three-votes/criteria, they become the winner.

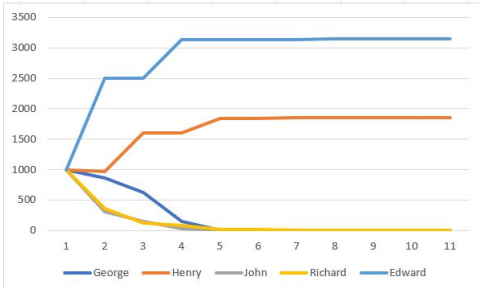

Figure 2. Domination with $\alpha = \beta = \gamma = \eta = 1$.

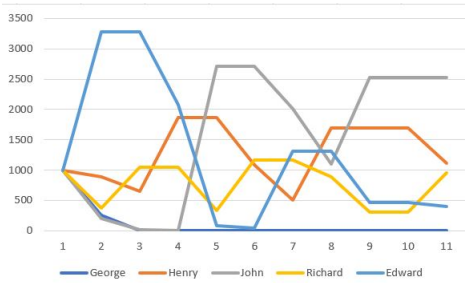

Figure 3. Domination with $\alpha = \gamma = \eta = 1$ and : $\beta = 2$.

- A candidate is designated if they win against all other candidates for the following combination: Age and Reputation, or Age and Token, or Age and Random

The repartition $\alpha = 1, \beta = 2, \gamma = 1$ and $\eta = 1$ fulfils this decision map.

F. Energy consumption

We would like to compare our algorithm with PoW used by Bitcoin. By analysing the Bitcoin blockchain (e.g. www.blockchain.com), we observed that one block (of 1.3 MB) contains up to 2500 transactions (tx). We will take this number as an assumption for our evaluation. To realise the consumption estimation, we use a personal computer with the following characteristics: Intel (4) Core i7 (max core speed 2500 Hz, bus speed 100 MHz) with 10 GB memory. The algorithm is written in Python 2.7. Different phases need to be correctly separated:

1) Access to the transactions pool in order to find candidates and their $\mathcal{E}$
2) Search in the chain the value of $\mathcal{A}$ and $\mathcal{R}$ for each candidate,
3) Compute $\mathcal{U}$ For each candidate,
4) Resolve the Condorcet-Tideman procedure
5) Broadcast the result
6) During this process each candidate creates their own block in parallel.

As the proposed consensus is not online (and thus it is not possible to evaluate the energy needed to assess and examine data from the chain), we will focus our analysis on the resolution of the Condorcet-Tideman procedure. The resolution of this procedure depends not only on the number of candidates but also on the relative weight factor between

TABLE XII
TABLE OF SIMULATION RESULTS FOR REQUIREMENT NO FAILURE TO REACH CONSENSUS

Simulations ref.	Beta	Alpha	Gamma	Eta	# candidates	% IC	%DR	%DA	%DE	%DU	%FT
1	1	1	1	1	1000	50.00%	22.00%				28.00%
2	2	1	1	1	1000	9.00%	0.02%				90.98%
6	20	10	5	1	1000	100.00%					
51	1	1	1	1	500	43.67%	31.33%				25.00%
52	2	1	1	1	500	11.30%	0.10%				88.60%
56	20	10	5	1	500	100.00%					
101	1	1	1	1	50	35.8%	57.5%				6.75%
102	2	1	1	1	50	35.90%	0.30%				63.80%
106	20	10	5	1	50	100.00%					
151	1	1	1	1	5	54.00%	44.00%				2.00%
152	2	1	1	1	5	83.80%	4.30%				11.90%
156	20	10	5	1	5	100.00%					

criteria. As already mentioned, we used the same weight factor as that used for the previous simulations. For instance, with these assumptions, we observed that the average time to find a winner from 5000 candidates was 180 seconds and the consumption of power was 6 Watts for only one miner (the previous designated one) during this process. So, this gives us $\frac{6W \times 180s}{2500tx} = 0.432J/tx = 1.2 \times 10^{-7} kWh/tx$. The evolution of the resolution time is virtually quadratic to the number of candidates, there is clearly room for improvement here.

VI. CONCLUSION AND FURTHER WORK

The contribution of this paper takes into account the observation of the complementary works of [2] and [9] to propose a more efficient miner selection procedure avoiding the concentration of decision-making power, consuming less energy than the PoW, while also providing the same resistance to a Sybil attack. The criteria Independence of Clones and Independence of Irrelevant Alternatives are the cornerstone of the Tideman's procedure to resolve a Sybil attack. This is the reason why it has been chosen as the main mechanism to determine the miner of the canonical block. The good relationship (vote weight) between the different criteria needs to be further investigated. We discover that a small advantage for the criteria 'Age' seems to be enough to avoid the concentration of decision-power (dominance), and also to avoid the risk of a tie in the Condorcet procedure and therefore to be more energy efficient. Further works are also needed to optimize the energy consumption and the resolution time when the number of candidates becomes very high.

Regarding further work, as explained in Figure 1, notification of the selected miner is done after the market period. There is a risk that while this information is effectively broadcast, it can arrive late to some candidates. In order to prevent information loss, these nodes will broadcast their own block. Other nodes that do not receive the information about the canonical block could accept this block and there is then the creation of a fork. As the miner selection procedure relies on the previous miner, there is a risk that it could act incorrectly. Some methods such as the Byzantine Fault Tolerance [6] can be used to resolve this issue.

REFERENCES

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

[2] D. Vangulick, B. Cornélusse, and D. Ernst, "Blockchain for peer-to-peer energy exchanges: design and recommendations," in *Proceedings of the XX Power Systems Computation Conference (PSCC2018)*, 2018.

[3] D. Vangulick, B. Cornélusse, T. Vanherck, O. Devolder, and D. Ernst, "E-cloud, the open microgrid in existing network infrastructure," in *Proceedings of the 24th International Conference on Electricity Distribution*, 2017.

[4] Z. Trifa and M. Khemakhem, "Sybil nodes as a mitigation strategy against sybil attack," *Procedia Computer Science*, vol. 32, pp. 1135–1140, 2014.

[5] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[6] M. Castro and B. Liskov, "Byzantine fault tolerance," Dec. 30 2003, uS Patent 6,671,821.

[7] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," 2014.

[8] A. de Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801 – 805, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2542435118301776

[9] D. Vangulick, B. Cornélusse, and D. Ernst, "Blockchain for peer-to-peer energy exchanges: Probabilistic approach of proof of stake," in *Proceedings of the CIRED WORKSHOP (CIRED 2018)*, 2018.

[10] S. J. Brams and P. C. Fishburn, "Voting procedures," *Handbook of social choice and welfare*, vol. 1, pp. 173–236, 2002.

[11] K. Arrow, "Individual values and social choice," *Nueva York: Wiley*, vol. 24, 1951.

[12] M. A. Satterthwaite, "Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions," *Journal of Economic Theory*, vol. 10, no. 2, pp. 187 – 217, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022053175900502

[13] J. A. N. de Caritat Condorcet, "Essais sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix," 1785.

[14] D. R. Woodall, "Monotonicity of single-seat preferential election rules," *Discrete Applied Mathematics*, vol. 77, no. 1, pp. 81–98, 1997.

[15] J. H. Smith, "Aggregation of preferences with variable electorate," *Econometrica: Journal of the Econometric Society*, pp. 1027–1041, 1973.

[16] T. N. Tideman, "Independence of clones as a criterion for voting rules," *Social Choice and Welfare*, vol. 4, no. 3, pp. 185–206, 1987.

[17] J. C. de Borda, "Mémoire sur les élections au scrutin," 1781.

[18] M. Schulze, "A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method," *Social Choice and Welfare*, vol. 36, no. 2, pp. 267–303, 2011.

[19] D. C. Parkes and L. Xia, "A complexity-of-strategic-behavior comparison between schulze's rule and ranked pairs." in *AAAI*, 2012.