Aggregating Conditionally Lexicographic Preferences on Multi-Issue Domains

Jérôme Lang, Jérôme Mengin, and Lirong Xia

Abstract

One approach to voting on several interrelated issues consists in using a language for compact preference representation, from which the voters' preferences are elicited and aggregated. A language usually comes with a domain restriction. We consider a well-known restriction, namely, *conditionally lexicographic preferences*, where both the relative importance between issues and the preference between values of an issue may depend on the values taken by more important issues. The naturally associated language consists in describing conditional importance and conditional preference by trees together with conditional preference tables. In this paper, we study the aggregation of conditionally lexicographic preferences, for several voting rules and several restriction of the framework. We characterizes computational complexity for some popular cases, and show that in many of them, computing the winner reduces in a very natural way to a MAXSAT problem.

1 Introduction

There are many situations where a group of agents have to make a common decision about a set of possibly interrelated issues, variables, or attributes. For example, this is the situation in the following three domains:

• *Multiple referenda*: there is a set of binary issues (such as building a sport centre, building a cultural centre etc.); on each of them, the group has to make a yes/no decision.

• *Committee elections*: there is a set of positions to be filled (such as a president, a vice-president, a secretary).

• *Group product configuration*: the group has to agree on a complex object consisting of several components.

Voting on several interrelated issues has been proven to be a challenging problem from both a social choice viewpoint and a computational viewpoint. If the agents vote separately on each issue, then paradoxes generally arise [6, 13]; this rules out this 'decompositional' way of proceeding, except in the restricted case when voters have separable preferences. A second way consists in using a sequential voting protocol: variables are considered one after another, in a predefined order, and the voters know the assignment to the earlier variables before expressing their preferences on later ones (see, e.g., [14, 15, 2]). This method, however, works (reasonably) well only if we can guarantee that there exists a common order over issues such that every agent can express her preferences unambiguously on the values of each issue at the time he is asked to report them. A third class of methods consists in using a language for compact preference representation, in which the voters' preferences are stored and from which they are aggregated. If the language is expressive enough to allow for expressing any possible preference relation, then the paradoxes are avoided, but at a very high cost, both in elicitation and computation. Therefore, when organizing preference aggregation in multiple interrelated issue, there will always be a choice to be made between (a) being prone to severe paradoxes, (b) imposing a domain restriction or (c) requiring a heavy communication and computation burden.

In this paper, we explore a way along the third class of methods. When eliciting, learning, and reasoning with preferences on combinatorial domains, a domain restriction often considered consists in assuming that preferences are lexicographic. Schmitt et al. [17] address the learning of lexicographic preferences, after recalling that the psychology literature shows evidence that lexicographic

preferences are often an accurate model for human decisions [10]. Learning such preferences is considered further in [8, 18], and then in [3] who learn more generally *conditionally lexicographic preferences*, where the importance order on issues as well as the local preferences over values of issues can be conditional on the values of more important issues. The aggregation of lexicographic preferences over combinatorial domains has received very little attention (the only exception we know of is [1]). Yet it appears to be – at least in some contexts – a reasonable way of coping with multiple elections. It does imply a domain restriction, and arguably an important one; but, as explained above, domain restrictions seem to be the only way of escaping both strong paradoxes and a huge communication cost, and conditionally lexicographic preference models are not so restrictive, especially compared to the most common domain restriction, namely separability.

The generic problem of aggregating conditionally lexicographic preferences can be stated as follows. The set of alternatives is a combinatorial domain $\mathcal{X}$ composed of a finite set of binary issues.[1] We have a set of voters, each providing a conditionally lexicographic preference over $\mathcal{X}$ under the compact and natural form of a lexicographic preference tree (LP-tree for short) [3], which we will define soon; therefore, a (compactly represented) profile P consists of a collection of LP-trees. Since each LP-tree $\mathcal{L}$ is the compact representation of one linear order $\succ_{\mathcal{L}}$ over $\mathcal{X}$, there is a one-to-one correspondence between P and the (extensively represented) profile P^* consisting of a collection of linear orders over $\mathcal{X}$. Finally, for a given voting rule r, we ask whether there is a simple way to compute the winner, namely $r(P^*)$, where 'simple' means that the winner should be computed directly (and efficiently) from P and in any case we must avoid to produce P^* *in extenso*, which would require exponential space. For many cases where winner determination is computationally hard, we show that these problems can be efficiently converted to MAXSAT problems and thus be solved by sat solvers.

The rest of the paper is organized as follows. Conditionally lexicographic preferences and their compact representation by LP-trees are defined and discussed in Section 2. In Section 3 we state the problem considered in this paper, namely the aggregation of conditionally lexicographic preferences by voting rules. As we will see, some voting rules are better than others in this respect. In the paper we focus on three families of rules. First, in Section 4, k-approval rules: we show that for many values of k, we can give a quite satisfactory answer to our question above, even for our most general models. Note that by 'satisfactory' we do not necessarily mean "computable in polynomial time": for instance, when deciding whether a given alternative is a winner is NP-complete but can be easily translated into a compact maximum (weighted) satisfiability problem, for which efficient algorithms exist, we still consider the answer as (more or less) positive. In Section 5 we then focus on the Borda rule, and show that the answer to our question is satisfactory for some of the simplest LP-tree models, but less so for some general models. We also provide a natural family of scoring rules for which the answer is positive in all cases. Then in Section 6 we consider the existence of a Condorcet winner, and show that for Condorcet-consistent rules, and in particular Copeland and maximin, the answer tends to be negative. Finally, Section 7 is devoted to the specific case of LP-trees with fixed local preferences. Due to the space constraint, most proofs are omitted.

2 Conditionally Lexicographic Preferences and LP-Trees

Let $\mathcal{I} = \{X_1, \ldots, X_p\}$ ($p \geq 2$) be a set of *issues*, where each issue X_i takes a value in a binary *local domain* $D_i = \{0_i, 1_i\}$. The set of alternatives is $\mathcal{X} = D_1 \times \cdots \times D_p$, that is, an alternative is uniquely identified by its values on all issues. Alternatives are denoted by $\vec{d}$, $\vec{e}$ etc. For any $Y \subseteq \mathcal{I}$ we denote $D_Y = \prod_{X_i \in Y} D_i$. Let $L(\mathcal{X})$ denote the set of all linear orders over $\mathcal{X}$.

Lexicographic comparisons order pairs of outcomes $(\vec{d}, \vec{e})$ by looking at the attributes in sequence, according to their importance, until we reach an attribute X such that the value of X in

[1] The assumption that variables are binary is made for the sake of simplicity due to the space constraint. Most of our results would easily extend to the non-binary case.

$\vec{d}$ is different from the value of X in $\vec{e}$; $\vec{d}$ and $\vec{e}$ are then ordered according to the *local preference* relation over the values of X. For such lexicographic preference relations we need both an *importance* relation, between attributes, and *local preference* relations over the domains of the attributes. Both the importance between attributes and the local preferences may be conditioned by the values of more important attributes. Such lexicographic preference relations can be compactly represented by *Lexicographic Preference trees (LP-trees)* [3], described in the next section.

2.1 Lexicographic Preference Trees

An LP-tree $\mathcal{L}$ is composed of two parts: (1) a tree $\mathcal{T}$ where each node t is labeled by an issue, denoted by $\mathsf{Iss}(t)$, such that each issue appears once and only once on each branch; each non-leaf node either has two outgoing edges, labeled by 0 and 1 respectively, or one outgoing edge, labeled by $\{0, 1\}$. (2) A *conditional preference table* $\mathsf{CPT}(t)$ for each node t, which is defined as follows. Let $\mathsf{Anc}(t)$ denote the set of issues labeling the ancestors of t. Let $\mathsf{Inst}(t)$ (respectively, $\mathsf{NonInst}(t)$) denote the set of issues in $\mathsf{Anc}(t)$ that have two (respectively, one) outgoing edge(s). There is a set $\mathsf{Par}(t) \subseteq \mathsf{NonInst}(t)$ such that $\mathsf{CPT}(t)$ is composed of the agent's local preferences over $D_{\mathsf{Iss}(t)}$ for all valuations of $\mathsf{Par}(t)$. That is, suppose $\mathsf{Iss}(t) = X_i$, then for every valuation $\vec{u}$ of $\mathsf{Par}(t)$, there is an entry in the CPT which is either $\vec{u} : 0_i \succ 1_i$ or $\vec{u} : 1_i \succ 0_i$. For any alternative $\vec{d} \in \mathcal{X}$, we let the *importance order* of $\vec{d}$ in $\mathcal{L}$, denoted by $\mathrm{IO}(\mathcal{L}, \vec{d})$, to be the order over $\mathcal{I}$ that gives $\vec{d}$ in $\mathcal{T}$. We use $\rhd$ to denote an importance order to distinguish it from agents' preferences $\succ$ (over $\mathcal{X}$). If in $\mathcal{T}$, each vertex has no more than one child, then all alternatives have the same importance order $\rhd$, and we say that $\rhd$ is the importance order of $\mathcal{L}$.

An LP-tree $\mathcal{L}$ represents a linear order $\succ_{\mathcal{L}}$ over $\mathcal{X}$ as follows. Let $\vec{d}$ and $\vec{e}$ be two different alternatives. We start at the root node t_{root} and trace down the tree according to the values of $\vec{d}$, until we find the first node t^* such that $\vec{d}$ and $\vec{e}$ differ on $\mathsf{Iss}(t^*)$. That is, w.l.o.g. letting $\mathsf{Iss}(t_{\mathrm{root}}) = X_1$, if $d_1 \neq e_1$, then we let $t^* = t_{\mathrm{root}}$; otherwise, we follow the edge d_1 to examine the next node, etc. Once t^* is found, we let $U = \mathsf{Par}(t^*)$ and let d_U denote the sub-vector of $\vec{d}$ whose components correspond to the nodes in U. In $\mathsf{CPT}(t^*)$, if $d_U : d_{t^*} \succ e_{t^*}$, then $\vec{d} \succ_{\mathcal{L}} \vec{e}$. We use $\mathcal{L}$ and $\succ_{\mathcal{L}}$ interchangeably.

Example 1 *Suppose there are three issues. An LP-tree $\mathcal{L}$ is illustrated in Figure 1. Let t be the node at the end of the bottom branch. We have* $\mathsf{Iss}(t) = X_2$, $\mathsf{Anc}(t) = \{X_1, X_3\}$, $\mathsf{Inst}(t) = \{X_1\}$, $\mathsf{NonInst}(t) = \{X_3\}$, *and* $\mathsf{Par}(t) = \{X_3\}$. *The linear order represented by the LP-tree is* $[001 \succ 000 \succ 011 \succ 010 \succ 111 \succ 101 \succ 100 \succ 110]$, *where* 000 *is the abbreviation for* $0_1 0_2 0_3$, *etc.* $\mathrm{IO}(\mathcal{L}, 000) = [X_1 \rhd X_2 \rhd X_3]$ *and* $\mathrm{IO}(\mathcal{L}, 111) = [X_1 \rhd X_3 \rhd X_2]$.

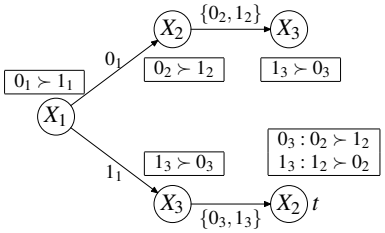

Figure 1: An LP-tree $\mathcal{L}$.

2.2 Classes of Lexicographic Preference Trees

The definition for LP-trees above is for the most general case. [3] also defined some interesting sub-classes of LP-trees by imposing a restriction on the local preference relations and/or on the conditional importance relation.

The local preference relations can be *conditional* (general case, as defined above), but can also be *unconditional* (the preference relation on the value of any issue is independent from the value of all other issues). The most restrictive case is *fixed*, which means that not only are the preferences unconditional, but that they are common to all voters. Formally, UP is the class of LP-trees with *unconditional local preferences*: for every issue X_i there exists a preference relation $\succ_i$ ($1_i \succ_i 0_i$ or $0_i \succ_i 1_i$) and for every node t with $X_i = \mathsf{Iss}(t)$, $\mathsf{Par}(t) = \emptyset$, and $\mathsf{CPT}(t) = \{\succ_i\}$. And FP is the class of LP-trees with *fixed local preferences* (FP): without loss of generality, for every node t (with $\mathsf{Iss}(t) = X_i$), $\mathsf{CPT}(t) = \{1_i \succ 0_i\}$.

Likewise, the importance relation over issues can be *conditional* (general case), or *unconditional*, of *fixed* when it is common to all voters: (UI) is the set of all linear LP-trees, i.e., every node has no more than one child. And (FI) is the set of all linear LP-trees with the (unconditional) importance order over issues $[X_1 \rhd \ldots \rhd X_p]$.

We can now combine a restriction on local preferences and a restriction on the importance relation. We thus obtain nine classes of LP-trees, namely, FI-FP, UI-FP, CI-FP, FI-UP, UI-UP, CI-UP, FI-CP, UI-CP, and CI-CP. For instance, UI-CP is defined as the class of all LP-trees with unconditional importance relation and conditional preferences. Note that the FI-FP class is trivial, as it contains a unique LP-tree.

Recall that a LP-tree is composed of a tree and a collection of conditional preference tables. The latter is reminiscent of CP-nets [4]. In fact, it can be viewed as some kind of generalized CP-net whose dependency relations between variables (induced from the importance relation) may be conditional on the values of their parent variables. However, in the case of an unconditional importance relation (UI), then the collection of CP-tables *is* a CP-net, and the LP-tree is a TCP-net [5]. In the general case however, a conditionally lexicographic preferences cannot be represented by a TCP-net.

3 Aggregating LP-trees by Voting Rules

We now consider n voters. A *(voting) profile* P over a set of alternatives $\mathcal{X}$ is a collection of n linear orders on $\mathcal{X}$. A *voting rule* r maps every profile P to a nonempty subset of $\mathcal{X}$: $r(P)$ is the set of *co-winners* for r and P.

A *scoring function* S is a mapping $L(\mathcal{X})^n \times \mathcal{X} \to \mathbb{R}$. Often, a voting rule r is defined so that $r(P)$ is the set of alternatives maximizing some scoring function S_r. In particular, *positional scoring rules* are defined via a *scoring vector* $\vec{v} = (v(1), \ldots, v(m))$, where m is the number of alternatives (here, $m = 2^p$): for any vote $V \in L(\mathcal{X})$ and any $c \in \mathcal{X}$, let $S_{\vec{v}}(V, c) = v(rank_V(c))$, where $rank_V(c)$ is the rank of c in V; then for any profile $P = (V_1, \ldots, V_n)$, let $S_{\vec{v}}(P, c) = \sum_{j=1}^{n} S_{\vec{v}}(V_j, c)$. The winner is the alternative maximizing $S_{\vec{v}}(P, \cdot)$. In particular, the k-approval rule App_k (with $k \leq m$), is defined by the scoring vector $v(1) = \cdots = v(k) = 1$ and $v(k+1) = \cdots = v(m) = 0$, the scoring function being denoted by S^k_{App}; and the *Borda* rule is defined by the scoring vector $(m-1, m-2, \ldots, 0)$, the scoring function being denoted by S_{Borda}.

An alternative α is the *Condorcet winner* for a profile P if for any $\beta \neq \alpha$, a (strict) majority of votes in P prefers α to β. A voting rule is *Condorcet-consistent* if it elects the Condorcet winner whenever one exists. Two prominent Condorcet-consistent rules are *Copeland* and *maximin*. The Copeland winners are the alternatives α that maximize the Copeland score $C(\alpha)$, defined as the number of alternatives β such that a majority of votes in P prefers α to β. The maximin winners are the alternatives α that maximize the maximin score $S_{\mathrm{MM}}(\alpha)$, defined as $S_{\mathrm{MM}}(P, \alpha) = \max\{N_P(\beta, \alpha) : \beta \in \mathcal{X}, \beta \neq \alpha\}$, where $N_P(\beta, \alpha)$ denotes the number of votes in P that rank α ahead of β.

3.1 Voting Restricted to Conditionally Lexicographic Preferences

The key problem addressed in this paper is the following. We know that applying voting rules to profiles consisting of arbitrary preferences on multi-issue domains is computationally difficult. Does it become significantly easier when we restrict to conditionally lexicographic preferences? The question, of course, may depend on the voting rule used.

A *conditionally lexicographic profile* is a collection of n conditionally lexicographic preferences over $\mathcal{X}$. As conditionally lexicographic preferences are compactly represented by LP-trees, we define a *LP-profile* P as a collection of n LP-trees. Given a class $\mathcal{C}$ of LP-trees, let us call *$\mathcal{C}$-profile* a finite collection of LP-trees in $\mathcal{C}$.

Given a LP-profile P and a voting rule r, a naive way of finding the co-winners would consists in determining the n linear orders induced by the LP-trees and then apply r to these linear orders. However, this would be very inefficient, both in space and time. We would like to know how feasible it is to compute the winners directly from the LP-trees. More specifically, we ask the following questions: (a) given a voting rule, how difficult is it to compute the co-winners (or, else, one of the co-winners) for the different classes of LP-trees? (b) for score-based rules, how difficult is it to compute the score of the co-winners? (c) is it possible to have, for some voting rules and classes of LP-trees, a compact representation of the set of co-winners?

Formally, we consider the following decision and function problems.

Definition 1 *Given a class $\mathcal{C}$ of LP-trees and a voting rule r that is the maximizer of scoring function S, in the S-*SCORE *and* EVALUATION *problems, we are given a $\mathcal{C}$-profile P and an alternative $\vec{d}$. In the S-*SCORE *problems, we are asked to compute whether $S(P, \vec{d}) > T$ for some given $T \in \mathbb{N}$. In the* EVALUATION *problem, we are asked to compute whether there exists an alternative $\vec{d}$ with $S(P, \vec{d}) > T$ for some given $T \in \mathbb{N}$. In the* WINNER *problem, we are asked to compute $r(P)$.*

When we say that WINNER for some voting rule w.r.t. some class $\mathcal{C}$ is in P, the set of winners can be compactly represented, and can be computed in polynomial time.

Note that if EVALUATION is NP-hard and the score of an alternative can be computed in polynomial time, then WINNER cannot be in P unless $\mathsf{P} = \mathsf{NP}$: if WINNER were in P, then EVALUATION could be solved in polynomial time by computing a winner and its score.

For the voting rules studied in this paper, if not mentioned specifically, EVALUATION is w.r.t. the score functions we present when defining these rules. In this paper, we only show hardness proofs, membership in NP or $\#\mathsf{P}$ is straightforward.

3.2 Two Specific Cases: Fixed Importance and Fixed Preference

It is worth focusing on the specific case of the class of profiles composed of LP-trees which have a fixed, linear structure: there is an order of importance among issues, which is common to all voters: X_1 is more important than X_2, which is itself more important than X_3, and so on.... Voters of course may have differing local preferences for the value for each issue, and their preferences on each issue may depend on the values of more important issues. A simple, easy to compute, and cheap in terms of communication, rule works as follows [14]: choose a value for X_1 according to the majority rule (possibly with a tie-breaking mechanism if we have an even number of voters); then, choose a value for X_2 using again the majority rule; and so on. The winner is called the *sequential majority winner*. When there is an odd number of voters, the sequential majority winner is the Condorcet winner (cf. Proposition 3 in [14], generalized in [7] to CI-profiles in which all voters have the same importance tree.). This, together with the fact that the sequential majority winner can be computed in polynomial time, shows that the winner of any Condorcet-consistent rule applied to FI profiles can be computed in polynomial time.

The case of fixed preferences is very specific for a simple reason: in this case, the top-ranked alternative is the same for all voters! This makes the winner determination trivial for all reasonable

voting rules. However, nontrivial problems arise if we have constraints that limit the set of feasible alternatives. We devote Section 7 to aggregating FP trees.

4 k-Approval

We start by the following lemma. Most proofs are omitted due to the space constraint.

Lemma 1 *Given a positive integer k' such that $1 \leq k' \leq 2^p$ written in binary, and an LP-tree $\mathcal{L}$, the k'-th preferred alternative of $\succ_{\mathcal{L}}$ can be computed in time $O(p)$ by Algorithm 1.*

Algorithm 1: $FindAlternative(\mathcal{L}, k')$

```
Let k* = (k*_{p-1}...k*_0)_2 = 2^p − k' and L* = L;
for i = p − 1 down to i = 0 do
    Let X_j be the root issue of L* with local preferences x_j ≻ x̄_j;
    if k*_i = 1 then
        Let L* ← L*(x_j) (the subtree of L* tracing the path X_j = x_j) and let a_j = x_j;
    end
    else Let L* ← L*(x̄_j) and let a_j = x̄_j;
end
return a⃗.
```

Similarly, the position of a given alternative $\vec{d}$ can be computed in time $O(p)$. It follows that the k-approval score of any alternative in a CI-CP profile can be computed in time $O(np)$. However, this does not mean that the winner can be computed easily, because the number of alternatives is exponential in p. For some specific values of k, though, computing the k-approval winner is in P.

Proposition 1 *Let k be a constant independent of p. When the profile is composed of n LP-trees, computing the k-approval co-winners for P can be done in time $O(knp)$.*

Proof: We compute the top k alternatives of each LP-tree in P; we store them in a table together with their k-approval score. As we have at most kn such alternatives, constructing the table takes $O(knp)$. □

A similar result also holds for computing the $(2^p - k)$-approval co-winners for any constant k.[2]

Theorem 1 (CI-CP) *For CI-CP profiles,* WINNER *for 2^{p-1}-approval can be computed in time $O(np)$.*

Proof: We note that an alternative $\vec{d}$ is among the first half of alternatives in $\mathcal{L}_j$ *iff* the root issue of $\mathcal{L}_j$ is assigned to the preferred value. We build a table with the following $2p$ entries $\{1_1, 0_1, \ldots, 1_p, 0_p\}$: for every $\mathcal{L}_j$ we add 1 to the score of 1_i (resp. 0_i) if X_i is the root issue of $\mathcal{L}_j$ and the preferred value is 1_i (resp. 0_i). When this is done, for each X_i, we instantiate X_i to 1_i (resp. 0_i) if the score of 1_i is larger than the score of 0_i (resp. vice versa); if the scores are identical, we do not instantiate X_i. We end up with a partial instantiation, whose set of models (satisfying valuations) is exactly the set of co-winners. □

Applying 2^{p-1}-approval here is both intuitive and cheap in communication (each voter only communicates her most important issue and its preferred value), and the output is computed very easily. On the other hand, it uses a very small part of the LP-trees. We may want to do better and take, say, the most important two issues into account, which comes down to using 2^{k-2}-approval or $(2^{k-1} + 2^{k-2})$-approval. However, this comes with a complexity cost. Let M be a constant independent of p and n and define $N(M, p)$ to be the set of all multiples of 2^{p-M} that are $\leq 2^p$, except 2^{p-1}. For instance, if $M = 3$ then $N(3, p) = \{2^{p-3}, 2^{p-2}, 2^{p-2}+2^{p-3}, 2^{p-1}+2^{p-3}, 2^{p-1}+2^{p-2}, 2^{p-1}+2^{p-2}+2^{p-3}\}$.

[2]However, there is little practical interest in using $2^p - k$ approval for a fixed (small) value of k, since in practice, we will have $kn \ll 2^p$, and almost every alternative will be a co-winner.

Theorem 2 (UI-UP) *For any $k \in N(M,p)$, for UI-UP profiles,* EVALUATION *for k-approval is* NP*-hard.*

Proof sketch: When $k = 2^{p-i}$ for some $i \geq 2$, the hardness of EVALUATION is proved by a reduction from the NP-complete problem MIN2SAT [12], where we are given a set Φ of clauses, each of which is the disjunction of two literals, and an integer T'. We are asked whether there exists a valuation that satisfy smaller than T' clauses in Φ. We next show the case $k = 2^{p-2}$ as an example. We note that $\vec{d}$ is among the first quarter of alternatives in $\mathcal{L}_j$ *iff* the root issue of $\mathcal{L}_j$ is assigned to the preferred value, and the second most important issue in $\mathrm{IO}(\mathcal{L}_j, \vec{d})$ is assigned to the preferred value as well. Now, we give a polynomial reduction from MIN2SAT to our problem: given a set Φ of 2-clauses, the negation $\neg C_i$ of each clause $C_i \in \Phi$ is mapped into a UI-UP LP-tree whose top quarter of alternatives satisfies $\neg C_i$ (for instance, $\neg X_3 \wedge X_4$ is mapped into a LP-tree whose two most important issues are X_3 and X_4, and their preferred values are 0_3 and 1_4). The set of co-winners is exactly the set of valuations satisfying a maximal number of clauses $\neg C_i$, or equivalently, satisfying a minimal number of clauses in Φ.

The hardness for any other k in $N(M,p)$ is proved by a reduction from special cases of the MAXSAT problem, which are omitted due to the space constraint. □

The hardness proofs carry over to more general models, namely $\{\text{UI,CI}\}\times\{\text{UP,CP}\}$. We next present an algorithm that converts winner determination for k-approval to a compact GENERALISED MAXSAT problem ("generalised" here means that the input is a set of formulas, and not necessarily clauses). The idea is, for each LP-tree $\mathcal{L}_j$, we construct a formula φ_j such that an alternative (valuation) is ranked within top k positions *iff* it satisfies φ_j. φ_j is further composed of the disjunction of multiple sub-formulas, each of which encodes a path from the root to a leaf in the tree structure, and the valuations that are ranked among top k positions.

Formally, for each path $\mathbf{u}$, we define a formula $C_{\mathbf{u}}$ that is the conjunction of literals, where there is an literal X_i (resp., $\neg X_i$) if and only if along the path $\mathbf{u}$, there is an edge marked 1_i (resp., 0_i). For any path with importance order $\mathcal{O}$ (w.l.o.g. $\mathcal{O} = X_1 \rhd X_2 \rhd \cdots \rhd X_p$) and $k = (k_{p-1} \ldots k_0)_2$ in binary, we define a formula $D_{\mathcal{O},k}$. Due to the space constraint, we only present the construction for the CI-UP case, but it can be easily extended to the CI-CP case. For each $i \leq p-1$, let $l_i = X_i$ if $1_i \succ 0_i$, and $l_i = \neg X_i$ if $0_i \succ 1_i$. Let $D_{\mathcal{O},k}$ be the disjunction of the following formulas: for every $i^* \leq p-1$ such that $k_{i^*} = 1$, there is a formula $(\bigwedge_{i>i^*:k_i=0} l_i) \wedge l_{i^*}$. To summarize, for each LP-tree $\mathcal{L}_j$ in the profile we have a formula φ_j, and we can use a (generalised) MAXSAT solver to find a valuation that maximizes the number of satisfied formulas $\{\varphi_j\}$. Note that there are efficient such solvers; see, e.g., [16] and the *Minimally Unsatisfiable Subset Track* of the 2011 Sat Competition, at http://www.satcompetition.org/2011/#tracks.

Example 2 *Let $\mathcal{L}$ denote the LP-tree in Example 1, except that the preferences for t is unconditionally $0_2 \succ 1_2$. Let $k = 5 = (101)_2$. For the upper path we have the following clause $(\neg X_1) \wedge (\neg X_1 \vee (\neg X_2 \wedge X_3))$. For the lower path we have the following formula $(X_1) \wedge (\neg X_1 \vee (X_3 \wedge \neg X_2))$.*

Theorem 3 *For any $k \leq 2^p - 1$ represented in binary and any profile P of LP-trees, there is a polynomial-size set of formulas Φ such that the set of k-approval co-winners for P is exactly the set of the models of* MAXSAT*(Φ).*

Therefore, though WINNER for k-approval is hard to compute for some cases, it can be done efficiently in practice by using a generalized MAXSAT solver.

Note that all polynomiality results for k-approval carry on to the *Bucklin* voting rule (that we do not recall): it suffices to apply k-approval dichotomously until we get the value of k for which the score of the winner is more than $\frac{n}{2}$.

Now, we focus on the specific case of fixed importance orders (FI).

Theorem 4 (FI-CP) *Let $k \in N(M,p)$. For FI-CP profiles,* WINNER *for k-approval can be computed in time $O(2^M \cdot n)$.*

Proof sketch: For simplicity, we only present the algorithm for the case $k = 2^{p-2}$. The other cases are similar. Let $X_1 > X_2 > \ldots$ be the importance order, common to all voters. There are four types of votes: those for which the 2^{p-2} top alternatives are those satisfying $\gamma_1 = X_1 \wedge X_2$ (type 1), those satisfying $\gamma_2 = X_1 \wedge \neg X_2$ (type 2), etc. Let α_i be the number of votes in P of type i ($i = 1, 2, 3, 4$). The 2^{p-2}-approval co-winners are the alternatives that satisfy γ_i such that $\alpha_i = \max\{\alpha_i, i = 1, \ldots, 4\}$. □

5 Borda

We start with a lemma that provides a convenient localized way to compute the Borda score for a given alternative in an LP-tree $\mathcal{L}$. For any $\vec{d} = (d_1, \ldots, d_p) \in \mathcal{X}$ and any $i \leq p$, we define the following notation, which is an indicator whether the i-th component of $\vec{d}$ is preferred to its negation in $\mathcal{L}$, given the rest of values in $\vec{d}$, denoted by $\vec{d}_{-i}$.

$$\Delta_i(\mathcal{L}, \vec{d}) = \begin{cases} 1 & \text{if in } \mathcal{L}, d_i \succ \overline{d_i} \text{ given } \vec{d}_{-i} \\ 0 & \text{Otherwise} \end{cases}$$

$\Delta_i(\mathcal{L}, \vec{d})$ can be computed in polynomial-time by querying the CPT of X_i along $\mathrm{IO}(\mathcal{L}, \vec{d})$. We let $\mathrm{rank}(X_i, \mathcal{L}, \vec{d})$ denote the rank of issue X_i in $\mathrm{IO}(\mathcal{L}, \vec{d})$.

Lemma 2 *For any LP-tree $\mathcal{L}$ and any alternative $\vec{d}$, we have the following calculation:*

$$S_{Borda}(\mathcal{L}, \vec{d}) = \sum_{i=1}^{p} 2^{p-rank(X_i, \mathcal{L}, \vec{d})} \cdot \Delta_i(\mathcal{L}, \vec{d})$$

Example 3 *Let $\mathcal{L}$ denote the LP-tree defined in Example 1. We have $S_{Borda}(\mathcal{L}, 011) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$ and $S_{Borda}(\mathcal{L}, 101) = 2^2 \cdot 0 + 2^0 \cdot 0 + 2^1 \cdot 1 = 2$.*

Hence, the Borda score of $\vec{d}$ for profile $P = (\mathcal{L}_1, \ldots, \mathcal{L}_n)$ is $S_{\text{Borda}}(P, \vec{d}) = \sum_{j=1}^{n} \sum_{i=1}^{p} 2^{p-\text{rank}(X_i, \mathcal{L}_j, \vec{d})} \cdot \Delta_i(\mathcal{L}_j, \vec{d})$.

Theorem 5 (CI-UP) *For CI-UP profiles,* EVALUATION *is* NP*-hard for Borda.*

Proof sketch: We prove the NP-hardness by a reduction from 3SAT. Given a 3SAT instance, we construct an EVALUATION instance, where there are $q + 2$ issues $\mathcal{I} = \{c, d\} \cup \{X_1, \ldots, X_q\}$. The clauses are encoded in the following LP-trees: for each $j \leq t$, we define an LP-tree $\mathcal{L}_j$ with the following structure. Suppose C_j contains variables $X_{i_1}, X_{i_2}, X_{i_3}$ ($i_1 < i_2 < i_3$), and $d_{i_1}, d_{i_2}, d_{i_3}$ are the valuations of the three variables that satisfy C_j. In the importance order of $\mathcal{L}_j$, the first three issues are $X_{i_1}, X_{i_2}, X_{i_3}$. The fourth issue is c and the fifth issue is d if and only if $X_{i_1} = d_{i_1}$, $X_{i_2} = d_{i_2}$, or $X_{i_2} = d_{i_2}$; otherwise the fourth issue is d and the fifth issue is c. The rest of issues are ranked in the alphabetical order (issues in C are ranked higher than issues in $\mathcal{S}$). Then, we set the threshold appropriately (details omitted due to the space constraint) such that the Borda score of an alternative is higher than the threshold if and only if its d-component is 1, and the its values for $X_1, \ldots, X_p$ satisfy all clauses. □

Finally, we show that WINNER for Borda can be converted to a weighted generalized MAXSAT problem. We note that $\Delta_i(\mathcal{L}_j, \vec{d})$ can be represented compactly by a formula φ_j^i such that a valuation $\vec{d}$ satisfies φ_j^i *iff* $\Delta_i(\mathcal{L}_j, \vec{d}) = 1$. The idea is similar to the logical formula for k-approval, where each path $\mathbf{u}$ corresponds to a clause $C_{\mathbf{u}}$, and there is another clause depicting whether $\Delta_i(\mathcal{L}_j, \vec{d}) = 1$ in $\mathbf{u}$. For example, let $\mathcal{L}$ denote the LP-tree in Example 1, then $\Delta_2(\mathcal{L}, \vec{d})$ can be presented by the disjunction of the clauses for the two paths: $\neg X_1 \wedge \neg X_2$ for the upper path, and $X_1 \wedge ((\neg X_3 \wedge \neg X_2) \vee (X_3 \wedge X_2))$ for the lower path.

Theorem 6 *For any profile P of LP-trees, there is a set of clauses Φ with weights such that the set of Borda co-winners for P is exactly the set of the models of* WEIGHTED MAXSAT*(Φ).*

Now, we focus on the specific case of unconditional importance orders (UI). When, for each $\mathcal{L}_j$ the importance order is unconditional, $\text{rank}(X_i, \mathcal{L}_j, \vec{d})$ does not depend on $\vec{d}$: let us denote it $\text{rank}(X_i, \mathcal{L}_j)$. It can be computed in polynomial time by a simple exploration of the tree $\mathcal{L}_j$.

If the preferences are unconditional, then the Borda winner is the alternative $\vec{d}$ that maximises $\sum_{i=1}^{p}\sum_{j=1}^{n} 2^{p-\text{rank}(X_i,\mathcal{L}_j)}\Delta_i(\mathcal{L}_j, \vec{d})$. We can choose in polynomial time the winning value for each issue independently: it is the d_i that maximizes

$$\sum_{j=1}^{n} 2^{p-\text{rank}(X_i,\mathcal{L}_j)}\Delta_i(\mathcal{L}_j, d_i) \quad \text{where } \Delta_i(\mathcal{L}_j, d_i) = \begin{cases} 1 \text{ if in } \mathcal{L}_j, d_i \succ \overline{d_i} \\ 0 \text{ otherwise.} \end{cases}$$

Note that this method still works if the voters have differing importance order – provided they still have unconditional importance.

Theorem 7 (UI-UP) *For UI-UP profiles,* WINNER *for Borda can be computed in polynomial time.*

However, if we allow conditional preferences, computing the Borda winner becomes intractable:

Theorem 8 (FI-CP) *For FI-CP profiles,* EVALUATION *is* NP*-hard for Borda.*

6 Condorcet-Consistent Rules

We start by studying the several classes of conditionally lexicographic preferences according to the existence of a Condorcet winner. We recall the following result from [7]:

Lemma 3 *[7] For FI-CP profiles, there always exists a Condorcet winner, and it can be computed in polynomial time.*

Proposition 2 *The existence of a Condorcet winner for our classes of conditionally lexicographic preferences is depicted on the table below, where* yes *(resp.* no*) means that the existence of a Condorcet winner is guaranteed (resp. is not guaranteed) for an odd number of voters.*

	FP	*UP*	*CP*
FI	*yes*	*yes*	*yes*
UI	*yes*	*no*	*no*
CI	*yes*	*no*	*no*

Proof: We know from [7] that for FI-CP profiles, there always exists a Condorcet winner, and it can be computed in polynomial time. For CI-FP profiles, since all voters have the same top alternative, the existence of a Condorcet winner is trivial. Finally, here is a UI-UP profile with two variables and three voters, that has no Condorcet winner:
– Voter 1: $[X \rhd Y], x \succ \bar{x}, y \succ \bar{y}$, and the linear order is $[xy \succ x\bar{y} \succ \bar{x}y \succ \bar{x}\bar{y}]$.
– Voter 2: $[Y \rhd X], \bar{x} \succ x, y \succ \bar{y}$, and the linear order is $[\bar{x}y \succ xy \succ \bar{x}\bar{y} \succ x\bar{y}]$.
– Voter 3: $[Y \rhd X], \bar{x} \succ x, \bar{y} \succ y$, and the linear order is $[\bar{x}\bar{y} \succ x\bar{y} \succ \bar{x}y \succ xy]$. □

Theorem 9 (UI-UP) *For UI-UP profiles, deciding whether a given alternative is the Condorcet winner is* coNP*-hard.*

Corollary 1 *For UI-UP profiles,* EVALUATION *for maximin is* coNP*-hard.*

7 Fixed Preferences

When the agents' local preferences are fixed (w.l.o.g. $1 \succ 0$), issues can be seen as objects, and every agent has a preference for having an object rather than not, everything else being equal. Obviously, the best outcome for every agent is $\vec{1}$, and applying any reasonable voting rule (that is, any voting rule that satisfies *unanimity*) will select this alternative, making winner determination trivial. However, winner determination ceases to be trivial if we have constraints that limit the set of feasible alternatives. For instance, we may have a maximum number of objects that we can take.

Let us start with the only tractability result in this section, with the Borda rule. Recall that, when, for each $\mathcal{L}_j$ the importance order is unconditional, $\text{rank}(X_i, \mathcal{L}_j)$ does not depend on $\vec{d}$. If, the preferences are fixed, $\Delta_i(\mathcal{L}_j, \vec{d}) = d_i$, and $S_{\text{Borda}}(P, \vec{d}) = \sum_{i=1}^{p} d_i \sum_{j=1}^{n} 2^{p-\text{rank}(X_i, \mathcal{L}_j)}$. We have the following theorem, which states that for the UI-FP case, computing the Borda winner is equivalent to computing the winner for a profile composed of importance orders, by applying some positional scoring rule. For any order $\rhd$ over $\mathcal{I}$, let $ext(\rhd)$ denote the UI-FP LP-tree whose importance order is $\rhd$.

Theorem 10 (UI-FP) *Let f_p denote the positional scoring rule over $\mathcal{I}$ with the scoring vector $(2^{p-1}, 2^{p-2}, \ldots, 0)$. For any profile $P_{\mathcal{I}}$ over $\mathcal{I}$, we have $ext(f_p(P_{\mathcal{I}})) = Borda(ext(P_{\mathcal{I}}))$.*

However, when the importance order is conditional, the Borda rule becomes intractable. We prove that using the following problem:

Definition 2 *Let voting rule r be the maximizer of scoring function S. In the K-*EVALUATION *problem, we are given a profile P that is composed of lexicographic preferences whose local preferences for all issues are $1 \succ 0$, a natural number K, and an integer T. We are asked to compute whether there exists an alternative $\vec{d}$ that takes 1 on no more than K issues and $S(P, \vec{d}) > T$.*

Theorem 11 (CI-FP) *For CI-FP profiles, K-*EVALUATION *is* NP*-hard for Borda.*

Theorem 12 (UI-FP) *Let $k \in N(M, p)$. For UI-FP profiles,* K-EVALUATION *for k-approval is* NP*-hard.*

Theorem 13 (UI-FP) *For UI-FP profiles, Copeland-*SCORE *is* #P*-hard.*

The proof is by polynomial-time counting reduction from #INDEPENDENT SET. Maximin, when the preferences are fixed (to be $1 \succ 0$ for all issues), the maximin score of $\vec{1}$ is 0 and the maximin score of any other alternative is $2^p - 1$. This trivialize the computational problem of winner determination even when with the restriction on the number of issues that take 1 (if $K \neq p$ then all available alternatives are tied). Following Lemma 3, for FI profiles, the winner can be computed in polynomial-time.

8 Summary and Future Work

Our main results are summarized in Table 1. In addition, we can also show that for k-approval (except $k = 2^{p-1}$), Copeland and maximin, there is no observation similar to Theorem 10, and the maximin score of a given alternative is APX-hard to approximate.

Our conclusions are partly positive, partly negative. On the one hand, there are voting rules for which the domain restriction to conditionally lexicographic preferences brings significant benefits: this is the case, at least, for k-approval for some values of k. The Borda rule can be applied easily provided that neither the importance relation and the local preference are unconditional, which is a very strong restriction. The hardness of checking whether an alternative is a Condorcet winner suggest that Condorcet-consistent rules appears to be hard to apply as well. However, as we have

	FP	UP	CP
FI	Trivial	P (Thm. 4)	
UI	NPC (Thm. 12)	NPC (Thm. 2)	
CI			

(a) k-approval, $k \in N(M, p)$.

	FP	UP	CP
FI	Trivial	P (Thm. 7)	NPC (Thm. 8)
UI	P		
CI	NPC (Thm. 11)	NPC (Thm. 5)	

(b) Borda.

	FP	UP	CP
FI	Trivial	Polynomial (Lemma 3)	
UI	#P-complete (Thm. 13)		
CI			

(c) Copeland score.

	FP	UP	CP
FI	Trivial	P (Lemma 3)	
UI		coNPC (Thm. 9, Coro. 1)	
CI			

(d) Maximin and Condorcet winner.

Table 1: Summary of computational complexity results.

shown that some of these problems can be reduced to a compact MAXSAT problem. From a practical point of view, it is important to test the performance of MAXSAT solvers on these problems. We believe that continuing studying preference representation and aggregation on combinatorial domains, taking advantages of developments in efficient CSP techniques, is a promising future work direction.

Acknowledgments

We thank all anonymous reviewers of AAAI-12, CP-12, and COMSOC-12 for their helpful comments and suggestions. This work has been partly supported by the project ComSoc (ANR-09-BLAN-0305-01). Lirong Xia is supported by NSF under Grant #1136996 to the Computing Research Association for the CIFellows Project.

References

[1] M. Ahlert. Aggregation of lexicographic orderings. *Homo Oeconomicus*, 25(3/4):301–317, 2008.

[2] S. Airiau, U. Endriss, U. Grandi, D. Porello, and J. Uckelman. Aggregating dependency graphs into voting agendas in multi-issue elections. In *Proceedings of IJCAI-11*, pages 18–23, 2011.

[3] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera. Learning conditionally lexicographic preference relations. In *Proceeding of ECAI-10*, pages 269–274, 2010.

[4] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191, 2004.

[5] R. I. Brafman, C. Domshlak, and S. E. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research (JAIR)*, 25:389–424, 2006.

[6] S. Brams, D. Kilgour, and W. Zwicker. The paradox of multiple elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

[7] V. Conitzer and L. Xia. Approximating common voting rules by sequential voting in multi-issue domains. In *Proceedings of KR-12*, 2012.

[8] J. Dombi, C. Imreh, and N. Vincze. Learning lexicographic orders. *European Journal of Operational Research*, 183:748–756, 2007.

[9] M. Garey and D. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[10] G. Gigerenzer and D. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

[11] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters*, 26(1):1–4, 1987.

[12] R. Kohli, R. Krishnamurti, and P. Mirchandani. The minimum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(2):275–283, 1994.

[13] D. Lacy and E. M. Niou. A problem with referendums. *Journal of Theoretical Politics*, 12(1):5–31, 2000.

[14] J. Lang and L. Xia. Sequential composition of voting rules in multi-issue domains. *Mathematical Social Sciences*, 57(3):304–324, 2009.

[15] G. D. Pozza, M. S. Pini, F. Rossi, and K. B. Venable. Multi-agent soft constraint aggregation via sequential voting. In *Proceedings of IJCAI-11*, pages 172–177, 2011.

[16] V. Ryvchin and O. Strichman. Faster extraction of high-level minimal unsatisfiable cores. In *Theory and Applications of Satisfiability Testing - SAT 2011*, volume 6695 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2011.

[17] M. Schmitt and L. Martignon. On the complexity of learning lexicographic strategies. *Journal of Machine Learning Research*, 7:55–83., 2006.

[18] F. Yaman, T. Walsh, M. Littman, and M. desJardins. Democratic approximation of lexicographic preference models. In *Proceedings of ICML-08*, pages 1200–1207, 2008.

Jérôme Lang
LAMSADE
Université Paris-Dauphine
France
Email: `lang@lamsade.dauphine.fr`

Jérôme Mengin
IRIT
Université de Toulouse
France
Email: `mengin@irit.fr`

Lirong Xia
School of Engineering and Applied Sciences
Harvard University
USA
Email: `lxia@seas.harvard.edu`