Unravelling Expressive Delegations: Complexity and Normative Analysis

Giannis Tyrovolas[1], **Andrei Constantinescu**[2], **Edith Elkind**[3]

[1]Independent
[2]ETH Zurich
[3]University of Oxford
johntyro@hotmail.com, aconstantine@ethz.ch, elkind@cs.ox.ac.uk

Abstract

We consider binary group decision-making under a rich model of liquid democracy: agents submit ranked delegation options, where each option may be a function of multiple agents' votes; e.g., "I vote yes if a majority of my friends vote yes." Such ballots are unravelled into a profile of direct votes by selecting one entry from each ballot so as not to introduce cyclic dependencies. We study delegation via monotonic Boolean functions, and two unravelling procedures: MINSUM, which minimises the sum of the ranks of the chosen entries, and its egalitarian counterpart, MINMAX. We provide complete computational dichotomies: MINSUM is hard to compute (and approximate) as soon as any non-trivial functions are permitted, and polynomial otherwise; for MINMAX the easiness results extend to arbitrary-arity logical ORs and ANDs taken in isolation, but not beyond. For the classic model of delegating to individual agents, we give asymptotically near-tight algorithms for carrying out the two procedures, and efficient algorithms for finding optimal unravellings with the highest vote count for a given alternative. These algorithms inspire novel tie-breaking rules for the setup of voting to change a status quo. We then introduce a new axiom, which can be viewed as a variant of the participation axiom, and use algorithmic techniques developed earlier in the paper to show that it is satisfied by MINSUM and a lexicographic refinement of MINMAX (but not MINMAX itself).

Introduction

Liquid democracy, also referred to as delegative democracy, is a decision-making mechanism that allows the voters more flexibility than representative democracy: While every voter may vote directly on an issue, voters may also delegate their vote to other voters. Crucially, delegations are *transitive*: if Alice delegates to Bob, and Bob delegates to Claire, then Claire votes with the combined power of all three.

Variants of this idea have been explored by many authors; see, e.g., the survey by Behrens (2017). In particular, Ford (2002) argues that large-scale direct democracy is infeasible and likely undesirable, whereas representative democracy is rather rigid: it requires holding elections every so often, with winners representing their entire constituency and losers gaining no representative power. Further, there is a hard limit on the total number of elected representatives, and every voter can pick from a limited number of candidates. Delegative democracy can then be seen as a balance of the two, by challenging the premise that the number of representatives needs to be kept small. In this model, voters can vote directly on issues if they wish to do so. Passive voters can delegate to representatives called *delegates*. Delegates are not chosen at fixed time points, but need to canvass voters continually, and acquire the combined power of all voters that delegate to them. They can vote on issues directly, or delegate to other delegates. Importantly, delegates need not win competitive elections, and their votes are public for the sake of accountability. Ford introduces several strengthenings of liquid democracy, such as voters delegating fractions of their votes to different delegates. A particularly interesting variant is to allow agents to submit "multiple delegation choices in order of preference." This is done partly to deal with the case of cycles, where Alice delegates to Bob, but also Bob delegates to Alice.

Colley, Grandi, and Novaro (2022) further extend this model, by allowing each delegation choice to be an arbitrary function of other agents' selections. That is, instead of delegating to Bob, Alice can declare that she supports option X if at least one of Bob, Charlie and Diana supports it (for a formal treatment of this model, see the Preliminaries). An agent's ballot is then a ranked list of such functions, in order of preference. This enhancement of the basic model means that cycles can be introduced in much more complicated ways.

Colley et al. propose two "unravelling" procedures to resolve these cycles. MINSUM is a utilitarian method that minimises the sum of preference levels used, and MINMAX is an egalitarian method that minimises the maximum of the preference levels used. They prove that MINSUM is NP-hard if agents are allowed to use arbitrary-arity logical ANDs and MINMAX is NP-hard for arbitrary Boolean functions. Also, they give polynomial-time algorithms to unravel instances where agents can only delegate directly to other agents.

Our Contribution

We focus on binary issues and strengthen the complexity results of Colley et al. to complete computational dichotomies: MINSUM is hard to compute (and approximate) as soon as any non-trivial functions are permitted, and polynomial otherwise; for MINMAX the easiness results extend

to arbitrary-arity logical ORs and ANDs taken in isolation, but not beyond. For the standard model of delegation to individual agents, we design algorithms that are faster than those of Colley et al. (indeed, nearly optimal). Then, we consider the problem of determining if there exists an unravelling where a particular alternative is selected, giving efficient algorithms in both cases. For MINSUM, we do this by leveraging Fulkerson's primal-dual algorithm for computing minimum-cost arborescences (Fulkerson 1974). Our results suggest novel tie-breaking rules for both unravelling procedures, which are particularly appealing when one of the two input alternatives is seen as a status quo. Interestingly, Fulkerson's ideas also turn out to be useful from a normative perspective: we put forward a new axiom, which can be seen as an adaptation of the classic participation axiom to the setting of liquid democracy, and use our algorithmic techniques to show that it is satisfied by MINSUM and a lexicographic refinement of MINMAX, but not MINMAX itself.

Related Work

The formal model of liquid democracy with ranked delegations was put forward by Kotsialou and Riley (2020), who were interested in identifying unravelling procedures that avoid cycles and have good normative properties. They study the properties of breadth-first search and depth-first search in this context. Their model inspired a number of recent papers (Colley, Grandi, and Novaro 2022; Brill et al. 2022; Kavitha et al. 2020; Utke and Schmidt-Kraepelin 2023); in particular, the work of Colley et al. extends the model of Kotsialou and Riley (2020) to more expressive ballots and is the direct predecessor of our work. Brill et al. (2022) define the concept of delegation graphs, for which unravelling procedures are called delegation rules. They study two important subclasses: sequence rules and confluent rules; the latter corresponding to selecting an arborescence of the graph. MINSUM and MINMAX are both confluent rules. Brill et al. provide a wide range of normative and experimental results. Of importance to us, MINSUM,[1] corresponding to minimum-cost arborescences, satisfies two attractive axioms—guru-participation and copy-robustness (the latter is similar to our new axiom)—if tie-breaking is performed lexicographically with respect to a fixed ordering of the agents. In contrast, our axiom is formulated for irresolute MINSUM, and can be satisfied by a resolute anonymous rule (i.e., treating all agents symmetrically), at the expense of neutrality. Moreover, building on the popular arborescences framework of Kavitha et al. (2020), the authors experimentally show that on most instances a majority of voters prefer MINSUM arborescences to other arborescences.

There is also a substantial body of work on liquid democracy in the setting with ground truth (Kahng, Mackenzie, and Procaccia 2021; Gölz et al. 2021; Caragiannis and Micha 2019; Becker et al. 2021; Alouf-Heffetz et al. 2022), as well as on game-theoretic aspects of delegation (Bloembergen, Grossi, and Lackner 2019; Escoffier, Gilbert, and Pass-Lanneau 2020; Zhang and Grossi 2021).

[1]Called BordaBranching in their paper to distinguish it from a different rule called MinSum.

Preliminaries

A finite set of agents/voters $N = \{1, \dots, n\}$ aim to select an outcome from a finite set D of alternatives. For most of our results, we focus on the *binary* setting $D = \{0, 1\}$.

Classic Liquid Democracy

We first describe the classic model of liquid democracy with ranked delegations; then, in the next section, we will delve into its generalisation proposed by Colley et al. (2022). We will refer to the latter model simply as *liquid democracy* whenever clear from the context.

The decision-making process proceeds as follows. First, each agent $a \in N$ casts an ordered ballot with size k_a: $B_a = B_a(0) \succ \dots \succ B_a(k_a - 1) \succ \tau_a$, where $B_a(i) \in N \setminus \{a\}$ for each $i = 0, \dots, k_a - 1$ and $\tau_a \in D$. We require $B_a(i) \neq B_a(j)$ for $0 \leq i < j < k_a$. For brevity, we write $B_a(k_a) = \tau_a$. Intuitively, agent a wants to delegate their vote first to $B_a(0)$, as a second option to $B_a(1)$, and so on; their final backup option is to vote directly for $\tau_a \in D$. When $k_a = 0$, the ballot of agent a is $B_a = \tau_a$, which indicates that a will vote directly for τ_a. After the ballots are cast, the electoral authority *unravels* the preferences by turning the collection of ballots $\mathbf{B} = (B_a)_{a \in N}$ into a collection of *concrete* votes $\mathbf{X} = (X_a)_{a \in N} \in D^N$. Then, a deterministic aggregation rule $agg : D^N \to D$ is applied to $\mathbf{X}$ to compute the final decision $agg(\mathbf{X})$. We will sometimes work with *partial* collections of concrete votes $\mathbf{X} \in (D \cup \{\perp\})^N$ where some entries are unspecified (as indicated by $X_a = \perp$).

To unravel the preferences, one can select, for each agent $a \in N$, a position $0 \leq c_a \leq k_a$ from their ballot and use the respective entry $B_a(c_a)$ to compute the concrete votes. This approach has the attractive feature of explainability: The collection of positions $\mathbf{c} = (c_a)_{a \in N}$ forms a *certificate*, which can be reported together with the electoral outcome as an explanation for it. Unravelling procedures that work in this fashion are called *confluent* (Brill et al. 2022). They provide a high degree of accountability, as each agent can see whether their direct vote τ_a was used, or which other agent their vote was delegated to.

Naturally, not all certificates are acceptable: e.g., if Alice and Bob want to delegate to each other as their first preference, the certificate $\mathbf{c} = (0, 0)$ leads to a delegation cycle, and hence to no concrete votes being computed for Alice and Bob. Following the terminology of Colley et al. (2022), we call such certificates *inconsistent*. It is convenient to formalise this concept in the language of graphs.

A tuple $(N, D, (B_a)_{a \in N})$ can be seen as a directed weighted graph called the *delegation graph*. Its vertex set is $N \cup D \cup \{r\}$, where r is a special vertex (the *root*). There is an edge $d \to r$ of weight 0 for each $d \in D$ as well as an edge $a \to B_a(i)$ of weight/cost[2] i for each $a \in A$ and $0 \leq i \leq k_a$. A certificate $\mathbf{c}$ is *consistent* if the set of edges $T_{\mathbf{c}} = \{(a, B_a(c_a)) \mid a \in A\} \cup \{(d, r) \mid d \in D\}$ is an r-*arborescence* (i.e., a directed spanning tree with sink r) of the delegation graph. In particular, every node can reach r along the edges in $T_{\mathbf{c}}$, so there are no delegation cycles. For

[2]We use these terms interchangeably.

brevity, we shorten "r-arborescence" to "simply arborescence". A consistent certificate $\mathbf{c}$ induces concrete votes $\mathbf{X}$ as follows: $X_a = d$ whenever d is on the a–r path in $T_{\mathbf{c}}$; we say that agent $a \in N$ *votes* $d \in D$ in $\mathbf{c}$.

When $\mathbf{B}$ admits multiple consistent certificates, it is natural to prefer ones selecting options that are ranked as highly as possible in the ballots. In particular, one can select the certificates that minimise $\sum_{a \in N} c_a$ or $\max_{a \in N} c_a$, i.e., maximise the *utilitarian* or *egalitarian* social welfare. We call such certificates MINSUM and MINMAX certificates, and extend this terminology to unravelling procedures that select certificates optimising the respective quantities. Notice that these procedures are irresolute, as a minimiser of $\sum_{a \in N} c_a$ or $\max_{a \in N} c_a$ need not be unique. In graph-theoretic terms, MINSUM and MINMAX certificates correspond to, respectively, minimum weight and minimum bottleneck arborescences. Given a (potentially irresolute) unravelling procedure F, e.g., MINSUM or MINMAX, and ballots $\mathbf{B}$, we write $F(\mathbf{B})$ for the set of collections of concrete votes $\mathbf{X}$ that can be output by F when executed with input $\mathbf{B}$. When F is resolute, we simply write $F(\mathbf{B}) = \mathbf{X}$ instead of $F(\mathbf{B}) = \{\mathbf{X}\}$.

Enhanced Liquid Democracy

We now present *smart voting*,[3] as defined by Colley et al. (2022): the extension of the classic delegation model that allows agents to delegate to complex functions of the other agents. For readability, we limit ourselves to the binary setting. We first take a short detour to introduce the notions of Boolean logic needed for our examples and results.

A *k-ary Boolean function* $f\colon \{0,1\}^k \to \{0,1\}$ is a mapping from k-tuples of binary values to binary values. Function f is *monotonic* if for all k-tuples $\mathbf{x}, \mathbf{y}$ such that $x_i \leq y_i$ for $1 \leq i \leq k$ it holds that $f(\mathbf{x}) \leq f(\mathbf{y})$. Two k-ary Boolean functions f and g are equal if they are *extensionally equal*, i.e., $f(\mathbf{x}) = g(\mathbf{x})$ for any k-tuple $\mathbf{x}$. Given a k-ary Boolean function f, substituting values for ℓ of its arguments yields a k'-ary function for $k' = k - \ell$. In a slight abuse of notation, we use 0 and 1 to denote the two constant functions (of arity zero). We express Boolean functions via *Boolean formulas* using connectives such as $\wedge$, $\vee$ and $\neg$; we often write $\neg x$ as $\overline{x}$. For instance, the ternary majority function $\textsc{Maj}(x, y, z)$ can be written as $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$. When a Boolean function is monotonic, it can be expressed by a *monotone* Boolean formula, i.e., without negations.

We will work with n-ary Boolean functions with arguments indexed by the set of agents N, i.e., functions $f : \{0,1\}^N \to \{0,1\}$. When a function does not require some of its arguments, we may omit them, and consider an "equivalent" function of smaller arity. Formally, if f only depends on its arguments indexed by $S \subsetneq N$ then we will view it as $f' : \{0,1\}^S \to \{0,1\}$. Conversely, in contexts requiring it, a function $f' : \{0,1\}^S \to \{0,1\}$ denotes the function $f : \{0,1\}^N \to \{0,1\}$ with redundant arguments added for $N \setminus S$. Given a (possibly partial) list of concrete votes $\mathbf{X} \in \{0,1,\bot\}^N$, we write $f(\mathbf{X})$ for the function obtained from f by substituting $a \mapsto X_a$ for all $a \in N$ with $X_a \neq \bot$.

[3]Not to be confused with the Swiss voting advice application `www.smartvote.ch`.

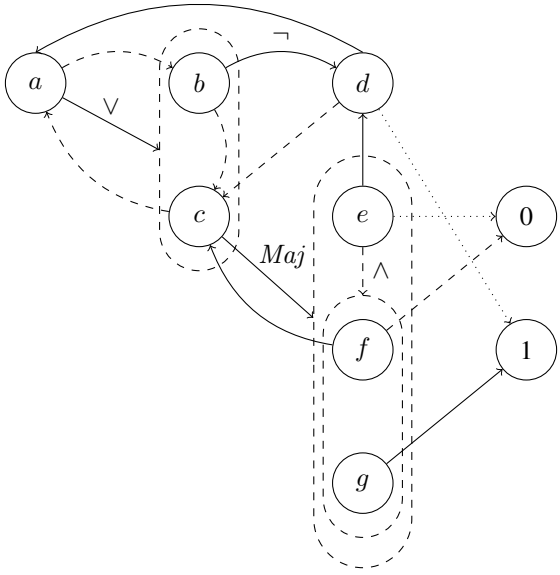

Figure 1: Preference profile in Example 1. Solid lines indicate first preferences, dashed lines indicate second preferences, and dotted lines indicate third preferences. To avoid clutter we have removed the third preferences of a, b and c.

In the enhanced liquid democracy model of Colley et al. (2022), for each $a \in N$ and $0 \leq i < k_a$ the i-th entry $B_a(i)$ in agent a's ballot is a Boolean function with arguments indexed by $N \setminus \{a\}$. The final entries $B_a(k_a) = \tau_a$ naturally correspond to one of the two constant functions.

Example 1. Consider agents $N = \{a, b, c, d, e, f, g\}$ casting ballots $\mathbf{B} = (B_a)_{a \in N}$ as follows (Figure 1):

$$B_a = b \vee c \succ b \succ 0 \qquad B_b = \overline{d} \succ c \succ 1$$
$$B_c = \mathit{Maj}\,(e, f, g) \succ a \succ 1 \qquad B_d = a \succ c \succ 1$$
$$B_e = d \succ f \wedge g \succ 0 \qquad B_f = c \succ 0$$
$$B_g = 1$$

The enhanced model generalises classic liquid democracy in that the latter only allows projection functions (direct delegations) and constant functions (direct votes). Most definitions from the classic model extend in a straightforward way, but a few important points need to be discussed. First, what language is used to express Boolean functions? Note that the validation check $B_a(i) \neq B_a(j)$ for all $0 \leq i < j \leq k_a$[4] has to be replaced with (extensional) function equality: a simple textual comparison would not suffice, as a Boolean function can be expressed by multiple Boolean formulas. We want the function-definition language to allow for this check to be performed efficiently, i.e., in polynomial time; hence, the language of arbitrary Boolean formulas is not suitable. Perhaps most importantly, the definition of the delegation graph does not extend to the enhanced model, and hence we need a new definition of a consistent certificate. As the enhanced model generalises the classic model, this definition

[4]Note the last $\leq$ instead of $<$.

should coincide with the classic one for instances that only use direct delegations and constants. It will be convenient to use the following reformulation of consistency for the classic model (the proof follows by considering arborescence traversals where vertices occur after their parents).

Lemma 2. In the classic model, a certificate $\mathbf{c}$ is consistent iff there exists a linear order $\lhd$ on N such that for all agents $a \in N$ it holds that either $B_a(c_a) \in D$ or $B_a(c_a) \lhd a$.

In other words, a certificate is consistent iff the concrete votes can be computed by starting with $\mathbf{X} = (\bot, \dots, \bot)$ and at each step picking an agent $a \in N$ with $X_a = \bot$ and resolving their vote, i.e., setting X_a either to $B_a(c_a) \in D$ or to $X_{B_a(c_a)}$; for the latter choice we need $X_{B_a(c_a)} \neq \bot$, i.e., $B_a(c_a) \lhd a$, which should be read as "the vote of $B_a(c_a)$ is resolved before that of a". Armed with the idea that concrete votes should be resolved sequentially, we extend the definition of consistency to the enhanced model as follows.

In the enhanced model, we say that a certificate $\mathbf{c}$ is consistent iff there is a linear order $\lhd$ on N that is *valid*, in the sense that the concrete votes can be computed in the following way. We start with $\mathbf{X} = (\bot, \dots, \bot)$ and go through the agents in the order given by $\lhd$. Whenever an agent $a \in N$ is reached, let $f = B_a(c_a)$ be their assigned function in the certificate. Then it should be the case that f simplifies to one of the two constant functions 0 or 1 when the values known for agents that appear earlier in $\lhd$ are substituted in. Formally, either $f(\mathbf{X}) = 0$ or $f(\mathbf{X}) = 1$ should hold, where equality denotes extensional equality. We set $X_a \leftarrow f(\mathbf{X})$ and move on to the next agent in order. That is, the way in which functions are evaluated is in the spirit of lazy function evaluation in some programming languages.

For instance, let $N = \{a, b, c\}$. Suppose that the function $f = B_a(c_a)$ of agent a assigned by the certificate is $b \vee c$, and consider the order $b \lhd a \lhd c$. If, when a is reached in the order, we have $\mathbf{X} = (\bot, 1, \bot)$, we can compute $f(\mathbf{X}) = f(\bot, 1, \bot) = 1 \vee c = 1$, even though the vote of c is still unknown. However, for $\mathbf{X} = (\bot, 0, \bot)$ the order $\lhd$ is not valid: $f(\mathbf{X}) = f(\bot, 0, \bot) = 0 \vee c = c$ is not extensionally equal to a constant function.

Crucially, the computed concrete votes $\mathbf{X}$ do not depend on the choice of the valid ordering $\lhd$: This is because every assignment performed is, in a sense, "forced."[5] Hence, for a consistent certificate, the concrete votes $\mathbf{X}$ are uniquely determined and are the unique solution to the system of simultaneous equations $X_a = B_a(c_a)(\mathbf{X})$ for all $a \in N$. One can ask whether the converse is also true. Namely, given a certificate $\mathbf{c}$ such that the system of simultaneous equations has a unique solution (i.e., the concrete votes are uniquely determined), is it consistent? If this were the case, this would be an argument in favor of consistent certificates as an attractive ground notion. The next theorem shows that this is indeed true when delegations are restricted to monotonic functions (see the full version of the paper for the proof, and all other omitted proofs), and we will argue that there are strong arguments for using monotonic functions from a normative perspective.

[5]To decide if a valid $\lhd$ exists it suffices to attempt to construct it iteratively, by picking, at each step, a new agent a whose vote can be determined based on the previous ones.

Theorem 3. *Assuming delegation is only permitted to monotonic functions, a certificate* $\mathbf{c}$ *is consistent iff the system of simultaneous equations* $X_a = B_a(c_a)(\mathbf{X})$ *for all* $a \in N$ *has a unique solution.*

We now return to Example 1 to illustrate the new definitions, together with MINSUM and MINMAX, whose definitions remain unchanged.

Example 1 (continued)**.** Observe that $\mathbf{c} = (0, \dots, 0)$ is not a consistent certificate. Indeed, the iterative process for this certificate starts with $\mathbf{X} = (\bot, \dots, \bot)$, sets $X_g = 1$ in the first step, but can not proceed further. This is because the knowledge of X_g does not immediately uniquely identify any other value in $\mathbf{X}$, since, considering first preferences only, g is only mentioned in the first preference of c, and $Maj(e, f, g) = Maj(e, f, 1)$ is not uniquely determined without knowing X_e and X_f first.

Hence, if there is a consistent certificate using only the first two preferences of all agents, then it is a MINMAX certificate. One such certificate is $\mathbf{c} = (0, 1, 0, 0, 1, 1, 0)$. To see this, start again the iterative process with $\mathbf{X} = (\bot, \dots, \bot)$. We immediately obtain $X_g \leftarrow 1$ and $X_f \leftarrow 0$. Subsequently, $X_e \leftarrow X_f \wedge X_g = 0 \wedge 1 = 0$. Then, $X_c \leftarrow Maj(X_e, X_f, X_g) = Maj(0, 0, 1) = 0$, which could not have been determined before X_e since $Maj(X_e, 0, 1) = X_e$. Afterwards, $X_b \leftarrow X_c = 0$. Then $X_a \leftarrow X_b \vee X_c = 0 \vee 0 = 0$ and $X_d \leftarrow X_a = 0$. So, the iterative procedure has successfully determined the concrete votes $\mathbf{X} = (0, 0, 0, 0, 0, 0, 1)$, so $\mathbf{c}$ is a MINMAX certificate; i.e., $\max_{a \in N} c_a = 1$. Note that the order used to resolve the votes is $g \lhd f \lhd e \lhd c \lhd b \lhd a \lhd d$. Notationally, the concrete votes satisfy $\mathbf{X} \in \text{MINMAX}(\mathbf{B})$.

The previous certificate had $\sum_{a \in N} c_a = 3$. In contrast, a MINSUM certificate is given by $\mathbf{c} = (0, 0, 2, 0, 0, 0, 0)$, which satisfies $\sum_{a \in N} c_a = 2$. Performing the unravelling in this case would yield the direct votes $\mathbf{X} = (1, 0, 1, 1, 1, 1, 1) \in \text{MINSUM}(\mathbf{B})$. Note that $\mathbf{c}$ is not the only MINSUM certificate: $\mathbf{c}' = (0, 0, 0, 2, 0, 0, 0)$ would be a different one.

We end these preliminaries by returning to the expressivity of the language used to specify functions. For computational tractability, all functions that appear in the ballots should be polynomial-time computable. Moreover, ballot validation, as well as checking whether a given certificate is consistent (say after it has been reported to the voters) both require that checking extensional function equality can be achieved in polynomial time. These are all rather demanding properties, and they notably fail unless P = NP even for functions expressed as monotone formulas in disjunctive normal form (DNF), as reported by Colley et al. (2022). As a resolution, they propose restricting the class of allowable functions to minimised DNF formulas, for which the problems become tractable. We do not make this assumption, but instead note that for our results these conditions will be either trivially satisfied or not necessary (the latter holds for the hardness parts of dichotomy results).

Complexity of the Enhanced Model

In this section, we provide complete computational dichotomies for MINSUM and MINMAX for the binary setting assuming that all allowable functions are monotonic. It is easy to see why non-monotonic delegations may cause issues that are more fundamental than computational complexity. E.g., suppose the supporters of a substantial-sized party delegate to the negation of the vote of a high-influence individual who votes directly for 1. This individual is then motivated to vote directly for 0 knowing that this flips the vote of many agents from 0 to 1, adding an undesirable strategic element to the setup.

For $F \in \{\text{MINSUM}, \text{MINMAX}\}$ and a class of Boolean functions $\mathcal{F}$ including projections and constants (i.e., direct delegations and direct votes), we write $F_{\mathcal{F}}$ for F restricted to delegations to functions in $\mathcal{F}$. Notable examples of $\mathcal{F}$ include BOOL, the class of all Boolean functions, MON, the class of all monotonic Boolean functions, LIQUID, which only allows projections and constants (i.e., classic liquid democracy), OR_2, which additionally allows binary disjunctions (similarly AND_2), OR, which allows arbitrary-arity disjunctions (similarly AND) and finally $\text{ORAND}_2 = \text{OR}_2 \cup \text{AND}_2$. We assume $\mathcal{F}$ to be invariant to permuting agents in N; e.g., if agent a can delegate to $(b \vee c) \wedge d$, then a' can delegate to $(b' \vee c') \wedge d'$ for any pairwise distinct a', b', c', d'. This assumption is mild given that anonymity is seen as a desirable preference aggregation axiom.

For brevity, we adopt the following notational conventions. We say that $\text{MINSUM}_{\mathcal{F}}$ is NP-hard if the decision problem "Given x, decide if there is a consistent certificate with $\sum_{a \in N} c_a \leq x$" is NP-hard. We say that it is polynomial-time computable if there exists a polynomial-time algorithm outputting a $\text{MINSUM}_{\mathcal{F}}$ certificate. We say that $\text{MINSUM}_{\mathcal{F}}$ is hard to approximate within a constant factor if, unless P = NP, for no constant $C \geq 1$ does there exist a polynomial-time algorithm outputting a consistent certificate $\mathbf{c}$ with $\sum_{a \in N} c_a$ at most C times more than this sum for a MINSUM certificate. Moreover, we say that it is hard to approximate within an $n^{1-\epsilon}$ factor if for no $\epsilon > 0$ the above holds for $C = n^{1-\epsilon}$. We extend this terminology to $\text{MINMAX}_{\mathcal{F}}$.

Colley et al. (2022) show that $\text{MINSUM}_{\text{LIQUID}}$ and $\text{MINMAX}_{\text{LIQUID}}$ are polynomial-time computable while $\text{MINSUM}_{\text{AND}}$ and $\text{MINMAX}_{\text{BOOL}}$ are NP-hard. We strengthen these results by giving a full characterisation of when the problems admit polynomial-time algorithms.

Complexity of $\text{MINSUM}_{\mathcal{F}}$. We start by showing that if either $\text{OR}_2 \subseteq \mathcal{F}$ or $\text{AND}_2 \subseteq \mathcal{F}$, then $\text{MINSUM}_{\mathcal{F}}$ is NP-hard and hard to approximate within an $n^{1-\epsilon}$ factor, even for ballots of maximum size two.

Theorem 4. *$\text{MINSUM}_{\text{OR}_2}$ and $\text{MINSUM}_{\text{AND}_2}$ are hard to $n^{1-\epsilon}$-factor approximate even for maximum ballot size 2.*

We prove this result by reducing from the decision version of VERTEXCOVER. We construct voter gadgets for each vertex and edge. Edge gadgets can be resolved using the voters' first preferences if and only if the edge is covered. We then use gap amplification to show hardness of approximation. The $n^{1-\epsilon}$ bound is tight for ballots of size at most 2 since there exists a simple n-approximation: unless everyone can get their first (cost 0) choice, the optimal solution has cost at least 1, so giving everyone their second choice is an n-approximation.

Complexity of $\text{MINMAX}_{\mathcal{F}}$. Next, we show that if $\text{ORAND}_2 \subseteq \mathcal{F}$ then $\text{MINMAX}_{\mathcal{F}}$ is NP-hard even for ballots of maximum size three,[6] and hard to approximate within a constant factor. In contrast, we prove that $\text{MINMAX}_{\text{OR}}$ and $\text{MINMAX}_{\text{AND}}$ are polynomial-time computable, showing the need for both $\wedge$ and $\vee$ to obtain hardness. We begin with the easiness result.

Theorem 5. *$\text{MINMAX}_{\text{OR}}$ and $\text{MINMAX}_{\text{AND}}$ are polynomial-time computable (for arbitrary ballot sizes).*

We now turn to the hardness results. The proofs are similar in spirit to those for MINSUM. We reduce from a version of 3SAT.

Theorem 6. *$\text{MINMAX}_{\text{ORAND}_2}$ is NP-hard even for maximum ballot size 3, and hard to constant-factor approximate.*

Computational Dichotomies. So far, we have proven results concerning $\vee$ and $\wedge$. We now extend our results to full computational dichotomies establishing tractability/intractability for every class of monotonic functions $\mathcal{F} \subseteq \text{MON}$. We implicitly assume $\text{LIQUID} \subseteq \mathcal{F}$ since LIQUID alone entails tractability anyway.

Theorem 7. *Assume $\mathcal{F} \subseteq \text{MON}$. If $\mathcal{F} = \text{LIQUID}$, $\text{MINSUM}_{\mathcal{F}}$ is poly-time computable. Otherwise, $\text{MINSUM}_{\mathcal{F}}$ is NP-hard and hard to $n^{1-\epsilon}$-approximate even for ballots of size at most 2. Also, If $\mathcal{F} \subseteq \text{OR}$ or $\mathcal{F} \subseteq \text{AND}$, $\text{MINMAX}_{\mathcal{F}}$ is poly-time computable. Otherwise, $\text{MINMAX}_{\mathcal{F}}$ is NP-hard even for ballots of size at most 3, and hard to constant-factor approximate.*

As an immediate corollary, if agents are allowed to delegate to some odd majority then MINSUM and MINMAX are hard, resolving a question left open by Colley et al. (2022).

Complexity of the Classic Model

We now revisit the classic model of liquid democracy with ranked delegations, as described in the Preliminaries. In particular, we study the MINSUM and MINMAX unravelling procedures, corresponding to minimum cost and minimum bottleneck arborescences in the delegation graph.

Efficient Computation of MINSUM and MINMAX Certificates

Our first results concern efficiently computing an arbitrary optimal certificate. Compared to previous works, our algorithm for MINMAX is asymptotically tight, while the one for MINSUM is tight up to a logarithmic factor. In this section only, we do not make the assumption that $D = \{0, 1\}$,

[6] Note the contrast with $\text{MINSUM}_{\mathcal{F}}$, where two sufficed for hardness. For maximum size two, finding a MINMAX certificate reduces to checking whether $\mathbf{c} = (0, \ldots, 0)$ is consistent, whose difficulty is given solely by the difficulty of testing function equivalence, which is easy for ORAND_2.

since our analysis applies to general domains with no additional complications. We work with a modified delegation graph: we remove the vertices that correspond to $D \subseteq V$, and redirect the edges pointing to them to point directly to the root r.[7] We then have $|V| = n+1$; we write $m = |E| = n + \sum_{a \in N} k_a$ for the total number of edges and ℓ for the maximum delegation ballot size. The algorithms proposed by Colley, Grandi, and Novaro (2022) run in time $\mathcal{O}(nm)$ for MINSUM and $\mathcal{O}(n^2\ell^2)$ for MINMAX.

Theorem 8. *A* MINSUM *certificate can be computed in time* $O(n \log n + m)$*, and a* MINMAX *certificate can be computed in time* $O(n + m)$.

The algorithm we propose for MINMAX can be described in purely graph-theoretic terms: given a directed graph with integer edge weights between 0 and W, it computes a minimum bottleneck arborescence with a given root in time $O(n + m + W)$. Despite the simplicity of the algorithm, we have not been able to find it in prior work.

Control for MINSUM and MINMAX

Thus far, we have concerned ourselves with the irresolute variants of MINSUM and MINMAX, where the electoral authority can choose any arborescence minimising the respective quantity and then apply an aggregation function. But what if they want to favour a specific alternative $d \in D = \{0, 1\}$? Without loss of generality, assume they want to favour alternative 1. Recall that for any arborescence T, the agents $a \in N$ that vote 1 in the concrete votes are the descendants of 1 in T. For both unravelling procedures, we will show that, surprisingly, if N_1 is the set of agents who vote for 1 in at least one optimal arborescence, then there is an optimal arborescence where all agents in N_1 vote for 1, and such an arborescence can be computed in polynomial time. This has immediate implications for the electoral authority attempting to exercise control: they can compute and select such an arborescence. The respective concrete votes subsume all possible concrete votes for 1 found in any optimal arborescence, so this leads to the best possible outcome for the authority, no matter which monotonic aggregation function is applied to the concrete votes. For MINMAX, the proof is not difficult.

Theorem 9. *Let* $N_1 \subseteq N$ *be the agents who vote 1 in some minimum bottleneck arborescence (i.e.,* MINMAX *certificate). Then, we can compute in* $O(n+m)$ *time a minimum bottleneck arborescence where all agents in* N_1 *vote 1.*

Proof. For convenience, we work on the transposed graph. Let w^* be the maximum edge weight used in a minimum bottleneck arborescence. This can be computed in $O(n+m)$ time using Theorem 8. Write $E_{\leq x} = \{e \in E \mid w(e) \leq x\}$ for the set of edges of weight at most x and $G_{\leq x} = (V, E_{\leq x})$ for the graph restricted to such edges. Then, any arborescence of $G_{\leq w^*}$ is a minimum bottleneck arborescence, and vice-versa. For readability, assume we remove all other edges and set $G = G_{\leq w^*}$. Consider the set S of vertices reachable from 1 in G. Note that in all arborescences vertices in $(V \setminus \{r\}) \setminus S$ will be descendants of 0, so, if we can find an arborescence where all vertices in S are descendants of 1, then the conclusion follows. Such an arborescence can indeed easily be constructed by running a depth-first search starting from the root and recursing along the edge to 1 before the edge to 0. This takes $O(n + m)$ time. Note that we implicitly get that $N_1 = S \setminus \{1\}$. □

[7] This gives slightly tighter and cleaner time bounds, whilst not impacting the certificate-arborescence correspondence.

The proof for MINSUM, while similar in spirit, requires insights from Fulkerson's primal-dual algorithm for computing minimum cost arborescences (Fulkerson 1974). Modern accounts of the algorithm are available in (Kamiyama 2014; Utke and Schmidt-Kraepelin 2023; Bernáth and Pap 2013). The algorithm outputs a collection $E' \subseteq E$ of edges of the graph, called the "tight" edges, as well as a *laminar family* $\mathcal{L}$ of subsets of $V \setminus \{r\}$. Laminarity means that for any $L_1, L_2 \in \mathcal{L}$ either $L_1 \subseteq L_2$, or $L_2 \subseteq L_1$ or $L_1 \cap L_2 = \varnothing$. Laminar families can be understood through their associated forest, where each set $L \in \mathcal{L}$ has a set of children sets $ch_{\mathcal{L}}(L)$ consisting of those $L' \in \mathcal{L}$ such that $L' \subsetneq L$ and there is no $L'' \in \mathcal{L}$ satisfying $L' \subsetneq L'' \subsetneq L$. The set of *roots* of the forest is denoted $rt_{\mathcal{L}}$ and consists of all maximal sets in $\mathcal{L}$. Fulkerson's algorithm is given in the full version. For this section, it suffices to understand a few key properties of its output known from previous works. Most importantly, the relationship between minimum cost arborescences and $(E', \mathcal{L})$ is given by the following powerful lemma, for which given a set of vertices $S \subseteq V$, we write $\rho(S) = \{(u, v) \in E \mid u \in S \text{ and } v \notin S\}$.

Lemma 10. An arborescence T has minimum cost iff the following two conditions hold: (i) $T \subseteq E'$; (ii) for any $L \in \mathcal{L}$ it holds that $|\rho(L) \cap T| = 1$.

This gives a combinatorial characterisation of minimum-cost arborescences without mentioning edge weights, enabling us to adapt our approach for MINMAX to MINSUM.

Theorem 11. *Let* $N_1 \subseteq N$ *be the set of agents who vote 1 in some min-cost arborescence (i.e.,* MINSUM *certificate). Then we can compute in polynomial time a min-cost arborescence where all agents in* N_1 *vote 1. Moreover,* $a \in N_1$ *iff 1 is reachable from* a *along edges in* E' *output by Fulkerson's algorithm.*

On the positive side, Theorems 9 and 11 suggest biased tie-breaking rules for MINMAX and MINSUM, which we denote by MINMAX^1 and MINSUM^1, selecting the collection of concrete votes $(X_a)_{a \in N}$ with the highest possible number of ones. Our results show that these procedures are resolute and subsume all possible votes for 1 among MINMAX (MINSUM) certificates. One can also define MINMAX^0 and MINSUM^0 analogously. These biased procedures can find use in settings where the decision to be made is for changing a status quo, where one can select between biasing for or against the status quo prior to the vote. We note the following attractive corollary of our results, for which given two collections of concrete votes $\mathbf{X}, \mathbf{Y}$ we write $\mathbf{X} \leq \mathbf{Y}$ to mean that $X_a \leq Y_a$ for all $a \in N$:

Corollary 12. For any ballot list $\mathbf{B}$ and $\mathbf{X} \in \text{MinMax}(\mathbf{B})$, $\mathbf{X}' \in \text{MinSum}(\mathbf{B})$, we have $\text{MinMax}^0(\mathbf{B}) \leq \mathbf{X} \leq \text{MinMax}^1(\mathbf{B})$ and $\text{MinSum}^0(\mathbf{B}) \leq \mathbf{X}' \leq \text{MinSum}^1(\mathbf{B})$.

Axiomatic Results

In this section we study MinSum and MinMax from a normative standpoint. We introduce a natural axiom, which we call *cast monotonicity*, and establish that it is satisfied by MinSum and a lexicographic refinement of MinMax (but not MinMax itself). This holds both for the irresolute procedures and for their biased resolute variants. Unless stated otherwise, in this section we consider the classic model.

Fix a set of agents N and $D = \{0, 1\}$. Let F be a (possibly irresolute) unravelling procedure mapping collections of ballots $\mathbf{B} = (B_a)_{a \in N}$ to collections of concrete votes $\mathbf{X} = (X_a)_{a \in N}$. Given an aggregation function $agg : \{0, 1\}^N \to \{0, 1\}$, the set of possible winning alternatives with respect to F on input $\mathbf{B}$ is $agg(F(\mathbf{B})) = \{agg(\mathbf{X}) \mid \mathbf{X} \in F(\mathbf{B})\}$.

Definition 13. An unravelling procedure F is *cast-monotonic* with respect to a monotonic aggregation function agg if for every ballot list $\mathbf{B}$, every agent $a \in N$, each alternative $d \in D$ and the ballot list $\mathbf{B}'$ obtained from $\mathbf{B}$ by changing B_a to a direct vote for d we have: (i) $d \in agg(F(\mathbf{B}))$ implies $d \in agg(F(\mathbf{B}'))$ and (ii) $1 - d \notin agg(F(\mathbf{B}))$ implies $1 - d \notin agg(F(\mathbf{B}'))$. We say that F is *(unconditionally) cast-monotonic* if it is cast-monotonic with respect to all monotonic aggregation functions agg.

Intuitively, cast monotonicity says that if an agent likes an alternative d, then she is best off voting for it directly: changing from an arbitrary ballot to a direct vote for d can neither remove d from the set of winners nor add $1 - d$ to the set of winners.

Given two lists of concrete votes $\mathbf{X}, \mathbf{X}'$, recall that $\mathbf{X} \leq \mathbf{X}'$ if $X_a \leq X'_a$ for all $a \in N$. We write $\leq_d$ to mean $\leq$ for $d = 1$ and $\geq$ for $d = 0$. Then, we can give the following characterisation of unconditional cast monotonicity.

Lemma 14. An unravelling procedure F is cast-monotonic iff for every ballot list $\mathbf{B}$, every agent $a \in N$, every alternative $d \in D$, and the ballot list $\mathbf{B}'$ obtained from $\mathbf{B}$ by changing B_a to a direct vote for d, the following conditions hold: (iii) for every $\mathbf{X} \in F(\mathbf{B})$ there is an $\mathbf{X}' \in F(\mathbf{B}')$ with $\mathbf{X} \leq_d \mathbf{X}'$; (iv) for every $\mathbf{X}' \in F(\mathbf{B}')$ there is an $\mathbf{X} \in F(\mathbf{B})$ with $\mathbf{X} \leq_d \mathbf{X}'$. Note that if F is resolute, (iii) is equivalent to (iv).

It is not hard to show that MinMax and its biased variants are not cast-monotonic.

Theorem 15. MinMax *fails cast monotonicity. The same holds for its resolute variants* MinMax^p *with* $p \in \{0, 1\}$.

Next, we show that MinSum and its variants are cast-monotonic. Our proof (see the full version) relies on somewhat technical arguments regarding Fulkerson's algorithm.

Theorem 16. MinSum, MinSum^0, MinSum^1 *satisfy cast monotonicity.*

The reader may wonder whether cast monotonicity results for MinSum extend to the more general setting where agents can delegate to monotonic Boolean functions. Sadly, this is not the case, even if we only allow binary $\vee$ or $\wedge$.

Theorem 17. $\text{MinSum}_{\mathcal{F}}$ *fails cast monotonicity as soon as either* $\text{Or}_2 \subseteq \mathcal{F}$ *or* $\text{And}_2 \subseteq \mathcal{F}$. *The same holds for the resolute variants* $\text{MinSum}^p_{\mathcal{F}}$ *for* $p \in \{0, 1\}$.

To end the section, we observe that the lexicographic refinement of MinMax satisfies cast monotonicity. In this refinement, we seek a certificate that minimises the number of agents $a \in N$ with $c_a = n - 1$, breaking ties by the number of agents with $c_a = n - 2$, and so on. Formally:

Definition 18 (LexiMin)**.** Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be two certificates. We say that $\mathbf{c}_1 \leq_{lex} \mathbf{c}_2$ if, after sorting the entries in $\mathbf{c}_1$ and $\mathbf{c}_2$ in non-increasing order, $\mathbf{c}_1$ is lexicographically no larger than $\mathbf{c}_2$. The unravelling procedure LexiMin returns a minimum consistent certificate with respect to $\leq_{lex}$.

LexiMin can be viewed as a variant of MinSum where the edge weights $0, 1, 2, \ldots, n - 1$ in the delegation graph are replaced with weights $1, X, X^2, \ldots, X^{n-1}$ for a large enough X (e.g., $X = n + 2$ suffices). Indeed, our results for MinSum do not require the edge weights to be drawn from $\{0, \ldots, n-1\}$; rather, they hold whenever the weights form a strictly increasing sequence. Accordingly, the two biased versions of LexiMin are resolute, easy to compute, and satisfy cast monotonicity. For more expressive function classes, LexiMin inherits its intractability from that of MinSum, as it coincides with it on ballots of maximum size 2.

Summary and Future Work

We have extended the work of Colley et al. (2022), by strengthening their complexity-theoretic results to a complete dichotomy for monotone Boolean functions. For the standard model, we provided near-optimal algorithms for the two unravelling procedures and efficient algorithms for computing an outcome where a given alternative wins, leading to attractive tie-breaking rules for the setup where one alternative is the status quo. We introduced the cast monotonicity axiom, which MinSum and the lexicographic refinement of MinMax satisfy, while standard MinMax does not; these results extend to the biased variants of these rules.

One interesting direction for future work is to extend the model by allowing agents to assign cardinal values to their preferences, instead of just ordering them. Moreover, our results show that computing optimal unravellings is NP-hard for most reasonable function classes. However, one can consider a variant of the model where agents are required to choose before the election whether they want to be delegates. The delegates have to disclose their votes, while the non-delegates can keep their votes secret. Suppose we allow non-delegates to vote for a single Boolean function of the delegates, but delegates can only delegate directly to other delegates. Then, hardness of unravelling no longer arises, as cycles may only appear in the pure fragment of the model. Thus, this model offers at attractive tradeoff between expressive power and tractability. An interesting question in this setting is under what conditions an agent prefers to join the delegates as opposed to remaining a voter.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive feedback and useful suggestions contributing to improving this paper. This work was supported by the AI Programme of The Alan Turing Institute. A preliminary version of this work has been presented at the Games, Agents, and Incentives Workshop of AAMAS'23, based on the first author's Master's thesis, which would not have been possible without the third author's supervision. We thank the workshop attendees for their interesting questions and remarks.

References

Alouf-Heffetz, S.; Armstrong, B.; Larson, K.; and Talmon, N. 2022. How should we vote? A comparison of voting systems within social networks. In *31st International Joint Conference on Artificial Intelligence*, 31–38.

Becker, R.; D'Angelo, G.; Delfaraz, E.; and Gilbert, H. 2021. Unveiling the truth in liquid democracy with misinformed voters. In *7th International Conference on Algorithmic Decision Theory*, 132–146.

Behrens, J. 2017. The origins of liquid democracy. *The Liquid Democracy Journal*, (5): 7–17.

Bernáth, A.; and Pap, G. 2013. Blocking Optimal Arborescences. In Goemans, M.; and Correa, J., eds., *Integer Programming and Combinatorial Optimization*, 74–85. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-36694-9.

Bloembergen, D.; Grossi, D.; and Lackner, M. 2019. On rational delegations in liquid democracy. In *33rd AAAI Conference on Artificial Intelligence*, 1796–1803.

Brill, M.; Delemazure, T.; George, A.-M.; Lackner, M.; and Schmidt-Kraepelin, U. 2022. Liquid democracy with ranked delegations. In *36th AAAI Conference on Artificial Intelligence*, 4884–4891.

Caragiannis, I.; and Micha, E. 2019. A contribution to the critique of liquid democracy. In *28th International Joint Conference on Artificial Intelligence*, 116–122.

Colley, R.; Grandi, U.; and Novaro, A. 2022. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems*, 36(1): 1–35.

Escoffier, B.; Gilbert, H.; and Pass-Lanneau, A. 2020. Iterative delegations in liquid democracy with restricted preferences. In *34th AAAI Conference on Artificial Intelligence*, 1926–1933.

Ford, B. A. 2002. Delegative democracy. Technical report.

Fulkerson, D. R. 1974. Packing Rooted Directed Cuts in a Weighted Directed Graph. *Math. Program.*, 6(1): 1–13.

Gölz, P.; Kahng, A.; Mackenzie, S.; and Procaccia, A. D. 2021. The fluid mechanics of liquid democracy. *ACM Transactions on Economics and Computation*, 9(4): 1–39.

Kahng, A.; Mackenzie, S.; and Procaccia, A. 2021. Liquid democracy: An algorithmic perspective. *Journal of Artificial Intelligence Research*, 70: 1223–1252.

Kamiyama, N. 2014. Arborescence Problems in Directed Graphs: Theorems and Algorithms. *Interdisciplinary Information Sciences*, 20(1): 51–70.

Kavitha, T.; Király, T.; Matuschke, J.; Schlotter, I.; and Schmidt-Kraepelin, U. 2020. Popular Branchings and Their Dual Certificates. In Bienstock, D.; and Zambelli, G., eds., *Integer Programming and Combinatorial Optimization*, 223–237. Cham: Springer International Publishing. ISBN 978-3-030-45771-6.

Kotsialou, G.; and Riley, L. 2020. Incentivising participation in liquid democracy with breadth-first delegation. In *19th International Conference on Autonomous Agents and Multiagent Systems*, 638–644.

Utke, M.; and Schmidt-Kraepelin, U. 2023. Anonymous and Copy-Robust Delegations for Liquid Democracy. *arXiv 2307.01174*.

Zhang, Y.; and Grossi, D. 2021. Power in liquid democracy. In *35th AAAI Conference on Artificial Intelligence*, 5822–5830.