Preference Trees: A Language for Representing and Reasoning about Qualitative Preferences

Xudong Liu and **Miroslaw Truszczynski**
Department of Computer Science
University of Kentucky
Lexington, KY USA
{liu,mirek}@cs.uky.edu

Abstract

We introduce a novel qualitative preference representation language, *preference trees*, or *P-trees*. We show that the language is intuitive to specify preferences over combinatorial domains and it extends existing preference formalisms such as *LP-trees*, *ASO-rules* and *possibilistic logic*. We study reasoning problems with P-trees and obtain computational complexity results.

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 much attention. 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 (Dubois, Lang, and Prade 1991), conditional preference networks (CP-nets) (Boutilier et al. 2004), lexicographic preference trees (LP-trees) (Booth et al. 2010), and answer-set optimization programs (ASO-theories) (Brewka, Niemelä, and Truszczynski 2003).

In this paper, we focus our study on combinatorial domains with binary issues. We denote by $\{x, \neg x\}$ the domain of each binary issue x. Thus, outcomes in the combinatorial domain determined by the set $\mathcal{I} = \{x_1, \ldots, x_n\}$ of binary issues, we denote it by $CD(\mathcal{I})$, are simply complete and consistent sets of literals over $\mathcal{I}$. 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 from the domain $CD(\mathcal{I})$. Namely, each formula φ represents the set of outcomes that satisfy φ (make φ true).

To give an example, we will consider preferences on possible ways to arrange a vacation. We will assume that vacations are described by four binary variables:

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

A complete and consistent set of literals $\{x_1, \neg x_2, x_3, x_4\}$ 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 or involve water sports can be described by the formula $x_3 \vee \neg x_1$, and vacations in Florida that we travel to by car by the formula $x_2 \wedge x_4$.

Explicitly specifying strict preference orders on $CD(\mathcal{I})$ becomes impractical even for 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. This way of specifying preferences is used (with minor modifications) in possibilistic logic and ASO programs. In our example, the expression

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

states that we prefer summer vacations where we drive by car to vacations in winter in Colorado, 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 preference can be most naturally captured by a form of a decision tree as presented in Figure 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.).

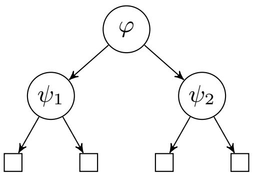
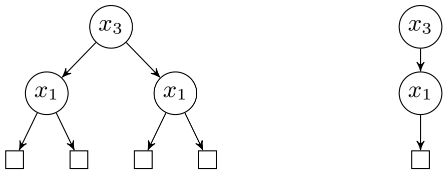


Figure 1: A decision tree

For instance, in the vacation example, a person may prefer summer vacations to winter vacations and, within each group, hiking to water sports. Such preferences can be represented by a decision tree (Figure 2a) which, in this case, can be collapsed, due to identical subtrees, into a compact representation in Figure 2b (we formally introduce collapsed representations below).



(a) Full (b) Compact

Figure 2: Vacations

Such tree representation of preferences, which we call *preference trees*, or P-trees, are reminiscent of LP-trees (Booth et al. 2010). In fact, preference trees generalize LP-trees. In this note, we formally introduce preference trees and their compact representation that exploits occurrences of identical subtrees (as illustrated in Figure 2). We discuss the relationships between preference trees, LP-trees, possibilistic logic theories and ASO preferences. We study the complexity of problems of comparing outcomes with respect to orders defined by preference trees, and of problems of finding optimal outcomes. We conclude by outlining some future research directions.

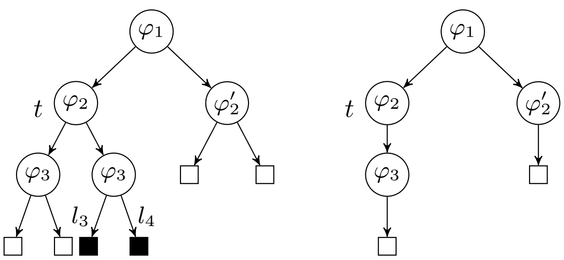
Preference Trees

In this section, we introduce preference trees and discuss their representation. Let $\mathcal{I}$ be a set of binary issues. A *preference tree* (*P-tree*, for short) over $CD(\mathcal{I})$ is a binary tree whose all nodes other than leaves are 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 that is reached by starting at the root of T and proceeding downwards. When at a node N labeled with φ, if $M \models \varphi$, we descend to the left child of N; otherwise, we descend to the right node of N. We denote the leaf of M in T by $l_T(M)$.

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 $M', M'' \in CD(\mathcal{I})$, we set $M' \succeq_T M''$ if $l_T(M') \succeq_T l_T(M'')$, and $M' \succ_T M''$, M' is *strictly preferred* to M'', if $l_T(M') \succ_T l_T(M'')$. (We overload the relations $\succeq_T$ and $\succ_T$ by using it both for the order on the leaves of T and the corresponding preorder on the outcomes from $CD(\mathcal{I})$). 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 consider a person planning her vacation. She prefers vacations that take place in summer or involve hiking ($\varphi_1 = x_1 \vee x_3$) to all others, and this is the most desirable property to her. Among those vacations that satisfy φ_1, our vacation planner prefers hiking vacations in Colorado ($\varphi_2 = x_1 \wedge \neg x_2$) over the remaining ones in that group. Provided φ_1 is satisfied, this is her second most important consideration. Her next concern for vacations satisfying φ_1 is the mode of transportation. She prefers driving to flying for summer vacations and flying to driving, otherwise ($\varphi_3 = (x_3 \rightarrow x_4) \vee (\neg x_3 \rightarrow \neg x_4)$). Among vacations that do not have the property φ_1, that is, the vacations that are in winter and involve water sports, the planner prefers to drive to Florida for her vacation ($\varphi_2' = x_2 \wedge x_4$). The resulting preference preorder on vacations can be represented as a P-tree T shown in Figure 3a. This preorder has six clusters of equivalent outcomes (vacation choices) represented by the six leaves, with the decreasing preference for clusters of outcomes associated the leaves as we move from left to right. To compare two outcomes, $M = (x_1, x_2, \neg x_3, \neg x_4)$ and $M' = (\neg x_1, \neg x_2, x_3, \neg x_4)$, we walk down the trees and find that $l_T(M) = l_3$ and $l_T(M') = l_4$. Thus, $M \succ_T M'$ since l_3 precedes l_4.



(a) Full representation (b) Compact representation

Figure 3: P-trees

Compact Representation of P-Trees. Sometimes P-trees have special structure that allows us to collapse subtrees of certain nodes. In many cases, it results in much smaller representations. A *compact P-tree over $CD(\mathcal{I})$* is a tree such that

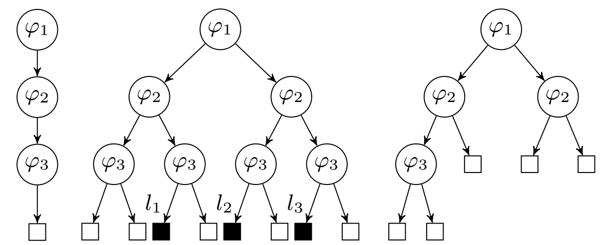
1. every non-leaf node is labeled with a Boolean formula over $\mathcal{I}$; and every leaf node is denoted by a box $\square$, and
2. every non-leaf node t labeled with φ has either one outgoing "straight down" edge, indicating the fact that the two subtrees of t are identical and the formulas labeling every pair of corresponding non-leaf nodes in the two subtrees are the same, or two outgoing edges, with the left out-

going edge representing that φ is satisfied and the right outgoing edge representing that $\neg\varphi$ is satisfied.

Let t be a non-leaf node in a P-tree T. We denote by $Inst(t)$ the set of ancestor nodes of t in T that have two outgoing edges. In the P-tree in Figure 3a, the two subtrees of node t and the formulas labeling the corresponding nodes are identical. Thus, we can collapse them and achieve a compact representation (Figure 3b), where $Inst(t)$ contains only the root of T.

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) size.

A compact P-tree T in Figure 4a represents a full binary tree T' in Figure 4b. The formulas labeling the non-leaf nodes in T are $\varphi_1 = x_1 \vee x_3$, $\varphi_2 = x_2 \vee \neg x_4$ and $\varphi_3 = x_2 \wedge x_3$. We can 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.



(a) T (b) T' (c) T'': pruned T'

Figure 4: P-trees with empty leaves

If we prune all empty leaves in T' (and the nodes whose all descendants are empty leaves), we obtain a P-tree T'' in Figure 4c. No leaf in T'' is empty. It is clear, that it has a larger representation size than the original tree T. That example generalizes and leads to the question of finding small sized representations of P-trees. From now on, we assume that P-trees are given in their compact representation.

P-Trees and Other Formalisms

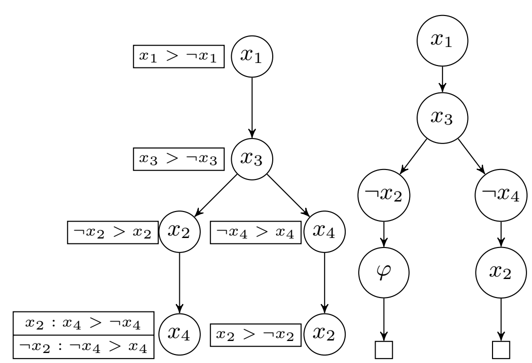
In this section we compare the preference representation language of P-trees with other preference languages.

P-Trees Extend LP-Trees. LP-trees (Booth et al. 2010; Lang, Mengin, and Xia 2012; Liu and Truszczynski 2013) offer a simple and intuitive way to represent strict total orders over combinatorial domains. An *LP tree* T over the set of issues $\mathcal{I}$ is a full *binary tree*. Each node t in T is labeled by an issue from $\mathcal{I}$, denoted by $Iss(t)$, and with *preference information* of the form $a > b$ or $b > a$ indicating which of the two values a and b comprising the domain of $Iss(t)$ is preferred. We require that each issue appears exactly once on each path from the root to a leaf.

Intuitively, the issue labeling the root of an LP tree is of highest importance. Outcomes with the preferred value of that issue are preferred over outcomes with the non-preferred one. The two subtrees refine that ordering. The left subtree determines the ranking of the preferred "upper half" and the right subtree determines the ranking of the non-preferred "lower half." In each case, the same principle is used, with the root issue being the most important one. The precise semantics of an LP tree T captures this intuition. Given an outcome M, we find its preference ranking in T by traversing the tree from the root to a leaf, moving to the left if M assigns the issue of the current node as the preferred value and to the right, otherwise.

In some cases, these decision trees can be collapsed to much smaller representations. For instance, if for some node t, its two subtrees are identical (that is, the corresponding nodes are assigned the same issue), they can be collapsed to a single subtree, with the same assignment of issues to nodes. To retain preference information, at each node t' of the subtree we place a *conditional preference table*, and each preference in it specifies the preferred value for the issue labeling that node given the value of the issue labeling t. In the extreme case when for every node its two subtrees are identical, the tree can be collapsed to a path.

Let us consider the vacation example and assume that to an agent planning a vacation, her most important issue is *activity*, for which she prefers hiking to water sports. No matter whether the vacation is about hiking or water sports, the next most important issue is *time*, where summer is preferred to winter. If the vacation is in summer, the next issue to consider is *destination*, with Colorado being more desirable. The least important issue is *transportation*, where preferences are conditioned upon how the *destination* is evaluated. On the other hand, if the vacation is in winter, the agent treats *transportation* as more important than *destination*, and gives unconditional preferences on both of them. These preferences can be described by an LP-tree L in Figure 5a. It induces a total order on vacations ranging from the leftmost leaf (most desirable vacation) to the rightmost one (least desirable vacation).



(a) An LP-tree L (b) The P-tree T_L

Figure 5: P-trees extend LP-trees

This LP-tree L can be translated into a P-tree T_L shown in Figure 5b, where $\varphi = (x_2 \rightarrow x_4) \vee (\neg x_2 \rightarrow \neg x_4)$. Clearly, the trees L and T_L represent the same ordering over vacations. The example generalizes and allows us to express

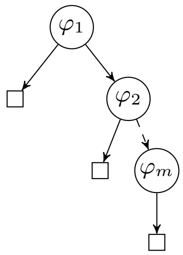


Figure 6: A P-tree T_r (T_P)

any LP-tree as a P-tree, compiling conditional preference tables into formulas labeling nodes. This discussion shows that P-trees encompass LP-trees and do so in a more uniform way, with no need for conditional preference tables.

P-Trees Extend ASO-Rules. The formalism of ASO-rules (Brewka, Niemelä, and Truszczynski 2003) 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 (1) determines the *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.[1]

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:

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

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

Given an ASO-rule r of form (1), we will show how r is encoded in a P-tree. From the ASO-rule r, we build a P-tree T_r in Figure 6, where $\varphi_1 = \neg B \vee C_1$, $\varphi_i = C_i$ ($2 \leq i \leq m$), and the dashed edge represents nodes labeled by the formulas $\varphi_3, \ldots, \varphi_{m-1}$ and every formula φ_i, $3 \leq i \leq m-1$, is constructed such that the parent of φ_i is φ_{i-1}, the left child of φ_i is $\Box$, and the right child of φ_i is φ_{i+1}.

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

Theorem 1. *Given an ASO-rule r, the P-tree T_r is built in time polynomial in the size of r such that for every two outcomes M and M'*

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

Proof. The P-tree T_r induces that the outcomes in the 1st cluster satisfy formula φ_1, the ones in the 2nd cluster satisfy formula $\neg\varphi_1 \wedge \varphi_2$, etc. Note that the right most leaf has formula $\neg\varphi_1 \wedge \ldots \wedge \neg\varphi_m$ which means that if outcome M satisfies B and falsifies all C_i's, M is the least preferred and belongs to cluster $m+1$. $\Box$

Clearly, the size of T_r is linear in the size of the input 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, $1 \leq i \leq m$, as before and setting $\varphi_{m+1} = B \wedge \neg C_1 \wedge \ldots \wedge \neg C_m$, we could proceed as flollows. First, create the root node N of T_r^b and label it with the formula $\bigvee_{1 \leq i \leq \lfloor \frac{m+2}{2} \rfloor} \varphi_i$. Then, proceed recursively to construct N's left subtree T_1 for $\varphi_1, \ldots, \varphi_{\lfloor \frac{m+2}{2} \rfloor}$, and N's right subtree T_2 for $\varphi_{\lfloor \frac{m+2}{2} \rfloor + 1}, \ldots, \varphi_{m+1}$.

For example, if $m = 6$, we build the P-tree T_r^b in Figure 7, 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.

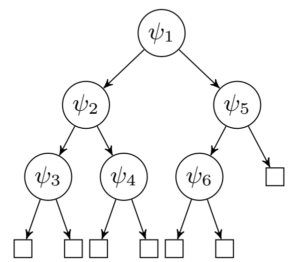


Figure 7: T_r^b when $m = 6$

This P-tree representation of a preference r of the form (1) is balanced and its height is $\lceil \log_2(m+1) \rceil$. Moreover, Theorem 1 also holds for the balanced T_r^b.

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. However, any P-tree can be represented as a set of *ranked* ASO-rules, or an RASO-theory (Brewka, Niemelä, and Truszczynski 2003), 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 $r \in P$, and 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 in some cases, total preorders can be obtained. Let us assume $P = \{P_1, \dots, P_g\}$ is a collection of ranked ASO preferences divided into g sets with each P_i consisting of ASO-rules of the same rank. A *rank* is a positive integer such that ASO-rules of smaller ranks are more important. We define $M \succeq_P^{rk} M'$ w.r.t P if for every i, $1 \le i \le g$, $M \approx_{P_i}^u M'$, or if there exists a rank i such that $M \approx_{P_j}^u M'$ for every j, $j < i$, and $M \succ_{P_i}^u M'$.

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, $\Phi_T = \Phi_T \cup \{\varphi_i \overset{d_i}{\leftarrow} conditions\}$, where φ_i is the formula labeling node t_i, d_i, rank of the ASO-rule, is the depth of node t_i starting with 1 as the depth of the root, and $conditions$ is the conjunction of formulas φ_j or $\neg\varphi_j$ for all nodes t_j such that $t_j \in Inst(t_i)$. Whether φ_j or $\neg\varphi_j$ is used depends on how the path from the root to t_i determines whether descending left (φ_j) or right ($\neg\varphi_j$) at t_j.

For instance, the P-tree T in Figure 3b gives rise to the following RASO-theory:

$$
\begin{array}{l}
x_1 \vee x_3 \overset{1}{\leftarrow} \\
x_1 \wedge \neg x_2 \overset{2}{\leftarrow} x_1 \vee x_3 \\
x_2 \wedge x_4 \overset{2}{\leftarrow} \neg(x_1 \vee x_3) \\
(x_3 \rightarrow x_4) \vee (\neg x_3 \rightarrow \neg x_4) \overset{3}{\leftarrow} x_1 \vee x_3
\end{array}
$$

Theorem 2. *Given a P-tree T, there exists an RASO-theory Φ_T of size polynomial in the size of T such that for every two outcomes M and M'*

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

Proof of Theorem 2 is omitted due to space constraint.

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

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

where every ϕ_i is a Boolean formula over $\mathcal{I}$, and every a_i is a real number such that $1 \ge a_1 > \dots > a_m \ge 0$. Intuitively, a_i means the importance of ϕ_i, the larger the more importance. Denote by $TD_{(\phi,a)}(M)$ the *tolerance degree* of outcome M with regard to preference pair (ϕ, a), and we have the following.

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

Denote by $TD_\Pi(M)$ the tolerance degree of outcome M with regard to the possibilistic logic theory Π, and we define that

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

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

For example, we have a theory of possibilistic theory $\{(x_1 \wedge x_3, 0.8), (x_2 \wedge x_4, 0.5)\}$ on the domain of vacations, which expresses that vacations satisfying both preferences are the most preferred, those satisfying $x_1 \wedge x_3$ but falsifying $x_2 \wedge x_4$ are the next preferred, and those falsifying $x_1 \wedge x_3$ are the worst.

Like the case with ASO-rules, we can apply different methods to encode a possibilistic logic theories in P-trees. Here we discuss one of them. Given a possibilistic logic theory Π, we now build an unbalanced P-tree T_Π (Figure 6 with different formula labels), where $\varphi_1 = \bigwedge_{1 \le i \le m} \phi_i$, $\varphi_2 = \bigwedge_{1 \le i \le m-1} \phi_i \wedge \neg\phi_m$, $\varphi_3 = \bigwedge_{1 \le i \le m-2} \phi_i \wedge \neg\phi_{m-1}$, and $\varphi_m = \phi_1 \wedge \neg\phi_2$.

Theorem 3. *Given a possibilistic theory Π, the P-tree T_Π is built in time polynomial in the size of Π such that for every two outcomes M and M'*

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

Proof. Note that both induce in general total preorders of $m+1$ clusters. It is clear that outcome M is in the i-th cluster induced by Π if and only if it is in the i-th cluster induced by T_Π. □

Reasoning Problems and Complexity

In this section, we study decision problems on reasoning about preferences described as P-trees, and provided computational complexity results of these problems.

Definition 1. Dominance-testing (DomTest): given a P-tree T and its 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 an optimal outcome.

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.

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, $M \approx_T M'$. □

One interesting reasoning problem on optimality is to decide if there exists an optimal outcome in a P-tree. This problem is trivial because its solution is the solution to the problem deciding whether there is any outcome at all.

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

Proof Sketch. Need to show that deciding whether the given outcome M is *not* an optimal outcome in a given P-tree T is NP-complete. This complement problem is in class NP because one can guess an outcome M' in polynomial time and verify in polynomial time that $M' \succ_T M$. Hardness follows from a polynomial time reduction from SAT (Garey and Johnson 1979). Details of the reduction is omitted due to limited space. □

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

Proof. (Membership) The problem is in 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[2] (Krentel 1988) is Δ_2^P-complete. We first show in Lemma 1 that the problem remains Δ_2^P-hard if we restrict the input to Boolean formulas that are satisfiable and has 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 when Φ is satisfiable and has models other than the all-false model.*

Proof. 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 M has $x_n = 1$. Extending M by $x_0 = 1$ yields a lexicographically maximum assignment satisfying Ψ and this assignment satisfies $x_n = 1$. Conversely, if M is a lexicographically maximum assignment satisfying Ψ and $x_n = 1$ is in M, it follows that $M \models \Phi$. Thus, restricted M to $\{x_1, \ldots, x_n\}$, the assignment is lexicographically maximal satisfying Φ. □

We 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.

(1). The P-tree T_Φ is shown in Figure 8, where every node is labeled by formula $\Phi \wedge x_i$.

(2). The property $\alpha = x_n$.

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 clusters. Moreover, all singleton clusters preceding the last one are ordered

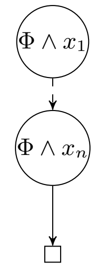


Figure 8: The P-tree T_Φ

lexicographically. Thus, the optimal outcome of T_Φ satisfies α if and only if the lexicographical maximum satisfying outcome of Φ satisfies x_n. □

Conclusion and Future Work

We introduced a novel qualitative preference representation language, *preference trees*, or *P-trees*. We studied the properties of the language and several reasoning problems, and obtained computational complexity results. For the future work, we will study the relationship between P-trees and CP-nets, and problems concerning preference learning and reasoning (e.g., preference aggregation) for P-trees. For aggregating P-trees, we will investigate the applicability of the Pareto method as well as voting rules in social choice theory.

References

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

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

Brewka, G.; Niemelä, I.; and Truszczynski, M. 2003. Answer set optimization. In *IJCAI*, 867–872.

Dubois, D.; Lang, J.; and Prade, H. 1991. A brief overview of possibilistic logic. In *ECSQARU*, 53–57.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Krentel, M. W. 1988. The complexity of optimization problems. *J. Comput. Syst. Sci.* 36(3):490–509.

Lang, J.; Mengin, J.; and Xia, L. 2012. Aggregating conditionally lexicographic preferences on multi-issue domains. In *CP*, 973–987.

Liu, X., and Truszczynski, M. 2013. Aggregating conditionally lexicographic preferences using answer set programming solvers. In *ADT*, 244–258.

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