Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-019-09645-7

Voting-based ensemble learning for partial lexicographic preference forests over combinatorial domains

Xudong Liu[1] · **Miroslaw Truszczynski**[2]

Published online: 06 July 2019

Abstract
We study preference representation models based on partial lexicographic preference trees (PLP-trees). We propose to represent preference relations as forests of small PLP-trees (PLP-forests), and to use voting rules to aggregate orders represented by the individual trees into a single order to be taken as a model of the agent's preference relation. We show that when learned from examples, PLP-forests have better accuracy than single PLP-trees. We also show that the choice of a voting rule does not have a major effect on the aggregated order, thus rendering the problem of selecting the "right" rule less critical. Next, for the proposed PLP-forest preference models, we develop methods to compute optimal and near-optimal outcomes, the tasks that appear difficult for some other common preference models. Lastly, we compare our models with those based on decision trees, which brings up questions for future research.

Keywords Lexicographic preference models · Preference learning · Preference modeling and reasoning · Social choice theory · Computational complexity theory · Voting theory · Maximum satisfiability

Mathematics Subject Classification (2010) 68T30

1 Introduction

Preferences are fundamental to decision making and have been researched in areas such as knowledge representation, decision theory, social choice, and constraint satisfaction.

This is an extension of the paper that appeared in the proceedings of the 10th International Symposium on Foundations of Information and Knowledge Systems [16].

✉ Xudong Liu
xudong.liu@unf.edu

Miroslaw Truszczynski
mirek@cs.uky.edu

[1] School of Computing, University of North Florida, Jacksonville, FL 32224, USA

[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

Preferences amount to a total order or preorder on a set of *outcomes* (*alternatives*). In some settings, for instance in voting theory, the number of outcomes is small enough to allow an explicit enumeration as a method to represent preference relations. However, in other settings outcomes are specified in terms of *attributes*, each with its own *domain*, where an outcome is a tuple of values, one for each attribute. Such outcome spaces are called *combinatorial domains*. If attribute domains have at least two values, the cardinality of a combinatorial domain is exponential in the number of attributes. Consequently, explicit enumeration of preference orders, even for combinatorial domains over as few as ten attributes, is infeasible.

To represent preferences over combinatorial domains, we use languages that concisely express the agent's criteria for preferring one outcome over another, thus determining preference orders on outcomes. Languages exploiting *lexicographic orders* have been especially extensively studied. They include lexicographic strategies [19], lexicographic preference trees [3], partial lexicographic preference trees [13] (PLP-trees for short, our focus in this paper), and preference trees [7, 14]. These models naturally support preference reasoning [20, 21]. Most recently, Bräuning et al. [4] studied learning of preference lists, a model orthogonal to the model of PLP-trees. On the one hand, preference lists can capture preferences that cannot be captured by PLP-trees. On the other hand, preference lists cannot capture conditional importances that can naturally be modeled by PLP-trees.

Lexicographic preference models have structure that factors the agent's preference order into the importance, sometimes conditional, of attributes, and preference orders, also sometimes conditional, on values of individual attribute domains. This structure can be exploited for preference elicitation. It also provides useful insights into what is important for an agent when choosing among available outcomes. In particular, it makes it easy to compare outcomes (dominance testing) and to identify outcomes that are most preferred.

Based on this structure, PLP-trees [13, 15] are grouped into four classes that capture different types of orders: *unconditional importance and unconditional preference* (UIUP) trees, *unconditional importance and conditional preference* (UICP) trees, *conditional importance and unconditional preference* (CIUP) trees, and *conditional importance and conditional preference* (CICP) trees.

PLP-trees that impose strong restrictions on the structure, for instance, UIUP trees, those with unconditional importance of attributes and unconditional preference orders on values of attribute domains, can be elicited effectively from the agents. However, in general, CICP trees are difficult to elicit directly and have to be *learned*, that is, built from examples of pairwise comparisons or other observed expressions of the agent's preference [15]. In this paper, we focus on this class of unrestricted trees, namely, CICP PLP-trees. CICP trees may have size of the order of the size of the underlying combinatorial domain. Such large trees offer no advantages over explicit enumerations of preference orders. However, PLP-trees learned from a set $\mathcal{E}$ of examples have size $O(|\mathcal{E}|)$. This gives us control over the size of learned trees, but the predictive power of trees learned from small sets of examples may be limited. Learning *forests* of small trees and using some voting aggregation method was proposed as a way to circumvent the problem. Following the formalism of random forests proposed by Breiman [5], where member decision trees are aggregated by the majority rule to produce the result of the random forest,[1] Liu and Truszczynski [15] studied learning forests of PLP-trees and used the Pairwise Majority rule (PMR) to obtain a new type of a lexicographic preference model [15].

[1] In general, predictions for unseen examples are made by taking the majority classifier from all the classification trees in the random forest, or by averaging the predicitions in the case of regression trees.

There are two main problems with this last approach. First, the PMR does not (in general) yield an order. Second, it does not lead to any obvious algorithms for reasoning tasks other than dominance testing. For instance, it does not seem to lead to natural approaches to preference optimization, that is, computing optimal or near-optimal outcomes. In this paper, we extend the results by Liu and Truszczynski [15] by replacing the PMR with several common *voting rules*. Using *voting* rules to aggregate preference orders defined by lexicographic models has drawn significant attention lately. Lang and Xia [11] studied sequential voting protocols. Lang, Mengin and Xia [10] established computational properties of voting-based methods to aggregate LP-trees, and Liu and Truszczynski [12] conducted an experimental study of aggregating LP-trees by voting using SAT-based tools.

Using voting rules to aggregate forests of PLP-trees turns out to yield preference models where dominance testing is as direct as with the PMR. However, preference optimization becomes feasible, too. As there are many voting rules that could be used, and they pose different computational challenges, it is important to study whether some rules are better than others. Earlier work in the standard voting setting showed significant robustness of the aggregated order to the choice of a voting rule. Comparing several common voting rules, researchers found that, except for Plurality, these voting methods show a high consensus on the resulting aggregated preference orderings [6, 17]. Our results on rank correlation in the setting when individual preferences are represented by PLP-trees over possibly large combinatorial domains also show high consensus among orders determined by the PLP-forest models, at levels consistent with those reported for the voting setting.

As long as we are interested in dominance testing only, one can build predictive models by learning decision trees.[2] We compare the quality of learned PLP-trees and forests with those of learned decision trees. Decision trees turn out to be more accurate for dominance testing. However, they have drawbacks. Decision trees do not in general represent order nor partial order relations. They do not provide any explicit information about underlying orders and so, do not provide insights into how agents whose preferences they aim to model make decisions. Lastly, they do not lend themselves easily to tasks involving preference optimization.

To summarize, our contributions are as follows. (1) We propose to model preferences by forests of PLP-trees, aggregated by voting rules. We study the computational complexity of key reasoning tasks for the resulting models. (2) We demonstrate that the models we studied had higher predictive accuracy than the models given by a single PLP-tree, and by a PLP-forest with the PMR. (3) We show that for several voting rules the orders obtained by aggregating PLP-forests are quite close to each other. This alleviates the issue of selecting the "right" rule. (4) For the proposed PLP-forest preference models, we develop methods to compute optimal and near-optimal outcomes, the reasoning task that has no natural solutions under models based on the PMR. (5) We compare our models with those based on decision trees. We show that the latter are more accurate but, as noted above, have shortcomings in other aspects.

The higher accuracy of models based on decision trees on the dominance testing task does not invalidate PLP-tree based approaches, as they have important advantages noted above. Rather, they suggest an intriguing question of whether PLP-trees (forests) could be combined with decision trees (forests) retaining the best features of each approach. One possibility might be to use PLP-trees to some top-level partitioning of outcomes, with decision trees used for low-level details.

[2]One can also learn random forests of decision trees. In our experiments, decision trees show high accuracy and seem robust to overfitting. Thus, we do not discuss here results we obtained for random forests.

2 Partial lexicographic preference trees and forests

Let $\mathcal{A} = \{X_1, \ldots, X_p\}$ be a set of attributes, each attribute X_i having a finite domain D_i, where $|D_i|$ is bounded by a constant. The corresponding *combinatorial domain* over $\mathcal{A}$ is the Cartesian product $CD(\mathcal{A}) = D_1 \times \ldots \times D_p$. We call elements of combinatorial domains *outcomes*.

A PLP-tree over $CD(\mathcal{A})$ is an ordered labeled tree, where: (1) every non-leaf node is labeled by some attribute from $\mathcal{A}$, say X_i, and by a *local preference* $>_i$, a total strict order on the corresponding domain D_i; (2) every non-leaf node labeled by an attribute X_i has $|D_i|$ outgoing edges; (3) every leaf node is denoted by $\Box$; and (4) on every path from the root to a leaf each attribute appears *at most once* as a label.

Each outcome $\alpha \in CD(\mathcal{A})$ determines in a PLP-tree T its *outcome path*, $H(\alpha, T)$. It starts at the root of T and proceeds downward. When at a node d labeled with an attribute X, the path descends to the next level based on the value $\alpha(X)$ of the attribute X in the outcome α and on the local preference order associated with d. Namely, if $\alpha(X)$ is the i-th most preferred value in this order, the path descends to the i-th child of d. We denote by $\ell^T(\alpha)$ the index of the leaf in which the outcome path $H(\alpha, T)$ ends. (The leaves are indexed from "left" to "right" with integers 0, 1, ..., in the order their traversal.)

We say that an outcome α is at least as good as an outcome β ($\alpha \succeq_T \beta$) if $\ell^T(\alpha) \leq \ell^T(\beta)$. The associated equivalence and strict order relations $\approx_T$ and $\succ_T$ are specified by the conditions $\ell^T(\alpha) = \ell^T(\beta)$ and $\ell^T(\alpha) < \ell^T(\beta)$, respectively. Preference relations modeled by PLP-trees are total preorders.

The leaves of a PLP-tree can be indexed in time $O(s(T))$, where $s(T)$ is the number of nodes in T, by adapting the inorder traversal to the task. After that, the value $\ell^T(\alpha)$ can be computed in time $O(h(T))$, where $h(T)$ is the height of tree T. Thus, assuming the indices were precomputed, all three relations can be decided in time $O(h(T))$.

To illustrate, let us consider the domain of cars described by four multi-valued attributes. The attribute *BodyType* (B) has three values: *minivan* (v), *sedan* (s), and *sport* (r). The attribute *Make* (M) can either have value *Honda* (h) or *Ford* (f). The *Price* (P) can be *low* (l), *medium* (d), or *high* (g). Finally, *Transmission* (R) can be *automatic* (a) or *manual* (m). An agent's preference order on cars from this space could be expressed by a PLP-tree T in Fig. 1.

The tree tells us that *BodyType* is the most important attribute to the agent and that she prefers minivans, followed by sedans and by sport cars. Her next most important attribute is contingent upon what type of cars the agent is considering. For minivans, her most important attribute is *Make*, where she likes Honda more than Ford. Among sedans, her most important

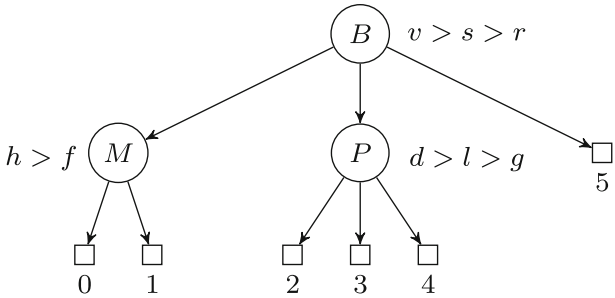


Fig. 1 A PLP-tree T over the car domain

attribute is *Price*, where she prefers medium-priced cars over low-priced ones, and those over high-priced ones. She does not differentiate between sport cars; they are least preferred.

To compare a Ford sedan with a middle-range price and an automatic transmission ($\langle s, f, d, a\rangle$, in our notation) and a Honda sedan with a high-range price and a manual transmission (that is, $\langle s, h, g, m\rangle$), we traverse the tree T. We see that the cars diverge on the node labeled by attribute P, and that the Ford car falls to leaf 2 and the Honda car leaf 4. Thus, the Ford car is preferred to the Honda car.

A *PLP-forest* is a finite set of PLP-trees. When extended with a voting rule to aggregate orders given by its constituent PLP-trees, a PLP-forest specifies a single preference order on the space of outcomes. In this way, PLP-forests with voting rules can be viewed as models of preference relations.

3 Voting in partial lexicographic preference forests

To aggregate PLP-forests we consider the voting rules Top-k Clusters, Plurality, Borda, Copeland, and Maximin. In our experiments, we also consider the earlier model of PLP-forests combined with the PMR. In general, the PMR does not yield a sensible preference relation as it suffers from the Condorcet paradox [8]. Nevertheless, it performs well in dominance testing [5, 15]. We consider it here as the baseline for the voting rules.

The five voting rules are *scoring* rules in the sense that, given a PLP-forest P, they assign to each outcome o the *score* $S_r(o, P)$ (where r refers to a voting rule). The scores define the preference relation $\succeq$ as follows: for outcomes o, o', we have $o \succeq o'$ if and only if $S_r(o, P) \geq S_r(o', P)$. Clearly, the relation defined in this way is a total preorder.[3]

The first three rules we discuss are versions of the well-known *positional scoring rules* used with total preference orders. They are adjusted here to the case of total preorders. Each tree T in a PLP-forest P determines the score $S_r(o, T)$ of an outcome o in the preference preorder given by T. This score depends on the position of the preorder "cluster" containing o, its size, and on the number of outcomes in the clusters that are more preferred than the one containing o. In each case we consider, namely, Top-k Clusters, Plurality and Borda the specific formula for $S_r(o, T)$ is a natural generalization of the corresponding formula for the standard case of total orders to total preorders. In each case, the sum of scores with respect to all trees in the forest P yields the score $S_r(o, P)$.

Below we introduce the five voting rules adjusted to the setting of total preorders (they are commonly defined for strict total orders), as well as the PMR.

Top-k clusters (Let k be a positive integer): For an outcome o, we define $S_{tkc}(o, T) = \max\{k - \ell^T(o), 0\}$ and set

$$S_{tkc}(o, P) = \sum_{T \in P} S_{tkc}(o, T).$$

Assuming that we precomputed indices of leaves in all trees, which can be accomplished in time $O(s(P))$, where $s(P)$ denotes the number of nodes in all trees in P, we can compute $S_{tkc}(o, P)$, for any outcome o, in time $O(t(P) \cdot \max\{h(T) : T \in P\})$, where $t(P)$ is the number of trees in P. We note that Top Cluster ($k = 1$) is a rule similar to approval, where

[3] While the preference models we consider here represent total preorders, arguably the most important class of preference relations, we note that some studies of preference relations allow for incomparability of outcomes, which leads to preference relations models by arbitrary preorders (not necessarily total).

each tree approves all outcomes in the leftmost cluster (and only those outcomes); and Top-k Cluster rules with $k > 1$ are its natural generalizations.

Plurality Let ℓ_0^T be the set of most preferred outcomes in a PLP-tree T (the set of all outcomes o with $\ell^T(o) = 0$). Next, let $\Delta^T(o) = 1$ if outcome o is a most preferred one in T, and $\Delta^T(o) = 0$, otherwise. We define the Plurality score $S_{pl}(o, P)$ by setting

$$S_{pl}(o, P) = \sum_{T \in P} \frac{\Delta^T(o)}{|\ell_0^T|}.$$

We can compute $\Delta^T(o)$ and $|\ell_0^T|$ in time $O(h(T))$. Thus, $S_{pl}(o, P)$ can be computed in time $O(t(P) \cdot \max\{h(T) : T \in P\})$.

Borda Let T be a PLP-tree. We define ℓ_i^T to be the set of all outcomes o with $\ell^T(o) = i$ (the i^{th} cluster in the order defined by T). Let $c(o)$ be the cluster containing o (in our notation, $c(o) = \ell^T_{\ell^T(o)}$). We define

$$S_b(o, T) = \frac{\sum\limits_{1 \le j \le |c(o)|} \left(n - j - \sum\limits_{0 \le i < \ell^T(o)} |\ell_i^T| \right)}{|c(o)|},$$

where n is the size of the combinatorial domain,[4] and set $S_b(o, P)$ as follows:

$$S_b(o, P) = \sum_{T \in P} S_b(o, T).$$

Assuming that the sizes $|\ell_i^T|$ of clusters and the quantities $\sum\limits_{0 \le i < \ell} |\ell_i^T|$) are precomputed, which can be done in time $O(s(P))$, we can compute $S_b(o, T)$ in time $O(h(T))$. Consequently, $S_b(o, P)$ can be computed in time $O(t(P) \cdot \max\{h(T) : T \in P\})$.

Copeland Let us define $N_P(o, o')$ to be the number of trees $T \in P$ such that $o \succ_T o'$. Informally, $N_P(o, o')$ is the number of trees that declare o more preferred to o'. If $N_P(o, o') > N_P(o', o)$, then o *wins* with o' in P. If $N_P(o, o') < N_P(o', o)$, then o *loses* to o' in P. The Copeland score $S_{cp}(o, P)$ is given by the difference between the number of pairwise wins and the number of pairwise losses of o:

$$\begin{aligned} S_{cp}(o, P) \; = \; & |\{o' \in C \setminus \{o\} : N_P(o, o') > N_P(o', o)\}| \\ & -|\{o' \in C \setminus \{o\} : N_P(o, o') < N_P(o', o)\}|. \end{aligned}$$

Maximin This method (also known as the Simpson-Kramer method) is considered in several variants in which the definition of the Maximin scoring function $S_{xn}(o, P)$ may include winning votes, margins, and pairwise oppositions. In this paper, we will define it in terms of the margin for an outcome, that is, the smallest difference between the numbers of pairwise wins and pairwise losses against all opponents.

$$S_{xn}(o, P) = \min_{o' \in C \setminus \{o\}} (N_P(o, o') - N_P(o', o)).$$

[4] This captures the idea that all outcomes in the top cluster in T have their score (with respect to T) equal to the average of $n-1, n-2, \ldots, n-i$ (with i being the number of outcomes in the top cluster), the outcomes in the next to top cluster have their scores equal to the average of $n-i-1, n-i-2, \ldots, n-i-j$ (with j being the number of elements in that cluster), etc.

Both the Copeland score and the Maximin score can be computed in time $O(n \cdot t(P) \cdot \max\{h(T) : T \in P\})$, where n is the size of the combinatorial domain.

Pairwise Majority Rule (PMR) The PMR is not a scoring rule. We use it to decide preferences between outcomes. Specifically, given two outcomes o and o', $o \succ_{pm} o'$ if $N_P(o, o') > N_P(o', o)$. Thus, deciding pairwise preferences takes time $O(t(P) \cdot \max\{h(T) : T \in P\})$.

4 Computational complexity

In the previous sections we listed estimates of the running time of algorithms that could be used to compute scores of the five scoring rules we consider. Here we complete the discussion by considering the complexity of the problems *SCORE*, *QUALITY*, and *OPTIMIZATION*.

SCORE (for a scoring rule r): Given a PLP-forest P, an outcome o, and a positive rational number s, decide whether $S_r(o, P) \geq s$.

QUALITY (for a scoring rule r): Given a PLP-forest P and a positive rational number ℓ, decide whether there is an outcome o such that $S_r(o, P) \geq \ell$.

OPTIMIZATION (for a scoring rule r): Given a PLP-forest P, compute an outcome with the highest score (an optimal outcome).

The picture for the rules Top-k Clusters, Plurality and Borda is complete. As we noted above, the *SCORE* problem for Borda is in the class P, and Lang et al. [10] proved that the *QUALITY* and *OPTIMIZATION* problems for Borda are NP-complete and NP-hard, respectively. The *SCORE* problem for Top-k Clusters and Plurality is in P (cf. our comments in the previous section) and the following two results show that in each case, the problems *QUALITY* and *OPTIMIZATION* are NP-complete and NP-hard, respectively.

Theorem 1 *The QUALITY problem for Top Cluster is NP-complete.*

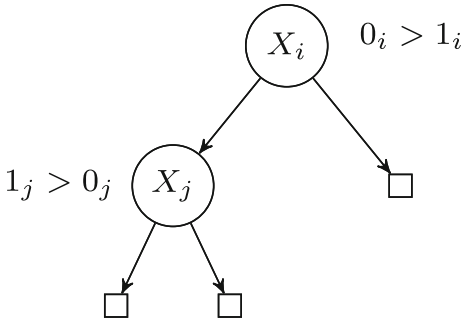
Proof Membership is obvious, as one can guess an outcome o in $O(p)$ time, and verify that $S_{tc}(o, P) \geq \ell$ in polynomial time in the size of P. Hardness is proved by reduction from MIN2SAT: Given a set Φ of n 2-clauses $\{C_1, \ldots, C_n\}$ over a set of propositional variables $\{X_1, \ldots, X_p\}$, and a positive integer g ($g \leq n$), decide whether there is a truth assignment that satisfies at most g clauses in Φ.

In the reduction, we take propositional variables X_i for the binary attributes with the domains $\{0_i, 1_i\}$. These attributes determine a combinatorial domain. Clearly, truth assignments over variables X_i and outcomes of the domain we obtain in this way are in one-to-one correspondence to each other.

Let now Φ be a collection of n 2-clauses. For each clause in Φ, say $C = X_i \vee \neg X_j$, we create a PLP-tree T_C in Fig. 2 (the form of the tree for other types of 2-clauses is evident from the example we selected for illustration; negated variables in a clause require that 1 is locally preferred to 0 and non-negated variables require that 0 is locally preferred to 1). We also set $\ell = n - g$.

A truth assignment v falsifies a clause C in Φ if and only if, when viewed as an outcome, it belongs to the top cluster of the corresponding tree T_C. Clearly, there is an assignment

Fig. 2 PLP-tree



satisfying at most g clauses of Π if and only if there is an assignment that falsifies at least ℓ clauses in Φ. The latter is equivalent to the existence of an outcome that belongs to the top cluster of at least ℓ trees T_C, $C \in \Phi$, that is, an outcome v such that $S_{tc}(v, P) \geq \ell$. □

The case of the Top-k Cluster rule with $k > 1$ is dealt with in the next theorem.

Theorem 2 *The QUALITY problem for Top-k Clusters, where $k > 1$, is NP-complete.*

Proof The membership in NP is evident. To prove NP-hardness for when $k > 1$, we again reduce from the MIN2SAT problem, but the construction is different. As before, we treat propositional variables as binary attributes of a combinatorial domain, and we identify truth assignments with outcomes.

For every clause $C \in \Phi$, say $C = X_i \vee \neg X_j \in \Phi$, we construct a set P_C of three PLP-trees shown in Fig. 3 (as in the previous proof, the construction is evident from the example we selected here for illustration).

We note that if an assignment (outcome) v satisfies a clause $C \in \Phi$, then, we have $S_{tkc}(v, P_C) = 3 \cdot k - 4$; otherwise, we have $S_{tkc}(v, P_C) = 3 \cdot k - 3$. We now set $P = \bigcup_{C \in \Phi} P_C$ and $\ell = 3n(k-1) - g$. We need to show that assignment v satisfies at most g clauses in Φ if and only if v scores at least ℓ in P according to the Top-k Clusters rule.

Let v be an assignment (outcome) satisfying g' clauses in Φ. Then, $S_{tkc}(v, P) = g'(3 \cdot k - 4) + (n - g')(3 \cdot k - 3) = 3n(k-1) - g' = \ell + g - g'$. It is clear now that for an assignment (outcome) v, $S_{tkc}(v, P) \geq \ell$ if and only if v satisfies no more than g clauses. Thus, NP-hardness follows. □

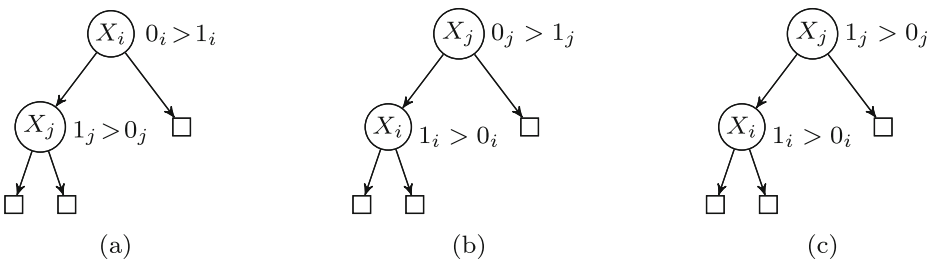


Fig. 3 Set P_C of PLP-trees for clause $C = X_i \vee \neg X_j$

Theorem 3 *The QUALITY problem for Plurality is NP-complete.*

Proof The membership in NP is clear. The NP-hardness can be proved by reduction from MIN2SAT. For each clause in Φ, we create a PLP-tree T_C as in the proof of Theorem 1, and we set $\ell = (n - g)/2^{p-2}$. One can show that there is a truth assignment satisfying at most g clauses in Φ if and only if there exists an outcome whose Plurality score is at least ℓ. □

Corollary 1 *The OPTIMIZATION problems for Top-k Cluster, $k \geq 1$, and for Plurality are NP-hard.*

Proof The claim follows as we can use an algorithm to solve the *OPTIMIZATION* problem for one of the rules in the assertion to decide the *QUALITY* version of the problem. Indeed, once the optimal outcome is known, we can compute its score and compare it to ℓ. If the score is ℓ or more, we return YES, otherwise, we return NO. □

The *SCORE*, *QUALITY* and *OPTIMIZATION* problems for Copeland and Maximin were studied by Lang et al. [10]. These results are partial and not tight. We studied the complexity of these problems for the scoring rules Top-k Clusters, Plurality and Borda. Our results are complete and tight. We summarize all these results in Table 1. Completing the complexity picture for Copeland and Maximin remains a challenging open problem.

5 Experiments and results

PLP-trees and forests are difficult to elicit from users directly. In practical settings they have to be learned from *examples*, that is, pairs (o, o') of outcomes, where o is strictly preferred to o' in the preference order we are trying to elicit (model). A method to learn PLP-trees was proposed by Liu and Truszczynski [15]. They also applied it to learn PLP-forests and aggregate them with the PMR. In this paper, we extend this work to the case when learned PLP-forests (forests of learned PLP-trees) are aggregated by means of voting rules.

In our main results, we evaluate the ability of PLP-forests extended with voting rules to approximate preference orders arising in practical settings. Further, we compare in this respect PLP-forest models with models based on decision trees, develop for PLP-forests effective techniques to compute optimal or near optimal outcomes, and study the effect of the choice of a specific voting rule on the quality of the preference model.

Table 1 Computational complexity results

	SCORE	*QUALITY*	*OPTIMIZATION*
Top-k Clusters	P	NPC (Theorems 1 and 2)	NPH (Corollary 1)
Plurality	P	NPC (Theorem 3)	NPH (Corollary 1)
Borda	P	NPC (cf. [10])	NPH (cf. [10])
Copeland	#PH (cf. [10])	?	?
Maximin	?	coNPH (cf. [10])	coNPH (cf. [10])

Table 2 Preference datasets in the preference learning library

Dataset	#Attributes	#Outcomes	#Examples
Breast Cancer Wisconsin (BCW)	9	270	9,009
Car Evaluation (CE)	6	1,728	682,721
Credit Approval (CA)	10	520	66,079
German Credit (GC)	10	914	172,368
Ionosphere (IN)	10	118	3,472
Mammographic Mass (MM)	5	62	792
Mushroom (MS)	10	184	8,448
Nursery (NS)	8	1,266	548,064
SPECT Heart (SH)	10	115	3,196
Tic Tac Toe (TTT)	9	958	207,832
Vehicle (VH)	10	455	76,713
Wine (WN)	10	177	10,322

5.1 Datasets, experimental set-up, and the greedy heuristic

We implemented the scoring rules discussed above as order aggregators for PLP-forests and experimented with them on the twelve preferential datasets used before by Liu and Truszczynski [15].[5] Their key characteristics are given in Table 2. The third column gives the number of pairs of outcomes from the corresponding domain with the first outcome being strictly better than the second one. As mentioned earlier, we refer to such pairs as examples. These datasets were generated from classification datasets, where every outcome is given a label value. When the values were reals, they were discretized, and examples were created once a total order was imposed on the discrete (and finete) set of labels.

The PLP-forest learning procedure works as follows. For each of the datasets, we randomly partition the set of examples $\mathcal{E}$, generating a training set of 70% of $\mathcal{E}$ and use the rest 30% as the testing set. In the training phase, we use the *greedy learning heuristic* [15] to learn a PLP-forest of a given number of PLP-trees, each of which is learned from M (a parameter) examples selected with replacement and uniformly at random from the training set. In the testing phase, the trees in the learned PLP-forest are aggregated using the seven voting methods, Top Cluster, Top-2 Clusters, Top-3 Clusters, Plurality, Borda, Copeland and Maximin, to predict testing examples and to compute the social welfare rankings. We repeat this procedure 20 times for each dataset.

We now describe the greedy heuristic as procedure *buildPLPTree* illustrated in Algorithm 1. It utilizes a recursive procedure *buildPLPTreeRecu*, in Algorithm 2, to learn a single (CICP) PLP-tree for a given set of examples over a set of attributes. (We refer to our previous paper [15] for the variations of the algorithm learning other types of PLP-trees.)

The *buildPLPTree* procedure creates an unlabeled node, builds configuration $(\mathcal{E}, \mathcal{A}, T)$ and pushes it to stack K, and invoke *buildPLPTreeRecur* to construct a PLP-tree. In each call of the recursive algorithm *buildPLPTreeRecur*, the algorithm removes one item $(\mathcal{E}, \mathcal{A}, n)$

[5] The datasets are available at http://www.unf.edu/~N01237497/preflearnlib.html

from the stack C.[6] The algorithm picks an attribute X_l and a preference order for X_l at n to maximize the number of examples in $\mathcal{E}$ *correctly* decided at this node. The algorithm updates $\mathcal{E}$ by removing all examples decided at n (correctly or incorrectly), and $\mathcal{A}$ by removing X_l. At this point, both outcomes of every example in $\mathcal{E}$ have the same value on X_l (otherwise, the example would be decided, correctly or not, by X_l). For each value $x_{l,i}$ of X_l, the algorithm creates a node n_i, makes n the parent of n_i, and sets $\mathcal{E}_i$ to consist of all examples in $\mathcal{E}$ whose both outcomes have value $x_{l,i}$ on X_l. It then adds $(\mathcal{E}_i, \mathcal{A}, n_i)$ to K. When the new stack K is computed, the algorithm starts the next recursion. Configurations with the empty set of examples become leaves and the process ends when there are no more configurations in the stack to process.

Algorithm 1 *buildPLPTree*$(\mathcal{E}, \mathcal{A})$ % learns a PLP-tree.

Input: $\mathcal{E}$ is the set of strict examples, $\mathcal{A}$ the set of available attributes
Output: A PLP-tree T over $\mathcal{A}$.
Procedure *buildPLPTree*$(\mathcal{E}, \mathcal{A})$*:*
  Create an unlabeled node T Push configuration $(\mathcal{E}, \mathcal{A}, T)$ to stack K
  buildPLPTreeRecur(K) **return** T

Algorithm 2 is a recursive procedure, where each recursion learns an attribute labeling node n using the available $\mathcal{A}$ and $\mathcal{E}$, and this recursion takes $O(|\mathcal{A}| \cdot |\mathcal{E}|)$ time. As aforementioned, the size of the learned tree is $O(|\mathcal{E}|)$. Therefore, algorithm *buildPLPTreeRecur* takes $O(|\mathcal{A}| \cdot |\mathcal{E}|^2)$ time, polynomial in $|\mathcal{A}|$ and $|\mathcal{E}|$. Experimentally, the greedy heuristic acheives an accuracy of 86.1% averaged over all datasets in the Preference Learning Library, with more than 90% accuracy for over half of them. These results, obtained for the CICP PLP-trees are considerably better than those for the other classes of trees. We refer to our earlier work [15] for detailed discussions on the performance of the greedy heuristic.

Algorithm 2 *buildPLPTreeRecur*(K) % the procedure used in Algorithm 1.

Input: K: a stack of configurations $(\mathcal{E}, \mathcal{A}, n)$, where $\mathcal{E}$ is the set of strict examples in the form of (α, β), $\mathcal{A}$ the set of available attributes, n an unlabeled node to consider next.
Procedure *buildPLPTreeRecur(K):*
  $(\mathcal{E}, \mathcal{A}, n) \leftarrow$ Pop an item from K **if** $\mathcal{E} = \emptyset$ **then**
    Label n as a leaf **if** K *is empty* **then**
      return
    end
  else
    $(X_l, \succ_l) \leftarrow$ Pick $X_l \in \mathcal{A}$ and $\succ_l$ that correctly decides the maximum number of examples in $\mathcal{E}$ Label n with tuple $(X_l, CPT(X_l))$
    $\mathcal{E} \leftarrow \mathcal{E} \backslash \{e \in \mathcal{E} : \alpha_e(X_l) \neq \beta_e(X_l)\}$ $\mathcal{A} \leftarrow \mathcal{A} \backslash \{X_l\}$ **for** $i \leftarrow 1$ *to* $|D_l|$ **do**
      Create an edge u_i and an unlabeled node n_i such that $u_i = \langle n, n_i \rangle$
      $\mathcal{E}_i \leftarrow \{e \in \mathcal{E} : \alpha_e(X_l) = \beta_e(X_l) = x_{l,i}\}$ Push item $(\mathcal{E}_i, \mathcal{A}, n_i)$ onto K
    end
  end
  buildPLPTreeRecur(K)

[6] Clearly, the stack makes the algorithm compute in a depth-first manner. We could use a different type of container, e.g., a queue, for a breadth-first manner computation. The result, however, will stay same, because $\mathcal{E}$ and $\mathcal{A}$ at a node do not change whether it is visited earlier or later by the learning algorithm.

The following three subsections present our experimental results. The first one concerns the task of predicting new preferences. For this task, we compute and report average *accuracy* results, where the accuracy is defined as the number of examples in the testing set that are in agreement with the learned model divided by the size of the testing set.

In the next subsection, we discuss computing optimal outcomes for PLP-forests using the Top-k Clusters rules. We show that the problem can be reduced to the weighted partial MAXSAT problem [2]. This allows one to use MAXSAT solvers to solve them. The specific MAXSAT solver we used in this work was *toulbar*2 [9].

Finally, we consider the effect of the choice of a scoring rule on the preference order. To this end, we calculate the Spearman's rho [17] for Top Cluster, Top-2 Clusters, Top-3 Clusters, Plurality, Borda and Maximin, all against Copeland. This allows us to quantify the similarity between orders generated by different rules.

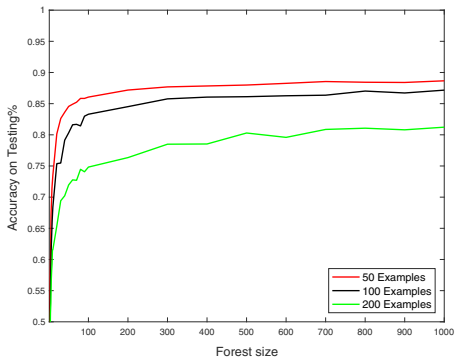
5.2 Preference prediction results

We focus on PLP-forests of trees learned from *small* sets of examples. This supports fast learning and leads to small constituent PLP-trees. In our experiments we learned PLP-trees from samples of 50, 100 and 200 examples. The results, averaged over all datasets for the Top-2 Clusters rule, are shown in Fig. 4a. They show that the testing accuracy is better when smaller PLP-trees are learned. We saw similar behavior for other scoring rules and so omitted the results from Fig. 4a. We attribute this to the fact that trees learned from these smaller samples of examples generated randomly out of the training set tend to be more diverse with different structures of attributes and different local preference orders at the attribute level. This diversity of trees becomes beneficial at the testing phase. Based on these experiments, we now restrict our discussion to PLP-forests with trees learned from samples of size 50.

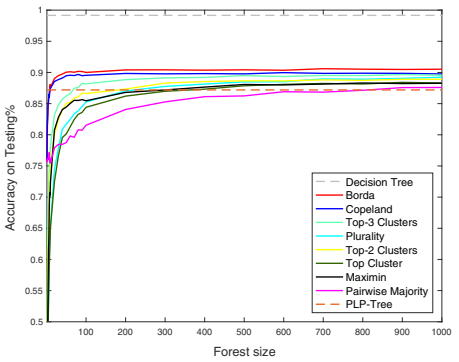
In Fig. 4b, we present the mean learning curves over all datasets for all 8 rules, where each curve shows how testing results (accuracy percentages) change with the PLP-forest size (the number of trees in the forest). We also show there the results for learning a single PLP-tree and a single decision tree using the whole training set (70% of $\mathcal{E}$). Decision trees in our experiments are classification trees trained using labeled instances, where an instance consists of two outcomes and has a binary label, 1 (0) indicating the first outcome is (is not, resp.) strictly preferred to the second. Given a decision tree D and two outcomes o, o', the dominance testing query asks if it is true that o is strictly preferred to o' in D. In testing, to answer such a query for two outcomes, they are input into a decision tree. If the tree predicts 1, we answer *yes* to the query; otherwise, *no*.

First, we observe that independently of the rule used the PLP-forest models across all datasets outperform the single PLP-tree model. This is most notable for the Borda rule, with a 4% improvement from 87% for single PLP-trees to 91% for PLP-forests. Pairwise Majority used by Liu and Truszczynski [15] turns out to be the worst aggregating method overall.

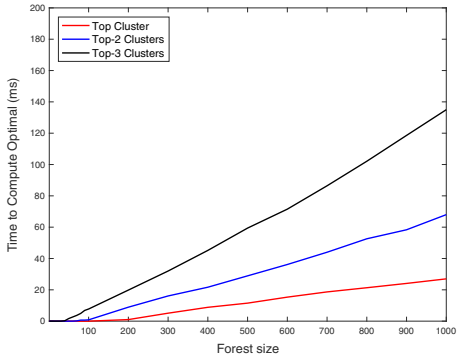
Moreover, looking at the results for 1000-tree forests, we see that, these forests have high accuracy of about 89∼91%, on the testing datasets, depending on the voting rule. This provides strong evidence for the adequacy of the PLP-forest model to represent user preferences over practical combinatorial domains. This also demonstrates that the differences between these voting rules in predicting new preferences are not significant. In particular, the Top-3 Clusters rule leads to accuracy of 90%, only a percentile point difference from Borda.



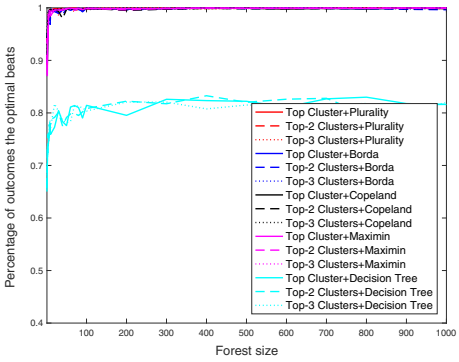
(a) Mean testing accuracy across all datasets for PLP-forests applying Top-2 Clusters, where every member tree is trained using random samples of sizes 50, 100 and 200.



(b) Mean testing accuracy across all datasets for decision trees, PLP-forests applying voting rules and PLP-trees.



(c) Mean time across all datasets computing optimal outcomes using Toulbar2 for PLP-forests applying Top Cluster, Top-2 Clusters and Top-3 Clusters.



(d) Mean results across all datasets evaluating optimal outcomes for PLP-forests applying Top Cluster, Top-2 Clusters and Top-3 Clusters, against other voting rules and decision trees.

Fig. 4 Preference learning and optimization results for PLP-forests

Our results also show that decision trees perform better on our datasets with accuracy of about 99%. We attribute the near-perfect performance of the decision-tree model to their large size (cf. Table 3) that enables classifying with high-granularity whether one outcome is preferred to another. However, the decision-tree model has drawbacks. It does not guarantee that the relation it determines is an order or a partial order, it does not offer clear

Table 3 Size comparison of PLP-trees and decision trees learned from the training data (70% of $\mathcal{E}$)

Dataset	BCW	CE	CA	GC	IN	MM	MS	NS	SH	TTT	VH	WN
Decision tree	188.8	683.0	897.2	2003.0	116.0	58	27.4	681.0	73.6	564.2	1549.0	36.2
PLP-tree	25.7	109.5	81.1	190.0	30.6	10.0	16.3	116.9	19.0	115.2	105.4	14.6
Ratio	7.6	6.2	11.1	10.5	3.8	5.8	1.7	5.8	3.9	4.9	14.7	2.5

explanations to what factors affect comparisons, and it does not support computing optimal and near-optimal outcomes. In each of these aspects PLP-forest models have an advantage.

5.3 Preference optimization results

PLP-forests with scoring rules allow for effective optimal outcome computation. We show it here for orders obtained by using Top, Top-2 and Top-3 Clusters rule to aggregate orders defined by individual PLP-trees in a PLP-forest.

For every dataset, we learn PLP-forests of up to 1000 PLP-trees. To compute the optimal outcome in each forest (under Top, Top-2 or Top-3 Clusters rule), we encode the problem as an instance of the weighted partial MAXSAT problem [2] and use *toulbar*2 [9] to solve it. A weighted partial MAXSAT instance Φ consists of two parts Φ_h and Φ_s, where Φ_h, called *hard constraints*, is a collection of clauses, and Φ_s, called *soft constraints*, is a collection of weighted clauses. If Φ_h is over-constrained and thus unsatisfiable, there is no solution to Φ. Otherwise, the solution to Φ is a truth assignment v that satisfies Φ_h and maximizes the sum of the weights of the clauses satisfied by v. We now briefly discuss the encoding for the case of binary attributes, which extends in a straightforward way to the general case of multi-valued attributes.

The encoding consists of two main steps and assumes that we are using the Top-k Clusters rule for aggregation. First, given a PLP-forest $P = \{T_1, \ldots, T_m\}$, we build a collection $\Psi = \{(B_0^1, k), \ldots, (B_{k-1}^1, 1), \ldots, (B_0^m, k), \ldots, (B_{k-1}^m, 1)\}$ of term-weight pairs. Each term B_j^i is the conjunction of literals x or $\neg x$, where x's are the names of the attributes labeling the nodes on the path in the tree T_i from its root to the leaf j. If the path follows to the left child of the node labeled by x, the term B_j^i includes the literal x, otherwise, it includes the literal $\neg x$. This collection of terms can be built in time that is linear in the size of the input. Clearly, an outcome o's score $S_{tkc}(o, P)$ for Top-k Clusters rule is equal to the sum of weights of the terms in Ψ satisfied by o. Hence, the winning outcome for P with respect to the Top-k Clusters rule is precisely the truth assignment with the maximum sum of weights of those terms in Ψ that it satisfies (and conversely).

Second, we translate Ψ to an equivalent weighted partial MAXSAT instance Φ of two parts Φ_s and Φ_h. Given $\Psi = \{(B_1, w_1), \ldots, (B_N, w_N)\}$, we build in linear time $\Phi_s = \{(c_1, w_1), \ldots, (c_N, w_N)\}$ where every c_i is a new atom, and $\Phi_h = \{CNF(B_i \leftrightarrow c_i) : (B_i, w_i) \in \Psi\}$ where *CNF* denotes the set of clauses of a given formula. Now, we show that the truth assignment with the maximum sum of weights of satisfied terms in Ψ is precisely the truth assignment that satisfies all clauses in Φ_h and for which the sum of weights of clauses in Φ_s that it satisfies is maximum (and the converse holds, too).

Theorem 4 *The truth assignment with the maximum sum of weights of satisfied terms in Ψ is the truth assignment that satisfies all clauses in Φ_h and for which the sum of weights of clauses in Φ_s that it satisfies is maximum.*

Proof Take an arbitrary truth assignment of Ψ, v, that satisfies terms $B_{s_1}, \ldots, B_{s_e}$ and falsifies terms $B_{t_1}, \ldots, B_{t_r}$, where $e+r = N$. We now have the sum of weights of v for Ψ being $S(v, \Psi) = w_{s_1} + \ldots + w_{s_e}$. Then, for Φ we can build a truth assignment v' that is an extension of v by setting variables $c_{s_1}, \ldots, c_{s_e}$ to true and $c_{t_1}, \ldots, c_{t_r}$ to false. Clearly, v' satisfies all the biconditionals in $\Phi_h = \{CNF(B_i \leftrightarrow c_i) : (B_i, w_i) \in \Psi\}$. That is, v' satisfies clauses $c_{s_1}, \ldots, c_{s_e}$ and falsifies $c_{t_1}, \ldots, c_{t_r}$. We have that $S(v', \Phi_s) = w_{s_1} + \ldots + c_{s_e} = S(v, \Psi)$.

Thus, the truth assignment with the maximum sum of weights of satisfied terms in Ψ is the truth assignment that satisfies all clauses in Φ_h and for which the sum of weights of clauses in Φ_s that it satisfies is maximum. □

Average computational time spent on searching for optimal outcomes for all datasets is shown in Fig. 4c. We see that, for any dataset and for any forest size up to 1000, a weighted partial MAXSAT instance encoding preference optimization can be solved within 0.2 sec.

The reductions are straightforward for the Top-k Clusters rules. It is not clear how to extend them to other scoring rules. Instead, we show that optimal outcomes computed for the three of the Top-k Clusters rules are close to optimal for orders obtained for other rules. Specifically, for every optimal outcome computed based on Top-k Clusters rule ($k = 1, 2, 3$) and every dataset, we randomly select 1000 outcomes and check how well the optimal outcome compares to them, when other voting rules (Plurality, Borda, Copeland and Maximin) are used. Average percentiles of the number of outcomes "beaten" by the optimal one are shown in Fig. 4d. We note that, when forests are big, the optimal outcomes based on the three Top-k Clusters rules are either very likely optimal (when the percentiles are exactly 100%) or very likely near-optimal (when the percentiles are not 100% but very close to), for all other voting rules. This is desirable because it shows that computing optimal outcomes for orders determined by Top-k Clusters rules, which we demonstrated to be computationally feasible, are likely optimal or near-optimal for rules where methods to optimize preferences are not straightforward. For decision trees, the results show that outcomes optimal for Top-k Rules are further from optimal but still within the top 20% of outcomes according to the decision-tree model.

5.4 Rank correlation results

In the standard voting setting, the rankings generated by different voting rules are quite close to each other [6, 17]. For the setting of combinatorial domain setting, when preference orders are given as PLP-forests (with scoring rules as aggregators), the results we discussed in the previous sections suggest that here, too, the choice of a voting rule does not affect the order significantly (all rules result in models of similar accuracy and outcomes highly preferred for one rule are highly preferred for other).

Specifically, we empirically studied the correlation to orders determined by the Copeland rule of orders determined by the other scoring rules we studied. As suggested in previous work on measuring rank correlation [6, 17, 18], we used the Spearman's rho (denoted by ρ) as the rank correlation coefficient. Given two total orders L_1 and L_2 of outcomes in C, we define

$$\rho(L_1, L_2) = 1 - \frac{6 \cdot \sum\limits_{1 \le i \le n} (i - D_2(L_1(i)))^2}{n \cdot (n^2 - 1)},$$

where i is the rank value between 1 and n, and $D_2(o)$ is the rank of outcome o in L_2. The value of $\rho(L_1, L_2)$ is in between -1 and 1, both inclusive. When L_1 and L_2 order C exactly the same, we have $\rho(L_1, L_2) = 1$. If $\rho(L_1, L_2) = -1$, it means L_1 and L_2 reversely order C. Furthermore, the closer the value to 0, the weaker the correlation between L_1 and L_2.

Our results (cf. Table 4) suggest that Borda-generated orders have a very high degree of consensus with those generated by Copeland, and that Plurality and Top Cluster lead to orders with the lowest degrees of agreement. Nevertheless, in all cases the Spearman's rho has high values, similar to those obtained for strict preference orders over non-combinatorial domains with few outcomes [6, 17, 18].

Table 4 Mean and standard deviation of the Spearman's rho results for voting rules against Copeland across all datasets in learning PLP-forests of size 1000

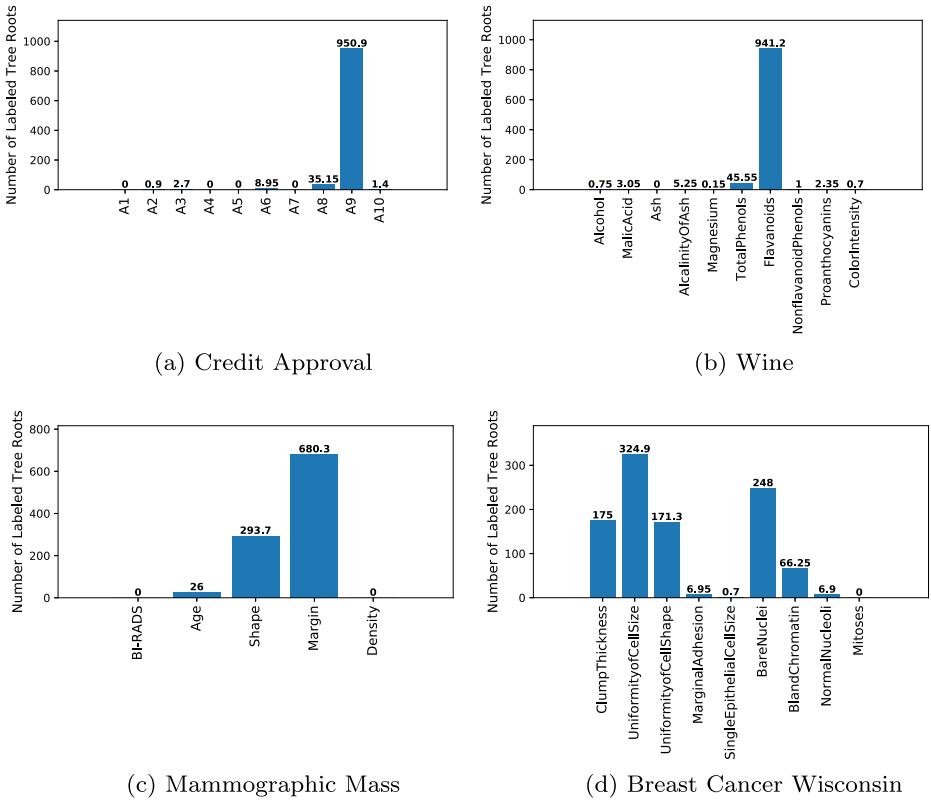
Dataset	Borda	Top-3	Top-2	Maximin	Plurality	Top
BCW	0.9616	0.8405	0.8456	0.8310	0.8288	0.8196
CE	0.7741	0.5532	0.5384	0.4716	0.4591	0.4665
CA	0.9847	0.9413	0.9354	0.9435	0.9260	0.9277
GC	0.9898	0.9159	0.9108	0.9165	0.9088	0.8742
IN	0.9240	0.9867	0.9786	0.9823	0.9763	0.9706
MM	0.8678	0.9331	0.9234	0.9109	0.9005	0.9054
MS	0.9712	0.9353	0.9115	0.9433	0.9025	0.8898
NS	0.9939	0.9908	0.9851	0.9766	0.9768	0.9806
SH	0.9729	0.9968	0.9810	0.9786	0.9747	0.9785
TTT	0.9900	0.9916	0.9841	0.9931	0.9847	0.9531
VH	0.9691	0.8612	0.8346	0.8588	0.8573	0.7958
WN	0.9937	0.9740	0.9266	0.9101	0.9090	0.9015
Mean	0.9494	0.9100	0.8963	0.8930	0.8837	0.8719
SD	0.0632	0.1181	0.1182	0.1358	0.1364	0.1347

5.5 Insights from learned forests

A PLP-tree specifies a total preorder over outcomes by means of the tree structure of labeled nodes and their local preferences. The PLP-tree partitions the universe of outcomes starting from its root, labeled by the most important attribute. A frequency with which an attribute appears as the most important one (or one of the two most important) can provide useful insights into what determines the quality of outcomes and what affects the results of direct comparisons between outcomes.

To see these distributions for our datasets, we set up experiments to learn forests of 1000 PLP-trees, each learned from 50 examples, selected with replacement and uniformly at random from the training set. This process is repeated 20 times and we present the averages for selected datasets in Fig. 5. In Fig. 5a, for the Credit Approval dataset, we see that over 95% of trees have the attribute A9 as the most important one, and so we could conclude that this is indeed a critical property when deciding on denying or approving credit applications.[7] Similarly, over 94% of the trees for comparing wines that we built based on the Wine dataset (cf. Fig. 5b) has Flavanoids at the root. This suggests that this is the key characteristics for the quality of a wine. For some other datasets, there is no single dominant attribute. Still the frequency of serving as the root attribute sheds light on what is important. For example, Fig. 5c shows that attributes Margin and Shape are critical for deciding the severity of mammographic masses. In other cases, the picture may be less clear. For instance for the Breast Cancer dataset, Fig. 5d, several attributes appear with high frequency as the roots. However, even in such cases, the results suggest a partition of attributes into two or more groups of varying importance.

[7] We derived this data set from the Credit Approval data set available at https://archive.ics.uci.edu/ml/datasets/Credit+Approval. That source does not provide information of what A9 or other attribute symbols stand for. However, in any real-life scenario, as well as for some other data sets we discuss here, attributes have a specific and known meaning corresponding to key properties pertinent to objects in the domain.



(a) Credit Approval (b) Wine

(c) Mammographic Mass (d) Breast Cancer Wisconsin

Fig. 5 Distributions of domain attributes to PLP-tree roots in learned PLP-forests of size 1000

6 Conclusions and future work

We proposed to use PLP-forests extended with a voting rule as a model of preference relations. We considered five voting rules, Top-k Clusters, Borda, Plurality, Copeland and Maximin, all adjusted to the case of total preorders. We studied the complexity of three key preference reasoning problems arising in this setting: *SCORE*, *QUALITY* and *OPTIMIZATION*. For Top-k Clusters, Borda and Plurality, our results, together with those obtained earlier in the literature, provide a complete picture. In all cases, the *SCORE* problem is in P, the *QUALITY* problem is NP-complete and the *OPTIMIZATION* problem is NP-hard. For the Copeland and Maximin rules, investigated by Lang at al. [10], only partial results are known. However, they suggest the two rules may be more demanding computationally.

We studied our PLP-forest models experimentally. Our results showed that using these voting rules for preferential datasets generated from real-world classification datasets yields models reflecting underlying preference relations with high accuracy, exceeding that of PLP-forest models utilizing the Pairwise Majority rule.

We also studied the correlation among the orders given by different PLP-forest models, extending to the setting of "votes" over combinatorial domains several earlier studies in the standard voting setting with a small number of alternatives. We found that when compared to the model given by PLP-forests with Copeland as an aggregator, all models showed high levels of correlation, similar to those reported in the literature for the standard voting setting.

Our results suggest that using rules such as Borda or Top-3 clusters (the two closest to Copeland) produces orders representative for all those that can be obtained by combining a PLP-forest with other scoring rules.

For the Top-k Clusters rule, we developed methods to compute optimal outcomes for orders they determine given a PLP-forest. Our experiments for when $k = 1, 2, 3$ showed that the methods are computationally feasible. They also show that optimal outcomes computed for the Top-k Clusters rules are near optimal for orders determined by all other scoring rules.

Our results suggest that PLP-forest preference models with scoring rules as aggregators, especially Top-k Clusters and Borda, have many attractive features. They can be learned so that to reflect underlying true preference relation with high accuracy. They well represent orders that result from using other scoring rules. Lastly, they support fast methods for computing optimal outcomes and these outcomes are likely to be near optimal for orders given by other scoring rules.

We also compare our PLP-forest models with the decision tree approach. The decision trees learned from examples can approximate underlying orders with higher accuracy (as high as 99% in our experiments). However, they do have drawbacks not present in PLP-forest models. First, unlike in the case of the PLP-forests, the relation defined by decision trees is not guaranteed to be a total order (not even a partial order). Second, decision trees do not provide any clear insights into key factors determining the underlying preference relations. In contrast, the PLP-tree and PLP-forest models yield information about attributes most significant for determining the preference order. For the PLP-tree model, it is the attribute that labels the root, for the PLP-forest model, the attributes appearing most frequently as the labels of the roots of its trees. Lastly they do not offer effective ways to solve optimization tasks (finding optimal or near optimal outcomes) while, as we show, PLP-tree and forest models do. These drawbacks of decision trees make PLP-forests, despite their lower accuracy, an attractive preference model for use in applications.

Our results provide evidence of low effect of the choice of a voting rule when aggregating preference orders determined by trees in a PLP-forest on the final preference order. Clearly, the strength of this observation has to be quantified by the range of the data sets we considered. Expanding the scope of experiments to other domains implied by practice, as well as to randomly generated ones is a goal for future research. It will provide a more detailed understanding of sensitivity of the model to the choice of a voting rule.

Improving the accuracy of the PLP-forest model is the main challenge for future work. There seem to be two natural directions. First, one can explore a possibility of combining the PLP-forest and decision-tree models, for instance, by using decision trees at leaf nodes of PLP-trees for comparison tests of outcomes in the corresponding clusters. Second, one can investigate other PLP-tree learning algorithms, possibly developing methods to find trees that best fit with given sets of examples, rather than to use heuristics, as we do now. Another promising direction is to extend the work of Bräuning et al. [4]. First, one can generalize the concept of a preference list to the tree of preference lists. In this way one can expand the ability of the preference list model to handle conditional preferences. Second, similarly as we do in this work considering PLP-forests, that is, collections of PLP-trees, one can study collections of preference list models.

To show usefulness and effectiveness of our preference tree and forest models, we are planning to collect real-world comparative preference data directly from human users and investigate the learning and reasoning aspects of the models using tools and techniques developed by Allen et al. [1].

Acknowledgments The work of the second author was supported by the NSF grant IIS-1618783.

References

1. Allen, J., Moussa, A., Liu, X.: Human-in-the-loop learning of qualitative preference models. In: The 32Nd International Florida Artificial Intelligence Research Society Conference. AAAI Press (2019)
2. Ansótegui, C., Bonet, M.L., Levy, J.: A new algorithm for weighted partial maxsat. In: Fox, M., Poole, D. (eds.) Proceedings of the 24th AAAI Conference on Artificial Intelligence. AAAI Press (2010)
3. Booth, R., Chevaleyre, Y., Lang, J., Mengin, J., Sombattheera, C.: Learning conditionally lexicographic preference relations. In: ECAI, pp. 269–274 (2010)
4. Bräuning, M., Hüllermeier, E., Keller, T., Glaum, M.: Lexicographic preferences for predictive modeling of human decision making: a new machine learning method with an application in accounting. Eur. J. Oper. Res. **258**(1), 295–306 (2017)
5. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
6. Felsenthal, D.S., Maoz, Z., Rapoport, A.: An empirical evaluation of six voting procedures: do they really make any difference? Br. J. Polit. Sci. **23**(01), 1–27 (1993)
7. Fraser, N.M.: Ordinal preference representations. Theor. Decis. **36**(1), 45–67 (1994)
8. Gehrlein, W.V.: Condorcet's paradox and the likelihood of its occurrence: different perspectives on balanced preferences. Theor. Decis. **52**(2), 171–199 (2002)
9. Hurley, B., O'Sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., De Givry, S.: Multi-language evaluation of exact solvers in graphical model discrete optimization. Constraints **21**(3), 413–434 (2016)
10. Lang, J., Mengin, J., Xia, L.: Aggregating conditionally lexicographic preferences on multi-issue domains. In: Principles and Practice of Constraint Programming, pp. 973–987. Springer (2012)
11. Lang, J., Xia, L.: Sequential composition of voting rules in multi-issue domains. Math. Soc. Sci. **57**(3), 304–324 (2009)
12. Liu, X., Truszczynski, M.: Aggregating conditionally lexicographic preferences using answer set programming solvers. In: Proceedings of the 3rd International Conference on Algorithmic Decision Theory, pp. 244–258. Springer (2013)
13. Liu, X., Truszczynski, M.: Learning partial lexicographic preference trees over combinatorial domains. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, pp. 1539–1545. AAAI Press (2015)
14. Liu, X., Truszczynski, M.: Reasoning with Preference Trees over Combinatorial Domains. In: Algorithmic Decision Theory, pp. 19–34. Springer (2015)
15. Liu, X., Truszczynski, M.: Learning partial lexicographic preference trees and forests over multi-valued attributes. In: Proceedings of the 2nd Global Conference on Artificial Intelligence (GCAI-16), EPiC Series in Computing, vol. 41, pp. 314–328. EasyChair (2016)
16. Liu, X., Truszczynski, M.: Preference learning and optimization for partial lexicographic preference forests over combinatorial domains. In: Proceedings of the 10th International Symposium on Foundations of Information and Knowledge Systems. Springer (2018)
17. Mattei, N.: Empirical evaluation of voting rules with strictly ordered preference data. In: International Conference on Algorithmic Decisiontheory, pp. 165–177. Springer (2011)
18. Myers, J.L., Well, A., Lorch, R.F.: Research design and statistical analysis. Routledge (2010)
19. Schmitt, M., Martignon, L.: Complexity of lexicographic strategies on binary cues. Preprint (1999)
20. Wilson, N.: Preference inference based on lexicographic models. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) Proceedings of the 21st European Conference on Artificial Intelligence, ECAI 2014, Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 921–926. IOS Press (2014)
21. Wilson, N., George, A.: Efficient inference and computation of optimal alternatives for preference languages based on lexicographic models. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 1311–1317 (2017)