Towards Fast Algorithms for the Preference Consistency Problem Based on Hierarchical Models

Abstract

In this paper, we construct and compare algorithmic approaches to solve the Preference Consistency Problem for preference statements based on hierarchical models. Instances of this problem contain a set of preference statements that are direct comparisons (strict and non-strict) between some alternatives, and a set of evaluation functions by which all alternatives can be rated. An instance is consistent based on hierarchical preference models, if there exists an hierarchical model on the evaluation functions that induces an order relation on the alternatives by which all relations given by the preference statements are satisfied. Deciding if an instance is consistent is known to be NP-complete for hierarchical models.

We develop three approaches to solve this decision problem. The first involves a Mixed Integer Linear Programming (MILP) formulation, the other two are recursive algorithms that are based on properties of the problem by which the search space can be pruned. Our experiments on synthetic data show that the recursive algorithms are faster than solving the MILP formulation and that the ratio between the running times increases extremely quickly.

1 Introduction

In many fields like recommender systems and multi-objective optimization, one wants to reason over user preferences. In these problems, it is often difficult or excessively time-consuming to elicit all user preferences. The Preference Deduction Problem (PDP) aims at eliciting only few preferences and inferring more preferences from the given ones; this might then be used in a conversational recommender system, for example, to help choose which items to show to the user next [Bridge and Ricci, 2007; Trabelsi et al., 2011]. Typically, an assumption is made on the type of order relation that the user (implicitly) uses to express the preference statements. Such order relations can be, e.g., comparing alternatives by the values of the evaluation functions lexicographically [Wilson, 2014], by Pareto order, weighted sums [Figueira et al., 2005] or based on hierarchical models (a generalization of lexicographic models) [Wilson et al., 2015]. Here, the choice of the order relation can lead to stronger or weaker inferences and can make solving PDP computationally more or less challenging. In a recommender system, or e.g., in a multi-objective decision making scenario, the user should only be presented with a relatively small number of solutions, hence, a strong order relation is required. Using PDP based on a lexicographic models has been shown to be successful in reducing the number of solutions, however computationally can be more expensive. See [Marinescu et al., 2013] and [George et al., 2015] for comparisons between order relations in a multi-objective optimization framework. While PDP based on hierarchical models yields an even lower number of solutions, it is coNP-complete and computationally expensive. The approach taken by the Preference Deduction Problem contrasts learning a single model that coincides with the user preferences as in [Fürnkranz and Hüllermeier, 2010; Dombi et al., 2007; Flach and Matsubara, 2007; Bräuning and Hüllermeier, 2012; Booth et al., 2010].

In this paper, we concentrate on a problem equivalent to PDP: the Preference Consistency Problem (PCP) based on hierarchical preference models. This is the problem of deciding whether given user preferences are consistent, i.e., not contradicting each other. In terms of hierarchical models, PCP determines whether there exists a hierarchical model of evaluation functions such that the induced order relation on the alternatives satisfies all preference statements given by the user. It is known that for hierarchical preference models, PCP is NP-complete and PCP and PDP are mutually expressible [Wilson et al., 2015]. The main issue in this paper is to find fast algorithms to solve PCP for hierarchical models by exploiting the problem’s structure. We compare the running time of the algorithmic approaches on a set of synthetic data.

The next section gives an introduction to hierarchical preference models, their induced order relation on alternatives, and the Preference Consistency Problem. Section 3 gives a Mixed Integer Linear Programming formulation for PCP. Section 4 discusses the exploitation
of properties of PCP instances yielding two recursive algorithms. In Section 5, the conducted experiments and their outcome are described. The last section concludes.

2 The Consistency Problem Based on Hierarchical Models

For a set of preference statements, consistency is defined to be the property that the statements don’t contradict each other. To formally define this term, we first describe the concept of hierarchical models.

2.1 Hierarchical Models

Hierarchical models will from here on be called HCLP models, where HCLP stands for “Hierarchical Constraint Linear Program” and points out the resemblance of the hierarchical order of the evaluation functions in HCLP models to the order of soft constraints in an HCLP. Here, we define HCLP structures, models and their implied order relation that is a kind of lexicographic order.

Definition 2.1 (HCLP structures). An HCLP structure \( \langle A, \oplus, C \rangle \) is a triple. Here, \( A \) is a finite set of alternatives and \( C \) is a set of evaluation functions from the alternatives \( A \) to the non-negative rational numbers \( \mathbb{Q}^{\geq 0} \). \( \oplus \) is an associative, commutative, and strictly monotonic operation on \( \mathbb{Q}^{\geq 0} \), i.e., \( x \oplus y < z \oplus y \) if and only if \( x < z \).

In an HCLP structure, the evaluations \( C \) as well as their combinations by \( \oplus \) provide ratings of the alternatives due to unfavorable aspects, e.g., costs, for which we assume that \( 0 \) is the best value. The notion of HCLP structures was first introduced in [Wilson et al., 2015] with \( \oplus \) as an associative, commutative and monotonic operation. In this paper, we consider \( \oplus \) to be a strictly monotonic operation as this yields interesting properties allowing us to formulate fast algorithms for checking consistency. Furthermore, we assume the operation to be computable in logarithmic time. These assumptions still allow interesting operators like addition and multiplication which seem to be a natural way of combining aspects like costs and distances, but exclude a minimum or maximum operator which could be desired sometimes.

For a subset \( C \subseteq C \) of evaluations, we define the weak order (transitive and complete binary order) \( \preceq_C \) on the set of alternatives in the following way: for \( \alpha, \beta \in A \), \( \alpha \preceq_C \beta \) if and only if \( \bigoplus_{c \in C} c(\alpha) \leq \bigoplus_{c \in C} c(\beta) \). The corresponding strict order \( \prec_C \) is given by \( \alpha \prec_C \beta \) if and only if \( \alpha \preceq_C \beta \) and \( \alpha \not\preceq_C \beta \), i.e., \( \bigoplus_{c \in C} c(\alpha) < \bigoplus_{c \in C} c(\beta) \).

Then, the equivalence relation \( \equiv_C \) is given by \( \alpha \equiv_C \beta \) if and only if \( \alpha \preceq_C \beta \) and \( \alpha \not\preceq_C \beta \), i.e., \( \bigoplus_{c \in C} c(\alpha) = \bigoplus_{c \in C} c(\beta) \). For \( C = \emptyset \), \( \alpha \equiv_C \beta \) for all \( \alpha, \beta \in A \).

Definition 2.2 (HCLP models). An HCLP model \( H \) for an HCLP structure \( \langle A, \oplus, C \rangle \) is an ordered partition of a subset \( \sigma(H) \subseteq C \) of evaluations. We write \( H \) as sequence \( (C_1, \ldots, C_k) \), where the (possibly empty) sets \( \emptyset \subseteq C_1, \ldots, C_k \subseteq C \) are disjoint and \( \bigcup_{i=1}^k C_k = \sigma(H) \).

We say \( C_i \) is the i-th level set in \( H \). We denote the empty HCLP model with \( \sigma(H) = \emptyset \) by \( H = \langle \rangle \).

An HCLP model can be viewed as a hierarchy on the evaluation functions. For HCLP model \( H = (C_1, \ldots, C_k) \) the level set \( C_1 \) contains the most important evaluation functions; the level set \( C_2 \) contains the second most important evaluation functions and so on. Evaluations \( C \setminus \sigma(H) \) that are not included in the HCLP model are irrelevant for rating the alternatives. Accordingly, we compare two alternatives first on a combination by \( \oplus \) of the most important evaluation functions; only if these are equal do we compare them on the combination of the next most important evaluations. Hence, each HCLP model \( H = (C_1, \ldots, C_k) \) implies a weak order \( \preceq_H \) on the alternatives that is a lexicographic order on combinations of evaluations. More specifically, for two alternatives \( \alpha, \beta \in A \), \( \alpha \preceq_H \beta \) if and only if

(I) for all \( i = 1, \ldots, k \), \( \alpha \equiv_{C_i} \beta \); or

(II) there exists some \( i \in \{1, \ldots, k\} \) such that

\[ \alpha \prec_{C_i} \beta \quad \text{and for all } 1 \leq j < i, \alpha \equiv_{C_j} \beta. \]

Similarly to \( \preceq_C \) and \( \equiv_C \), we define the strict weak order \( \prec_H \) and the equivalence relation \( \equiv_H \). For \( \beta \in A \), \( \alpha \prec_H \beta \) if and only if \( \alpha \preceq_H \beta \) and \( \alpha \not\preceq_H \beta \) (i.e., condition (II) holds). Analogously, \( \alpha \equiv_H \beta \) if and only if \( \alpha \preceq_H \beta \) and \( \alpha \not\preceq_H \beta \) (i.e., condition (I) holds). Note, that by these definitions level sets of HCLP models can be empty, but empty level sets do not affect the relations \( \preceq_H \), \( \prec_H \) and \( \equiv_H \) or any statements based on these relations.

In this paper, we consider special classes of HCLP models where the sizes of the level sets are bounded by some constant. The class \( C(t) \) is defined to be the set of HCLP models with level sets that contain at most \( t \) evaluations, i.e., if \( H = (C_1, \ldots, C_k) \) is in \( C(t) \) then \( |C_i| \leq t \) for all \( i = 1, \ldots, k \). Note, that the model classes satisfy the relation \( C(s) \subseteq C(t) \) for \( s \leq t \). Class \( C(1) \) contains standard lexicographic models.

Example. Consider the choice of alternatives Apple Pie (AP), Chocolate Cake (CC) and Ice Cream (IC). The desserts are rated by the evaluation functions: calories \((c)\), sugar \((s)\), and fat \((f)\). The values of \( C = \{c, s, f\} \) are percentages of the recommended daily intake as shown in Table 1. Here, 0 is the best possible value, meaning 0% sugar, calories or fat of the recommended daily intake is contained in the dessert. Let \( \oplus \) be the standard addition on \( \mathbb{Q} \).

<table>
<thead>
<tr>
<th>C</th>
<th>AP</th>
<th>CC</th>
<th>IC</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>10</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>s</td>
<td>23</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>f</td>
<td>20</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>f + s</td>
<td>43</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 1: Values of \( c, s, f, f \oplus s \) evaluated on AP, CC, IC.
Deduction (decision) Problems for classes

\( C \) does \( C \) Then (1) is a strict preference statement, (2) is a non-\( \beta \) \( C \) \( \langle A \rangle \langle A \rangle \) \( H \) say \( H \) Proposition 2.5.

but never satisfies a strict preference statement. Thus, \( s \prec t \) preference statements in \( \text{noted} \) We say \( \text{for} \) \( r \) version of preference statements \( \Gamma \) \( \phi \) \( \{ \} \) \( \alpha \phi \leq \beta \phi \), and \( \varphi \in \mathcal{L}_2^\phi \) as \( \alpha \phi < \beta \phi \) for \( \alpha \phi, \beta \phi \in \mathcal{A} \). We denote the non-strict version of preference statements \( \Gamma \subseteq \mathcal{L}^A \) by \( \Gamma^{(\leq)} \), i.e., \( \Gamma^{(\leq)} = \{ \alpha \phi \leq \beta \phi \mid \varphi \in \Gamma \} \).

Definition 2.3 (Satisfaction of Preference Statements). Let \( H \) be an HCLP model for HCLP structure \( \langle A, \oplus, C \rangle \). We say \( H \) satisfies a preference statement \( \varphi \in \mathcal{L}_2^\phi \) denoted \( H \models \varphi \), if \( \alpha \phi \preceq \beta \phi \) for \( \varphi \in \mathcal{L}_2^\phi \); or \( \alpha \phi \not\preceq \beta \phi \) for \( \varphi \in \mathcal{L}_2^\phi \). Furthermore, \( H \) satisfies a set of preference statements \( \Gamma \subseteq \mathcal{L}^A \), denoted \( H \models \Gamma \), if \( H \) satisfies all preference statements in \( \Gamma \), i.e., \( H \models \varphi \) for all \( \varphi \in \Gamma \).

Definition 2.4 (Consistency). Let \( \langle A, \oplus, C \rangle \) be an HCLP structure and \( t \in \{ 1, \ldots, |C| \} \) a constant. We say \( \Gamma \subseteq \mathcal{L}^A \) is \( \mathcal{C}(t) \)-consistent, if there exists an HCLP model \( H \in \mathcal{C}(t) \) such that \( H \models \Gamma^{(\leq)} \).

It is easy to show that \( \Gamma \) is \( \mathcal{C}(t) \)-consistent if \( \Gamma \) is \( \mathcal{C}(s) \)-consistent for some \( s < t \). Note, that the empty model \( H = \emptyset \) always satisfies a non-strict preference statement, but never satisfies a strict preference statement. Thus, a set \( \Gamma \subseteq \mathcal{L}_2^\phi \) is always consistent.

Example (continued). Consider preference statements:

(1) I strictly prefer ice cream to apple pie (\( IC < AP \)).
(2) I prefer chocolate cake to apple pie (\( CC \leq AP \)).

Then (1) is a strict preference statement, (2) is a non-strict preference statement. As before, HCLP model \( H = \{(c, s), \{f\}\} \) satisfies \( IC \prec_H AP \) and \( CC \preceq_H AP \). Thus, (1) and (2) together are consistent.

We can now formulate the Preference Consistency and Deduction (decision) Problems for classes \( \mathcal{C}(t) \).

\( \mathcal{C}(t) \) Preference Consistency Problem (\( \mathcal{C}(t) \)-PDP): Given HCLP structure \( \langle A, \oplus, C \rangle \), constant \( t \in \{ 1, \ldots, |C| \} \) and set of preference statements \( \Gamma \subseteq \mathcal{L}^A \). Is \( \Gamma \) \( \mathcal{C}(t) \)-consistent?

\( \mathcal{C}(t) \) Preference Deduction Problem (\( \mathcal{C}(t) \)-PDP): Given HCLP structure \( \langle A, \oplus, C \rangle \), constant \( t \in \{ 1, \ldots, |C| \} \), preference statements \( \Gamma \subseteq \mathcal{L}^A \) and \( \varphi \in \mathcal{L}^A \). Does \( H \models \varphi \) hold for all \( H \in \mathcal{C}(t) \) with \( H \models \Gamma^{(\leq)} \)?

The next proposition by [Wilson et al., 2015] shows \( \mathcal{C}(t) \)-PCP and \( \mathcal{C}(t) \)-PDP are mutually expressive.

Proposition 2.5. \( H \models \varphi \) for all \( H \in \mathcal{C}(t) \) with \( H \models \Gamma^{(\leq)} \) if and only if \( \Gamma \cup \{-\varphi\} \) is \( \mathcal{C}(t) \)-inconsistent, where \( \neg \varphi = \beta \phi < \alpha \phi \leq \alpha \phi \) for \( \varphi \in \mathcal{L}_2^\phi \).

Furthermore, [Wilson et al., 2015] established that \( \mathcal{C}(t) \)-PDP is coNP-complete for any \( t \geq 2 \) (even if \( \oplus \) is strictly monotonic). Thus \( \mathcal{C}(t) \)-PDP is NP-complete. The special cases \( \mathcal{C}(1) \)-PDP and \( \mathcal{C}(1) \)-PDP can be solved in time \( O(|C| \cdot |\Gamma|) \) by a greedy algorithm.

3 MILP Formulation

We describe a Mixed Integer Linear Programming (MILP) formulation for \( \mathcal{C}(t) \)-PDP. Here, we assume \( \oplus \) to be the standard addition on the integers and the evaluation functions to be integral.

For HCLP structure \( \langle A, \oplus, C \rangle \) and preference statements \( \Gamma \subseteq \mathcal{L}^A \), let \( n = |C| \) be the number of evaluations.

Assigning Evaluations to Level Sets: We introduce a matrix of Boolean variables \( Y \in \{0, 1\}^{n \times n} \) such that \( y_{i,j} = 1 \) if and only if evaluation \( i \) is included in the \( j \)-th level set of the solution HCLP model. For \( \mathcal{C}(t) \)-PDP, every evaluation is contained in at most one level set and the cardinality of the level sets is bounded by \( t \).

\[
\sum_{j=1}^{n} y_{i,j} \leq 1 \quad \text{and} \quad \sum_{i=1}^{n} y_{i,j} \leq t \quad \forall i, j = 1, \ldots, n. \tag{1}
\]

Maintaining Values of \( \oplus \)-combined Level Sets: The matrix of integer variables \( X \in \mathbb{Q}^{n \times |\Gamma|} \) contains the values of the combined evaluation functions in the level sets for the alternatives of the preference statements, i.e., \( x_{i,\varphi} = \bigoplus_{\varphi \in \mathcal{C}(t) \subseteq \varphi \subseteq H} c(\alpha \phi)\). Thus, \( x_{i,\varphi} \) maintains the exact degree of support/opposition of the preference statement \( \varphi \) in the \( i \)-th level set and

\[
\sum_{i=1}^{n} y_{i,j}(c(\alpha \phi) - c(\beta \phi)) = x_{j,\varphi} \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma. \tag{2}
\]

We define upper and lower bounds \( M_{\varphi} \geq x_{j,\varphi} \geq m_{\varphi} \) for all \( j = 1, \ldots, n \) and \( \varphi \in \Gamma \) by

\[
m_{\varphi} = \min_{E \subseteq C} \sum_{\varphi \in E} c(\alpha \phi) - c(\beta \phi) = \sum_{E \subseteq C} c(\alpha \phi) - c(\beta \phi), \quad \text{and} \quad M_{\varphi} = \max_{E \subseteq C} \sum_{\varphi \in E} c(\alpha \phi) - c(\beta \phi) = \sum_{E \subseteq C} c(\alpha \phi) - c(\beta \phi).
\]

Maintaining the Sign for Supporting, Opposing and Indifferent Level Sets: We introduce Boolean variables \( s_{j,0}^0, s_{j,0}^+ \) and \( s_{j,0}^- \) to express the sign for each variable \( x_{j,\varphi} \). In particular, \( s_{j,0}^0 = 1 \) if and only if \( x_{j,\varphi} < 0 \), \( s_{j,0}^+ = 1 \) if and only if \( x_{j,\varphi} > 0 \), and \( s_{j,0}^- = 1 \) if and only if \( x_{j,\varphi} = 0 \). Since only one of the relations can hold,

\[
s_{j,\varphi}^0 + s_{j,\varphi}^+ + s_{j,\varphi}^- = 1 \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma. \tag{3}
\]

To enforce the equivalences, we make use of the bounds \( M_{\varphi} \) and \( m_{\varphi} \) and the integrity of the evaluations. In particular, we use that the lowest positive value is 1 and the highest negative value \( x_{j,\varphi} \) can take is \(-1\).

For the implication \( s_{j,0}^0 = 1 \Rightarrow x_{j,\varphi} < 0 \), we formulate:

\[
x_{j,\varphi} + s_{j,\varphi}^0(M_{\varphi} + 1) \leq M_{\varphi} \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma. \tag{4}
\]
For the implication \( s^{0}_{j,\varphi} > 0 \Rightarrow x_{j,\varphi} > 0 \), we formulate:
\[
x_{j,\varphi} + s^{0}_{j,\varphi}(m_{\varphi} - 1) \geq m_{\varphi} \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma.
\]
Finally, we enforce \( s^{0}_{j,\varphi} = 1 \Rightarrow x_{j,\varphi} = 0 \) by
\[
x_{j,\varphi} - (1 - s^{0}_{j,\varphi})m_{\varphi} \geq 0 \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma \text{ and} \quad (6)
\]
\[
x_{j,\varphi} - (1 - s^{0}_{j,\varphi})M_{\varphi} \leq 0 \quad \forall j = 1, \ldots, n, \forall \varphi \in \Gamma. \quad (7)
\]
Since only one of the relations \( x_{j,\varphi} < 0 \), \( x_{j,\varphi} > 0 \) or \( x_{j,\varphi} = 0 \) can be true, the equivalences follow.

**Satisfaction of Strict and Non-strict Statements:**
Following the definition of \( \leq_{H} \), the HCLP model corresponding to the variable assignments of \( Y \) satisfies a non-strict statement \( \varphi \) in \( \Gamma \) if and only if
\[
(\forall i) \text{ for all } i = 1, \ldots, n, s^{0}_{i,\varphi} = 1; \text{ or}
\]
\[
(\forall i) \text{ there exists some } i \in \{1, \ldots, n\} \text{ such that } s^{0}_{i,\varphi} = 1
\]

Analogously, a strict statements \( \varphi \) in \( \Gamma \) is satisfied if and only if \( (\forall i) \) holds. It is easy to check that conditions \((\forall i)\) or \((\forall i)\) holding for all \( \varphi \in \Gamma \) is equivalent to
\[
\sum_{j=1}^{n} s^{0}_{j,\varphi} \geq s^{0}_{i,\varphi} \quad \forall i = 1, \ldots, n, \forall \varphi \in \Gamma. \quad (8)
\]
Inequality \((9)\) yields the satisfaction of \( \Gamma^{(\leq)} \). We enforce satisfaction of all strict statements in \( \Gamma \) by
\[
\sum_{j=1}^{n} s^{0}_{j,\varphi} \geq 1 \quad \forall \varphi \in \Gamma \cap \mathcal{L}^{\leq}_{<}. \quad (9)
\]

The constraints \((1)-(9)\) form a complete Mixed Integer Linear Program formulation for \( \mathcal{C}(t)\)-PCP. Note that constraints \((3)-(9)\) could be replaced by weighted sums with extreme weights to enforce the lexicographic order on the alternatives. These inequalities, however, lead to numerical difficulties for the solver.

4 **Recursive Algorithms**

In the following, we describe two recursive search algorithms for \( \mathcal{C}(t)\)-PCP. Both algorithms are based on properties of the problem that can be used to prune the search space. In general, we try constructing a \( \Gamma \)-satisfying HCLP model by sequentially adding new level sets to the model that do not oppose any preference statement that is not strictly satisfied so far. Thus, during the algorithm the current model always satisfies \( \Gamma^{(\leq)} \) the non-strict version of \( \Gamma \). We backtrack when the current model cannot be extended further and the model does not satisfy all strict preference statements. The two approaches aim to reduce the number of \( \Gamma^{(\leq)} \)-satisfying HCLP models constructed by the algorithm before deciding consistency. In particular, we try to identify and ignore level sets which cannot lead to a \( \Gamma \)-satisfying HCLP model although not opposing the preference statements.

**Utilising Sequences of Singleton Level Sets:** The first approach is based on the idea of including as many singleton level sets as possible. This seems computationally less challenging since a \( \Gamma^{(\leq)} \)-satisfying sequence of singleton level sets that is maximal in the number of level sets can be found in time \( O(|\mathcal{C}(\Gamma)|) \) [Wilson et al., 2015].

Remember, that \( \Gamma^{(\leq)} \) is defined to be the non-strict version of \( \Gamma \), i.e. \( \Gamma^{(\leq)} = \{\alpha_{\varphi} \leq \beta_{\varphi} \mid \varphi \in \Gamma\} \). In the following we show that for strictly monotonic operators \( \oplus \) the recursive search algorithm never needs to backtrack over the choice of such singleton sequences. We first establish the following property for strictly monotonic operators \( \oplus \) which can be shown by a short technical proof.

**Lemma 4.1.** Let \( \oplus \) be a strictly monotonic operator and let \( X, Y \subseteq \mathcal{C}(t) \) be sets of evaluation functions with \( X \subseteq Y \). Let \( \alpha, \beta \in \mathcal{A} \) be alternatives such that \( X \) is indifferent under \( \alpha \) and \( \beta \), i.e., \( \alpha \equiv_{Y} \beta \). Then \( \alpha \leq_{Y} \beta \) if and only if \( \alpha \leq_{Y} Y \beta \). Hence, \( \alpha \equiv_{Y} \beta \) if and only if \( \alpha \equiv_{Y} \beta \) if and only if \( \alpha \leq_{Y} \beta \). Hence, \( \alpha \equiv_{Y} \beta \) if and only if \( \alpha \leq_{Y} \beta \).

We define the (non-commutative) combination of two HCLP models \( H = (C_{1}, \ldots, C_{l}) \) and \( H' = (C'_{1}, \ldots, C'_{l}) \) in \( C(t) \) as \( H \circ H' = (C'_{1}, \ldots, C_{l}) \), where \( \sigma_{H} = \bigcup_{i=1}^{l} C_{i} \). It is easy to see that \( H \circ H' \) is an HCLP model in \( C(t) \) as well. The following proposition shows how the satisfaction of preference statements \( \Gamma \) of \( H \) persists under combination with sequences of singleton level sets that only satisfy \( \Gamma^{(\leq)} \).

**Proposition 4.2.** Let \( \langle \mathcal{A}, \mathcal{C}, \oplus \rangle, \Gamma \rangle \) be an instance of \( \mathcal{C}(t)\)-PCP. If \( H = (c_{1}, \ldots, c_{l}) \) is a \( \Gamma^{(\leq)} \)-satisfying model in \( \mathcal{C}(1) \) and \( H' = (C'_{1}, \ldots, C'_{l}) \) is a \( \Gamma \)-satisfying model in \( C(t) \), then \( H \circ H' \) is a \( \Gamma \)-satisfying model in \( C(t) \).

**Proof.** We show, that \( H \circ H' \) satisfies \( \Gamma^{(\leq)} \) and strictly satisfies the same preference statements as \( H' \). Hence, \( H \circ H' \) is a \( \Gamma \)-satisfying HCLP model in \( C(t) \).

Recall that a preference statement \( \varphi \) is strictly satisfied when there exists a level set \( C \) supporting \( \varphi \), i.e., \( \alpha_{\varphi} \sim C \beta_{\varphi} \), and all preceding level sets \( C' \) are indifferent under \( \varphi \), i.e., \( \alpha_{\varphi} \equiv_{C'} \beta_{\varphi} \). Hence, the preference statements in \( \Gamma \) that are strictly satisfied by \( H = (c_{1}, \ldots, c_{l}) \) are also strictly satisfied by \( H \circ H' = (c'_{1}, \ldots, c_{l}) \setminus \sigma_{H} \). Let \( \Gamma' \) be the set of remaining preference statements that are not strictly satisfied by \( H \). Since \( H \) satisfies \( \Gamma^{(\leq)} \), it is indifferent under all statements in \( \Gamma' \), i.e., \( c_{i} \alpha_{\varphi} = c_{i} \beta_{\varphi} \) for all \( 1 \leq i \leq l \). Consider an arbitrary level set \( C = C'_{i} \) in \( H' \) and a preference statement \( \varphi \in \Gamma' \). Repeatedly applying Lemma 4.1 for the singleton level sets in \( H \cap C \) yields: \( c_{i} \alpha_{\varphi} \sim C_{i} \beta_{\varphi} \) if and only if \( \alpha_{\varphi} \sim_{H \cap C} \beta_{\varphi} \), where \( \sim \) is one of the relations \( \sim, \equiv \) or \( \succ \). Thus, the level sets \( C_{i} \setminus \sigma_{H} \) in \( H \circ H' \) have the same relations towards some \( \varphi \in \Gamma' \) as the level sets \( C'_{i} \setminus \sigma_{H} \). Since the initial singleton sequence in \( H \circ H' \) is indifferent under preference statements \( \varphi \in \Gamma' \), \( H \circ H' \) satisfies \( \varphi \) if and only if \( H' \) satisfies \( \varphi \). Also, all statements \( \Gamma \setminus \Gamma' \) are strictly satisfied by \( H \circ H' \). Hence, \( H \circ H' \) satisfies \( \Gamma^{(\leq)} \) and strictly satisfies all statements in \( \Gamma \) that \( H \) strictly satisfies. \( \square \)

Proposition 4.2 immediately leads to the next result.
Proposition 4.3. Let \( H \) be a \( \Gamma(\leq) \)-satisfying model in \( C(1) \) that consists of a maximal number of singleton level sets. If \( \Gamma \) is \( C(t) \)-consistent, then there exists a \( \Gamma \)-satisfying model with \( H \) as initial sequence.

Based on Proposition 4.3, we describe the algorithm PC-check that solves PCP by trying to construct a \( \Gamma \)-satisfying HCLP model. This method is summarised in Algorithm 1 (disregarding the framed parts). After finding an initial singleton sequence \((c_1, \ldots, c_k)\) that is maximal while satisfying \( \Gamma(\leq) \), we consider every possible (not opposing) level set \( C \) of size \( 2 \leq |C| \leq t \). Let \( \Gamma' \) be the set of preference statements in \( \Gamma \) that are not strictly satisfied by \( H = (c_1, \ldots, c_k, C) \). We try to extend the sequence \( H \) by a \( \Gamma' \)-satisfying HCLP model such that the resulting sequence satisfies \( \Gamma \). We construct this extending model by recursively calling the method for the subproblem with statements \( \Gamma' \) and evaluations \( C' = C - \{c_1, \ldots, c_k\} - C \). If no such extension exists, we backtrack over the last chosen level set \( C \) and try a new level set. Note, that by Lemma 4.3, we never have to backtrack over the choice of singleton level sets which can be a significant advantage over solving the MILP model. As soon as the currently considered sequentialisation yields: If there exists no \( \Gamma \)-satisfying HCLP model in \( C \), Proposition 4.2 using Lemma 4.1. Negating the state- assumption of \( \Gamma \) in \( C \) yields: If there exists no \( \Gamma \)-satisfying HCLP model in \( C \), we stop and return the sequence, \( \Gamma \)-satisfying model in \( C \).

Maintaining Conflicting Sets: In the following, we extend the algorithm PC-check\((C, \Gamma, \oplus, t)\) by maintaining conflicting sets that cannot be contained in the next chosen level sets and thus reduce the number of backtracks. Proposition 4.4 is necessary to solve the most general problems. For our experiments we considered running times for different instance sizes, to observe the effect on the running time by the number of evaluations \( n \) and the number of preference statements \( q \). We generated 100 instances uniformly at random with integral evaluation functions with domains \( \{0, 1, 2, 3, 4, 5\} \) for each of the problem sizes \( n, q \in \{5, 10, 15, \ldots, 50\} \) where we fixed the number of alternatives that the preference statements are based on to \( m = 25 \). Note, that not all alternatives generated are involved in preference statements. Thus, \( m \) has no direct influence on the size of the search space or the running time. For time reasons, some experiments were not conducted for all instance sizes.

Algorithm 1: PC-check\((C, \Gamma, \oplus, t, S = \emptyset, s)\)

<table>
<thead>
<tr>
<th>Algorithm 1: PC-check((C, \Gamma, \oplus, t, S = \emptyset, s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H \leftarrow (c_1, \ldots, c_k) ) some ( \Gamma(\leq) )-satisfying singleton sequence with ( k ) maximal;</td>
</tr>
<tr>
<td>if ( H \models \Gamma ) then stop and return ( H );</td>
</tr>
<tr>
<td>for all the ( C \subseteq C - {c_1, \ldots, c_k} ) with ( 2 \leq</td>
</tr>
<tr>
<td>such that there exists no ( S \subseteq S ) with (</td>
</tr>
<tr>
<td>and ( \alpha \models_{H} \beta ) for all ( \alpha ) with ( \alpha \models \beta ) do</td>
</tr>
<tr>
<td>( H \leftarrow (c_1, \ldots, c_k, C) );</td>
</tr>
<tr>
<td>if ( H \models \Gamma ) then stop and return ( H );</td>
</tr>
<tr>
<td>( \Gamma' = { \varphi \in \Gamma \mid \alpha \models \beta } );</td>
</tr>
<tr>
<td>( \Gamma' = C - \sigma(H) );</td>
</tr>
<tr>
<td>( H \leftarrow (c_1, \ldots, c_k, C, \text{PC-check}(C', \Gamma', \oplus, t, S, s)) );</td>
</tr>
<tr>
<td>if ( H \models \Gamma ) then stop and return ( H );</td>
</tr>
<tr>
<td>else ( S \leftarrow S \cup C )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>return Inconsistent</td>
</tr>
</tbody>
</table>

5 Experimental Runtime Comparisons

In our experiments, we compare the three approaches for solving PCP described in Section 3 and 4 by their running time. Here, we expect solving the MILP to be outperformed by the two recursive approaches as they directly exploit the problem structure to perform less backtracks in a way that is not recognized by CPLEX. Furthermore, we are interested in the relation of the two recursive algorithms in practice. Though PC-check\((C, \Gamma, \oplus, t, S = \emptyset, s)\) prunes the search space further than PC-check\((C, \Gamma, \oplus, t)\), it maintains a possibly exponentially large list of conflicting sets. Thus, it is not obvious if maintaining conflicting sets is advantageous.

Instances:

For our experiments we considered running times for different instance sizes, to observe the effect on the running time by the number of evaluations \( n \) and the number of preference statements \( q \). We generated 100 instances uniformly at random with integral evaluation functions with domains \( \{0, 1, 2, 3, 4, 5\} \) for each of the problem sizes \( n, q \in \{5, 10, 15, \ldots, 50\} \) where we fixed the number of alternatives that the preference statements are based on to \( m = 25 \). Note, that not all alternatives generated are involved in preference statements. Thus, \( m \) has no direct influence on the size of the search space or the running time. For time reasons, some experiments were not conducted for all instance sizes.

Implementation: We implemented all three approaches in Java Version 1.8 using the IBM ILOG CPLEX library for the MILP formulation (which is a satisfaction problem only). All experiments were conducted on a 2.66GHz quad-core processor with 12GB memory.

We choose \( \oplus \) as the standard addition on the integers. For the recursive algorithms we enumerate next level sets with lower cardinality before ones with higher cardinality, and level sets containing evaluations with smaller indexes before ones with higher indexes. Also, we allow the cardinality bound on the level sets to be \( t = n \), the number of evaluations, and fix the cardinality bound on the maintained conflicting sets to be \( s = 5 \). Since
\( C(k') \subseteq C(k) \) for all \( k' < k \), we expect the running times to be lower for smaller \( t \). Also, \( C(k) = C(n) \) for all \( k \geq n \). Thus, the running times are the same for bigger \( t \).

**Experimental Results:** Solving the MILP formulation of PCP (as presented in Section 3) by the CPLEX solver is much slower than by the two recursive algorithms PC-check (as presented in Section 4), see Table 2. Furthermore, the ratio between the mean times of solving the MILP and PC-check grows extremely quickly with the number of statements and evaluations in the instances.

<table>
<thead>
<tr>
<th>( g )</th>
<th>( n )</th>
<th>PC-check</th>
<th>MILP</th>
<th>PC-check</th>
<th>MILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>0.002</td>
<td>0.019</td>
<td>0.002</td>
<td>0.011</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.003</td>
<td>0.011</td>
<td>0.002</td>
<td>0.017</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>0.002</td>
<td>0.011</td>
<td>0.002</td>
<td>0.017</td>
</tr>
</tbody>
</table>

Table 2: Mean times in seconds for solving the MILP and PC-check(\( C, \Gamma, \oplus, t \)) fixing \( m = 25 \).

The two recursive algorithms PC-check(\( C, \Gamma, \oplus, t \)) and PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) show similar behavior of running times for different instance sizes. Figure 1 shows this for a scale of \( n \) while fixing \( g = 10 \) and for a scale of \( g \) while fixing \( n = 20 \).

![Figure 1: Mean times in milliseconds for PC-check(\( C, \Gamma, \oplus, t \)) and PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) fixing \( g = 10, m = 25 \) (left) and \( n = 20, m = 25 \) (right).](image)

By analysing the difference of the mean running times, we observed that the gap between the two algorithms increases, i.e. PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) becomes slower than PC-check(\( C, \Gamma, \oplus, t \)). This could mean maintaining conflicting sets (of size \( < 5 \)) takes so much space that even the smaller number of backtracks does not improve the running time. However, the ratios between the mean running times of PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) and PC-check(\( C, \Gamma, \oplus, t \)) don’t seem to depend on the instance sizes and PC-check(\( C, \Gamma, \oplus, t \)) is at most 2.87 times faster than PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) over all sizes, \( g = 5, 10, \ldots, 25 \) and \( n = 5, 10, \ldots, 25 \) with \( m = 25 \).

We can observe similar trends for the running times for a scale of \( n \) while fixing \( g = 15 \) and for a scale of \( g \) while fixing \( n = 15 \) as the ones shown in Figure 1. To find an explanation for this behavior of the running times, we observed the occurrence of instances that are in \( C(1) \), in \( C(t) \) with \( t > 1 \), or inconsistent for all \( t \). The whole search space has to be explored until deciding inconsistency for all \( t \), which could lead to high running times.

PC-check solves instances in \( O(ng) \) in polynomial time. The instance distribution, however, does not show a connection to the solving times. For example, the distribution of instances with \( n = 5, \ldots, 50 \) and \( g = 10, m = 25 \) as displayed in Figure 2, cannot explain the behavior of the mean running times in Figure 1 which has a peak at \( n = 30 \). However, it is likely that the number of \( C(1) \)-consistent instances (solved in \( O(ng) \)) increases when scaling up the number of evaluations. Thus, the running times will be lower for large \( n \) and behave polynomially. For a large number of statements \( g \), the number of inconsistent instances (for all \( t \)) will increase. At the same time these inconsistent instances are likely to be solved very fast, as fewer non-opposing level sets can be found. Thus, the running times will be lower for large \( n \).

![Figure 2: Percentages of instance classes for 100 random instances with \( n = 5, \ldots, 50 \) and \( g = 10, m = 25 \).](image)

**6 Conclusion**

Exploiting the theoretical results on properties of consistent instances developed in Section 4 allow the algorithms PC-check to prune the search space much further than a MILP solver could do for the MILP formulation given in Section 3. The experimental results confirm, that the algorithms PC-check are solving the instances faster than CPLEX. Even more, the ratios between the mean solution times of the MILP and PC-check increase extremely quickly with the number of evaluations and the number of statements.

There is no significant difference between the mean running times of the two recursive algorithms PC-check. This means that the additional pruning of the search space that is done in PC-check(\( C, \Gamma, \oplus, t, S = \emptyset, s \)) is not worth maintaining a list \( S \) of (possibly exponentially many) conflicting sets. An extensive analysis of the instance consistency types could not explain the behavior of the running times for algorithms PC-check.

A further analysis could involve the sizes of the search spaces, i.e., counting the number of \( \Gamma^{(2)} \)-satisfying HCLP models. Future work could also investigate if the mean running times for PC-check on larger instances behave like the curves displayed in Figure 1 for a scale over the number of evaluations or statements, respectively.
References


