An Analysis of smart voting in liquid democracy

Giannis Tyrovolas

March 4, 2022

Contents

1 Preliminaries

1.1 Ballots

An election consists of a finite set of voters, voting on a finite set of issues. For each issue there is a finite set of alternatives. A special alternative is the abstention represented by $*$. Finally, there is an aggregation function that decides what the result of the election is. Formally:

Definition 1.1 (Election)**.** An *election* consists of a tuple $\langle N, D, r\rangle$ where $N = \{1, ..., n\}$ is a finite non-empty set of voters. The set D is a finite set with $|D| \geq 1$. The function $r\colon D^n \longrightarrow D$ is a resolute aggregation function that inputs the votes of every voter and outputs the outcome of the election.

The model we will consider allows each voter to submit a smart ballot. A smart ballot is a preference list of smart votes. Each smart vote is a function with domain the other voters. A special requirement is that the final preference in the preference list is a direct vote on an alternative in D. Formally:

Definition 1.2 (Smart Ballots)**.** A smart ballot of an agent a is an ordering $((S^0, F^0) > \ldots > (S^{k-1}, F^{k-1}) > d)$ where $k \geq 0$. Each S^h for $h \leq k$ is a subset of N and $F^h\colon D^{S^h} \longrightarrow D$ is a resolute non-trivial aggregation function. We also have that $d \in D$.

Further when relevant we will consider F^k to be the constant function with output d. Now, in most cases the sets S^h are implicit and we will drop any mention to them. That is supported by the fact that we will treat two functions F, G as identical if they are extensionally equal. Additionally we will disallow a voter to delegate to themselves. This is formalised by the following definition:

Definition 1.3 (Valid Smart Ballot)**.** A valid smart ballot of an agent a is a smart ballot B_a such that for all $0 \leq s < t \leq k$ F^s is not equivalent to F^t. Additionally $a \notin S_t$.

We collect the n smart ballots into a smart profile $\mathbf{B}$.

1.2 Unravellings

Now that we have defined each agents preferences we need to formalise how to make sense of these preferences. To do so we use *unravelling procedures* to determine each agent's vote.

Definition 1.4 (Unravelling Procedure)**.** An unravelling procedure is any computable function $\mathcal{U}$ where $\mathbf{B} \mapsto^{\mathcal{U}} \mathbf{d}$ with $\mathbf{d} \in D^n$.

Now, every agent would like to know which preference level was used to compute their vote. To do so we introduce the notion of a certificate:

Definition 1.5 (Certificate)**.** A certificate $\mathbf{c} \in \mathbb{N}^n$ for a profile $\mathbf{B}$ is a vector where for each $a \in N$ such that $B_a = (B_a^0 > \ldots > B_a^{k_a})$, the entry $\mathbf{c}_a \in [0, k_a]$ corresponds to the preference level for agent a.

Until now we have not actually used the agent's preferences. So, we need to introduce the concept of a consistent certificate. Consistent certificates are certificates where the vote of each agent is determined by the votes of other agents using the functions in their smart ballots. Formally:

Definition 1.6 (Consistent certificate)**.** For a profile $\mathbf{B}$, a certificate $\mathbf{c}$ is consistent if there is an ordering $\sigma\colon N \longrightarrow N$ of agents which starting from vector $X^0 = \{\Delta\}^n$ with placeholder values Δ for all agents, iteratively constructs an outcome vector of direct votes $X \in D^n$ as follows for $\sigma(a) = z \in [1, n]$:

$$X_a^z = F_a^{\mathbf{c}_a}(X^{z-1} \restriction_{S_a^{\mathbf{c}_a}})$$

where X_a represents agent a's entry in X and $X \restriction_S = \prod_{s \in S} X_s$.

Now what allows the computation of F on a partial input is the concept of a *necessary winner*. A necessary winner occurs when the available input of F is enough to decide the output of F. For example, $Maj(1, 1, a)$ will always resolve to 1 regardless of agent a's vote. So, in this case the necessary winner is 1.

Now, we are only interested in consistent certificates as these are the ones that respect the agents' votes. We will denote the set of consistent certificates of a profile $\mathbf{B}$ as $C(\mathbf{B})$

It is good that there is only one way to unravel consistent certificates. The following proposition is proven in .

Proposition 1.7. If a consistent certificate $\mathbf{c}$ can be given by two orderings σ and σ' of the agents, then the orderings yield the same outcome $X_{\mathbf{c}} \in D^n$.

So, that when we refer to consistent certificates we do not need to specify the votes of each agent. Now that we have settled the framework it's time to consider some "good" certificates. There is a very natural "cost" in this scenario and that is using a lot of the later preferences of each agent. So, we can set two very natural ways of minimising this cost. We can attempt to minimise the sum of the cost or in a more utilitarian setting attempt to minimise the maximum of the cost. Formally:

Definition 1.8 (MinSum)**.** For a given profile $\mathbf{B}$, the MinSum unravelling procedure is defined as:

$$\textsc{MinSum}(\mathbf{B}) = \left\{ X_{\mathbf{c}} \mid \arg \min_{\mathbf{c} \in C(\mathbf{B})} \sum_{i=1}^{n} c_i \right\}$$

Definition 1.9 (MinMax)**.** For a given profile $\mathbf{B}$, the MinMax unravelling procedure is defined as:

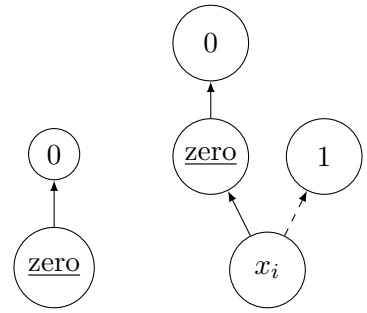
$$\textsc{MinMax}(\mathbf{B}) = \left\{ X_{\mathbf{c}} \mid \arg \min_{\mathbf{c} \in C(\mathbf{B})} \max(\mathbf{c}) \right\}$$

A natural extension of MinMax is that of LexiMin that selects from the MinMax certificates the ones with minimum sum. Formally:

Definition 1.10 (LexiMin)**.** For a given profile $\mathbf{B}$, the LexiMin unravelling procedure s defined as:

$$\textsc{LexiMin}(\mathbf{B}) = \left\{ X_{\mathbf{c}} \mid \arg \min_{\mathbf{c} \in minC(\mathbf{B})} \sum_{i=1}^{n} c_i \right\}$$

where $minC(\mathbf{B})$ is the set of consistent certificates minimise the maximum c_i.



(a) zero voter (b) Voter x_i

Figure 1: Setting up the variables voters

2 Complexity Results

2.1 Function classes strictly larger than LIQUID

Definition 2.1 (BoundedMinSum)**.** Let BoundedMinSum$_{\mathcal{F}}$ be the decision problem with input a target constant M and a smart profile $\mathbf{B}$ which uses functions in the class $\mathcal{F}$. The YES instances are those with a consistent certificate $\mathbf{c}$ with $\sum_i c_i \leq M$.

Definition 2.2 (BoundedMinMax)**.** Let BoundedMinMax$_{\mathcal{F}}$ be the decision problem with input a target constant M and a smart profile $\mathbf{B}$ which uses functions in the class $\mathcal{F}$. The YES instances are those with a consistent certificate $\mathbf{c}$ with $\max(\mathbf{c}) \leq M$.

Definition 2.3 (BoundedLexiMin)**.** Let BoundedLexiMin$_{\mathcal{F}}$ be the decision problem with input target constants M, S and a smart profile $\mathbf{B}$ which uses functions in the class $\mathcal{F}$. The YES instances are those with a consistent certificate $\mathbf{c}$ with $\max(\mathbf{c}) \leq M$ and $\sum_i c_i \leq S$.

Let $\vee$ the binary logical OR and $\wedge$ the binary logical AND. Then:

Proposition 2.4 (Hardness of BoundedMinSum)**.** Suppose $LIQUID \cup \{\vee, \wedge\} \subseteq \mathcal{F}$ and that every voter is allowed only 1 non-constant delegation. Then BoundedMinSum$_{\mathcal{F}}$ is NP-hard.

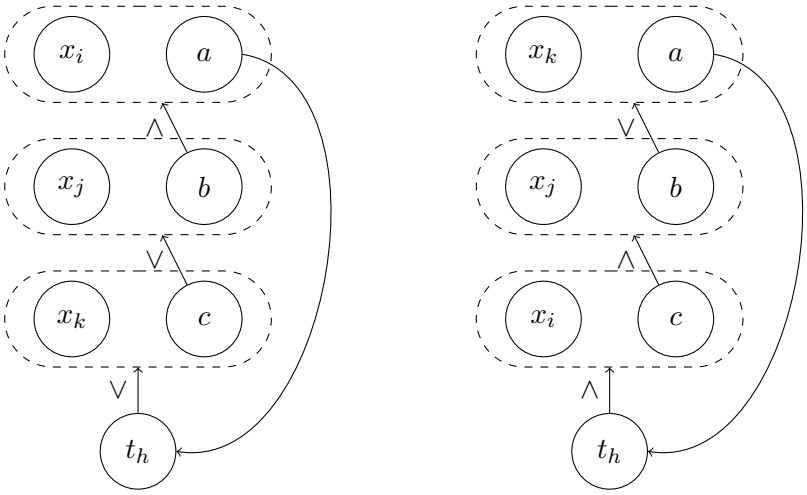
Proof. We reduce from the NP-hard problem of NAE-SAT. An instance of NAE-SAT is an instance of 3-SAT without terms of the forms $x_i \vee x_j \vee x_k$ and $\overline{x_i} \vee \overline{x_j} \vee \overline{x_k}$. Let $\varphi = \bigwedge_{i=1}^{k} t_i$ a NAE-SAT instance on boolean variables $x_1, \ldots, x_n$ with $t_i = l_a \vee l_b \vee l_c$ where l_a, l_b, l_c correspond to literals of variables x_a, x_b, x_c or their negations.

We first define constant voter zero, that always votes for 0. So that $B_{\text{zero}} = (0)$.

For each variable x_i we construct a voter x_i with voting profile $B_{x_i} = (\text{zero} > 1)$. These are drawn in Figure 1 We denote the first preferences as full lines and second preferences as dashed lines.

We then construct gadgets for each term t_h. These gadgets will have the property that they incur no additional cost if t_h is satisfied and a cost of at least one if t_h is satisfied.

Case $\overline{x_i} \vee x_j \vee x_k$: We construct additional voters t_h and a, b and c, with smart profiles:



(a) Case $\overline{x_i} \vee x_j \vee x_k$ (b) Case $\overline{x_i} \vee \overline{x_j} \vee x_k$

Figure 2: Gadgets for MinSum

$$B_a = (t_h > 0)$$
$$B_b = (x_i \wedge a > 0)$$
$$B_c = (x_j \vee b > 0)$$
$$B_{t_h} = (x_k \vee c > 0)$$

This is demonstrated by Figure 2.

Now, we analyse when first preferences of the additional voters can be resolved without producing cycles. If $x_k = 1$ then t_h can resolve to 1 and so will a. As x_i and a are set to a value b can resolve its first preference. As x_j and b are set, c can resolve its first delegation as well. Similarly if $x_j = 1$ then c is immediately resolved to 1 and so t_h is resolved. Hence a and then b can be resolved. Similarly if $x_i = 0$ the logical AND of b is resolved to 0 and all additional voters are resolved.

Now, suppose $x_i = 0, x_j = 0, x_k = 1$. Then t_h cannot immediately resolve the logical OR as it is dependent on the vote of c, similarly c is waiting for b to decide, and b is waiting for a. But a is waiting for t_h to decide. There is no way to resolve this cycle and so one of the additional voters we've introduced will have to vote for their second preference. This will incur an additional cost of at least 1.

Case $\overline{x_i} \vee \overline{x_j} \vee x_k$: The proof is absolutely symmetrical for this case. We need to switch some ANDs to ORs and vice versa but other than that it is identical. For completeness we show the resulting gadget in Figure 2. This is the result of a smart profile of:

$$B_a = (t_h > 0)$$
$$B_b = (x_k \vee a > 0)$$
$$B_c = (x_j \wedge b > 0)$$
$$B_{t_h} = (x_i \wedge c > 0)$$

Now suppose that for each term we construct $n+1$ such gadgets. Then if every term is satisfied by some assignment of the variables the maximum cost will be that of n. If a term is not satisfied then at least $n+1$ gadgets will incur a cost of 1 so that the cost is at least $n+1$. Hence we have reduced the satisfiability of a NAE-SAT instance φ to querying if there is a certificate $\mathbf{c}$ for the above election with $\sum_i c_i \leq n$. Thus BoundedMinSum$_{\mathcal{F}}$ is NP-hard. □

Corollary 2.5 (Inapproximability of BoundedMinSum)**.** A constant factor approximation of BoundedMinSum$_{\mathcal{F}}$ is NP-hard.

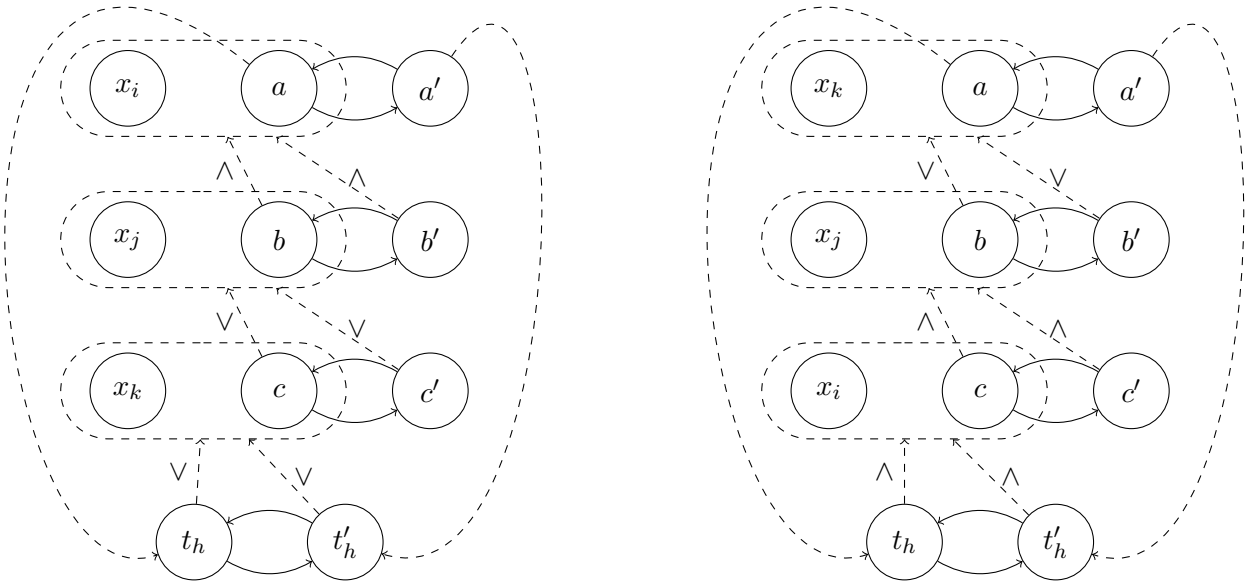
Proof. We can adapt the above proof to prove that BoundedMinSum is not constant-factor approximable. Following our construction above we can incur a cost of k for when the expression φ is not satisfiable by simply creating k gadgets for each term. So for a MinSum outcome $\mathbf{c}$: $\sum_i \mathbf{c}_i \leq n$ if and only if φ is satisfiable and $\sum_i c_i \geq k$ if and only if φ is not satisfiable. So, any constant factor approximator would solve 3-SAT, thus proving hardness. □

Proposition 2.6 (Hardness of BoundedMinMax)**.** Suppose $\mathit{LIQUID} \cup \{\vee, \wedge\} \subseteq \mathcal{F}$. Then BoundedMinMax is NP-hard even if the maximum size of a smart ballot is 3.

Proof. We follow a similar construction as above. We reduce from the NP-hard problem NAE-SAT. Suppose φ is an instance of NAE-SAT as above on k terms and n variables. We again introduce constant voter $\underline{\text{zero}}$ with smart ballot $B_{\underline{\text{zero}}} = (0)$. For each variable x_i we create a voter x_i with smart ballot $B_{x_i} = (\underline{\text{zero}} > 1)$. Now for each term t_h we create the following gadgets depending on the form of t_h.

Case $t_h = \overline{x_i} \vee x_j \vee x_k$: For each term we construct additional voters a, a', b, b', c, c', t_h. With voting profiles:

$$B_a = (a' > t_h > 0)$$
$$B_{a'} = (a > t_h > 0)$$
$$B_b = (b' > x_i \wedge a > 0)$$
$$B_{b'} = (b > x_i \wedge a > 0)$$
$$B_c = (c' > x_j \vee b > 0)$$
$$B_{c'} = (c > x_j \vee b > 0)$$
$$B_{t_h} = (t'_h > c \vee x_k > 0)$$
$$B_{t'_h} = (t_h > c \vee x_k > 0)$$



(a) Case $\overline{x_i} \vee x_j \vee x_k$

(b) Case $\overline{x_i} \vee \overline{x_j} \vee x_k$

Figure 3: Gadgets for MinMax

We present this smart ballot in this much easier to parse figure where filled lines indicate first preferences and dashed lines indicate second preferences:

Now, we claim that if t_h is satisfied then this component can be resolved with at most the second preference being used. If $x_k = 1$ then we can resolved t_h, t'_h to 1 using their second preferences. Hence, we can then resolve agents in order (a, a', b, b', c, c'). Similarly if $x_j = 1$ we can resolve in order $(c, c', t_h, t'_h, a, a, b, b')$ using only the agents first two preferences. Now, if $x_i = 0$ the same argument holds as the logical AND is resolved to 0.

Now consider the case where $x_i = 1, x_j = 0$ and $x_k = 0$. Let $\sigma \colon N \longrightarrow \{1, ..., m\}$ be an ordering of the m agents from which a consistent certificate arises. We can set without loss of generality $\sigma(\underline{\text{zero}}) = 1$ as $\underline{\text{zero}}$ is a constant function. After $\underline{\text{zero}}$ we can place all voters that represent variables as they have no dependencies other than $\underline{\text{zero}}$. So, for each x_i, $\sigma(x_i) = i + 1$. Then, without loss of generality every primed agent appears after his unprimed counterpart so for instance $\sigma(t_h) < \sigma(t'_h)$. Then, $\sigma(t_h) > \sigma(c)$ as $x_k \vee a$ cannot be resolved with $x_k = 0$. Similarly, $\sigma(c) > \sigma(b) > \sigma(a)$ but $\sigma(a) < \sigma(t_h)$. So we the term is not satisfied there is no consistent certificate using only the first

two preferences.

Case $\overline{x_i} \vee \overline{x_j} \vee x_k$: This case is symmetrical to the above. We only need to permute the agents and replace the logical ANDs with ORs and vice versa. For completeness the smart voting profiles are:

$$
\begin{aligned}
B_a &= (a' > t_h > 0) \\
B_{a'} &= (a > t_h > 0) \\
B_b &= (b' > x_k \vee a > 0) \\
B_{b'} &= (b > x_k \vee a > 0) \\
B_c &= (c' > x_j \wedge b > 0) \\
B_{c'} &= (c > x_j \wedge b > 0) \\
B_{t_h} &= (t'_h > c \wedge x_i > 0) \\
B_{t'_h} &= (t_h > c \wedge x_i > 0)
\end{aligned}
$$

So, if and only if φ is satisfiable then there exists a consistent certificate $\mathbf{c}$ with $\max(\mathbf{c}) \leq 2$. So we have reduced NAE-SAT to an instance of BoundedMinMax$_{\mathcal{F}}$, proving that BoundedMinMax$_{\mathcal{F}}$ is NP-hard. □

Proposition 2.7 (BoundedMinMax is not approximable)**.** If $\mathit{LIQUID} \cup \{\vee, \wedge\} \subseteq \mathcal{F}$ then a non-trivial approximation of BoundedMinMax$_{\mathcal{F}}$ is NP-hard.

Proof. This result can be proven by modifying the above proof. Instead of constructing a cycle of size 2 for voters a, b, c, t_h we construct a cycle of size $k+1$. So every voter a, b, c, t_h can be thought of as a voter v with $B_v = (v' > u > 0)$. Then we construct voters $v_1, \ldots, v_k$ and set $B_v = (v_1 > v_2 > \ldots > v_k > u > 0)$ and for v_i, $B_{v_i} = (v > v_1 > \ldots > v_{i-1} > v_{i+1} > \ldots > v_k > u > 0)$. Now, since the maximum size of the smart ballots is $k+2$ then clearly for the MinMax certificate $\mathbf{c}$, $\max \mathbf{c} \leq k+2$. But suppose that an algorithm was able to decide if there was $\mathbf{c}$ with $\max \mathbf{c} \leq k+1$. Then this algorithm would be able to determine if the original instance is satisfied. So, it would solve NAE-SAT. □

Corollary 2.8 (Hardness of BoundedLexiMin)**.** If $\mathit{LIQUID} \cup \{\vee, \wedge\} \subseteq \mathcal{F}$ then BoundedLexiMin$\mathcal{F}$ is NP-hard.

Proof. Every instance of BoundedMinMax is an instance of BoundedLexiMin. For target maximum M and smart profile $\mathbf{B}$ a MinMax query, we can consider a LexiMin query with target maximum M, target sum S and smart profile $\mathbf{B}$. Simply set S to be the sum of all preference levels, i.e. $S = \sum_{a \in N} k_a$. Then the sum requirement of LexiMin is trivially satisfied and so we have reduced BoundedMinMax to BoundedLexiMin. □

2.2 For function classes equal to LIQUID

Proposition 2.9. Suppose that r is monotone and $D = \{0, 1\}$. Then for any outcome $d \in D$ there is an algorithm that determines if there is a MinMax certificate such that the election resolves in favour of d.

Proof. Our algorithm iteratively constructs a graph and tests if there is a path from each voter to an outcome. □

3 Axiomatisation

We introduce the axiom of cast participation. That is that voters that have a preference for an outcome benefit from voting directly for it. We denote that agent a prefers outcome d to e as $d >_a e$. We also infer that if agent a votes directly for outcome d then they weakly prefer d over any other outcome. Formally if $B_a = (d)$ then for all $d' \in D$, $d \geq_a d'$. Now, an issue is that irresolute unravelling procedures might return multiple outcomes. So MinSum can return outcomes for 0 and for 1. So, we need to consider the set of outcomes from irresolute unravelling procedures. The simplest ordering of these sets of outcomes is that a voter a that votes for d prefers every set of outcomes that includes d to every set of outcomes that doesn't. Further, $\{d\}$ is the preferred outcome of voter a. We assume that voters are indifferent between other sets of outcomes.

Then we can formalise cast-participation for irresolute procedures:

Definition 3.1 (Cast-Participation for irresolute procedures)**.** A resolute voting rule r and a irresolute unravelling procedure $\mathcal{U}$ satisfy cast-participation if for all valid smart profiles $\mathbf{B}$ and agents $a \in N$ such that $B_a \in D \setminus \{*\}$ we have that for all $B'_a \neq B_a$:

$$r(\mathcal{U}(\mathbf{B})) \geq_a r(\mathcal{U}(\mathbf{B}_{-a}, B'_a))$$

Where $\mathbf{B}_{-a}, B'_a$ denotes replacing B_a with B'_a.

Cast participation can be thought of disallowing tactical voting. Unfortunatelly it does not always hold. We first prove the cases of non-monotone r and non-monotone delegating functions. We prove these results for all unravelling procedures that would respect the voters first preference if the preferences didn't introduce cycles.

We now prove a series of negative results for cast participation.

Lemma 3.2. Suppose r is not monotone and $\mathcal{U}$ is any unravelling procedure, then cast participation doesn't hold.

Proof. Suppose r is not monotone. Then for some vector $\mathbf{u} \in D^n$ with $\mathbf{u}_i = 0$ there exists a $\mathbf{u}' = (\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, 1, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_n)$ with $r(\mathbf{u}) = 1$ and $r(\mathbf{u}') = 0$. This violates cast participation by the following profile: let $B_a = (\mathbf{u}_a)$, so in particular voter i prefers 0 to 1. Then voter i strictly prefers to vote for 1 than 0. So, to achieve cast participation, r must be monotone. □

Lemma 3.3. Suppose r is a monotone rule, $\mathcal{U}$ respects first preferences and $f \in \mathcal{F}$ is a function on $D^k \longrightarrow D$ that is not monotone. Now suppose additionally that for some n there exists a "deciding" subset of voters S with $|S| \leq n - k$ such that if every voter in S votes for 0, r votes for 0 and if every voter in S votes for 1, r votes for 1. Then cast-participation doesn't hold.

Proof. Let S and f as defined above. Suppose $\mathbf{u} < \mathbf{u}'$ with only $\mathbf{u}_i \neq \mathbf{u}_i$ and $f(\mathbf{u}) > f(\mathbf{u}')$. Then, enumerate voters $v_1, \ldots, v_k$ not in S and let $B_{v_i} = (\mathbf{u}_i)$. For all other agents a set $B_a = (f(v_1, \ldots, v_k) > d)$ for some arbitrary d. Then the first preferences of the voters do not introduce cycles. So r resolves to vote for 1 as all voters in S vote for 1. Now if v_i were to switch their vote from 0 to 1 everyone in S would vote for 0 and so r would resolve to 0. This breaks cast participation for voter v_i. □

Lemma 3.4. Let r be a rule such that for $n \geq 5$ voters if $n-2$ voters vote for d then r assigns d. Let $\mathcal{F}$ contain LIQUID. Then there are examples that unravelling with MinMax violates cast-participation for any n.

Proof. Let $N = \{v, v', a, u_1, \ldots, u_{n-3}\}$ and r as described. Then, let $B_v = (0), B_{v'} = (v > 0), B_a = (1), B_{u_i} = (a > 0)$. Then MinMax would simply assign each individual to first preferences and the majority votes for 1. So the outcome set is $\{1\}$.

But, if $B_v = (v' > 0)$ then a cycle is formed and so necessarily, MinMax will have to use some second preferences. Hence setting $v, u_1, \ldots, u_{n-3}$ to their second preference would be a valid solution. So every voter except a vote for 0, so that r resolves to 0. Note that we can still assign first preference to all voters except than v so that $(2, 1, \ldots, 1)$ is a consistent certificate and would result to the majority again voting for 1. So the set of outcomes is $\{0, 1\}$ which is better than $\{1\}$ for agent a. □

We can further prove that MinSum doesn't hold when using three symbols. To do this set $D = \{0, 1, *\}$ where $*$ denote abstentions. In the case where abstentions are allowed we denote $Maj(0, *) = 0, Maj(1, *) = 1$ and $Maj(0, 1) = *$.

Lemma 3.5. Let $D = \{0, 1, *\}$ and r such that if a strict majority votes for outcome d then r supports outcome d. Let voters that directly vote for 0 have preference $0 > * > 1$. Then if voters are allowed to delegate to even majorities, cast-participation doesn't hold for MinSum.

Proof. Let $N = \{a, b, c, d, e\}$. Let $B_a = (0), B_b = (1), B_c = (Maj(a, b) > 0), B_d = B_e = (c > 0)$. Then we can resolve this smart profile by assigning everyone to first preferences. Then r delegates to $*$. So the set of outcomes is $\{*\}$.

Now if $B'_a = (c > d > 0)$ then the unique result of MinSum is c votes for 0. Further, a, d, e delegate to c using first preferences and vote for 0 as well. This results in a strict majority for 0. So the set of outcomes is $\{0\}$ which is a preferred set of outcomes. □

Note, that in the above case Maj is a monotone rule. So, it is not the case that monotone rules guarantee cast participation.

4 Proposals