# Preference Learning and Optimization for Partial Lexicographic Preference Forests over Combinatorial Domains

**Xudong Liu**
School of Computing
University of North Florida, USA
xudong.liu@unf.edu

**Miroslaw Truszczynski**
Department of Computer Science
University of Kentucky, USA
mirek@cs.uky.edu

### 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.

## Introduction

Preferences are fundamental to decision making and have been researched in areas such as decision theory, social choice, knowledge representation, and constraint satisfaction. 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 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 (Schmitt and Martignon 1999), lexicographic preference trees, or LP-trees (Booth et al. 2010), partial lexicographic preference trees, or PLP-trees (Liu and Truszczynski 2015a), and preference trees (Fraser 1994; Liu and Truszczynski 2015b). These models naturally support preference reasoning (Wilson 2014; Wilson and George 2017).

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.

In this paper, we focus on lexicographic models given by PLP-trees (Liu and Truszczynski 2015a). PLP-trees that impose strong restrictions on the structure, for instance, 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, PLP-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 (Liu and Truszczynski 2016). Unrestricted PLP-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 $E$ of examples have size $O(|E|)$. 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 can circumvent the problem. Following ideas proposed by Breiman (2001), Liu and Truszczynski (2016) proposed to learn forests of PLP-trees and use the Pairwise Majority rule (PMR) to obtain a new type of a lexicographic preference model (Liu and Truszczynski 2016).

The main problems with this last approach are that the PMR does not (in general) yield an order and 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 (2016) 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 (2009) studied sequential voting protocols. Lang, Mengin and Xia (2012) established computational properties of voting-based methods to aggregate LP-trees, and Liu and Truszczynski (2013) 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 aggregates preference ordering (Mattei 2011; Felsenthal, Maoz, and Rapoport 1993). 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.[1] We compare the quality of learned PLP-trees and forests with those of learned decision trees. Decisions 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 proposed to model preferences by forests of PLP-trees, aggregated by voting rules. We studied computational complexity of key reasoning tasks for the resulting models. (2) We demonstrated that the models we studied had higher predictive accuracy than that given by a single PLP-tree, and by PLP-forest with the PMR. (3) We showed 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 developed methods to compute optimal and near-optimal outcomes, the reasoning task that has no natural solutions under models based on the PMR. (5) We compared our models with those based on decision trees. We showed that the latter are more accurate but, as noted, 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 — they shed light on how agents make decisions and support preference optimization. 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.

[1]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.

## 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 $D_i$. 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 $\alpha$ 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 the outcome path $H(\alpha, T)$ ends in (the leaves are indexed from left to right with integers $0, 1, \ldots$).

We say that outcome $\alpha$ is at least as good as $\beta$ ($\alpha_T \succeq_T \beta$) 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.

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* ($T$) 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 Figure 1.

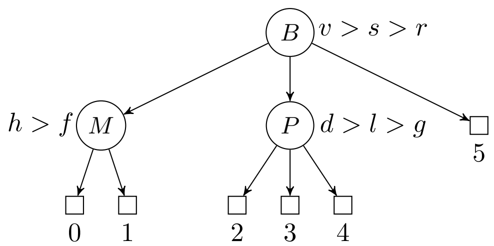



Figure 1: A PLP-tree $T$ over the car domain

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 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 orders on the space of outcomes. In this way, PLP-forests with voting rules can be viewed as models of preference relations.

## 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 (Gehrlein 2002). Nevertheless, it performs well in dominance testing (Breiman 2001; Liu and Truszczynski 2016). We consider it here as the baseline for the voting rules.

The five voting rules are *scoring* rules. In our setting, 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 every 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.

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

**Top-$k$ Clusters** (where $k$ is 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 $O(t(P) * \max\{h(T) : T \in P\})$, where $t(P)$ is the number of trees in $P$. We note that Top Cluster ($k = 1$) is a rule similar to approval, where 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 $\ell_0^T$ 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 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 $\ell_i^T$ 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

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

where $n$ is the size of the combinatorial domain. 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 $\sum\limits_{0\leq i<\ell} |\ell_i^T|)$ 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 $O(t(P) * \max\{h(T) : T \in P\})$.

**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 $O$:

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

**Maximin:** This method (also 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.

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

Both the Copeland score and the Maximin score can be computed in time $O(n * t(P) * \max\{h(T) : T \in P\})$, 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 $\alpha$ and $\beta$, $\alpha \succ_{pm} \beta$ if $N_P(\alpha,\beta) > N_P(\beta,\alpha)$. Thus, deciding pairwise preferences takes time $O(t(P) * \max\{h(T) : T \in P\})$.

## 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 two other problems *QUALITY* and *OPTIMIZATION*. Let $r$ be a scoring rule. Given a PLP-forest $P$ and an integer $\ell$, the *QUALITY* problem asks whether there exists an outcome $o$ such that $S_r(o,P) \geq \ell$. Similarly, given a PLP-forest $P$, the *OPTIMIZATION* problem consists of computing 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. (2012) 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 (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-k Clusters, where k is a positive integer, is NP-complete.*

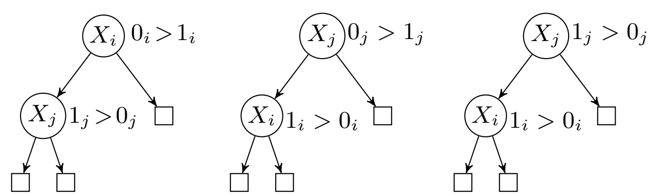
Table 1: Computational complexity results

| | *SCORE* | *QUALITY* | *OPTIMIZATION* |
|---|---|---|---|
| Top-$k$ Clusters | P | NPC (Theorem 1) | NPH |
| Plurality | P | NPC (Theorem 2) | NPH |
| Borda | P | NPC[3] | NPH |
| Copeland | #PH[3] | ? | ? |
| Maximin | ? | coNPH[3] | coNPH |

*Proof.* (Sketch) Membership is obvious, as one can guess an outcome $o$ in $O(p)$ time, and verify that $S_{tkc}(o,P) \geq l$ in polynomial time in the size of $P$.

Hardness for the case of $k = 1$ is straightforward. We omit details due to space constraint. To prove hardness for when $k > 1$, we reduce from the NP-complete problem MIN2SAT.[2] For every clause $C = X_i \vee \neg X_j \in \Phi$, we construct a set $P_C$ of three PLP-trees shown in Figure 2a, Figure 2b and Figure 2c. Here we note that, if $v \models C$, 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 $l = g(3 \cdot k - 4) + (n-g)(3 \cdot k - 3) = 3n \cdot (k-1) - g$. We need to show that assignment $v$ satisfies at most $g$ clauses in $\Phi$ if and only if $v$ scores at least $l$ in $P$ according to the Top-$k$ Clusters rule.

Assuming there is an assignment $v$ satisfying $g' \leq g$ clauses in $\Phi$, we have $S_{tkc}(v,P) = g'(3 \cdot k-4) + (n-g')(3 \cdot k - 3) = 3n \cdot (k-1) - g' \geq 3n \cdot (k-1) - g = l$. Similarly, if $v$ satisfies less than $g$ clauses, we have $S_{tkc}(v,P) < l$. □




(a) (b) (c)

Figure 2: Set $P_C$ of PLP-trees for clause $C = X_i \vee \neg X_j$

The next theorem concerns the Plurality rule. The proof is similar to that of Theorem 1 when $k = 1$.

**Theorem 2.** *The QUALITY problem for Plurality is NP-complete.*

Theorems 1 and 2 also show that the problem *OPTIMIZATION* for Plurality are NP-hard.

The *SCORE*, *QUALITY* and *OPTIMIZATION* problems for Copeland and Maximin were studied by Lang et al. (2012). However, the complexity bounds were established in some cases only. We summarize in Table 1 the computational complexity picture for all voting rules we study in this paper.

[2]Given a set $\Phi$ 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 $\Phi$.

[3]cf. (Lang, Mengin, and Xia 2012)

## Experiments and Results

PLP-trees and forests are difficult to elicit from users directly. In practical settings they have to be learned from examples of pairwise comparisons. A method to learn PLP-trees was proposed by Liu and Truszczynski (2016). They also applied it 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.

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

### Datasets and Experimental Set-up

We implemented the scoring rules discussed above as order aggregators for PLP-forests and experimented with them on the twelve preferential datasets[4] used by Liu and Truszczynski (2016).

The PLP-forest learning procedure works as follows. For each of the datasets, we randomly partition the set of strict examples $\mathcal{E}^{\succ}$, generating a training set of 70% of $\mathcal{E}^{\succ}$ and use the rest 30% as the testing set. In the training phase, we use the greedy learning heuristic (Liu and Truszczynski 2016) 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.

For the task of predicting new preferences, we compute and report average *accuracy* results, where the accuracy is defined as the number of strict examples in the testing set that are in agreement with the learned model divided by the size of the testing set.

As we showed above, computing optimal outcomes for PLP-forests using the Top-$k$ Clusters rules is closely related and can be reduced to solving MAXSAT problems. Thus, we proposed reductions of score computation for the three Top-$k$ Clusters rules to the *weighted partial MAXSAT* problem and used the MAXSAT solver *toulbar*2 (Hurley et al. 2016) to solve them.

For the task of understanding the relationship among the scoring rules, we calculate the Spearman's rho for Top Cluster, Top-2 Clusters, Top-3 Clusters, Plurality, Borda and Maximin, all against Copeland.

### 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 Figure 3a. 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 Figure 3a. Based on these experiments, from now on we restrict our discussion to PLP-forests with trees learned from samples of size 50.

In Figure 3b, we present the mean learning curves over all datasets for all 8 rules, where each curve shows how testing results (accuracy percentages) change as 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. 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 $D$. To answer a such query for testing, two outcomes are fed to a decision tree. If the predicted is 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 (2016) turns out to be the worst aggregating method overall.

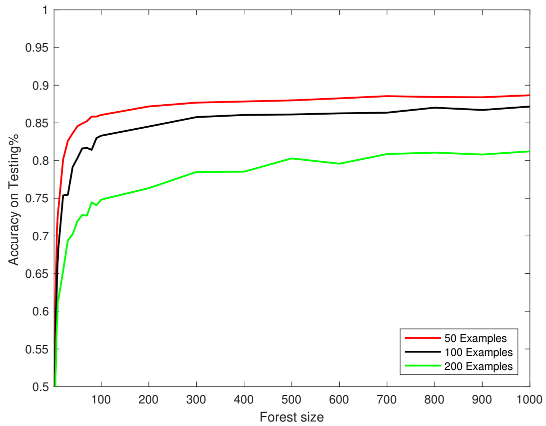
Moreover, looking at the results for 1000-tree forests, we see that, these forests have high accuracy of about 89∼91%, on the testing datasets, depending on the voting rule. This provides strong evidence for the adequacy of the PLP-forest model to represent user preferences over practical combinatorial 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 finishes 90%, only a percentile point difference from Borda.

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 enabling 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 explanations 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.

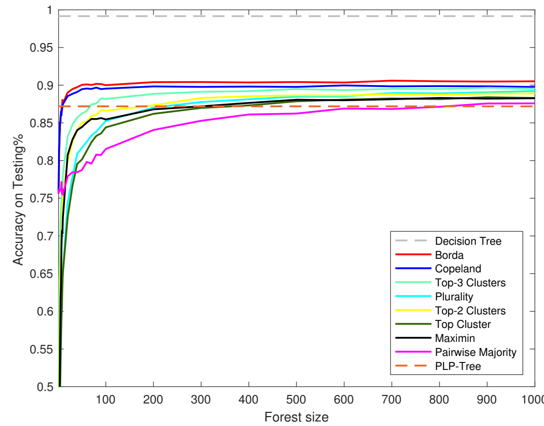
### Preference Optimization Results

PLP-forests with scoring rules allow for effective optimal outcome computation. We demonstrate it below for orders obtained by using Top, Top-2 and Top-3

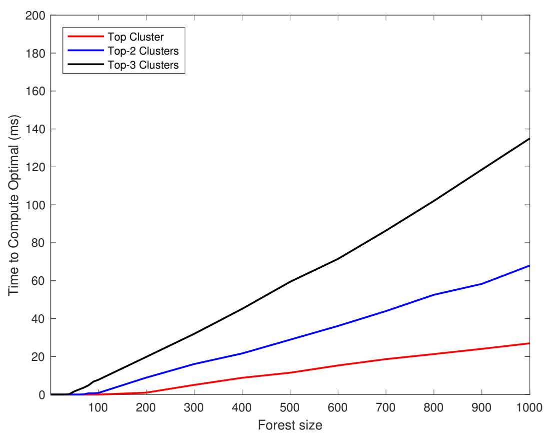
[4] https://www.unf.edu/∼N01237497/preflearnlib.php



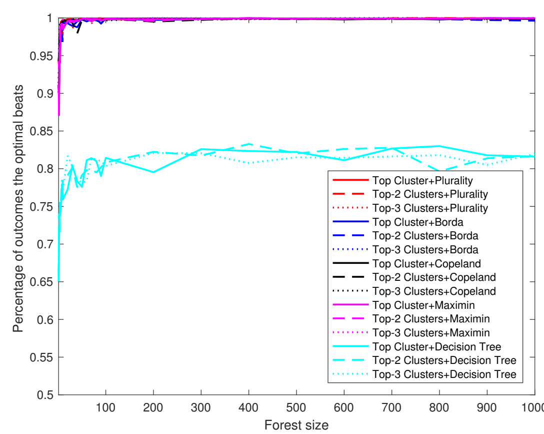

(a) Mean testing accuracy across all datasets for PLP-forests applying Top-2 Clusters, where every member tree is trained using random samples of sizes 50, 100 and 200.




(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.

Figure 3: Preference learning and optimization results for PLP-forests

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 a weighted partial MAXSAT instance and use *toulbar2* to solve it. Average computational time spent on searching for optimal outcomes using for all datasets is shown in Figure 3c. We see that, for any dataset 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 Figure 3d. 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 op-

Table 2: 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 |
|---|---|---|---|---|---|---|
| Mean | 0.95 | 0.91 | 0.90 | 0.89 | 0.88 | 0.87 |
| SD | 0.14 | 0.14 | 0.06 | 0.13 | 0.12 | 0.12 |

timize 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.

## Rank Correlation Results

In the standard voting setting, the rankings generated by different voting rules are quite close to each other (Mattei 2011; Felsenthal, Maoz, and Rapoport 1993). 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 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 (Myers, Well, and Lorch 2010; Mattei 2011; Felsenthal, Maoz, and Rapoport 1993), we used the Spearman's rho (denoted by $\rho$) as the rank correlation coefficient.

Our results (cf. Table 2) 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 (Felsenthal, Maoz, and Rapoport 1993; Myers, Well, and Lorch 2010; Mattei 2011).

## 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. (2012), only some 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 a scoring rule.

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 represent well orders they 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 compared out 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, the relation they define is not guaranteed to be a total order (not even a partial order). Second, they do not provide any clear insights into key factors determining the underlying preference relations. Lastly they do not offer any effective ways to solve optimization tasks (finding optimal or near optimal outcomes). These drawbacks make PLP-forests, despite their lower accuracy, an attractive preference model for use in applications.

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 new PLP-tree learning algorithms.

## References


Booth, R.; Chevaleyre, Y.; Lang, J.; Mengin, J.; and Sombattheera, C. 2010. Learning conditionally lexicographic preference relations. In *ECAI*, 269–274.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Felsenthal, D. S.; Maoz, Z.; and Rapoport, A. 1993. An empirical evaluation of six voting procedures: do they really make any difference? *British Journal of Political Science* 23(01):1–27.

Fraser, N. M. 1994. Ordinal preference representations. *Theory and Decision* 36(1):45–67.

Gehrlein, W. V. 2002. Condorcet's paradox and the likelihood of its occurrence: different perspectives on balanced preferences. *Theory and decision* 52(2):171–199.

Hurley, B.; OSullivan, B.; Allouche, D.; Katsirelos, G.; Schiex, T.; Zytnicki, M.; and De Givry, S. 2016. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints* 21(3):413–434.

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

Lang, J.; Mengin, J.; and Xia, L. 2012. Aggregating conditionally lexicographic preferences on multi-issue domains. In *Principles and Practice of Constraint Programming*, 973–987. Springer.

Liu, X., and Truszczynski, M. 2013. Aggregating conditionally lexicographic preferences using answer set programming solvers. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*, 244–258. Springer.

Liu, X., and Truszczynski, M. 2015a. Learning partial lexicographic preference trees over combinatorial domains. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 1539–1545. AAAI Press.

Liu, X., and Truszczynski, M. 2015b. Reasoning with preference trees over combinatorial domains. In *Algorithmic Decision Theory*. Springer. 19–34.

Liu, X., and Truszczynski, M. 2016. Learning partial lexicographic preference trees and forests over multi-valued attributes. In *Proceedings of the 2nd Global Conference on Artificial Intelligence (GCAI-16)*, volume 41 of *EPiC Series in Computing*, 314–328. EasyChair.

Mattei, N. 2011. Empirical evaluation of voting rules with strictly ordered preference data. In *International Conference on Algorithmic DecisionTheory*, 165–177. Springer.

Myers, J. L.; Well, A.; and Lorch, R. F. 2010. *Research design and statistical analysis*. Routledge.

Schmitt, M., and Martignon, L. 1999. Complexity of lexicographic strategies on binary cues. *Preprint*.

Wilson, N., and George, A. 2017. 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*, 1311–1317.

Wilson, N. 2014. Preference inference based on lexicographic models. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence, ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 921–926. IOS Press.