EPiC Series in Computing
Volume 41, 2016, Pages 81–94
GCAI 2016. 2nd Global Conference on Artificial Intelligence

Learning Importance of Preferences

Ying Zhu and Miroslaw Truszczynski

University of Kentucky, Lexington, Kentucky, U.S.A.
ying.zhu@uky.edu, mirek@cs.uky.edu

Abstract

We study the problem of learning the importance of preferences in preference profiles in two important cases: when individual preferences are aggregated by the *ranked* Pareto rule, and when they are aggregated by positional scoring rules. For the ranked Pareto rule, we provide a polynomial-time algorithm that finds a ranking of preferences such that the ranked profile correctly decides *all* the examples, whenever such a ranking exists. We also show that the problem to learn a ranking maximizing the number of correctly decided examples is NP-hard. We obtain similar results for the case of weighted profiles when positional scoring rules are used for aggregation.

1 Introduction

Preferences appear in every scenario requiring selecting among several alternatives. When choosing a restaurant to have dinner at, people consider the menu, price, location, opening hours, and table size. When selecting a college to attend, students take into account the rank of the college, the reputation of the department, its location, the faculty, and the cost. When buying a car, buyers may have in mind safety, fuel economy and size.

To design an effective automated system to help an agent make optimal choices over large collections of alternatives, one needs a model to represent the agent's preferences. Explicit representations are impractical. The sheer number of choices often makes agents unable to fully specify their preference relation on them. To get around the problem, we rely on concise implicit preference models. They specify a preference order that captures the agent's preferences exactly or, more commonly, offers a good approximation.

There are two key problems here: what expressions to use as preference models, and how to elicit these models from the users. Both problems have received substantial attention. Researchers developed and studied several preference representation languages [15, 12, 20][1]. Some of these languages, for instance CP-nets, are explicitly designed to support structured preference elicitation. However, in most cases, eliciting preference models relies on preference learning. It has been used to build quantitative preference representations based on utility functions [7, 16, 4, 6, 18], as well as models for pairwise comparisons of choices (the dominance relation) [9, 13, 19], lexicographic orders, strategies and trees [11, 26, 24], and CP-nets [10, 21, 22, 8, 25, 17].

[1]These publications offer surveys of preference reasoning and are good sources of additional references.

We consider a preference model that arranges outcomes into several total preorders, each characterizing preferences wrt a particular feature, and aggregates these orders into a single preference preorder based on the importance of the features. For instance, in the car-buying example one could have several preference orderings according to the safety record, size and fuel economy, and a relative importance of these factors, where safety may be ranked as most important, followed by size and fuel economy. Such preference models were considered for combinatorial domains over multivalued attributes [1, 3, 23]. In that research, preorders based on preferences on values of individual attributes are aggregated into a single overall preference order in a lexicographic manner determined by the importance of the attributes. A similar preference model also arises in the voting setting, where individual preference orders come from the voters. The voters may have different importance levels expressed by real non-negative weights, and the votes are aggregated by weighted versions of positional scoring rules [2].

Our goal is to study how to learn such models. We assume that individual orders (or votes) are known (because they are "natural" and common to all agents, or because they can be elicited or learned), and we focus only on learning the importance ranks or weights, as appropriate. For instance, it is reasonable to assume that a typical user prefers higher safety ratings and better mileage, and that preferences of particular agents in a voting group can be elicited or accurately predicted (pools of candidates in elections are limited, preferences of voters often are correlated to their income, education levels, and other demographic factors). However, the importance of features (safety vs fuel economy) to a user, or the standing of particular voters in a voting group may be unknown. One way to establish that is by learning. This is the focus of our paper. More precisely, given a *preference profile*, that is, a collection of total preorders over a set of options, *and* a set of examples reflecting the aggregated preference relation, we want to determine whether preferences in the profile can be ranked (weighted) so that the resulting ranked (weighted) profile correctly orders all (or "almost" all) examples in the set.

While learning preferences has been studied extensively, learning importance of preferences has not yet received much attention. We show that the problems to decide the existence of a rank or weight assignment so that all examples in the example set were decided correctly is in P. To this end, we present polynomial-time algorithms for the corresponding two settings that produce a "witness" rank or weight assignment, if one can be found. However, the problems asking for assignments maximizing the number of decided examples are harder. We show that in each setting (ranked and weighted) they are NP-hard, leaving a study of practical algorithms for these optimization problems for the future.

2 Preliminaries

A *preference order* (or, simply, a *preference*) over a set X of *options* is a total preorder on X. We denote it by $\succeq$, possibly with annotations. Each preference $\succeq$ determines two associated relations: *strict preference* $\succ$, where $x \succ y$ if and only if $x \succeq y$ and $y \not\succeq x$, and *equivalence* $\approx$ (also called *indifference*), where $x \approx y$ if and only if $x \succeq y$ and $y \succeq x$. The relation $\approx$ is an equivalence relation on X and partitions X into equivalence classes, $X_1, \ldots, X_m$, which we always enumerate from the most to the least preferred. Using this notation, we can describe a total preorder $\succeq$ by the expression $X_1 \succ X_2 \succ \cdots \succ X_m$. Given such an expression, for every $x \in X$, we define the *satisfaction degree* of x in $\succeq$, written $d_{\succeq}(x)$, as the unique i such that $x \in X_i$. When we refer to a preference by a letter, say p, we write $\succeq_p$ for the corresponding total preorder and d_p for the corresponding satisfaction degree function.

The first class of profiles we consider are ranked profiles. A *profile* over X is a set of preferences over X. A *ranked* profile is a pair (P, r), where P is a profile and r is a *ranking function* that assigns a positive integer (a *rank*) to every preference in the profile. The preferences with a lower rank are more important than the preferences with a higher rank. An *unranked* profile is a special case of the ranked profile where all preferences have the same rank.

To obtain the collective preorder over options, we will aggregate preferences in ranked profiles by the *ranked Pareto principle* [5] which, speaking informally, applies the Pareto rule lexicographically. The rule is defined as follows. For a ranked profile (P, r), and options $x, y \in X$, we define $x \succeq_{P,r} y$ to hold if

1. $x \approx_p y$ ($d_p(x) = d_p(y)$) for every preference $p \in P$, or
2. there is a preference $p \in P$ such that $x \succ_p y$ ($d_p(x) < d_p(y)$) and for every preference $p' \in P$ such that $r(p') \leq r(p)$, $x \succeq_{p'} y$ ($d_p(x) \leq d_p(y)$).

Intuitively, the ranked Pareto rule considers preferences rank by rank, starting with the lowest ranked (most important) preferences. If these preferences consistently make an option x at least as preferred as an option y, and at least one of these preferences makes x strictly more preferred to y, x is more preferred to y in the entire profile (the Pareto rule applied to this group of preferences). If some preferences in this group strictly prefer x to y and some other of them strictly prefer y to x, x and y are incomparable in the profile (again, consistently with the Pareto rule applied to this group of preferences). Otherwise, the lowest ranked (most important) preferences do not differentiate between x and y, and we move on to the preferences of the next rank to see how x and y should be ordered. Thus, the Pareto rule is used at each level, and levels are treated lexicographically, that is, the first level that distinguishes between two options determines their order in the entire profile.[2]

The relation $\succeq_{P,r}$ is a preorder on X. In general, it is not total. We write $\succ$, $\approx$, and $|$ (preserving annotations) for the corresponding strict preference, equivalence, and *incomparability* relations.[3] If P is an unranked profile, we drop a reference to the ranking function from the notation and write $\succeq_P$ instead of $\succeq_{P,r}$.

For example, let $X = \{a, b, c, d\}$ be the set of outcomes, and let us consider the profile over X consisting of these three preferences:

$$1\colon \{a, b\} \succ \{c, d\} \qquad 2\colon \{a\} \succ \{b, c\} \succ \{d\} \qquad 3\colon \{c, d\} \succ \{b\} \succ \{a\}.$$

If all preferences have the same rank (the profile is unranked), then the only two outcomes that are strictly ordered in the aggregated order $\succeq_P$ are c and d (we have $c \succ_P d$). All other pairs of distinct elements are incomparable. This example shows, in particular, that our aggregation procedure may produce preorders that are not total.

Next, let us consider a ranking function r that assigns rank 1 to the first two preferences and rank 3 to the third preference. Now, $a \succ_{P,r} b \succ_{P,r} c \succ_{P,r} d$, that is all pairs of distinct elements are strictly ordered. The first two preferences determine this order; the third preference, which conflicts with it, is ignored as it is less important than the first two. It would only be used if there were any ties in the order determined by the more important preferences. It is clear that the ranking function r' assigning rank 1 to the first two preferences and rank 2 to the last one, works in the same way as r, that is the orders $\succeq_{P,r}$ and $\succeq_{P,r'}$ are the same.

Finally, let us consider a ranking function s that assigns rank 1 to the preference 3, rank 2 to preference 1 and rank 3 to the preference 2. Now the third preference is the most important

[2]The ranked Pareto rule we consider is related to the way lexicographic preference trees order outcomes [1, 3, 23]. The difference is that lexicographic trees concern combinatorial domains and aggregate preferences of a particular type, those that are induced by preferences on values of attribute (or issue) domains. Our setting is more general.

[3]The first two are defined as for total orders; for the incomparability, we set $x \mid y$ (x and y are incomparable) if neither $x \succeq_{P,r} y$ nor $y \succeq_{P,r} x$ holds.

one and determines almost the entire collective order $\succeq_{P,s}$. It only leaves the relation between c and d unresolved. To determine this relation, we look next at the preference 1 (the next in importance). It does not help as c and d are equivalent in it. We proceed then to the preference 2 (the least important one) and see that it prefers c to d, thus resolving the matter of the order of c and d in the aggregated order $\succeq_{P,s}$.

The *number of ranks* in a ranked profile (P, r) is denoted by $range(P, r) = max_{p \in P} r(p)$. For example, the range of the profiles (P, r) and (P, s) in the example above is 3. A ranked profile (P, r) is *gap-free* if for every $i = 1, \ldots, range(P, r)$, there exists $p \in P$ such that $r(p) = i$. Clearly, in the example above, the profile (P, r) is not gap-free but the profile (P, r') is. For a gap-free ranked profile (P, r), $range(P, r) \leq |P|$. Gap-free profiles are as expressive as arbitrary ranked profiles, as shown by the following result formalizing our observation about profiles (P, r) and (P, r') in the example.[4]

Theorem 1. *Let P be a preference profile over a set of outcomes O. For every ranking function r, there is a ranking function r' such that (P, r) and (P, r') define the same total order over O and (P, r') is gap-free.*

Given a set O of outcomes, an *example* is an expression $\langle x, y, R\rangle$, where $x, y \in O$ and $R \in \{\succ, \approx, |\}$. An example $\langle x, y, R\rangle$ is *decided* by a ranked profile (P, r) if $xR_{P,r}y$ (that is, $\langle x, y, \succ\rangle$ is decided by (P, r) if $x \succ_{P,r} y$, etc). If $R \in \{\succ, |\}$, we also say $\langle x, y, R\rangle$ is *decided on rank i by* (P, r) if for every $p \in P$ such that $r(p) < i$, $x \approx_p y$, and if $R = \succ$, $x \succ_p y$ for some $p \in P$ such that $r(p) = i$, and $x \succeq_p y$ for every $p \in P$ such that $r(p) = i$ or, if $R = |$, $x \succ_p y$ and $y \succ_q x$ for some $p, q \in P$ such that $r(p) = r(q) = i$.

Let P be a preference profile, E a set of examples (we assume P and E are over a set O of outcomes), and r a ranking function. The ranked profile (P, r) is *consistent* with E if (P, r) decides all examples in E. In our example, the profile (P, r) is consistent with $E = \{\langle a, c, \succ\rangle, \langle b, d, \succ\rangle\}$ while the profile (P, s) is not. A profile P is *consistent* with E if there is a ranking function r such that (P, r) is consistent with E. Thus, the profile P in our running example is consistent with $E = \{\langle a, c, \succ\rangle, \langle b, d, \succ\rangle\}$. If a profile is consistent with a set of examples, one can always find a ranking function such that the corresponding ranked profile is consistent with the examples and the range of the ranking function is polynomially bounded.

Theorem 2. *Let P be a preference profile and E be a set of examples. If P is consistent with E, there exists a ranking function r such that (P, r) is consistent with E and $range(P, r) \leq min(|P|, |E| + 1)$.*

The two problems we will study for ranked profiles can be stated as follows.

RANK-CONSISTENCY: Given a preference profile P and a set E of examples, decide if P is consistent with E.

A more practical version of the problem, its approximation version, asks for a ranking function such that the corresponding ranked profile decides at least k examples.

RANK-MATCH-MANY: Given a preference profile P, a set E of examples, and an integer $k \geq 1$, decide if there is a ranking function r such that (P, r) decides at least k examples from E.

In the second part of the paper, we consider quantitative representation of the importance of preferences. A *weighted* profile (P, w) is a profile in which every preference $p \in P$ is assigned a non-negative real *weight* $w(p)$. The preferences with larger weights are more important than those with smaller ones. Importantly, weights allow us to express quantitatively the difference in importance, and the magnitude of the weight determines the impact of a preference. In our

[4]The proofs of this and several other results are omitted due to space requirements.

study of this setting, we restrict preferences to be *strict* total orders, that is, *votes*, and call a preference profile a *voting* profile. We use *positional scoring rules* [2] studied in social choice as the mechanism to aggregate preferences (votes).[5] As we will study positional scoring rules in the computational setting, we present them in terms appropriate for complexity and algorithm design considerations. We use the term *effective* to underscore this.

Definition 1. An effective positional scoring rule F is given by a function $f : \{(n, m) \colon n \geq m \geq 1\} \rightarrow \{0, 1, \ldots\}$ such that (1) for every m, m', if $1 \leq m < m' \leq n$ then $f(n, m) \geq f(n, m')$; and (2) f can be computed in time bounded by a polynomial in the first argument of f.

Let f be an effective positional scoring rule. Given a vote p over a set O of n outcomes, which are called *candidates* in this setting, the score $s^F_p(c)$ of a candidate c in p is defined by $s^F_p(c) = f(n, \alpha_p(c))$, where $\alpha_p(c)$ is the *position* of c in p (the number of candidates ahead of c in p plus 1). The score of a candidate in a weighted profile (P, w), $s^F_{P,w}(c)$, is the weighted sum of individual scores. Specifically,

$$s^F_{P,w}(c) = \sum_{p \in P} s^F_p(c) \cdot w(p).$$

The scores in a profile determine a total preorder $\succ^F_{P,w}$ on candidates; the higher the score, the more preferred the candidate. We write $\alpha^F_{P,w}(c)$ for the position of a candidate c in that preorder. Whenever the rule F is clear from the context, we drop it from the notation.

Given an effective positional scoring rule F and a weighted profile (P, w), the preorder $\succ^F_{P,w}$ and the position function $\alpha^F_{P,w}$ can be computed in polynomial time (in the size of the profile).

For a voting profile over a set O of candidates, an *example* is an expression $\langle x, y, R\rangle$, where $x, y \in O$ and $R \in \{\succ, \approx\}$. Since $\succ^F_{P,w}$ is a total preorder for any effective positional scoring rule F, we do not allow incomparability examples, as they cannot be satisfied by any weighted profile. Let (P, w) be a weighted profile and F an effective positional scoring rule. A strict example $\langle x, y, \succ\rangle$ is *decided* by (P, w) under F if $s^F_{P,w}(x) > s^F_{P,w}(y)$, and an equivalence example $\langle x, y, \approx\rangle$ is *decided* by (P, w) under F if $s^F_{P,w}(x) = s^F_{P,w}(y)$. Further, let E be a set of strict and equivalence examples (over O). A weighted profile (P, w) is *consistent with E under F* if all examples in E are decided by (P, w) under F; and a voting profile P is *consistent with E under F* if there exists a weight assignment w such that (P, w) is *consistent with E under F*.

Let F be an effective positional scoring rule. The two problems discussed above in the ranked profile setting can be stated as follows for the setting of weighted profiles of votes.

WEIGHT-CONSISTENCY-F: Given a voting profile P and a set of strict and equivalence examples E, decide if P is consistent with E under F.

WEIGHT-MATCH-MANY-F: Given a voting profile P, a set of strict and equivalence examples E, and an integer $k \geq 1$, decide if there is a weight assignment w such that (P, w) decides at least k examples from E under F.

3 Learning Ranks of Preferences

3.1 The consistency problem

For the RANK-CONSISTENCY problem, we show that it can be solved by a polynomial time algorithm. If a profile P is consistent with the examples E, the algorithm finds a ranking

[5]Positional scoring rules can be extended to preorders and so the restriction to strict total orders is not critical. We adopt it here for simplicity and leave the general case for the future work.

function so that all examples are decided by the ranked profile. Moreover, the range of the rank assignment (the number of different ranks used) is minimized.

Our first result shows that a simple preprocessing allows us to reduce the RANK-CONSISTENCY problem to the case without equivalence examples.

Proposition 1. *Let (P, r) be a ranked profile and $\langle x, y, \approx\rangle$ be an example. Then $\langle x, y, \approx\rangle$ is decided by (P, r) if and only if $x \approx_p y$ for every $p \in P$.*

Thus, an algorithm to decide the consistency of a profile P with a set E of examples could first check all equivalence examples. If for some equivalence example $\langle x, y, \approx\rangle \in E$ and some preference $p \in P$, $x \not\approx_p y$ then, by Proposition 1, P is inconsistent with E. Otherwise, also by Proposition 1, for any ranking function r, all equivalence examples are decided by (P, r) and so the consistency of P with E depends only on the consistency of P with the set of non-equivalence examples in E.

In view of this observation, from now on we assume that all examples in E involve either $\succ$ or $|$ relations. To solve the consistency problem under this assumption, we first describe an algorithm to determine those preferences that cannot have rank 1 in any ranked profile consistent with the examples. The algorithm is based on two auxiliary facts that follow directly from the definitions of the relation $\succeq_{P,r}$ and its derived variants $\succ_{P,r}$ and $|_{P,r}$.

Proposition 2. *Let (P, r) be a ranked profile and $\langle x, y, \succ\rangle$ an example decided by (P, r). For every $p \in P$ such that $y \succ_p x$, $r(p) > 1$.*

Proposition 3. *Let (P, r) be a ranked profile and $\langle x, y, |\rangle$ an example decided by (P, r). If $x \succeq_p y$ holds for every $p \in P$ such that $r(p) = 1$, then for every $q \in P$ such that $x \succ_q y$, $r(q) > 1$. Moreover, if $y \succeq_p x$ holds for every $p \in P$ such that $r(p) = 1$, then for every $q \in P$ such that $y \succ_q x$, $r(q) > 1$.*

These results suggest the following method to determine preferences in a given profile P that cannot have rank 1 under any ranking function r for which (P, r) is consistent with a given set E of examples. Let U denote the set of all such preferences identified so far (initially empty). First, let us consider an example $\langle x, y, \succ\rangle \in E$. By Proposition 2, if (P, r) is consistent with E (and so, decides $\langle x, y, \succ\rangle$), for each preference p such that $y \succ_p x$, $r(p) > 1$. Thus, we add p to U. Once all strict examples in E have been considered, we have that for every ranking function s, such that (P, s) is consistent with E, $\{p \in P : s(p) = 1\} \subseteq P \setminus U$.

We now move on to incomparability examples. Let $\langle x, y, |\rangle \in E$. If (P, r) is consistent with E, (P, r) decides $\langle x, y, |\rangle$. Let us assume that $x \succeq_p y$ holds for every $p \in P \setminus U$. By the inclusion above, it follows that $x \succeq_p y$ holds for every preference p such that $r(p) = 1$. By Proposition 3, for every preference q such that $x \succ_q y$, $r(q) > 1$. Similarly, if $y \succeq_p x$ holds for every $p \in P \setminus U$, for every preference q with $y \succ_q x$, $r(q) > 1$, too (also by Proposition 3). Therefore such preferences q are included in U and we move on to the next incomparability example.

After checking all incomparability examples, we iteratively repeat this process checking all incomparability examples again. The process terminates when there is no preference added to U in some iteration. These iterations are necessary because when U is extended (and so, $P \setminus U$ gets smaller), the condition $x \succeq_p y$, for all $p \in P \setminus U$, may now hold (similarly for the dual condition $y \succeq_p x$, for all $p \in P \setminus U$), and an incomparability example that has not "pushed" a preference out of rank 1 before, may do so now.

We call this algorithm *NotTopRanked*. Its formal description follows (P is a given preference profile and E is a given set of examples).

1. $U = \varnothing$.

```
2. For each $\langle x, y, \succ\rangle \in E$:   if $y \succ_p x$, for some $p \in P$, $U = U \cup \{p\}$.
3. Loop until no change in $U$
   (a) For each $\langle x, y, |\rangle \in E$,
       i. if $x \succeq_p y$ for every $p \in P\backslash U$, $U = U \cup \{q : q \in P, x \succ_q y\}$.
       ii. if $y \succeq_p x$ for every $p \in P\backslash U$, $U = U \cup \{q : q \in P, y \succ_q x\}$.
4. Return $U$.
```

Since U is expanded by at least one preference in each iteration, there are at most $|P|$ iterations. Thus, the algorithm terminates and runs in polynomial time. The algorithm *NotTopRanked* plays a key role in deciding the RANK-CONSISTENCY problem. Here are some useful properties of this algorithm.

Proposition 4. *Let P be a preference profile, E be a set of strict and incomparability examples, and $U = NotTopRanked(P, E)$. For every ranking function r such that (P, r) is consistent with E, $r(p) > 1$ for every $p \in U$.*

Proposition 4 establishes the fundamental property of the set U returned by the algorithm *NotTopRanked*: it consists of preferences in P that do not have rank 1 under any rank function r such that (P, r) is consistent with E.

Proposition 5. *Let P be a preference profile and E a set of strict and incomparability examples. If $P = NotTopRanked(P, E)$, P is inconsistent with E.*

This result gives a condition allowing us to determine that the RANK-CONSISTENCY problem has the negative answer (no ranking function r exists such that (P, r) is consistent with E).

Proposition 6. *Let P be a preference profile, E a set of strict and incomparability examples, and $U = NotTopRanked(P, E)$. For every example $\langle x, y, R\rangle$ in E, either $x \approx_{P\backslash U} y$ or $xR_{P\backslash U}y$.*

This result allows us to reduce the RANK-CONSISTENCY problem for P and E to its version for the set U of preferences and the set of all examples not decided by preferences in $P \setminus U$, and to proceed by recursion. Namely, if $U \neq P$ (otherwise, the problem has no solution by Proposition 5), we assign rank 1 to the preferences in $P\backslash U$. If $xR_{P\backslash U}y$ holds for every example $\langle x, y, R\rangle$ in E, (P, r) is consistent with E, where r is a ranking function such that $r(p) = 1$ for every $p \in P\backslash U$ and $r(p) = 2$ for every $p \in U$. Otherwise, some examples in E are not decided by $P \setminus U$. By Proposition 6, for outcomes x and y in each such undecided example we have $x \approx_p y$, for every $p \in P \setminus U$. It follows that P is consistent with E if and only if U is consistent with the set E' of examples in E undecided by $P \setminus U$. Thus, we can recurse.

The formal description of the algorithm follows. We call it $Rank(P, E, r)$. It returns false if P is not consistent with E, and returns true, otherwise. In this latter case, it also sets r to be a ranking function r such that (P, r) is consistent with E.

```
1. If $E = \varnothing$, set $r(p) = 1$ for each $p \in P$ and return true.
2. If $P = \varnothing$, return false.
3. Set $U = NotTopRanked(P, E)$, and $E' = \{e = \langle x, y, R\rangle : e \in E, x \approx_{P\backslash U} y\}$.
4. If $U = P$, return false.
5. If $Rank(U, E', r') = false$, return false.
   Otherwise, set $r(p) = 1$ for all $p \in P\backslash U$ and $r(p) = r'(p) + 1$ for all $p \in U$; return true.
```

Clearly, the algorithm *Rank* runs in polynomial time. Its correctness is stated in the following result.

Theorem 3. *Let P be a preference profile and E be a set of strict and incomparability examples. If P is consistent with E, then $Rank(P, E, r)$ returns true and (P, r) is consistent with E. If P is inconsistent with E, then $Rank(P, E, r)$ returns false.*

Moreover, if a preference profile P is consistent with a set E of examples, the ranking function computed by the algorithm *Rank* minimizes the number of ranks.

Theorem 4. *Let P be a preference profile and E be a set of strict and incomparability examples. If $Rank(P, E, r)$ returns true, $range(P, r) \leq range(P, r')$ for any ranking function r' such that (P, r') is consistent with E.*

Based on Theorems 4 and 2, the ranking function r computed by the algorithm *Rank* has at most $min(|P|, |E|+1)$ ranks and each rank except the lowest one decides at least one example.

3.2 The optimization problem

If a preference profile is inconsistent with a set of examples, i.e., for no ranking function the corresponding ranked profile decides all examples, it becomes important to find a ranking function so that the corresponding ranked profile decides as many examples as possible. This problem is an optimization version of the RANK-MATCH-MANY problem which we discuss now.

Theorem 5. *The* RANK-MATCH-MANY *problem is NP-complete.*

Proof. The problem is in NP as for every preference $p \in P$, we can guess its rank $r(p)$ (and all these ranks can be chosen from the set $\{1, \ldots, |P|\}$ according to Theorem 1), and then verify in polynomial time whether the ranked profile (P, r) decides at least k examples from E (exploiting the fact that dominance testing in ranked profiles is a polynomial-time task).

For the hardness part, we reduce from the *vertex cover* problem in graphs, which is known to be NP-complete [14]. Let $G = (V, E)$ be an undirected graph. A set $V' \subseteq V$ is a *vertex cover* of G if every edge in E is incident to at least one vertex in V'. Given a graph G and an integer $k \leq |V|$, the *vertex cover* problem is to decide whether G has a vertex cover of size at most k.

Let $G = (V, E)$ be an undirected graph with n vertices and m edges (that is, $n = |V|$ and $m = |E|$). We denote by E_v the set of edges in E incident to v in G. For every $e \in E$, we introduce $n + 1$ copies of e, say $e_1, \ldots, e_{n+1}$. We denote that set by C_e. Finally, we also introduce one more object, say c, different from all elements in $V \cup \bigcup_{e \in E} C_e$. We now define a set O of outcomes, a profile P over O, and a set A of examples over O by setting

$$O = V \cup \bigcup_{e \in E} C_e \cup \{c\}, \qquad P = \{\bigcup_{e \in E_v} C_e \succ v \succ REM : v \in V\} \cup \{c \succ REM\}, \text{ and}$$

$$A = \{\langle c, v, \succ \rangle : v \in V\} \cup \{\langle x, c, \succ \rangle : x \in C_e, e \in E\}.$$

We adopt the convention to write REM as the last option in a preference. By default, REM stands for the set of all outcomes not included in the more preferred options. We will denote each preference $\bigcup_{e \in E_v} C_e \succ v \succ REM$, $v \in V$, by p_v, and the preference $c \succ REM$ by p_c.

Clearly, A consists of $n + m(n+1)$ examples. Let $k \leq n$. We claim that G contains a vertex cover V' such that $|V'| \leq k$ if and only if there exists a ranking function r on preferences in P such that the ranked profile (P, r) decides at least $m(n+1) + n - k$ examples.

($\Rightarrow$) Let V' be a vertex cover for G and $|V'| \leq k$. We define the ranking function r on P by setting

$$r(p_v) = \begin{cases} 1 & \text{if } v \in V' \\ 3 & \text{if } v \in V \backslash V' \end{cases}$$

and $r(p_c) = 2$.

Let us consider an example $\langle x, c, \succ \rangle$, for some $x \in C_e$ and $e \in E$. Since V' is a vertex cover of G, there is a vertex $v \in V'$ such that $e \in E_v$. Thus, $d_{p_v}(x) = 1$ and $r(p_v) = 1$. Since for every $p \in P$ with $r(p) = 1$, $d_p(c) = 3$ and $d_p(x) \leq 3$, the example $\langle x, c, \succ \rangle$ is decided by (P, r).

Next, let us consider an example $\langle c, v, \succ \rangle$, where $v \in V$. If $v \in V'$, the ranked profile (P, r) does not decide $\langle c, v, \succ \rangle$. Indeed, $r(p_v) = 1$, $d_{p_v}(v) = 2$ and $d_{p_v}(c) = 3$. Thus, $c \not\succ_{P,r} v$. On the other hand, if $v \notin V'$, then $r(p_v) = 3$. Moreover, $r(p_c) = 2$, $d_{p_c}(c) = 1$, $d_{p_c}(v) = 2$, and for every other preference p (it is of the form p_w for some $w \in V$, $w \neq v$), $d_p(v) = d_p(c)$. Thus, the ranked profile (P, r) decides $\langle c, v, \succ \rangle$.

It follows that the ranked profile (P, r) decides $n + m(n+1) - |V'|$ examples. Since $|V'| \leq k$, (P, r) decides at least $n + m(n+1) - k$ examples, as claimed.

($\Leftarrow$) Let us assume that r is a ranking function such that (P, r) decides at least $m(n+1) + n - k$ examples. If there is $e \in E$ and $x \in C_e$ such that (P, r) does not decide $\langle x, c, \succ \rangle$, then (P, r) does not decide any of the examples $\langle y, c, \succ \rangle$, where $y \in C_e$. Thus the number of examples decided by (P, r) is at most $n + m(n+1) - (n+1) < n + m(n+1) - k$, a contradiction.

It follows that (P, r) decides all examples $\langle x, c, \succ \rangle$, where $e \in E$, $x \in C_e$. Moreover, (P, r) decides at least $n - k$ examples $\langle c, v, \succ \rangle$, where $v \in V$. Let V' consist of all elements $v \in V$ such that (P, r) does not decide $\langle c, v, \succ \rangle$. By our observation, $|V'| \leq k$. We will show that V' is a vertex cover for G.

Let us consider an edge $e \in E$ and $x \in C_e$, and let v and w be the two endpoints of e. Since (P, r) decides $\langle x, c, \succ \rangle$, and $c \succ_{p_c} x$, it follows that there is a preference p_u, for some $u \in V$, such that $x \succ_{p_u} c$ and $r(p_u) < r(p_c)$. It follows that $u = v$ or $u = w$. Without loss of generality, we may assume that $u = v$. Thus, $r(p_v) < r(p_c)$ and $x \succ_{p_v} v \succ_{p_v} c$. Since for all preferences p other that p_v and p_c, $v \approx_p c$, $v \succ_{P,r} c$, that is, (P, r) does not decide the example $\langle c, v, \succ \rangle$. Thus, $v \in V'$. It follows that V' is a vertex cover for G. □

The results in this section are the first steps towards practical rank-learning algorithms. They provide insights into the complexity of the problems. In particular, the complexity of the RANK-MATCH-MANY problem implies that computing rankings maximizing the number of decided examples is NP-hard. Nevertheless, techniques such as Boolean satisfiability checking, answer-set programming or constraint satisfaction might prove effective in handling it, and fast approximation techniques may be possible, too. We leave these questions for future work.

We also note that if k is fixed, the problem can trivially be decided in polynomial time by solving the RANK-CONSISTENCY problem for every set of examples with size k. Whether faster methods exist (in particular, of order independent of k, which would imply fixed-parameter tractability of the problem) is another direction for future work.

4 Learning Weights for Voting

Next we consider the learning problems for voting profiles with an effective positional scoring rule as the aggregation method. In a weighted voting scenario, each vote is a strict total order, and a candidate receives scores from every vote based on its position in the vote and on the weight of the vote. The weights are represented by numerical values and we consider both real and integer weights in our work.

4.1 The consistency problem

Theorem 6. *For any effective positional scoring rule F, the* WEIGHT-CONSISTENCY-F *problem is in P (both when weights are allowed to be reals and when they are restricted to be integers).*

Proof. This problem can be modeled as a linear programming problem. We construct a linear program over variables w_p, $p \in P$, and ϵ. Variables w_p, $p \in P$, are meant to represent the weights we seek.

For every example $\langle a, b, \succ\rangle$, we include in the program the linear inequality

$$\Sigma_{p \in P} w_p s_p(a) \geq \Sigma_{p \in P} w_p s_p(b) + \epsilon,$$

and for every example $\langle a, b \approx\rangle$, similarly, we include the inequalities

$$\Sigma_{p \in P} w_p s_p(a) \geq \Sigma_{p \in P} w_p s_p(b) \qquad \text{and} \qquad \Sigma_{p \in P} w_p s_p(b) \geq \Sigma_{p \in P} w_p s_p(a).$$

We also include in the program the inequalities $w_p \geq 0$, for every $p \in P$, $0 \leq \epsilon$ and $\epsilon \leq 1$. We take ϵ as the objective function and we want to maximize it.

If P is consistent with E under F, there is a solution for this linear program with ϵ satisfying $0 < \epsilon \leq 1$. Thus, an optimal solution also has a positive ϵ. Conversely, if an optimal solution for the linear programming problem has a positive ϵ, the profile P is consistent with examples in E under F. It is also clear that if the linear programming problem has no solutions or if the optimal solution has $\epsilon = 0$, P is inconsistent with E under F. Thus, the consistency of P with E can be decided by solving the linear programming problem we constructed.

The linear programming problem can be solved in polynomial time. Therefore, the consistency problem can be decided in polynomial time. Moreover, if the answer is yes, there is a rational solution to the LP problem (as the coefficients are integers) and so, a rational weight assignment showing the consistency of P. Thus, also an integer weight assignment showing the consistency of P exists (it can be obtained by scaling up the rational weight assignment). $\square$

This result has a corollary limiting the size of the weights, in the case when a profile is consistent with a set of examples.

Corollary 1. *Given a set P of votes, a set E of strict and equivalence examples, and an effective positional scoring rule F, if P is consistent with E under F, then there exists an integer weight assignment w such that (P, w) decides E under F and the size of the representation of all weights $w(p)$ is polynomial in the size of the input.*

Proof. From Theorem 6, we know that we can decide whether P is consistent with E, and compute an integer weight assignment w in polynomial time in the size of the input. Thus, the size of the representation of all weights $w(p)$ is polynomial in the size of the input. $\square$

4.2 The optimization problem

In this section, we consider the optimization problem for voting profiles, which decides whether a voting profile can be weighted to decide at least k examples.

We first discuss a simple instance of the optimization problem where the positional scoring rule is veto, the number of votes is equal to the number of candidates and every candidate is placed at the last (lowest) position by exactly one vote. Under veto, every candidate gets one point from a vote except the candidate placed at the very end of that vote. The *veto* rule can be formally defined by the function f such that $f(n,m) = 1$ for every $n \geq 1$ and $1 \leq m < n$, and $f(n,m) = 0$ for every $n \geq 1$ and $m = n$. Given a set O of candidates, we use $O \setminus \{c\} \succ c$ to denote the vote over O that positions the candidate c at the end and the remaining candidates in an arbitrary order.

To prove the complexity of the optimization problem for this special voting profile, we need a result bounding the size of weights for the profile.

Proposition 7. *Let O be a set of candidates and $P = \{O \setminus \{c\} \succ c : c \in O\}$ a profile over O. If E is a set of strict and equivalence examples over O such that some total preorder on O is consistent with E, then there is a weight assignment w such that (P, w) is consistent with E under the veto rule, and for every $p \in P$, $w(p) \leq |O|$.*

We use this result to prove that the optimization problem for this special voting profile under veto is NP-complete.

Theorem 7. *The following problem is NP-complete: Given a set of candidates O, a set of votes $P = \{O \setminus \{c\} \succ c : c \in O\}$, a set of strict and equivalence examples E over O, and an integer k, $1 \leq k \leq |E|$, decide whether there is a subset E' of E such that $|E'| \geq k$ and P is consistent with E' under the veto rule.*

This theorem is a stepping stone to establishing the complexity of the optimization problem under a general effective positional scoring rule.

An effective positional scoring rule F is called trivial if $f(n,m) = f(n,m')$ for every $1 \leq m, m' \leq n$. The optimization problem is in P under a trivial effective positional scoring rule since every candidate gets the same score no matter how to assign the weights to votes. For any non-trivial effective positional scoring rule, the optimization problem is NP complete.

Theorem 8. *The* WEIGHT-MATCH-MANY-F *problem is NP-complete for every non-trivial effective positional scoring rule F.*

Proof. The problem is in NP because we can guess a weight assignment w of size polynomial in the size of input (cf. Corollary 1), compute the scores of every candidate, and use these scores to verify whether at least k examples are decided. As each of the latter two tasks can be accomplished in polynomial time, the membership in NP follows.

For the hardness part, we use a reduction from the problem in Theorem 7. In that problem we are given a set O of n candidates, a set $P = \{p_c : c \in O\}$ of votes, where $p_c = O \setminus \{c\} \succ c$, a set E of strict and equivalence examples, and an integer k; the objective is to decide whether there is a weight assignment w such that at least k examples from E are decided by (P, w) under the veto rule.

We construct the sets O' and E' of candidates and examples by setting $O' = O$ and $E' = E$. Next, for each $c \in O$, we fix an enumeration $c_1, \ldots, c_{n-1}$ of elements in $O \setminus \{c\}$ and define votes p_c^i, $i = 1, \ldots, n-1$ by setting

$$p_c^i = c_i \succ \ldots \succ c_{n-1} \succ c_1 \succ \ldots \succ c_{i-1} \succ c.$$

Finally, we define $P_c = \{p_c^i \colon i = 1, \ldots, n-1\}$ and $P' = \bigcup_{c \in O} P_c$. We will show that there is a weight assignment w on P such that at least k examples in E are decided by (P, w) under the veto rule if and only if there is a weight assignment w' on P' such that at least k examples in E' $(= E)$ are decided by (P', w') under the rule F.

($\Rightarrow$) Let w be a weight assignment for P such that at least k examples in E are decided by (P, w) under the veto rule. Clearly, for every $x \in O$, $s^{veto}_{P,w}(x) = W - w(p_x)$, where $W = \sum_{c\in O} w(p_c)$.

We now define a weight assignment w' on P'. Namely, for every $p \in P_c$, we set $w'(p) = w(p_c)$. Clearly,

$$s^F_{P',w'}(x) = \sum_{c\in O} s^F_{P_c,w'}(x).$$

Since x appears in the last position (position n) in every vote in P_x, and since each of these votes has the same weight $w(p_x)$,

$$s^F_{P_x,w'}(x) = \sum_{p\in P_x} \alpha_n w(p_x) = (n-1)\alpha_n w(p_x)$$

where we denote $f(n, m)$ by α_m. For every $c \in O \setminus \{x\}$, x appears exactly once in each position i, $1 \leq i \leq n-1$, in votes from P_c. Since each of these votes has the same weight $w(p_c)$,

$$s^F_{P_c,w'}(x) = (\alpha_1 + \cdots + \alpha_{n-1})w(p_c)$$

Thus,

$$\begin{aligned} s^F_{P',w'}(x) = \sum_{c\in O} s^F_{P_c,w'}(x) &= (n-1)\alpha_n w(p_x) + A \sum_{c\in O, c\neq x} w(p_c) \\ &= (n-1)\alpha_n w(p_x) + A(W - w(p_x)) = AW - (A - (n-1)\alpha_n)w(p_x), \end{aligned}$$

where we denote $\alpha_1 + \cdots + \alpha_{n-1}$ by A (and we recall that we defined $W = \sum_{c\in O} w(p_c)$).

We now observe that since F is non-trivial, $A - (n-1)\alpha_n > 0$. Consequently, for every $x, y \in C$,

$$s^{veto}_{P,w}(x) > s^{veto}_{P,w}(y) \quad \text{if and only if} \quad s^F_{P_c,w'}(x) > s^F_{P_c,w'}(y).$$

Since (P, w) decides at least k examples in E (under the veto rule), (P', w') decides at least k examples in E' $(= E)$ under the rule F.

($\Leftarrow$) Let w' be a weight assignment on P' such that at least k examples in E' $(= E)$ are decided by P' under F. Let $E'' \subseteq E$ be the set of examples that are decided by (P', w') under F. It follows that the preorder $\succ^F_{P',w'}$ is consistent with E''. By Proposition 7, there is a weight assignment w on P such that (P, w) is consistent with E'' under veto. In other words, (P, w) decides at least k examples in E under veto. □

As in the ranked setting, our results here are the first step towards weight learning. They provide an understanding of the complexity of the basic decision problems but leave an open question whether learning weight assignments to preferences can be implemented in ways that would be effective in practice.

5 Conclusions

In this work, we studied the problem of learning the importance of preferences in a preference profile based on a set of examples. We investigated two preference settings here. In one of them, preferences are ranked and are aggregated by the ranked Pareto principle, in the other they are weighted and aggregated by positional scoring rules. We proved that in each of these settings, one can decide in polynomial time whether a rank/weight assignment exists such that all examples are decided by the corresponding ranked/weighted profile. For the ranked profile we have described an algorithm based on a procedure that identifies preferences that cannot have rank 1. The basic idea to solve the weight assignment problem is to cast it in linear programming. We also considered the optimization versions of the problems, where the goal is to find rank/weight assignments maximizing the number of decided examples. For both settings we proved that the decision versions of the optimization problem are NP-complete.

Our work opens several directions for future work. First, it brings up the key problem of whether practical rank/weight learning algorithms are possible. In the future, we will study both exact optimization algorithms by exploiting satisfiability and answer-set programming solvers. We will also investigate fast heuristic algorithms to learn good rank/weight assignments. Another problem of interest is to extend the results of this paper to the case of conditional preferences. Finally, we note that in our study we assume that all examples are given up-front. The *active* learning setting, when we query users for preferences as we learn the model (ranks and weights in our case) is another possible direction for further research.

Acknowledgments

This work was partially supported by the NSF grant IIS-1618783.

References

[1] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. Learning conditionally lexicographic preference relations. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 269–274, 2010.

[2] Steven J. Brams and Peter C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*, volume 1 of *Handbook of Social Choice and Welfare*, chapter 4, pages 173–236. Elsevier, 2002.

[3] Michael Bräuning and Hüllermeier Eyke. Learning conditional lexicographic preference trees. In *Proceedings of the ECAI-12 Workshop on Preference Learning: Problems and Applications in AI (PL-12)*, page 11, 2012.

[4] Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. *CoRR*, abs/1207.1361, 2012.

[5] G. Brewka, Ilkka Niemelä, and Tommi Syrjänen. Implementing ordered disjunction using answer set solvers for normal programs. In Sergio Flesca, Sergio Greco, Giovambattista Ianni, and Nicola Leone, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence, JELIA 2002*, pages 444–456, Berlin, 2002. Springer.

[6] Urszula Chajewska, Lise Getoor, Joseph Norman, and Yuval Shahar. Utility elicitation as a classification problem. *CoRR*, abs/1301.7367, 2013.

[7] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 2000.

[8] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. Learning ordinal preferences on multiattribute domains: The case of cp-nets. In *Preference Learning.* 2010.

[9] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Int. Res.*, 10:243–270, 1999.

[10] Yannis Dimopoulos, Loizos Michael, and Fani Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, 2009.

[11] József Dombi, Csanád Imreh, and Nándor Vincze. Learning lexicographic orders. *European Journal of Operational Research*, 183:748 – 756, 2007.

[12] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in AI: An overview. *Artif. Intell.*, 175:1037–1052, 2011.

[13] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In *ECML*, 2003.

[14] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[15] Judy Goldsmith and Ulrich Junker, editors. *Special Issue on Preferences*, volume 29(4) of *AI Magazine*. 2008.

[16] Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, 2004.

[17] Joshua T. Guerin, Thomas E. Allen, and Judy Goldsmith. Learning cp-net preferences online from user queries. In *Late-Breaking Developments in the Field of Artificial Intelligence, Bellevue, Washington, USA, July 14-18, 2013*, 2013.

[18] Vu A. Ha and Peter Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. *CoRR*, abs/1302.1544, 2013.

[19] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172:1897–1916, 2008.

[20] Souhila Kaci. *Working with Preferences: Less Is More*. Springer, 2011.

[21] Frédéric Koriche and Bruno Zanuttini. Learning conditional preference networks with queries. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1930–1935, 2009.

[22] Frdric Koriche and Bruno Zanuttini. Learning conditional preference networks. *Artificial Intelligence*, 174:685 – 703, 2010.

[23] Xudong Liu and Miroslaw Truszczynski. Preference trees: A language for representing and reasoning about qualitative preferences. In *Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF)*, pages 55–60. AAAI Press, 2014.

[24] Xudong Liu and Miroslaw Truszczynski. Learning partial lexicographic preference trees over combinatorial domains. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1539–1545. AAAI Press, 2015.

[25] Xudong Liu and Miroslaw Truszczynski. Reasoning with preference trees over combinatorial domains. In Toby Walsh, editor, *Proceedings of the 4th International Conference on Algorithmic Decision Theory, ADT 2015*, volume 9346 of *LNCS*, pages 19–34. Springer, 2015.

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