An Analysis of Smart Voting in Liquid Democracy

Giannis Tyrovolas

March 24, 2022

Contents

1 Preliminaries

1.1 Ballots

A single-issue election consists of a finite set of voters that vote on a single issue. Each voter is presented with a finite set of alternatives to pick from. A special alternative is the *abstention* represented by $*$. Finally, there is an aggregation function that decides the result of the election. Formally:

Definition 1.1 (Single-issue election)**.** A *single-issue 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.

Throughout this work we will focus on single issues with a binary set of outcomes. That is because all of our hardness results hold for binary issues and we can easily extend them to n-ary issues. Therefore, unless otherwise stated we will consider $D = \{0, 1\}$.

The model we will consider generalises the above and allows each voter to submit a smart ballot. A smart ballot is a preference list of smart votes. Each smart vote is a function whose domain is a subset of N. 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-constant 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 of 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 for all $0 \leq t \leq k$, $a \notin S_t$.

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

Throughout this work it will be meaningful to restrict the functions agents can delegate from. The most notable classes of functions we will consider are:

- Direct delegations to voter v denoted by $LIQUID = \{id_v \mid v \in N\}$, where id_v is the identity function applied to the vote of voter v.
- Boolean functions in conjunctive normal form denoted by *BOOL*.
- Monotone boolean function in conjunctive normal form denoted by *MON-BOOL*.

1.2 Unravellings

Now that we have defined each agent's 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 $\mathbf{d} \in D^n$.

The reason we need to consider unravelling procedures is that smart ballots can create cycles. Different choices on how to "unravel" cycles will produce different outcomes. For the majority of this work we will focus on the mathematical and computational properties of different unravelling procedures.

Now as each outcome is calculated it is important for agents 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 $c_a \in [0, k_a]$ corresponds to the preference level for agent a.

Something that will be of high importance is that functions of interest can be calculated on partial input. For example, consider majority rule denote as Maj. For a variable a, $Maj(1, 1, a)$ will always resolve to 1 regardless of agent a's vote. We refer to these cases as *necessary winners* as in Konczak's and Lang's work [2]. We formalise this in the following definition where we use Δ to denote placeholder values.

Definition 1.6 (Necessary winner extensions)**.** Let $F\colon D^n \longrightarrow D$ be a function. We define $F'\colon \{D \cup \{\Delta\}\}^n \longrightarrow D \cup \{\Delta\}$ function F's *necessary winner extension*. We set for all $\mathbf{d} \in D^n$, $F'(\mathbf{d}) = F(\mathbf{d})$. Now for $\mathbf{d} \in \{D \cup \{\Delta\}\}^n$, let $i_1, i_2, \ldots, i_m$ be the indices such that $d_{i_j} = \Delta$. Let $\mathbf{d}[x_1, \ldots, x_m]$ denote the vector $\mathbf{d}$ where x_j replaces the value at index i_j. Then, if for some $y \in D$ and for all $\mathbf{x} \in D^m$, $F(\mathbf{d}[\mathbf{x}]) = y$ then $F'(\mathbf{d}) = y$. Otherwise $F'(\mathbf{d}) = \Delta$.

To actually use our agents' preferences 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 1.7 (Consistent certificate)**.** For a profile $\mathbf{B}$, a certificate $\mathbf{c}$ is *consistent* if there is an ordering $\sigma\colon N \longrightarrow \{1, \ldots, n\}$ of agents starting from vector $\mathbf{X}^0 = \{\Delta\}^n$ with placeholder values Δ for all agents, iteratively constructs an outcome vector of direct votes $\mathbf{X} \in D^n$ as follows for $\sigma(a) = z \in [1, n]$. For ease of notation we abbreviate agent a's c_ath function, $F_a^{c_a}$ to F.

$$X_a^z = F'(\mathbf{X}^{z-1} \restriction_{S_a^{c_a}}).$$

Here, X_a represents agent a's entry in $\mathbf{X}$. The restriction of vector $\mathbf{X}$ in S, written as $\mathbf{X} \restriction_S$, is a vector indexed by S where for each $s \in S$, $(\mathbf{X} \restriction_S)_s = X_s$.

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 outcome matched to each consistent certificate regardless of the ordering σ. The following proposition is proven by Colley et al. [1].

Proposition 1.8. 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 $X_{\mathbf{c}} \in D^n$.

Consequently, when we refer to consistent certificates are enough to determine every agent's vote. 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 costs or in a more egalitarian spirit attempt to minimise the maximum cost. Formally:

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

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

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

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

The above unravelling procedures have already been introduced and studied by Colley et al. [1]. A natural extension of MinMax, not studied is that of MinMaxSum that selects from the MinMax certificates the ones with minimum sum. Formally:

Definition 1.11 (MinMaxSum)**.** For a given profile $\mathbf{B}$, the MinMaxSum unravelling procedure s defined as:

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

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

2 Complexity Results

2.1 Function classes strictly larger than LIQUID

In order to study the computational properties of the unravelling procedures we set them up as decision problems.

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 (BoundedMinMaxSum)**.** Let BoundedMinMaxSum$_{\mathcal{F}}$ 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 $\sum_i c_i \leq S$.

Now, BoundedMinSum and BoundedMinMax have been studied by Colley et al [1]. They proved that BoundedMinSum$_{LIQUID}$ and BoundedMinMax$_{LIQUID}$ are poly-time computable and that BoundedMinSum$_{MON\text{-}BOOL}$ and BoundedMinMax$_{BOOL}$ are NP-complete. We improve on these hardness results and conclude some inapproximability results.

Before delving any further we can consider the trivially easy cases. For any function class $\mathcal{F}$, BoundedMinSum$_{\mathcal{F}}$ and BoundedMinMax$_{\mathcal{F}}$ are trivially solvable if the maximum size of the ballot is 1.

Now, let $\vee$ be the binary logical OR and $\wedge$ be the binary logical AND. Then:

Proposition 2.4 (Hardness of BoundedMinSum)**.** Suppose $LIQUID \cup \{\vee, \wedge\} \subseteq \mathcal{F}$. Then BoundedMinSum$_{\mathcal{F}}$ is NP-hard, even if the maximum size of a smart ballot is 2.

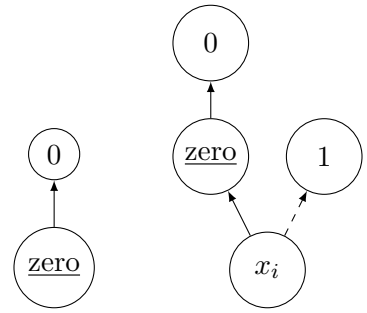
Proof. We reduce from the NP-hard problem of 3-SAT. Let $\varphi = \bigwedge_{i=1}^{k} t_i$ a 3-SAT instance on boolean variables $x_1, \ldots, x_n$ with $t_i = l_a \vee l_b \vee l_c$ where l_a, l_b, l_c correspond to literals of variables x_a, x_b, x_c or their negations.

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

For each variable x_i we construct a voter x_i with voting profile $B_{x_i} = (\underline{\text{zero}} > 1)$. These are drawn in Figure 1. We denote the first preferences as solid 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 not satisfied. We prove this for the four different structures of a term t_h.

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



(a) zero voter (b) Voter x_i

Figure 1: Setting up the variables voters

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

Now, we analyse when first preferences of the additional voters can be resolved without producing cycles. If $x_i = 1$ then $x_i \vee a = 1$ and b resolves to vote 1 using the first preference. Hence, $b \vee x_j = 1$ and c resolves to 1 using first preference. Similarly, t_h resolves to 1 and then a resolves to 1 using only first preferences. Similarly, if $x_j = 1$ or $x_k = 1$, agents c and t_h respectively will resolve to 1 and so all fresh agents will resolve their votes using only first preferences.

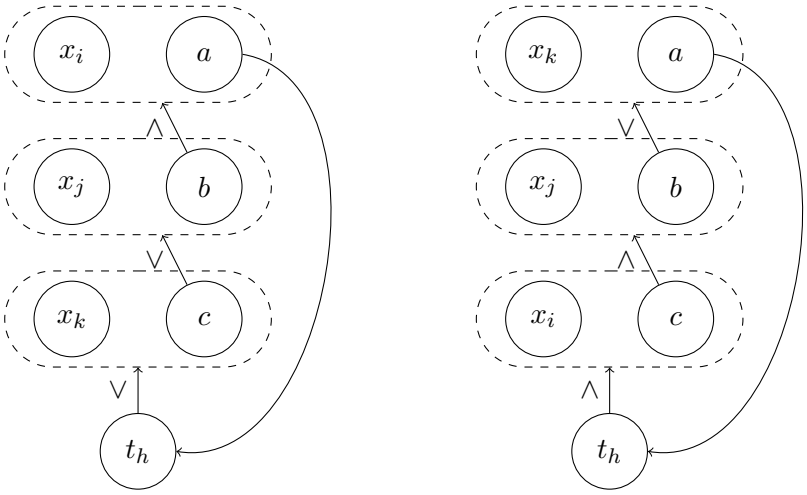
Now, suppose $x_i = x_j = x_k = 0$. Suppose there is an ordering $\sigma \colon N \longrightarrow \{1, \ldots, m\}$ that gives rise to a consistent certificate using only the first preferences of agents a, b, c, t_h. Then, $\sigma(t_h) < \sigma(a)$ as a needs t_h to be resolved. Further, $\sigma(c) < \sigma(t_h)$ as $x_k = 0$ and so c determines the clause $x_k \vee c$. Similarly, $\sigma(b) < \sigma(c)$ and $\sigma(a) < \sigma(b)$. This leads to the contradiction that $\sigma(a) < \sigma(a)$. Therefore, no such ordering exists and if $x_i = x_j = x_k = 0$, one of the fresh voters a, b, c, t_h will need to incur a cost of at least 1.

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

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

This is demonstrated by Figure 2a.

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



(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

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 $x_i = 0$ $x_i \wedge a = 0$ and so b resolves to 0 and all additional voters are resolved.

Now, suppose $x_i = 1, x_j = 0, x_k = 0$. 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 have 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 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 2b. This is the result of a smart profile of:

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

Case $\overline{x_i} \vee \overline{x_j} \vee \overline{x_k}$: The proof is symmetrical to the case of $x_i \vee x_j \vee x_k$. We only need to switch $\vee$ to $\wedge$. This is achieved by the following smart ballot:

$$
\begin{aligned}
B_a &= (t_h > 0) \\
B_b &= (x_i \wedge a > 0) \\
B_c &= (x_j \wedge b > 0) \\
B_{t_h} &= (x_k \wedge c > 0).
\end{aligned}
$$

Now suppose that for each term we construct $n+1$ such gadgets for each term. Then if every term is satisfied by some assignment of the variables $x_1, x_2, \ldots, x_n$ then the total cost incurred will be at most n. That is because the fresh voters incur no additional cost but each voter x_i can incur a cost of at most 1. If a term is not satisfied then at least $n+1$ gadgets will incur a cost of at least 1 so that the cost is at least $n+1$. Hence we have reduced the satisfiability of any 3-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 $\mathbf{c}$: $\sum_i c_i \leq n$ 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. □

Now we shift our focus on BoundedMinMax. We first note that there is an additional easiness lemma for BoundedMinMax.

Proposition 2.6. Let $\mathcal{F}$ be any computable family of functions and $\mathbf{B}$ a smart profile where each ballot has size at most 2. Suppose for each function $f \in \mathcal{F}$, f' can be computed instantly. Then, there is an algorithm that decides BoundedMinMax$_{\mathcal{F}}$ which runs in $\mathcal{O}(n^2)$ steps.

Proof. Let m_i the size of the ballot of agent i, as always $\mathbf{c} = (m_1, \ldots, m_n)$ is a consistent certificate, as each delegate votes for a constant. As each $m_i \leq 2$, the only possible certificate that improves on $\mathbf{c}$ is $\mathbf{1} = (1, \ldots, 1)$. To check if $\mathbf{1}$ is consistent, we first set all agents a with F_a^1 constant and equal to d to d. Then, we check iteratively if any of the unset agents can vote using their first preference. We repeat this step until running this iteration makes no changes. We return that $\mathbf{1}$ is a consistent certificate if and only if all agents are set at this final point.

We now prove correctness of the above. If the algorithm returns that $\mathbf{1}$ is a consistent certificate then it truly is so. It can be proved by constructing $\sigma \colon N \longrightarrow \{1, \ldots, n\}$ where $\sigma(a)$ is the position in which a was activated by our algorithm. By our definition, a then uses only the values of agents that were set before a in this ordering. Now consider every case where $\mathbf{1}$ is a consistent certificate with a corresponding ordering σ. Then, without loss of generality all the agents that vote for constants are placed first in positions $1, \ldots, k$. Then, for $\sigma(a) = k+1$, F_a^1 can be calculated using only agents v with $\sigma(v) \leq k$, hence our algorithm will set a to vote for F_a^1. Inductively, it will reconstruct an equivalent ordering to σ.

□

Now that we have the easiness result the following hardness result will be tight. Our construction is similar to the one for BoundedMinSum. Instead of using multiple gadgets to amplify the effect of cycles we create some additional cycles by introducing primed voters a', b', c' and t'_h.

Proposition 2.7 (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 3-SAT. Suppose φ is an instance of 3-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 $B_{x_i} = (\underline{\text{zero}} > 1)$. For each term t_h we create gadgets with the property that if t_h is satisfied the gadget uses only the first two preference levels. If t_h is not satisfied, the gadget uses the third preference level. We prove this for the following four cases.

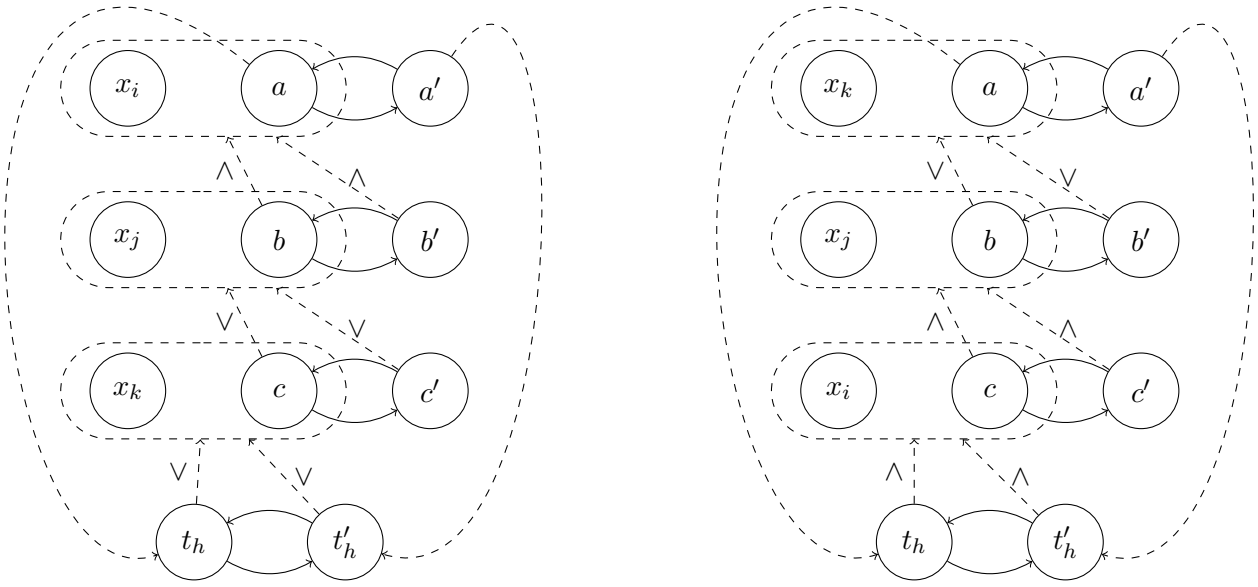
Case $t_h = x_i \vee x_j \vee x_k$: For each term we construct additional voters a, a', b, b', c, c', t_h. With voting profiles:

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

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 resolve t_h, t'_h to 1 using their second preferences. Hence, we can then resolve agents in order (a, a', b, b', c, c') using the agents' second preference. 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. Similarly, for $x_i = 1$.

Now consider the case where $x_i = x_j = x_k = 0$. Let $\sigma \colon N \longrightarrow \{1, \ldots, m\}$ be an ordering of the m agents from which a consistent certificate arises. Suppose, this only uses the agents' first two preferences. As the fresh agents a, b, c, t_h are identical with the primed versions a', b', c', t'_h, without loss of generality every unprimed agent appears before their primed counterpart so for instance $\sigma(t_h) < \sigma(t'_h)$. Then, $\sigma(c) < \sigma(t_h)$ as $x_k \vee c$ cannot be resolved with $x_k = 0$. Similarly, $\sigma(b) < \sigma(c)$ and $\sigma(a) < \sigma(b)$ but $\sigma(t_h) < \sigma(a)$. This leads to the contradiction $\sigma(t_h) < \sigma(t_h)$. Hence, if t_h is not satisfied, one of the fresh agents will use their third preferences, and if t_h is satisfied, all fresh agents will use only their first two preferences.

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



(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

voting profiles:

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

We present this smart ballot in the much easier to parse Figure 3a where solid 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 resolve 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 $\sigma \colon N \longrightarrow \{1, \ldots, m\}$ be an ordering of the m agents from which a consistent certificate arises. Suppose, this only uses the agents' first two preferences. As the fresh agents a, b, c, t_h are identical with the primed versions a', b', c', t'_h, without loss of generality every unprimed agent appears before their primed counterpart, so for instance $\sigma(t_h) < \sigma(t'_h)$. Then, $\sigma(c) < \sigma(t_h)$ as $x_k \vee c$ cannot be resolved with $x_k = 0$. Similarly, $\sigma(b) < \sigma(c)$. Again, $\sigma(a) < \sigma(b)$ as $x_i \wedge a = a$ as $x_i = 1$. But $\sigma(t_h) < \sigma(a)$. This leads to the contradiction $\sigma(t_h) < \sigma(t_h)$. Hence, if t_h is not satisfied one of the fresh agents will use their third preferences and if t_h is satisfied all fresh agents will use only their 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:

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

Case $\overline{x_i} \vee \overline{x_j} \vee \overline{x_k}$: The proof is symmetrical to the case $x_i \vee x_j \vee x_k$. The smart ballot used reverses $\vee$ to $\wedge$. This is accomplished by the following smart profile:

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

So, φ is satisfiable if and only if there exists a consistent certificate $\mathbf{c}$ with $\max(\mathbf{c}) \leq 2$. Hence, we have reduced 3-SAT to BoundedMinMax$_{\mathcal{F}}$, proving that BoundedMinMax$_{\mathcal{F}}$ is NP-hard. □

Proposition 2.8 (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 3-SAT. □

Corollary 2.9 (Hardness of BoundedMinMaxSum)**.** If $\mathit{LIQUID} \cup \{\vee, \wedge\} \subseteq \mathcal{F}$ then BoundedMinMaxSum$_{\mathcal{F}}$ is NP-hard.

Proof. Every instance of BoundedMinMax is an instance of BoundedMinMaxSum. For target maximum M and smart profile $\mathbf{B}$ a MinMax query, we can consider a MinMaxSum 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 MinMaxSum is trivially satisfied and so we have reduced BoundedMinMax to BoundedMinMaxSum. □

2.2 For function classes equal to LIQUID

Proposition 2.10. 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.

3 Axiomatisation

We introduce the axiom of cast participation. This axiom states that voters preferring outcome d benefit from voting directly for d. This axiom is introduced by Kotsialou and Riley in [3] for binary issues and by Colley et al. in [1] for binary issues with abstentions. We use the definition for binary issues with abstentions i.e. $D = \{0, 1, *\}$. We denote that agent a prefers outcome d to e as $d >_a e$. As in Colley et al. we infer that if agent a votes directly for outcome $d \in \{0, 1\}$ then agent a prefers d over any other outcome. Formally if $B_a = (d)$ then $d >_a 1 - d$ and $d >_a *$. Additionally, we consider that if agent a votes directly for d then $* >_a 1 - d$. Now, an issue we face is that irresolute unravelling procedures might return multiple outcomes. For example, MinSum can return outcomes for 0 and for 1. So, we need to assign preferences over sets of outcomes rather than just for single outcomes. For the case $D = \{0, 1, *\}$ we posit that reasonable extensions of the order $1 - d <_a * <_a d$ would include the partial order:

$$\{1-d\} < \{1-d, *\} < \{*\} \sim \{1-d, d\} \sim \{1-d, *, d\} < \{*, d\} < \{d\}.$$

Here $S_1 \sim S_2$ denotes that S_1 is not comparable to S_2. 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 $B'_a \neq B_a$:

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

Here $\mathbf{B}_{-a}, B'_a$ denotes replacing B_a with B'_a and r is applied to subsets of $S \subseteq D^n$ as follows: $r(S) = \{r(\mathbf{d}) \mid \mathbf{d} \in S\}$.

Cast participation can be thought of as disallowing tactical voting. Unfortunately, it is not always satisfied. We first consider the cases of some non-monotone r and non-monotone delegating functions. To do so we first define monotonicity as follows. Consider $D = \{0, 1, *\}$ to be ordered totally as $0 < * < 1$. Then, we can define a partial order on D^n where $\mathbf{u} \leq \mathbf{v}$ if for all i, $u_i \leq v_i$. Additionally, $\mathbf{u} < \mathbf{v}$ if $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$.

Definition 3.2 (Monotonicity)**.** Function $f\colon D^n \longrightarrow D$ is *monotone* if for all $\mathbf{u}, \mathbf{v} \in D^n$, $\mathbf{u} \leq \mathbf{v}$ implies that $f(\mathbf{u}) \leq f(\mathbf{v})$.

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

Lemma 3.3. Suppose that the voting rule r is not monotone. Additionally the counterexample to monotonicity is not caused by flipping a $*$ to a 0 or 1. That is there exists $\mathbf{u} = (u_1, \ldots, u_{i-1}, 0, u_{i+1}, \ldots, u_n)$ and $\mathbf{u}' = (u_1, \ldots, u_{i-1}, e, u_{i+1}, \ldots, u_n)$ with $0 \neq e$ but $r(\mathbf{u}) > r(\mathbf{u}')$. Then cast participation does not hold.

Proof. Consider the smart profile where for each agent a, $B_a = (u_a)$. In particular $0 >_i * >_i 1$. We are allowed to deduce this because i does not vote for an abstention. But, by setting a smart ballot of $B_i = (e)$ we obtain $r(\mathbf{u}') < r(\mathbf{u}')$. So voter i prefers to vote directly for e than d despite preferring outcome d. This violates cast-participation. □

We consider that reasonable unravelling procedures will not needlessly violate their agents' first preferences. We formally say that $\mathcal{U}$ respects first preferences if whenever $\mathbf{c} = (1, 1, \ldots, 1)$ is a consistent certificate, $\mathcal{U}$ returns an outcome with certificate $\mathbf{c}$. Note that MinMax, MinSum and MinMaxSum respect first preferences as well as any Pareto optimal unravelling procedure.

Lemma 3.4. Suppose r is a monotone rule, $\mathcal{U}$ respects first preferences. Suppose additionally, that for some n and $k > 0$ there exists a "deciding" subset of voters S with $|S| \leq n - k$ such that if every voter in S votes for d, r votes for d. Then, cast-participation does not hold.

Proof. Let S as defined above and $f : \{0, 1, *\}^k \longrightarrow \{0, 1, *\}$. Further, suppose there exist $\mathbf{u} < \mathbf{u}'$ with only $u_i < u_i'$, $u_i = 0$ and $f(\mathbf{u}) > f(\mathbf{u}')$. Then, enumerate voters $v_1, \ldots, v_k$ not in S and let $B_{v_i} = (u_i)$. 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. As $\mathcal{U}$ respects first preferences it picks an outcome where every agent votes for their first preference. Hence, r resolves to vote for $f(\mathbf{u})$ as all voters in S vote for $f(\mathbf{u})$. Now if agent v_i were to switch their vote from 0 to u_i' everyone in S would vote for $f(\mathbf{u}')$ and so r would resolve to $f(\mathbf{u}')$. This breaks cast participation for voter v_i as they prefer outcome 0 but are better not voting for 0. □

Lemma 3.5. Let r be a rule such that for $n \geq 5$ voters if $n - 2$ voters vote for d then r assigns d. Then there are examples that unravelling with MinMax violates cast-participation for any n. This result holds even if we only allow agents to vote using LIQUID.

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. Therefore, the set of outcomes is $\{0, 1\}$ which is better than $\{1\}$ for agent a. □

Do note that the above counterexample works even when consider issues with binary outcomes. Additionally, the above proof would not work in the case of MinMaxSum, as MinMaxSum would only return the certificate $(2, 1, \ldots, 1)$ We can further prove that MinSum does not satisfy cast-participation. In the case where abstentions are allowed we denote $Maj(0, *) = 0, Maj(1, *) = 1$ and $Maj(0, 1) = *$.

Lemma 3.6. Let r be such that if a strict majority votes for outcome d then r supports outcome d. 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 $B_a = (1), B_b = (0), B_c = (Maj(a, b) > 1), B_d = B_e = (c > 1)$. So that agent a prefers outcome 1. Then we can resolve this smart profile by assigning everyone to first preferences. Then r resolves to $*$. So the set of outcomes is $\{*\}$.

Now if $B'_a = (c > d > 1)$ then the unique result of MinSum is c votes for 1. Further, a, d, e delegate to c using first preferences and vote for 1 as well. This results in a strict majority for 1. So the set of outcomes is $\{1\}$ which is preferred to $\{*\}$. □

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

References

[1] Rachael Colley, Umberto Grandi, and Arianna Novaro. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–35, 2022.

[2] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20. Citeseer, 2005.

[3] Grammateia Kotsialou and Luke Riley. Incentivising participation in liquid democracy with breadth-first delegation. *arXiv preprint arXiv:1811.03710*, 2018.