Improving Delegated Proof-of-Stake

January 8, 2020

Abstract

1 Introduction

Developing protocols for self-interested agents to come to agreement on what data is saved, shared and valid can be a challenging aspect of multi-agent systems. Distributed Ledger Technology (DLT) [1 AAMAS paper cite] provides a unique solution to this problem through its consensus protocols that allow a possibly unlimited and anonymous number of self-interested agents (each running a node of the DLT) to maintain consensus on data without a central authority. A consensus protocol consists of two parts: (1) a sybil control mechanism, which give no advantage to agents who create multiple accounts; and (2) a data agreement protocol that details the rules regarding how the next valid block of data is agreed.

In this article, we focus on a special type of DLT, called blockchains [11 AAMAS paper cite]. A blockchain is a series of blocks of data (containing multiple transactions) linked together via cryptographic hashes. A blockchain increases in size over time as new blocks keep being added. Each node of the blockchain network maintains its own copy of the blockchain. When a node receives a new block, the data within it needs to be validated according to the data agreement protocol. If a node deems a block valid, then the block is added to that nodes copy of the blockchain. Therefore, everyone who has the same copy of the blockchain has agreed to all the data within it

Proof-of-work was the first Sybil control mechanism used by Bitcoin [31], the first blockchain implementation. The idea behind this Sybil control mechanism is that nodes race to solve a complex computational problem, where the first node to solve it correctly is allowed to publish a block of data to be added to the blockchain, where the block consists of a set of previously unconfirmed transactions. The likelihood of an account solving this computational problem first is proportional to the computational power that account has access to (compared with the total computational power of the network). This meaning that creating multiple accounts gives a person no advantage as computational power cannot be used in parallel by multiple accounts.

The main issue of Proof-of-Work is that it is energy intensive due to multiple computational entities attempting to solve a difficult computational problem at the same time. Another issue is the number of transactions per second a Proof-of-Work system

allows, typically no more than 10 CITE. This is significantly less then something like the SWIFT payment network that allow for XXXXX transactions per second.

But there has been many other proposed consensus protocols in the distributed ledger technology domain. One such protocol is Delegated-Proof-of-Stake (DPoS). In a DPoS blockchain, there are a limited number of nodes who can produce blocks. These block producers (the committee) can change over time through elections. In DPoS, every cryptocurrency coinholder is allowed to cast a ballot of his preferred block producer candidates. The weight of a ballot depends on the number of coins in her possession. Then all of the ballots are counted and the top k candidate block producers with the most votes become the elected block producers.

This voting system is known as approval voting CITES. Yet the academic literature has identified two main flaws in approval voting: (1) non representation of the voters; and (2) easy to strategically manipulate. Therefore in this paper, we introduce a method to solve both of these flaws. Firstly we develop an election algorithm that generates committees that satisfy representation axioms in the literature, specifically *justified representation* and *extended justified representation*... and secondly...?

This paper is structured as follows. Section two introduces the election notation, the committee selection rules and how a committee is judged on the quality of representation. Note that the notation presented has been designed to allow voters to have multiple votes, as they would in a cryptocurrency environment. Section 3 details our algorithm for searching for an elected committee that satisfies extended justified representation. This algorithm has been modified for speed reasons, from one recently presented in the literature... Section 4 shows how the algorithm can be modified to make it difficult to strategically manipulate. Finally Section 5 concludes.

2 Notation

We consider an election (N, C) where there is a set of nominators N and a set of candidates C, which we can represent as a bipartite graph $(N \cup C, E)$. There is an edge $(n, c) \in E$ whenever nominator $n \in N$ votes for candidate $c \in C$.

We look to elect $1 \leq k \leq |C|$ candidates into the committee S. To do so, each nominator $i \in N$ provides a list of approved candidates $A_i \subseteq C$ where the size is restricted by a variable x, i.e. $|A_i| \leq x$. We refer to the collection of approval ballots as A, where $\forall n \in N$ then $A_n \in A$. To understand who is elected from these approval ballots, we will use three 'helper' vectors. Firstly, we have a vector $b \in \mathbb{R}^N_{\geq 0}$, recording the *budget* of each nominator, i.e. the amount of cryptocurrency held. Secondly, a vector $w \in \mathbb{R}^C_{\geq 0}$ records the vote *weight* assigned to each candidate. And thirdly a vector $r \in \mathbb{R}^N_{\geq 0}$ records the number of *representatives* each nominator has in the elected committee.

Committee Selection Rules

Given an election (N, C) and the nominator's budget b, we define the `PAV-score` of a committee $W \subseteq C$ as:

$$\texttt{PavSc} \;=\; \sum_{i=1}^{|N|} \Big(\sum_{j=1}^{|A_i \cap W|} \frac{1}{j} \Big) \cdot b_i$$

Proportional Approval Voting (PAV) is the committee selection rule that, given an election (N, C) and a committee size k, outputs all size-k committees with the highest PavSc; finding a committee in the output of this rule is NP-hard.

Committee Quality

Different committees have different possible qualities of representation. To define if a committee represents the electorate well, we introduce the following axioms:

Definition 1. *A committee S of size k satisfies* Justified Representation *if there does not exist a group of nominators $M \subseteq N$ where:*

- $\sum_{m \in M} b_m \geq \sum_{n \in N} b_n$
- $|\bigcap_{m \in M} A_m| \geq 1$, *and*
- *For all* $m \in M$, $|S \cap A_m| < 1$.

The rationale behind the justified representation definition is that if k candidates are to be selected, then intuitively, each group of nominators that has a budget of $\frac{\sum_{n \in N} b_n}{k}$ deserves a representative. So this group should not be completely unrepresented.

We can also generalise justified representation into the following axiom:

Definition 2. *A committee S of size k satisfies* Extended Justified Representation *if for any integer $1 \leq j \leq k$, there does not exist a group of nominators $M \subseteq N$ where:*

- $\sum_{m \in M} b_m \geq \frac{j}{k} \cdot \sum_{n \in N} b_n$
- $|\bigcap_{m \in M} A_m| \geq j$, *and*
- *For all* $m \in M$, $|S \cap A_m| < j$.

The rationale behind the extended justified representation definition is that if k candidates are to be selected, then intuitively, each group of nominators that has a budget of $\ell \cdot \frac{\sum_{n \in N} b_n}{k}$ deserves ℓ representatives. So there should be a member of this group with at least ℓ representatives.

3 Local Search - PAV Algorithm

We have currently made the heuristic decision to start the algorithm from the committee that contains the block producer candidates with the most votes and generate new committees by replacing the least popular block producer candidate first. This starting point is subject to change because having a fixed starting committee probably means that block producer candidates would still be able to strategically vote (perhaps using vote buying). This issue will be tackled by replacing the fixed committee starting

point with a randomly chosen committee and deciding on a method to compare different returned committees produced by the algorithm. Given different starting points, the algorithm could return different committees due to: (a) the algorithm's approximate nature; and (b) it is possible that more than one committee could have the same score.

Algorithm 1: The Local Search Proportional (LS-PAV) Algorithm

function `LS-PAV (N, C, A, S, b)` **returns** a committee S;
Input: $\langle N, C, A, S, b\rangle$, $S \subseteq C$, $b \in \mathbb{R}^N_{\geq 0}$; where N is the set of nominators, C is the set of candidates, A is the set of approval ballots, S is the currently selected committee (indexed from $0, ..., |S|-1$) and b is the vector of nominator budgets.
Output: $\langle S^*\rangle$; where S^* is the definitive selected committee.
begin:
double `PavSc` $= 0.0$; //initialising PAV score counter
vector r, w; //declaring helping vectors: representatives and vote weight
bipartite graph $G := (N \cup C, E)$; //saves the nominator/candidate relationships
int `SPointer` $:= |S| - 1$; //counter that tracks attempts to improve the committee starting from the end. Everytime we replace someone, the algorithm goes back to the end (i.e. back to $|S|-1$).
for all $(n \in N)$ **do**
  for all $(c \in A_n)$ **do**
    add edge (n, c) **in** E; //connect each candidate with it's nominators in the graph
    $w_c = w_c + b_n$; //update the maximum vote weight of a candidate
  end for
  $r_n := |S \cap A_n|$; //the number of current representatives for each nominator
  `PavSc` $:=$ `PavSc` $+ ((\sum_{q=1}^{r_n} \frac{1}{q}) \cdot b_n)$; //update current PAV score
end for
//now we have the PAV score of the most popular committee
//Now we attempt to find new committees that improve the PAV score
while (`SPointer`$>= 0$) **do**
  $c^- = S_{|\texttt{SPointer}|}$; //the candidate to remove
  double `lostPavSc` $:= 0.0$; //the PAV score lost by removing a candidate
  vector $r' = r$; //keep a temporary number of representatives r': if we find a worst committee, we will just keep r instead r'
  for all $((n, c^-) \in E)$ **do**
    `lostPavSc` $:=$ `lostPavSc` + $(\frac{1}{r_n} \cdot b_n)$; //updating the lost PAV score
    $r'_n - -$; //updating the temporary vector
  end for
  for all $(c^+ \in C \backslash S$, where $w_{c^+} \geq$ `lostPavSc` $+ \frac{|N|}{|S|^2})$ **do**
    double `addedPavSc` $:= 0.0$; //PAV score gained by adding this candidate
    for all $(n, c^+) \in E$ **do**
      `addedPavSc` $:=$ `addedPavSc` + $(\frac{1}{r'_n+1} \cdot b_n)$; //updating the added PAV
      $r'_n + +$; //updating the temporary vector
    end for
    if (`PavSc` $-$ `lostPavSc` $+$ `addedPavSc` $\geq$ `PavSc` $+ \frac{|N|}{|S|^2}$) **then**
      $S := (S \backslash \{c^-\}) \cup \{c^+\}$; $r := r'$; //save the improved committee
      `PavSc` $:=$ `PavSc` $-$ `lostPavSc` $+$ `addedPavSc`; //update the score
      `SPointer` $:= |S|$; //counter is reset to check again
      break for;
    end if
  end for
  `SPointer`$--$; //continue the search to look for an elected candidate to replace
end while
return $\langle S\rangle$;
end.