Reasoning with Preference Trees over Combinatorial Domains

Xudong Liu$^{(\boxtimes)}$ and Miroslaw Truszczynski

Department of Computer Science, University of Kentucky,
Lexington, KY, USA
{liu,mirek}@cs.uky.edu

Abstract. Preference trees, or *P-trees* for short, offer an intuitive and often concise way of representing preferences over combinatorial domains. In this paper, we propose an alternative definition of P-trees, and formally introduce their compact representation that exploits occurrences of identical subtrees. We show that P-trees generalize lexicographic preference trees and are strictly more expressive. We relate P-trees to answer-set optimization programs and possibilistic logic theories. Finally, we study reasoning with P-trees and establish computational complexity results for the key reasoning tasks of comparing outcomes with respect to orders defined by P-trees, and of finding optimal outcomes.

1 Introduction

Preferences are essential in areas such as constraint satisfaction, decision making, multi-agent cooperation, Internet trading, and social choice. Consequently, preference representation languages and algorithms for reasoning about preferences have received substantial attention [8]. When there are only a few objects (or *outcomes*) to compare, it is both most direct and feasible to represent preference orders by their explicit enumerations. The situation changes when the domain of interest is *combinatorial*, that is, its elements are described in terms of combinations of values of *issues*, say $x_1, \ldots, x_n$ (also called *variables* or *attributes*), with each issue x_i assuming values from some set D_i — its *domain*.

Combinatorial domains appear commonly in applications. Since their size is exponential in the number of issues, they are often so large as to make explicit representations of preference orders impractical. Therefore, designing languages to represent preferences on elements from combinatorial domains in a concise and intuitive fashion is important. Several such languages have been proposed including penalty and possibilistic logics [4], conditional preference networks (CP-nets) [2], lexicographic preference trees (LP-trees) [1], and answer-set optimization (ASO) programs [3].

In this paper, we focus our study on combinatorial domains with *binary* issues. We assume that each issue x has the domain $\{x, \neg x\}$ (we slightly abuse the notation here, overloading x to stand both for an issue and for one of the elements of its domain). Thus, outcomes in the combinatorial domain determined by the set $\mathcal{I} = \{x_1, \ldots, x_n\}$ of binary issues are simply complete and consistent

sets of literals over $\mathcal{I}$. We denote the set of all such sets of literals by $CD(\mathcal{I})$. We typically view them as truth assignments (interpretations) of the propositional language over the vocabulary $\mathcal{I}$. This allows us to use propositional formulas over $\mathcal{I}$ as concise representations of sets of outcomes over $\mathcal{I}$. Namely, each formula φ represents the set of outcomes that satisfy φ (make φ true).

For example, let us consider preferences on possible ways to arrange a vacation. We assume that vacations are described by four binary variables:

1. *activity* (x_1) with values *water sports* (x_1) and *hiking* ($\neg x_1$),
2. *destination* (x_2) with *Florida* (x_2) and *Colorado* ($\neg x_2$),
3. *time* (x_3) with *summer* (x_3) and *winter* ($\neg x_3$), and
4. the mode of *travel* (x_4) could be *car* (x_4) and *plane* ($\neg x_4$).

A complete and consistent set of literals $\neg x_1 \neg x_2 x_3 x_4$ represents the hiking vacation in Colorado in the summer to which we travel by car.

To describe sets of vacations we can use formulas. For instance, vacations that take place in the summer (x_3) or involve water sports (x_1) can be described by the formula $x_3 \vee x_1$, and vacations in Colorado ($\neg x_2$) that we travel to by car (x_4) by the formula $\neg x_2 \wedge x_4$.

Explicitly specifying strict preference orders on $CD(\mathcal{I})$ becomes impractical even for combinatorial domains with as few as 7 or 8 issues. However, the setting introduced above allows us to specify total preorders on outcomes in terms of desirable properties outcomes should have. For instance, a formula φ might be interpreted as a definition of a total preorder in which outcomes satisfying φ are preferred to those that do not satisfy φ (and outcomes within each of these two groups are equivalent). More generally, we could see an expression (a sequence of formulas)

$$\varphi_1 > \varphi_2 > \ldots > \varphi_k$$

as a definition of a total preorder in which outcomes satisfying φ_1 are preferred to all others, among which outcomes satisfying φ_2 are preferred to all others, etc., and where outcomes not satisfying any of the formulas φ_i are least preferred. This way of specifying preferences is used (with minor modifications) in possibilistic logic [4] and ASO programs [3]. In our example, the expression

$$x_3 \wedge x_4 > \neg x_3 \wedge \neg x_2$$

states that we prefer summer vacations (x_3) where we drive by car (x_4) to vacations in winter ($\neg x_3$) in Colorado ($\neg x_2$), with all other vacations being the least preferred.

This linear specification of preferred formulas is sometimes too restrictive. An agent might prefer outcomes that satisfy a property φ to those that do not. Within the first group that agent might prefer outcomes satisfying a property ψ_1 and within the other a property ψ_2. Such *conditional* preference can be naturally captured by a form of a decision tree presented in Fig. 1. Leaves, shown as boxes, represent sets of outcomes satisfying the corresponding conjunctions of formulas ($\varphi \wedge \psi_1$, $\varphi \wedge \neg \psi_1$, etc.).
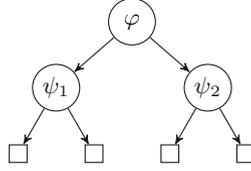
Fig. 1. A preference tree

Trees such as the one in Fig. 1 are called *preference trees*, or *P-trees*. They were introduced by Fraser [5,6], who saw them as a convenient way to represent conditional preferences. Despite their intuitive nature they have not attracted much interest in the preference research in AI. In particular, they were not studied for their relationship to other preference formalisms. Further, the issue of compact representations received only an informal treatment by Fraser (P-trees in their full representation are often impractically large), and the algorithmic issues of reasoning with P-trees were also only touched upon.

In this paper, we propose an alternative definition of preference trees, and formally define their compact representation that exploits occurrences of identical subtrees. P-trees are reminiscent of LP-trees [1]. We discuss the relation between the two concepts and show that P-trees offer a much more general, flexible and expressive way of representing preferences. We also discuss the relationship between P-trees and ASO preferences, and between P-trees and possibilistic logic theories. We study the complexity of the problems of comparing outcomes with respect to orders defined by preference trees, and of problems of finding optimal outcomes.

Our paper is organized as follows. In the next section, we formally define P-trees and a compact way to represent them. In the following section we present results comparing the language of P-trees with other preference formalisms. We then move on to study the complexity of the key reasoning tasks for preferences captured by P-trees and, finally, conclude by outlining some future research directions.

2   Preference Trees

In this section, we define preference trees and discuss their representation. Let $\mathcal{I}$ be a set of binary issues. A *preference tree* (*P-tree*, for short) over $\mathcal{I}$ is a binary tree with all nodes other than leaves labeled with propositional formulas over $\mathcal{I}$. Each P-tree T defines a natural strict order $\succeq_T$ on the set of its leaves, the order of their enumeration from left to right.

Given an outcome $M \in CD(\mathcal{I})$, we define the *leaf of M in T* as the leaf reached by starting at the root of T and proceeding downwards. When at a node t labeled with φ, if $M \models \varphi$, we descend to the left child of t; otherwise, we descend to the right child of t. We denote the leaf of M in T by $l_T(M)$.
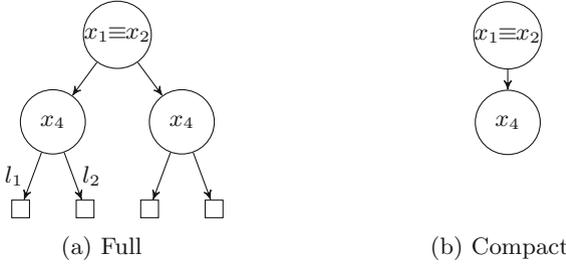
Fig. 2. P-trees on vacations

We use the concept of the leaf of an outcome M in a P-tree T to define a total preorder on $CD(\mathcal{I})$. Namely, for outcomes $M, M' \in CD(\mathcal{I})$, we set $M \succeq_T M'$ (M is *preferred* to M'), if $l_T(M) \succeq_T l_T(M')$, and $M \succ_T M'$ (M is *strictly preferred* to M'), if $l_T(M) \succ_T l_T(M')$.[1] We say that M is *equivalent* to M', $M \approx_T M'$, if $l_T(M) = l_T(M')$. Finally, M is *optimal* if there exists no M' such that $M' \succ_T M$.

Let us come back to the vacation example and assume that an agent prefers vacations involving water sports in Florida or hiking in Colorado over the other options. This preference is described by the formula $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$ or, more concisely, as an equivalence $x_1 \equiv x_2$. Within each of the two groups of vacations (satisfying the formula and not satisfying the formula), driving (x_4) is the preferred transporting mode. These preferences can be captured by the P-tree in Fig. 2a. We note that in this example, the preferences at the second level are *unconditional*, that is, they do not depend on preferences at the top level.

To compare two outcomes, $M = \neg x_1 \neg x_2 \neg x_3 x_4$ and $M' = x_1 x_2 x_3 \neg x_4$, we walk down the tree and find that $l_T(M) = l_1$ and $l_T(M') = l_2$. Thus, we have $M \succ_T M'$ since l_1 precedes l_2.

The key property of P-trees is that they can represent any total preorder on $CD(\mathcal{I})$.

Proposition 1. *For every set $\mathcal{I}$ of binary issues, for every set $D \subseteq CD(\mathcal{I})$ of outcomes over $\mathcal{I}$, and for every total preorder $\succeq$ on D into no more than 2^n clusters of equivalent outcomes, there is a P-tree T of depth at most n such that the preorder determined by T on $CD(\mathcal{I})$ when restricted to D coincides with $\succeq$ (that is, $\succeq_{T|D} = \succeq$).*

Proof. Let $\succeq$ be a total preorder on a subset $D \subseteq CD(\mathcal{I})$ of outcomes over $\mathcal{I}$, and let $D_1 \succ D_2 \succ \ldots \succ D_m$ be the corresponding strict ordering of clusters of equivalent outcomes, with $m \leq 2^n$. If $m = 1$, a single-leaf tree (no decision nodes, just a box node) represents this preorder. This tree has depth 0 and so, the assertion holds. Let us assume then that $m > 1$, and let us define $D' = D_1 \cup \ldots \cup D_{\lceil m/2 \rceil}$ and $D'' = D \setminus D'$. Let $\varphi_{D'}$ be a formula such that models of D' are

---

[1] We overload the symbols $\succeq_T$ and $\succ_T$ by using them both for the order on the leaves of T and the corresponding preorder on the outcomes from $CD(\mathcal{I})$.

precisely the outcomes in D' (such a formula can be constructed as a disjunction of conjunctions of literals, each conjunction representing a single outcome in D'). If we place $\varphi_{D'}$ in the root of a P-tree, that tree represents the preorder with two clusters, D' and D'', with D' preceding D''. Since each of D' and D'' has no more than 2^{n-1} clusters, by induction, the preorders $D_1 \succ \ldots \succ D_{\lceil m/2 \rceil}$ and $D_{\lceil m/2 \rceil + 1} \succ \ldots \succ D_m$ can each be represented as a P-tree with depth at most $n - 1$. Placing these trees as the left and the right subtrees of $\varphi_{D'}$ respectively results in a P-tree of depth at most n that represents $\succeq$.   □
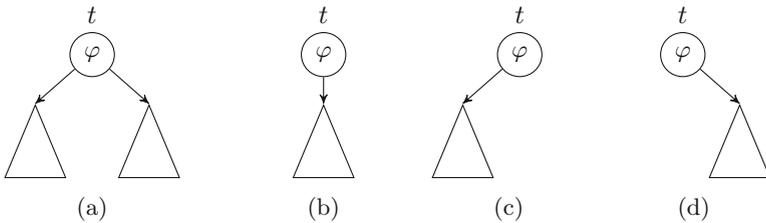
Compact Representation of P-trees. Proposition 1 shows P-trees to have high expressive power. However, the construction described in the proof has little practical use. First, the P-tree it produces may have a large size due to the large sizes of labeling formulas that are generated. Second, to apply it, one would need to have an explicit enumeration of the preorder to be modeled, and that explicit representation in practical settings is unavailable.

However, preferences over combinatorial domains that arise in practice typically have structure that can be elicited from a user and exploited when constructing a P-tree representation of the preferences. First, decisions at each level are often based on considerations involving only very few issues, often just one or two and very rarely more than that. Moreover, the subtrees of a node that order the "left" and the "right" outcomes are often identical or similar.

Exploiting these features often leads to much smaller representations. A *compact P-tree over* $\mathcal{I}$ is a tree such that

1. every node is labeled with a Boolean formula over $\mathcal{I}$, and
2. every non-leaf node t labeled with φ has either two outgoing edges, with the left one meant to be taken by outcomes that satisfy φ and the right one by those that do not (Fig. 3a), or one outgoing edge pointing
   - straight-down (Fig. 3b), which indicates that the two subtrees of t are *identical* and the formulas labeling every pair of corresponding nodes in the two subtrees are the *same*,
   - left (Fig. 3c), which indicates that right subtree of t is a leaf, or
   - right (Fig. 3d), which indicates that left subtree of t is a leaf.

The P-tree in Fig. 2a can be collapsed as both subtrees of the root are the same (including the labeling formulas). This leads to a tree in Fig. 2b with a


Fig. 3. Compact P-trees

straight-down edge. We note that we drop box-labeled leaves in compact representations of P-trees, as they no longer have an interpretation as distinct clusters.

Empty Leaves in P-trees. Given a P-tree T one can prune it so that all sets of outcomes corresponding to its leaves are non-empty. However, keeping empty clusters may lead to compact representations of much smaller (in general, even exponentially smaller) size.

A full P-tree T in Fig. 4a uses labels $\varphi_1 = \neg x_1 \vee x_3$, $\varphi_2 = x_2 \vee \neg x_4$, and $\varphi_3 = x_2 \wedge x_3$. We check that leaves l_1, l_2 and l_3 are empty, that is, the conjunctions $\varphi_1 \wedge \neg \varphi_2 \wedge \varphi_3$, $\neg \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ and $\neg \varphi_1 \wedge \neg \varphi_2 \wedge \varphi_3$ are unsatisfiable. Pruning T one obtains a compact tree T' (Fig. 4b) that is smaller compared to T, but larger than T'' (Fig. 4c), another compact representation of T, should we allow empty leaves and exploit the structure of T.

(a) T (b) T': pruned T (c) T''
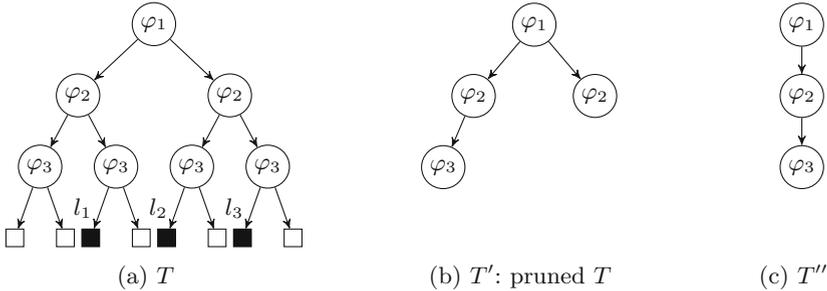
Fig. 4. P-trees with empty leaves

That example generalizes and leads to the question of finding small-sized representations of P-trees (we conjecture that the problem in its decision version asking about the existence of a compact representation of size at most k is NP-complete). From now on, we assume that P-trees are given in their compact representation.

3   P-trees and Other Formalisms

In this section we compare the preference representation language of P-trees with other preference languages.

P-trees Generalize LP-trees. As stated earlier, P-trees are reminiscent of LP-trees, a preference language that has received significant attention recently [1,10,11]. In fact, LP-trees over a set $\mathcal{I} = \{x_1, \ldots, x_n\}$ of issues are simply special P-trees over $\mathcal{I}$. Namely, an LP-tree over $\mathcal{I}$ can be defined as a P-tree over $\mathcal{I}$, in which all formulas labeling nodes are atoms x_i or their negations $\neg x_i$, depending on whether x_i or $\neg x_i$ is preferred, and every path from the root to a

leaf has all atoms x_i appear on it exactly once. Clearly, LP-trees are full binary trees of depth n (assuming they have an implicit extra level of "non-decision" nodes representing outcomes) and determine strict *total orders* on outcomes in $CD(\mathcal{I})$ (no indifference between different outcomes). An example of an LP-tree over $\{x_1, x_2, x_3, x_4\}$ for our vacation example is given in Fig. 5.
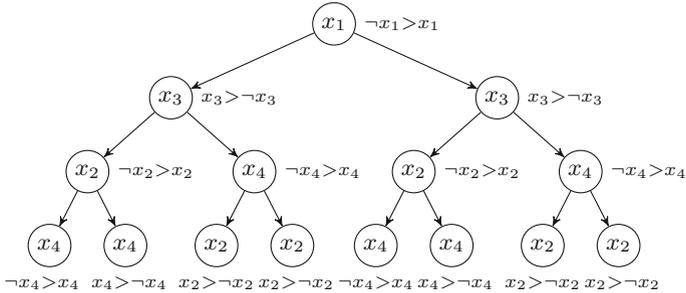

Fig. 5. A full LP-tree on vacations

In general representing preferences by LP-trees is impractical. The size of the representation is of the same order as that of an explicit enumeration of the preference order. However, in many cases preferences on outcomes have structure that leads to LP-trees with similar subtrees. That structure can be exploited, as in P-trees, to represent LP-trees compactly. Figure 6a shows a compact representation of the LP-tree in Fig. 5. We note the presence of conditional preference tables that make up for the lost full binary tree structure. Together with the simplicity of the language, compact representations are essential for the practical usefulness of LP-trees. The compact representations of LP-trees translate into compact representations of P-trees, in the sense defined above. This matter is not central to our discussion and we simply illustrate it with an example. The compactly represented P-tree in Fig. 6b is the counterpart to the compact LP-tree in Fig. 6a, where $\varphi = (x_2 \wedge x_4) \vee (\neg x_2 \wedge \neg x_4)$.

The major drawback of LP-trees is that they can capture only a very small fraction of preference orders. One can show that the number, say $G(n)$, of LP-trees over n issues is

$$G(n) = \prod_{k=0}^{n-1} (n-k)^{2^k} \cdot 2^{2^k}$$

and is asymptotically much smaller than $L(n) = (2^n)!$, the number of all preference orders of the corresponding domain of outcomes. In fact, one can show that

$$\frac{G(n)}{L(n)} < \frac{1}{2^{(2^n \cdot (n - \log n - 2))}}.$$

This is in stark contrast with Proposition 1, according to which every total preorder can be represented by a P-tree.

(a) A compact LP-tree   (b) The corresponding P-tree
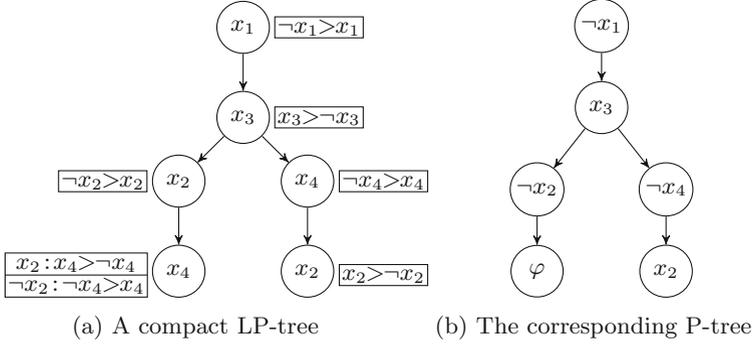
Fig. 6. A compact LP-tree as a compact P-tree

Even very natural orderings, which have simple (and compact) representations by P-trees often cannot be represented as LP-trees. For instance, there is no LP-tree on $\{x_1, x_2\}$ representing the order $00 \succ 11 \succ 01 \succ 10\}$. However, the P-trees (both full and compact) in Fig. 2 do specify it.

P-trees Extend ASO-Rules. The formalism of ASO-rules [3] provides an intuitive way to express preferences over outcomes as total preorders. An ASO-rule partitions outcomes into ordered clusters according to the semantics of the formalism. Formally, an ASO-rule r over $\mathcal{I}$ is a preference rule of the form

$$C_1 > \ldots > C_m \leftarrow B, \tag{1}$$

where all C_i's and B are propositional formulas over $\mathcal{I}$. For each outcome M, rule r of the form (1) determines its *satisfaction degree*. It is denoted by $SD_r(M)$ and defined by

$$SD_r(M) = \begin{cases} 1, & M \models \neg B \\ m + 1, & M \models B \wedge \bigwedge_{1 \leq i \leq m} \neg C_i \\ min\{i : M \models C_i\}, & \text{otherwise.} \end{cases}$$

We say that an outcome M is weakly preferred to an outcome M' ($M \succeq_r M'$) if $SD_r(M) \leq SD_r(M')$. Thus, the notion of the satisfaction degree (or, equivalently, the preference r) partitions outcomes into (in general) $m + 1$ clusters.[2]

Let us consider the domain of vacations. An agent may prefer hiking in Colorado to water sports in Florida if she is going on a summer vacation. Such preference can be described as an ASO-rule:

$$\neg x_1 \wedge \neg x_2 > x_1 \wedge x_2 \leftarrow x_3.$$

Under the semantics of ASO, this preference rule specifies that the most desirable vacations are summer hiking vacations to Colorado and all winter vacations, the

---

[2] This definition is a slight adaptation of the original one.

next preferred vacations are summer water sports vacations to Florida, and the least desirable vacations are summer hiking vacations to Florida and summer water sports vacations to Colorado.

It is straightforward to express ASO-rules as P-trees. For an ASO-rule r of form (1), we define a P-tree T_r as shown in Fig. 7. That is, every node in T_r has the right child only (the left child is a leaf representing an outcome and is not explicitly shown). Moreover, the labels of nodes from the root down are defined as follows: $\varphi_1 = \neg B \vee C_1$, and $\varphi_i = C_i$ $(2 \le i \le m)$.
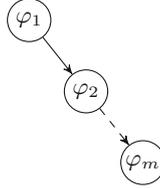

Fig. 7. A P-tree T_r

Theorem 1. *Given an ASO-rule r, the P-tree T_r has size linear in the size of r, and for every two outcomes M and M'*

$$M \succeq_r^{ASO} M' \;\; iff \;\; M \succeq_{T_r} M'$$

Proof. The P-tree T_r induces a total preorder $\succeq_{T_r}$ where outcomes satisfying φ_1 are preferred to outcomes satisfying $\neg\varphi_1 \wedge \varphi_2$, which are then preferred to outcomes satisfying $\neg\varphi_1 \wedge \neg\varphi_2 \wedge \varphi_3$, and so on. The least preferred are the ones satisfying $\bigwedge_{1 \le i \le m} \neg\varphi_i$. Clearly, the order $\succeq_{T_r}$ is precisely the order $\succeq_r^{ASO}$ given by the ASO rule r. □

There are other ways of translating ASO-rules to P-trees. For instance, it might be beneficial if the translation produced a more balanced tree. Keeping the definitions of φ_i, $1 \le i \le m$, as before and setting $\varphi_{m+1} = B \wedge \neg C_1 \wedge \ldots \wedge \neg C_m$, we could proceed as in the proof of Proposition 1.

For example, if $m = 6$, we build the P-tree T_r^b in Fig. 8, where $\psi_1 = \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4$, $\psi_2 = \varphi_1 \vee \varphi_2$, $\psi_3 = \varphi_1$, $\psi_4 = \varphi_3$, $\psi_5 = \varphi_5 \vee \varphi_6$, and $\psi_6 = \varphi_5$. The indices i's of the formulas ψ_i's indicate the order in which the corresponding formulas are built recursively.

This P-tree representation of a preference r of the form (1) is balanced with the height $\lceil \log_2(m + 1) \rceil$. Moreover, the property in Theorem 1 also holds for the balanced tree T_r^b. The size of T_r^b is in $O(s_r \log s_r)$, where s_r is the size of rule r. It is then larger by the logarithmic factor than T_r but has a smaller depth.

Representing P-trees as RASO-Theories. Preferences represented by compact P-trees cannot in general be captured by ASO preferences without a significant (in some cases, exponential) growth in the size of the representation.
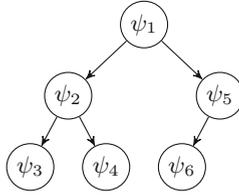
Fig. 8. T_r^b when $m = 6$

However, any P-tree can be represented as a set of *ranked* ASO-rules, or an RASO-theory [3], aggregated by the Pareto method.

We first show how Pareto method is used to order outcomes with regard to a set of *unranked* ASO-rules. Let M and M' be two outcomes. Given a set P of unranked ASO-rules, M is weakly preferred to M' with respect to P, $M \succeq_P^u M'$, if $SD_r(M) \leq SD_r(M')$ for every $r \in P$. Moreover, M is strictly preferred to M', $M \succ_P^u M'$, if $M \succeq_P^u M'$ and $SD_r(M) < SD_r(M')$ for some $r \in P$. Finally, M is equivalent to M', $M \approx_P^u M'$, if $SD_r(M) = SD_r(M')$ for every $r \in P$.

In general, the resulting preference relation is not total. However, by ranking rules according to their importance total preorders can in some cases be obtained. Let us assume $P = \{P_1, \ldots, P_g\}$ is a collection of ranked ASO preferences divided into g sets P_i, with each set P_i consisting of ASO-rules of rank d_i so that $d_1 < d_2 < \ldots < d_g$. We assume that a lower rank of a preference rule indicates its higher importance. We define $M \succeq_P^{rk} M'$ w.r.t P if for every i, $1 \leq i \leq g$, $M \approx_{P_i}^u M'$, or if for some i, $1 \leq i \leq g$, $M \succ_{P_i}^u M'$, and $M \approx_{P_j}^u M'$ for every j, $j < i$.

Given a P-tree T, we construct an RASO-theory Φ_T as follows. We start with $\Phi_T = \emptyset$. For every node t_i in a P-tree T, we update $\Phi_T = \Phi_T \cup \{\varphi_i \overset{d_i}{\leftarrow} conditions\}$, where φ_i is the formula labeling node t_i, d_i, the rank of the ASO-rule, is the depth of node t_i, and *conditions* is the conjunction of formulas φ_j or $\neg\varphi_j$ labeling all nodes t_j that have two children and that are ancestors of t_i in T. We use φ_j in the conjunction if the path from the root to t_i descends from t_j to its left child. Otherwise, we use $\neg\varphi_j$.

For instance, the P-tree T in Fig. 6b gives rise to the following RASO-theory:

$$\neg x_1 \overset{1}{\leftarrow}.$$
$$x_3 \overset{2}{\leftarrow}.$$
$$\neg x_2 \overset{3}{\leftarrow} x_3. \qquad \neg x_4 \overset{3}{\leftarrow} \neg x_3.$$
$$(x_2 \wedge x_4) \vee (\neg x_2 \wedge \neg x_4) \overset{4}{\leftarrow} x_3. \qquad x_2 \overset{4}{\leftarrow} \neg x_3.$$

Theorem 2. *For every P-tree T, the RASO-theory Φ_T has size polynomial in the size of T, and for every two outcomes M and M'*

$$M \succeq_{\Phi_T}^{RASO} M' \text{ iff } M \succeq_T M'$$

Proof. The claim concerning the size of Φ_T is evident from the construction.

($\Leftarrow$) Let us assume $M \succeq_T M'$. Denote by $(\varphi_{i_1}, \ldots, \varphi_{i_j})$ the order of formulas labeling the path determined by M from the root to a leaf. Let φ_{i_k}, $1 \leq k \leq j$, be the first formula that M and M' evaluate differently. Then, $M \models \varphi_{i_k}$ and $M' \not\models \varphi_{i_k}$. Denote by d the depth of φ_{i_k} in T. Based on the construction of Φ_T, for every RASO-rule r of rank less than d, we have $M \approx_r^{ASO} M'$. For every RASO-rule r of rank d, we have $M \succ_r^{ASO} M'$ if r comes from φ_{i_k}, and we have $M \approx_r^{ASO} M'$ for other rules of rank d (in fact, the satisfaction degrees of M and M' on all these other rules are equal to 1). Thus, $M \succeq_{\Phi_T}^{RASO} M'$. If M and M' evaluate all formulas φ_{i_k}, $1 \leq k \leq j$, the same, then $M \approx_{\Phi_T}^{RASO} M'$ and so, $M \succeq_{\Phi_T}^{RASO} M'$, too.

($\Rightarrow$) Towards a contradiction, let us assume that $M \succeq_{\Phi_T}^{RASO} M'$ and $M' \succ_T M$ hold. We again denote by $(\varphi_{i_1}, \ldots, \varphi_{i_j})$ the order of formulas labeling the path determined by M from the root to a leaf. There must exist some formula φ_{i_k}, $1 \leq k \leq j$, such that $M' \models \varphi_{i_k}$, $M \not\models \varphi_{i_k}$, and all formulas φ_ℓ, $1 \leq \ell \leq k-1$, are evaluated in the same way by M and M'. Based on RASO ordering, we have $M' \succ_{\Phi_T}^{RASO} M$, contradiction. $\square$

Hence, the relationship between P-trees and ASO preferences can be summarized as follows. Every ASO preference rule can be translated into a P-tree, and every P-tree into a theory of ranked ASO preference rules. In both cases, the translations have size polynomial in the size of the input. Examining the inverse direction, the size of the ASO rule translated from a P-tree could be exponential, and the orders represented by ranked ASO theories *strictly include* the orders induced by P-trees, as RASO-theories describe *partial* preorders in general.

P-trees Extend Possibilistic Logic. A possibilistic logic theory Π over a vocabulary $\mathcal{I}$ is a set of *preference pairs*

$$\{(\phi_1, a_1), \ldots, (\phi_m, a_m)\},$$

where every ϕ_i is a Boolean formula over $\mathcal{I}$, and every a_i is a real number such that $1 \geq a_1 > \ldots > a_m \geq 0$ (if two formulas have the same importance level, they can be replaced by their conjunction). Intuitively, a_i represents the importance of ϕ_i, with larger values indicating higher importance.

The *tolerance degree* of outcome M with regard to preference pair (ϕ, a), $TD_{(\phi,a)}(M)$, is defined by

$$TD_{(\phi,a)}(M) = \begin{cases} 1, & M \models \phi \\ 1-a, & M \not\models \phi \end{cases}$$

Based on that, the tolerance degree of outcome M with regard to a *set Π* of preference pairs, $TD_\Pi(M)$, is defined by

$$TD_\Pi(M) = min\{TD_{(\phi_i, a_i)}(M) : 1 \leq i \leq m\}.$$

The larger $TD_\Pi(M)$, the more preferred M is.

For example, for the domain of vacations, we might have the following set of preference pairs $\{(\neg x_1 \wedge x_3, 0.8), (x_2 \wedge x_4, 0.5)\}$. According to the possibilistic logic interpretation, vacations satisfying both preferences are the most preferred, those satisfying $\neg x_1 \wedge x_3$ but falsifying $x_2 \wedge x_4$ are next in the preference order, and those falsifying $\neg x_1 \wedge x_3$ are the worst.

Similarly as for ASO-rules, we can apply different methods to encode a possibilistic logic theory in P-trees. Here we discuss one of them. We define T_Π to be an unbalanced P-tree shown in Fig. 7 with labels φ_i defined as follows: $\varphi_1 = \bigwedge_{1 \le i \le m} \phi_i$, $\varphi_2 = \bigwedge_{1 \le i \le m-1} \phi_i \wedge \neg \phi_m$, $\varphi_3 = \bigwedge_{1 \le i \le m-2} \phi_i \wedge \neg \phi_{m-1}$, and $\varphi_m = \phi_1 \wedge \neg \phi_2$.

Theorem 3. *For every possibilistic theory Π, the P-tree T_Π has size polynomial in the size of Π, and for every two outcomes M and M'*

$$M \succeq_\Pi^{Poss} M' \;\; iff \;\; M \succeq_{T_\Pi} M'.$$

Proof. It is clear that the size of the P-tree T_Π is polynomial in the size of Π. Let $mi(M, \Pi)$ denote the maximal index j such that M satisfies all $\phi_1, \ldots, \phi_j$ in Π. (If M falsifies all formulas in Π, we have $mi(M, \Pi) = 0$.) One can show that $M \succeq_\Pi^{Poss} M'$ if and only if $mi(M, \Pi) \ge mi(M', \Pi)$, and $mi(M, \Pi) \ge mi(M', \Pi)$ if and only if $M \succeq_{T_\Pi} M'$. Therefore, the theorem follows.   □

4   Reasoning Problems and Complexity

In this section, we study decision problems on reasoning about preferences described as P-trees, and provide computational complexity results for the three reasoning problems defined below.

Definition 1. *Dominance-testing (*DomTest*): given a P-tree T and two distinct outcomes M and M', decide whether $M \succeq_T M'$.*

Definition 2. *Optimality-testing (*OptTest*): given a P-tree T and an outcome M of T, decide whether M is optimal.*

Definition 3. *Optimality-with-property (*OptProp*): given a P-tree T and some property α expressed as a Boolean formula over the vocabulary of T, decide whether there is an optimal outcome M that satisfies α.*

Our first result shows that P-trees support efficient dominance testing.

Theorem 4. *The* DomTest *problem can be solved in time linear in the height of the P-tree T.*

Proof. The DomTest problem can be solved by walking down the tree. The preference between M and M' is determined at the first non-leaf node n where M and M' evaluate φ_n differently. If such node does not exist before arriving at a leaf, $M \approx_T M'$.   □

An interesting reasoning problem not mentioned above is to decide whether there exists an optimal outcome with respect to the order given by a P-tree. However, this problem is trivial as the answer simply depends on whether there is any outcome at all. However, optimality *testing* is a different matter. Namely, we have the following result.

Theorem 5. *The* OptTest *problem is coNP-complete.*

Proof. We show that the complementary problem, testing non-optimality of an outcome M, is NP-complete. The membership is obvious. A witness of non-optimality of M is any outcome M' such that $M' \succ_T M$, a property that can be verified in linear time (cf. Theorem 4). NP-hardness follows from a polynomial time reduction from SAT [7]. Given a CNF formula $\Phi = c_1 \wedge \ldots \wedge c_n$ over a set of variables $V = \{X_1, \ldots, X_m\}$, we construct a P-tree T and an outcome M as follows.

1. We choose $X_1, \ldots, X_m, unsat$ as issues, where $unsat$ is a new variable;
2. we define the P-tree T_Φ (cf. Fig. 9) to consist of a single node labeled by $\Psi = \Phi \wedge \neg unsat$;
3. we set $M = \{unsat\}$.

We show that $M = \{unsat\}$ is not an optimal outcome if and only if $\Phi = c_1 \wedge \ldots \wedge c_n$ is satisfiable.

($\Rightarrow$) Assume that $M = \{unsat\}$ is not an optimal outcome. Since $M \not\models \Psi$, M belongs to the right leaf and there must exist an outcome M' such that $M' \succ M$. This means that $M' \models \Phi \wedge \neg unsat$. Thus, Φ is satisfiable.

($\Leftarrow$) Let M' be a satisfying assignment to Φ over $\{X_1, \ldots, X_m\}$. Since no $c_i \in \Phi$ mentions $unsat$, we can assume $unsat \notin M'$. So $M' \models \Psi$ and M' is optimal. Thus, $M = \{unsat\}$ is not optimal. $\square$

Fig. 9. The P-tree T_Φ

Theorem 6. *The* OptProp *problem is* Δ_2^P*-complete.*

Proof. (Membership) The problem is in the class Δ_2^P. Let T be a given preference tree. To check whether there is an optimal outcome that satisfies a property α, we start at the root of T and move down. As we do so, we maintain the information about the path we took by updating a formula ψ, which initially is set to $\top$ (a generic tautology). Each time we move down to the left from a node t, we update ψ to $\psi \wedge \varphi_t$, and when we move down to the right, to $\psi \wedge \neg\varphi_t$. To decide whether to move down left or right form a node t, we check if $\varphi_t \wedge \psi$ is satisfiable by making a call to an NP oracle for deciding satisfiability. If $\varphi_t \wedge \psi$

is satisfiable, we proceed to the left subtree and, otherwise, to the right one. We then update t to be the node we moved to and repeat. When we reach a leaf of the tree (which represents a cluster of outcomes), this cluster is non-empty, consists of all outcomes satisfying ψ and all these outcomes are optimal. Thus, returning YES, if $\psi \wedge \alpha$ is satisfiable and NO, otherwise, correctly decides the problem. Since the number of oracle calls is polynomial in the size of the tree T, the problem is in the class Δ_2^P.

(Hardness) The maximum satisfying assignment (MSA) problem[3] [9] is Δ_2^P-complete. We first show that MSA remains Δ_2^P-hard if we restrict the input to Boolean formulas that are satisfiable and have models other than the all-false model (i.e., $\neg x_1 \ldots \neg x_n$).

Lemma 1. *The MSA problem is Δ_2^P-complete under the restriction to formulas that are satisfiable and have models other than the all-false model.*

Proof. The membership in Δ_2^P is evident. Given a Boolean formula Φ over $\{x_1, \ldots, x_n\}$, we define $\Psi = \Phi \vee (x_0 \wedge \neg x_1 \wedge \ldots \wedge \neg x_n)$ over $\{x_0, x_1, \ldots, x_n\}$. It is clear that Ψ is satisfiable, and has at least one model other than the all-false one. Let M be a lexicographically maximum assignment satisfying Φ and assume that M has $x_n = 1$. Extending M by $x_0 = 1$ yields a lexicographically maximum assignment satisfying Ψ and this assignment obviously satisfies $x_n = 1$, too. Conversely, if M is a lexicographically maximum assignment satisfying Ψ and $x_n = 1$ holds in M, then it follows that $M \models \Phi$. Thus, M restricted to $\{x_1, \ldots, x_n\}$ is a lexicographically maximal assignment satisfying Φ and $x_n = 1$. Thus, the unrestricted problem has a polynomial reduction to the restricted one. That proves Δ_2^P-hardness. □

We now show the hardness of the OPTPROP problem by a reduction from this restricted version of the MSA problem. Let Φ be a satisfiable propositional formula over variables $x_1, \ldots, x_n$ that has at least one model other than the all-false one. We construct an instance of the OPTPROP problem as follows. We define the P-tree T_Φ as shown in Fig. 10, where every node is labeled by formula $\Phi \wedge x_i$, and we set $\alpha = x_n$.

Fig. 10. The P-tree T_Φ

---
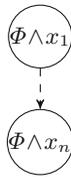
[3] Given a Boolean formula Φ over $\{x_1, \ldots, x_n\}$, the maximum satisfying assignment (MSA) problem is to decide whether $x_n = 1$ in the lexicographically maximum satisfying assignment for Φ. (If Φ is unsatisfiable, the answer is *no*.)

Our P-tree T_Φ induces a total preorder consisting of a sequence of singleton clusters, each containing an outcome satisfying Φ, followed by a single cluster comprising all outcomes that falsify Φ and the all-false model. By our assumption on Φ, the total preorder has at least two non-empty clusters. Moreover, all singleton clusters preceding the last one are ordered lexicographically. Thus, the optimal outcome of T_Φ satisfies α if and only if the lexicographical maximum satisfying outcome of Φ satisfies x_n.   □

5   Conclusion and Future Work

We investigated the qualitative preference representation language of *preference trees*, or *P-trees*. This language was introduced in early 1990s (cf. [5,6]), but have not received a substantial attention as a formalism for preference representation in AI. We studied formally the issue of compact representations of P-trees, established its relationship to other preference languages such as lexicographic preference trees, possibilistic logic and answer-set optimization. For several preference reasoning problems on P-trees we derived their computational complexity.

P-trees are quite closely related to possibilistic logic theories or preference expressions in answer-set optimization. However, they allow for much more structure among formulas appearing in these latter two formalisms (arbitrary trees as opposed to the linear structure of preference formulas in the other two formalisms). This structure allows for representations of conditional preferences. P-trees are also more expressive than lexicographic preference trees. This is the case even for P-trees in which every node is labeled with a formula involving just two issues, as we illustrated with the $00 \succ 11 \succ 01 \succ 01$ example. Such P-trees are still simple enough to correspond well to the way humans formulate hierarchical models of preferences, with all their decision conditions typically restricted to one or two issues.

Our paper shows that P-trees form a rich preference formalism that deserves further studies. Among the open problems of interest are those of learning P-trees and their compact representations, aggregating P-trees coming from different sources (agents), and computing optimal consensus outcomes. These problems will be considered in the future work.

References

1. Booth, R., Chevaleyre, Y., Lang, J., Mengin, J., Sombattheera, C.: Learning conditionally lexicographic preference relations. In: ECAI, pp. 269–274 (2010)
2. Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., Poole, D.: CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. **21**, 135–191 (2004)
3. Brewka, G., Niemelä, I., Truszczynski, M.: Answer set optimization. In: IJCAI, pp. 867–872 (2003)
4. Dubois, D., Lang, J., Prade, H.: A brief overview of possibilistic logic. In: ECSQARU, pp. 53–57 (1991)

5. Fraser, N.M.: Applications of preference trees. In: Proceedings of IEEE Systems Man and Cybernetics Conference, pp. 132–136. IEEE (1993)
6. Fraser, N.M.: Ordinal preference representations. Theory Decis. **36**(1), 45–67 (1994)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
8. Kaci, S.: Working with Preferences: Less Is More. Cognitive Technologies. Springer, Berlin (2011)
9. Krentel, M.W.: The complexity of optimization problems. J. Comput. Syst. Sci. **36**(3), 490–509 (1988)
10. Lang, J., Mengin, J., Xia, L.: Aggregating conditionally lexicographic preferences on multi-issue domains. In: CP, pp. 973–987 (2012)
11. Liu, X., Truszczynski, M.: Aggregating Conditionally lexicographic preferences using answer set programming solvers. In: Perny, P., Pirlot, M., Tsoukiàs, A. (eds.) ADT 2013. LNCS, vol. 8176, pp. 244–258. Springer, Heidelberg (2013)