This paper was selected by a process of
anonymous peer reviewing for presentation at

COMMONSENSE 2007

8th International Symposium on Logical Formalizations of Commonsense Reasoning

Part of the AAAI Spring Symposium Series, March 26-28 2007,
Stanford University, California

Further information, including follow-up notes for some of the
selected papers, can be found at:

www.ucl.ac.uk/commonsense07

On Domain-Independent Heuristics for Planning with Qualitative Preferences

Jorge A. Baier and **Sheila A. McIlraith**
Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

This paper describes a method for planning with rich qualitative, temporally extended preferences (QTEPs) using lookahead heuristics inspired by those employed in state-of-the-art classical planners. Key to our approach is a transformation of the planning domain into an equivalent but simplified planning domain. First, compound preference formulae are transformed into simpler, equivalent preference formulae. Second, temporally extended preferences are replaced by equivalent, atemporal preferences. These two simplifications enable us to propose a number of simple heuristic strategies for planning with QTEPs. We propose an algorithm that uses these heuristics and that furthermore is provably k-optimal, i.e. it finds all optimal plans of length no greater than a parameter k. We compare our planner against the PPLAN planner, which does not use lookahead heuristics. Preliminary results show a significant improvement in performance.

Introduction

Planners typically generate plans that satisfy a specified goal formula, but the goal formula provides no information about how to distinguish between multiple successful plans. Preferences convey additional information about desirable properties of a plan, enabling a planner to evaluate the relative quality of the plans that it generates. Planning with *temporally extended preferences* (TEPs), i.e., preferences that refer to properties that can range over the entire plan execution, has been the subject of recent research, e.g., [7, 16, 5]. It was also a theme of the 2006 International Planning Competition (IPC-5).

The problem of planning with TEPs was popularized by IPC-5. Nevertheless IPC-5 focused effort on planning with preferences specified in PDDL3 [11], a preference language that was ultimately quantitative, requiring a planner to optimize a numeric objective function. In contrast to PDDL3, there have been several proposals for preference languages that are *qualitative* or ordinal, rather than quantitative (e.g., [5, 16, 7]). Because such languages do not have to employ numbers, they provide a natural and compelling means for users to specify preferences over properties of plans. Unfortunately, existing qualitative preference planners such as PPLAN [5] and Son & Pontelli's planner [16] that deal with qualitative temporal preferences (QTEPs) do not demonstrate performance comparable to the PDDL3-based TEP planners. To be fair to the developers of these systems, efficiency was not their objective. Both planners were proof-of-concept systems that had not been highly optimized. Nevertheless, our analysis of their behaviour has led to observations that motivate the work presented here. In particular, PPLAN, the more efficient of the two planners, exploits a best-first heuristic search technique. Nevertheless, the heuristic it exploits does not provide guidance based on a measurement of achievement of the preferences.

In this paper, we study the problem of planning with QTEPs specified in a dialect of LPP, the qualitative preference language proposed by [5] and exploited by their planner PPLAN. Our objective is to improve the efficiency of QTEP planning by exploiting lookahead domain-independent heuristic search, such as that existing in state-of-the-art classical planners. To do so, we propose a two-step process to transform our QTEP planning problem into a simplified planning problem. In the first step, we transform LPP preferences into equivalent, more uniform, primitive preferences that enables a simple adaptation of heuristic approaches to classical planning. Next we compile temporally extended preferences into equivalent preferences that refer to (non-temporal) predicates of the domain

With this simplified planning problem in hand, we are now able to exploit heuristic search. To this end, we propose domain-independent heuristic strategies tailored to QTEP planning, that employ a provably sound strategy for pruning states from the search space. We prove that our planner finds all optimal plans of length bounded by a parameter k. We conduct a preliminary experimental investigation in a domain where qualitative preferences are natural. We compare our planner against the PPLAN planner, which does not use lookahead heuristics. Our results demonstrate a significant gain in performance.

Preliminaries

In this section we review the LPP preference language and define the problem of planning with preferences. We use the situation calculus as the formal framework, as presented by Reiter [15].

Planning in the Situation Calculus

A *planning problem* is a tuple $\langle \mathcal{D}, G \rangle$ where $\mathcal{D}$ is a basic situation calculus action theory and G is a goal formula, representing properties that must hold in the final situation. In the situation calculus, planning is characterized as deductive plan synthesis. Given a planning problem $\langle \mathcal{D}, G \rangle$, the task is to determine a situation $s = do(a_n, \ldots, do(a_1, S_0)))$[1] (or equivalently a plan

[1] which we abbreviate to $do([a_1, \ldots, a_n], S_0)$, or $do(\vec{a}, S_0)$

$a_1 \cdots a_n$) such that $\mathcal{D} \models (\exists s).executable(s) \wedge G(s)$ where $executable(s) \stackrel{\text{def}}{=} (\forall a, s').do(a, s') \sqsubseteq s \supset Poss(a, s')$, and $s' \sqsubseteq s$ iff s' does not succeed s in the situation tree.

The Preference Language LPP

In this section, we describe the syntax of LPP [5] [2], a first-order language for specifying user preferences. We provide a brief description of LPP repeating basic definitions and examples from [5]. The reader is directed to this paper for further details. LPP enables the specification of preferences over properties of state as well as temporally extended preferences over multiple states. Unlike many preference languages, LPP provides a total order on preferences. It is qualitative in nature, facilitating elicitation.

To illustrate LPP, we present the dinner example domain.

The Dinner Example: It's dinner time and Claire is tired and hungry. Her goal is to be at home with her hunger sated. Claire can get food by cooking, ordering take-out food, or by going to a restaurant. Because she is tired, she'd rather stay home, and Italian food is her most desired meal.

To understand the preference language, consider the plan we are trying to generate to be a situation as defined earlier. A user specifies his or her preferences in terms of a single, so-called *General Preference Formula*. This formula is an aggregation of preferences over constituent properties of situations. The basic building block of a preference formula is a *Basic Desire Formula* which describes properties of situations. In the context of planning, situations can be thought of as partial plans.

Definition 1 (Basic Desire Formula (BDF)) *A basic desire formula is a sentence drawn from the smallest set $\mathcal{B}$ where:*

1. $\mathcal{F} \subset \mathcal{B}$
2. *$f \in \mathcal{F}$, then* ***final***$(f) \in \mathcal{B}$
3. *If $a \in \mathcal{A}$, then* ***occ***$(a) \in \mathcal{B}$
4. *If φ_1 and φ_2 are in $\mathcal{B}$, then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $(\exists x)\varphi_1$, $(\forall x)\varphi_1$,* ***next****(φ_1),* ***always****(φ_1),* ***eventually****(φ_1), and* ***until****(φ_1, φ_2).*

final(f) states that fluent f holds in the final situation, **occ**(a) states that action a occurs in the present situation, and **next**(φ_1), **always**(φ_1), **eventually**(φ_1), and **until**(φ_1, φ_2) are basic linear temporal logic (LTL) constructs.

BDFs establish preferred situations. By combining BDFs using boolean and temporal connectives, we are able to express a wide variety of properties of situations. E.g,

$$(\exists x).(\exists y).\textbf{eventually}(\textbf{occ}(orderTakeout(x,y))) \quad \text{(P1)}$$

$$(\exists x).(\exists y).\textbf{eventually}(\textbf{occ}(orderRestaurant(x,y))) \quad \text{(P2)}$$

P1 – P2 tell us respectively that Claire prefers to order take-out food, or order at a restaurant, at some point in the plan.

To define preference orderings over alternative properties of situations, we define *Atomic Preference Formulae*. Each alternative being ordered comprises two components: the property of the situation, specified by a BDF, and a *value* term which stipulates the relative strength of the preference.

Definition 2 (Atomic Preference Formula (APF)) *Let $\mathcal{V}$ be a totally ordered, finite set with minimal element v_{min} and maximal element v_{max}. An atomic preference formula is a formula $\varphi_0[v_0] \gg \varphi_1[v_1] \gg ... \gg \varphi_n[v_n]$, where each φ_i is a BDF, each $v_i \in \mathcal{V}$, $v_i < v_j$ for $i < j$, and $v_0 = v_{min}$. When $n = 0$, atomic preference formulae correspond to BDFs.*

In what follows, we let $\mathcal{V} = [0, 1]$ for parsimony (we could have chosen a strictly qualitative set like $\{best < good < indifferent < bad < worst\}$ instead). Returning to our example, the following APF expresses Claire's preference over what to eat (pizza followed by spaghetti):

$$\textbf{eventually}(\textbf{occ}(eat(pizza)))[0] \gg \textbf{eventually}(\textbf{occ}(eat(spag)))[0.4] \quad \text{(P3)}$$

To allow the user to aggregate preferences, General Preference Formulae extend LPP to conditional, conjunctive, and disjunctive preferences.

Definition 3 (General Preference Formula (GPF)) *A formula Φ is a general preference formula if one of the following holds:*

- *Φ is an atomic preference formula*
- *Φ is $\gamma : \Psi$, where γ is a BDF and Ψ is a general preference formula [Conditional]*
- *Φ is one of*
  - *$\Psi_0 \,\&\, \Psi_1 \,\&\, ... \,\&\, \Psi_n$ [General Conjunction]*
  - *$\Psi_0 \mid \Psi_1 \mid ... \mid \Psi_n$ [General Disjunction]*

where $n \geq 1$ and each Ψ_i is a general preference formula.

For example, the GPF $P1 : P3$ specifies a preference for pizza over spaghetti when ordering takeout food.

Semantics Informally, the semantics of LPP is achieved through assigning a weight to a situation s with respect to a GPF, Φ, written $w_s(\Phi)$. This weight is a composition of its constituents. For BDFs, a situation s is assigned the value v_{min} if the BDF is satisfied in s, v_{max} otherwise. Similarly, given an APF, and a situation s, s is assigned the weight of the best BDF that it satisfies within the defined APF. Finally GPF semantics follow the natural semantics of boolean connectives. As such General Conjunction yields the maximum of its constituent GPF weights and General Disjunction yields the minimum of its constituent GPF weights.

The following definition shows us how to compare two situations with respect to a GPF.

Definition 4 (Preferred Situations) *A situation s_1 is at least as preferred as a situation s_2 with respect to a GPF Φ, written $pref(s_1, s_2, \Phi)$ if $w_{s_1}(\Phi) \leq w_{s_2}(\Phi)$.*

Planning with preferences A preference-based planning problem can be characterized by a tuple $\langle \mathcal{D}, G, \Phi \rangle$, where Φ is a GPF, and $\mathcal{D}$ is a theory of action and G is a goal formula. The problem of finding an optimal plan can be defined also as a deductive task in the situation calculus.

Definition 5 (Optimal Plan, k-Optimal Plan) *Let $P = \langle \mathcal{D}, G, \Phi \rangle$ be a preference-based planning problem. Then $\vec{a}$ is an optimal plan (resp. k-optimal plan) for P iff $\vec{a}$ is a plan (resp. a plan of length at most k) for $\langle \mathcal{D}, G \rangle$, and for every plan (resp. every plan of length at most k) $\vec{b}$ for $\langle \mathcal{D}, G \rangle$, $pref(do(\vec{a}, S_0), do(\vec{b}, S_0), \Phi)$.*

Simplifying the Planning Problem

In this section we propose a means of transforming planning problems with LPP preferences into planning problems in which preferences are described in a simplified but equivalent form. We start by motivating the need for heuristics in

[2] The name LPP was coined after publication of [5].

planning with preferences, and then we propose two simplifications to the LPP representation of preferences that together enable the development and exploitation of new heuristic search techniques for planning with QTEPs expressed in LPP.

The Need for Heuristics and Simplification

A common property of existing planners for QTEPs like PPLAN and Son and Pontelli's planner [16] is that they do not actively guide search towards actions that satisfy preferences. This tends to result in poor performance even on problems with very simple preferences. To understand why this happens, we focus on how PPLAN operates.

PPLAN is a best-first search forward chaining planner. Search is guided by an admissible evaluation function that evaluates partial plans with respect to whether they satisfy a user-specified GPF, Φ. This function is the optimistic evaluation of the preference formula with the pessimistic evaluation and the plan length used as tie breakers where necessary, in that order. A GPF is evaluated over intermediate states of a partial plan by exploiting progression [5]. Evaluation of a GPF with respect to a partial plan results in assignment of a weight to that partial plan. This weight is used to guide search towards plans with better (lower) weights.

To illustrate the limitations of this approach, and the motivation for a lookahead-style of heuristic search, consider how PPLAN processes the following GPF Φ,

$$[\textbf{eventually}(\varphi_1) \wedge \textbf{eventually}(\varphi_2)][v_1] \gg \textbf{always}(\varphi_3)[v_2].$$

Here, φ_1 might be $\textbf{occ}(clean(kitchen))$, φ_2 might be $\textbf{occ}(eat(pizza))$ and φ_3 might be $at(home)$. As its name suggests, the optimistic evaluation of a component predicate in a GPF assumes the predicate to be true, until proven false. As such, the BDF $\textbf{eventually}(\varphi_1) \wedge \textbf{eventually}(\varphi_2)$ will be true whether or not either of φ_1 or φ_2 have actually been satisfied. $\textbf{eventually}(\varphi_i)$ can never be falsified, since there is always hope that it will be achieved in a subsequent state of the plan. Thus, there is no distinction between a partial plan in which one or both of φ_1 or φ_2 is true and one in which they are both false, and as such no measure of progress towards satisfaction of the BDF. In contrast, the BDF $\textbf{always}(\varphi_3)$ is falsifiable as soon as φ_3 is false in some state.

An APF is assigned a weight equal to the smallest weight BDF that is optimistically satisfied. Since BDF $\textbf{eventually}(\varphi_1) \wedge \textbf{eventually}(\varphi_2)$ is always optimistically satisfied, our example Φ is always evaluated to weight v_1.

In this case and other cases the optimistic evaluation function provides poor guidance. First,the optimistic evaluation function used in PPLAN cannot distinguish between partial plans that make progress towards satisfying preferences and those that do not. Second, the evaluation function provides no estimate of the number of actions required to satisfy BDF $\textbf{eventually}(\varphi_1) \wedge \textbf{eventually}(\varphi_2)$ nor does it have a way of determining actions to select that will make progress towards satisfaction of preferences.

In classical planning, heuristic approaches have proved to be quite successful (all winners of non-optimal tracks of recent planning competitions use heuristics to guide their search). Unfortunately, there are several barriers to immediate application of these techniques to planning with QTEPs. First, these techniques have been developed for single goals. Second, preferences, as specified in LPP, can interact in rather complex ways (consider for example a conjunction of conditional GPFs). Characterization of these complex interactions is difficult with existing heuristic search formalisms for classical planning. Finally, classical heuristic techniques are tailored to final-state goals. In our case, preferences are temporal formulae.

To adapt classical heuristic techniques for the case of QTEP planning with LPP preferences, we propose to transform the QTEP planning problem into an equivalent problem that is more amenable to these techniques. First we simplify the syntax of LPP by transforming GPFs into an equivalent APF representation, and then we represent temporal first-order preference formulae into equivalent atemporal formula.

Simplifying GPFs into APFs

Here we prove that it is possible to significantly simplify the syntax of GPFs. In fact, the conditional, conjunctive, and disjunctive GPFs can all be simplified into simple APFs.

Theorem 1 *Let Ψ be an arbitrary GPF over the set of preference values $\mathcal{V}$, then it is possible to construct an equivalent APF ϕ_Ψ, over $\mathcal{V}$.*

Proof sketch: By induction in the number of operators of the GPF. We prove, for each type of GPF Ψ, that there exists an equivalent APF $\phi_\psi = \varphi_0[v_0] \gg \varphi_1[v_1] \gg \cdots \gg \text{TRUE}[v_n]$, where v_0 is the minimum element in $\mathcal{V}$ and v_n is the maximum. For brevity, we omit the resulting formulae for each case. Nevertheless, the size of the resulting formulae is linear in $|\Psi|$ for conditional and disjunctive GPFs, however, its size is may be exponential in the number of conjunctive operators. □

This simplification will be key when defining heuristics for planning with LPP preferences. We will focus on computing an estimation of each BDF composing the APF. Since there are no general conjunctions or disjunctions the heuristics do not need to handle complex interactions between preferences.

Simplifying Temporal Formulae

We use techniques presented in [3] to represent the achievement of first-order temporally extended formulae within a classical planning domain. This results in a new augmented planning domain in which for each temporally extended BDF φ, there is a *new* domain predicate, Acc_φ that is true in the final state of a plan if and only if the plan satisfies the temporally extended formula φ.

The compilation process first constructs a parametrized nondeterministic finite state automata (PNFA) A_φ for each temporally extended preference or hard constraint expressed as an LTL formula φ.[3] The PNFA represents a family of nondeterministic finite state automata. Its transitions are labeled by sets of first-order formulae. Its states intuitively

[3]The construction works for an expressive a subset of LTL, i.e. those formulae in extended prenex normal form [3].

"monitor" the progress towards satisfying the original temporal formula. A PNFA A_φ accepts a sequence of domain states iff such sequence satisfies φ.

We then represent the automata within the planning domain. To that end, for each automaton, we define a predicate specifying the automaton's current set of states. When the automaton is parametrized, the predicate represents the current set of automaton states for a particular *tuple of objects*. Moreover, for each automaton we define an *accepting predicate*. This predicate is true of a tuple of objects if the plan has satisfied the temporal formula for such a tuple.

Planning for LPP with Heuristic Search

With the new compiled problem in hand, we propose several heuristics for planning with LPP preferences using forward search. These heuristics are inspired by those used in state-of-the-art heuristic-search classical planners. They provide a way of measuring progress towards the goal and the preferences. The rest of this section describes these heuristics, and a planning algorithm for planning with preferences.

Guiding the Search

Our heuristics for preferences and goals utilize the additive heuristic proposed for classical planning by [6]. To compute them, we use a well-known artifact for classical planning: the *relaxed planning graph* [13]. We can view this graph as composed of *relaxed states*. A relaxed state at depth $n+1$ is generated by *adding* all the effects of actions that can be performed in the relaxed state of depth n, and then by copying all facts that appear in layer n.

Moreover, each fact f in layer i is assigned a heuristic cost $h(f,i)$. All facts in the first layer of the graph have cost 0. If a fact does not appear in layer i, then $h(f,i) = \infty$. If the fact f is added by action a to layer $n+1$, then,

$$h(f,n+1) = \min\{h(f,n), 1 + \sum_{\ell \in \Gamma_{a,f}} h(\ell,n)\},$$

where $\Gamma_{a,f}$ is a minimal set of facts in layer n that are needed to produce effect f from action a. On the other hand, if fact f was copied from layer n to $n+1$ then $h(f,n+1) = h(f,n)$.

Intuitively, any mechanism for guiding search when planning with preferences should guide the search towards (1) satisfying the goal, and (2) generating good-quality plans. Nevertheless, low-weight preferences may be hard to achieve, and therefore this fact should be considered by the heuristics. Below we describe 3 heuristic functions that we use to build search strategies for planning with QTEPs.

Goal distance function (G) This function is a measure of how hard it is to reach the goal. Let $\mathcal{G}$ be a set of goal facts, and let N be the last layer of the expanded relaxed graph.[4] The goal distance for a state s is $G(s) = \sum_{g \in \mathcal{G}} h(g,N)$.

*Preference distance function (**P**)* Suppose the APF describing our preferences is $\varphi_0[v_0] \gg \cdots \gg \varphi_n[v_n]$. We can estimate how hard it is to achieve each of the formulae $\varphi_0, \ldots, \varphi_n$ in a similar way to the processing of the goal. Thus, **P** is a function returning a vector such that its i-th component is $p_i = h(Acc_{\varphi_i}, N)$, where Acc_{φ_i} is the accepting predicate of φ_i, and N is the depth of the relaxed graph. If φ_i is not temporal, we use the heuristic cost of φ_i.

Best relaxed preference weight (B) This function is a lower-bound on the preference weight that a successor of the current state can achieve when completed to satisfy the goal. Although it is similar to the optimistic weight by [5], by using the relaxed planning graph, we can often obtain a better estimate. We compute the preference weight in each of the relaxed states. The B function corresponds to the lowest of these. Intuitively, by using the relaxed graph, we are sometimes able to detect some accepting predicates that can never be made true from the current state.

Strategies for Guiding Search With the heuristic functions defined above, we are ready to propose strategies to heuristically guide search for QTEP planning. Each strategy corresponds to a particular way the search frontier is ranked. Below, we define 4 different strategies to guide search.

Since in planning with preferences it is mandatory to achieve the goal, all strategies we propose here guide the search in some way towards the goal. Before we introduce the strategies, we define two ways of comparing the preference distance vectors.

Definition 6 ($<_{\text{VALUE}}$) *Let* $\mathbf{P} = (p_0, \ldots, p_{max})$ *and* $\mathbf{Q} = (q_0, \ldots, q_{max})$ *be preference distance vectors. Then we say that* $\mathbf{P} <_{\text{VALUE}} \mathbf{Q}$ *if* $\mathbf{P}$ *is lexicographically smaller than* $\mathbf{Q}$. *Formally,* $\mathbf{P} <_{\text{VALUE}} \mathbf{Q}$ *iff* $p_0 < q_0$, *or* $p_0 = q_0$ *and* $(p_1, \ldots, p_{max}) <_{\text{VALUE}} (q_1, \ldots, q_{max})$.

Intuitively $\mathbf{P} <_{\text{VALUE}} \mathbf{Q}$ means that the best-weighted BDF preference of $\mathbf{P}$ has been estimated easier than $\mathbf{Q}$. Ties are resolved by looking at the next best-weighted BDF.

Definition 7 ($<_{\text{EASY}}$) *Let* $\mathbf{P} = (p_0, \ldots, p_{max})$ *and* $\mathbf{Q} = (q_0, \ldots, q_{max})$ *be preference distance vectors. Moreover, let* $best_{\mathbf{P}}$ *be the smallest i such that* $p_i = min_j\{p_j\}$, *and let* $best_{\mathbf{Q}}$ *be defined analogously. Then, we say that* $\mathbf{P} <_{\text{EASY}} \mathbf{Q}$ *iff* $p_{best_{\mathbf{P}}} < q_{best_{\mathbf{Q}}}$, *or* $p_{best_{\mathbf{P}}} = q_{best_{\mathbf{Q}}}$ *and* $best_{\mathbf{P}} < best_{\mathbf{Q}}$.

Intuitively, $\mathbf{P} <_{\text{EASY}} \mathbf{Q}$ means that either $\mathbf{P}$ contains a preference formula that has been estimated to be easier than all those in $\mathbf{Q}$, or the easiest preferences of both vectors have been estimated to be equally hard but $\mathbf{P}$'s easiest preference has a better associated weight.

Now, when ranking the search frontier we say that the state s_1 is better than the state s_2 (denoted by $s_1 \prec s_2$) using four different criteria. These criteria are shown in Table 1, and they correspond to a prioritization of some of the functions defined above. For example, under strategy goal-value first we check whether the distance to the goal from s_1 is less than that from s_2; in case of a tie we check whether s_1's preference vector is better than s_2's with respect to $<_{\text{VALUE}}$.

Our proposed strategies are based on intuitions and hands-on experience. The "value" family of strategies are greedy in the sense that they strive to create a highly-preferred plan first. Although this is intuitively desirable, it can be the case that low-weight BDFs are difficult to achieve, requiring very long plans, and therefore a lot of search effort. With that in mind, the "easy" family of heuristics attempt to gradually satisfy those preferences that are estimated as easily achiev-

[4] To simplify the explanation, we assume that the goal is a conjunction of facts. Our planner can also handle the general case.

Strategy	Check whether	If tied, check whether
goal-value	$G_1 < G_2$	$\mathbf{P}_1 <_{\text{VALUE}} \mathbf{P}_2$
goal-easy	$G_1 < G_2$	$\mathbf{P}_1 <_{\text{EASY}} \mathbf{P}_2$
value-goal	$\mathbf{P}_1 <_{\text{VALUE}} \mathbf{P}_2$	$G_1 < G_2$
easy-goal	$\mathbf{P}_1 <_{\text{EASY}} \mathbf{P}_2$	$G_1 < G_2$

Table 1: Four strategies to determine whether $s_1 \prec s_2$. G_1 and G_2 are the *goal distances*, and $\mathbf{P}_1$ and $\mathbf{P}_2$ are the preference distance vectors of s_1 and s_2.

able. These strategies guide the search towards rapidly finding a plan, no matter how good it is. However, finding a plan is always good, since the algorithm is able to use its weight as a upperbound to prune the search space for subsequent better plans, as we see in the next section.

The Planning Algorithm

Our planning algorithm, depicted in Figure 1, performs a best-first search in the space of states, incrementally generating plans of ever better quality. Additionally, the algorithm prunes states from the search space in two cases: (1) when the plan violates a user-defined hard constraint, or (2) when an estimate of the lowerbound on the weight of all its successors (computed by the function PREFWEIGHTBOUNDFN) is no better than the weight of the best plan that has been found so far. In our implementation, PREFWEIGHTBOUNDFN corresponds to the B function proposed above. Henceforth, we refer to pruning using PREFWEIGHTBOUNDFN as the ***pruning strategy***.

Theoretical Results

We have investigated two relevant properties of the proposed algorithm: whether the *pruning strategy is sound*, and whether the algorithm is able to produce k-optimal plans. We now elaborate on these notions and our results.

Soundness of Pruning Strategy We say that a pruning strategy is *sound* if whenever it prunes a state s from the search space then no successor of s has a weight that is better than that of the best plan found so far.

Theorem 2 *The* best relaxed preference weight function *is a sound pruning strategy.*

This property of the pruning is very important, since it will allow the algorithm to sometimes prove that an optimal solution has been found without exploring the entire search space.

***k*-Optimality** We say that a planning algorithm is k-optimal, if it eventually returns the best-weighted plan among all those of length bounded by k.

Theorem 3 *The algorithm of Figure 1 is k-optimal.*

Proof sketch: This is straightforward from Theorem 2 and the fact that the algorithm exhausts the space of plans of length up to k.

It is important to note here that this result does not mean that the *first* plan found by HPLAN-QP is k-optimal. This is an important difference with respect to the PPLAN planner, where effectively the first (and only) plan returned is a k-optimal plan.

```
Input : init: initial state, goal: goal formula, hardConstraints: a
        formula for hard constraints, φ: an APF, STRATEGY: a
        ranking function, k: a bound for the plan length
begin
  frontier ← INITFRONTIER(init)
  bestWeight ← ∞; while frontier ≠ ∅ do
    current ← REMOVEBEST(frontier)
    f ←Progress hardConstraints over to last state of current
    if f is not false then
      if current is a plan and its weight is < bestWeight then
        Output the current plan
        if this is first plan found then
          hardConstraints ← hardConstraints∪
                {always(PREFWEIGHTBOUNDFN < bestWeight)}
        bestWeight ← WEIGHT(φ, current)
      if LENGTH (succ)< k then
        succ ← EXPAND(current)
        COMPUTEHEURISTICS (succ)
        frontier ← MERGE(succ, frontier, STRATEGY)
end
```

Figure 1: HPLAN-QP's search algorithm.

Implementation and Evaluation

We implemented the proof-of-concept planner HPLAN-QP. The planner consists of two modules. The first is a preprocessor that reads problems in an extended PDDL3 language, which allows the definition of APFs through an additional construct. The second module is a modified version of TLPLAN [1] which is able to compute the heuristic functions and implements the search algorithm of Figure 1.

We performed a preliminary evaluation of the different strategies we proposed over a *dinner domain* originally introduced in [5]. In this domain, there is an agent that is able to drive to restaurants and stores, cook, and eat food. In all our experiments, the agent is initially at home and her goal is to be sated; availability of ingredients to cook and weather conditions vary across individual initial states. Different problems are obtained by adding preferences about things she would like to eat or places she would like to visit. In the most complex problems, preferences state that she would like to eat several types of food or visit different places.

Table 2 contains a summary of the results. It shows the number of states visited by the planner (equivalent to the number of times the main loop of the algorithm of Figure 1 has been executed) and the length of the final plan. We also show the same metrics for the PPLAN planner. Problems that contain a star (*) are those where preferences cannot be fully satisfied, and so the most preferred plan is found.

The results show that in most cases, at least one of our strategies outperforms PPLAN in the number of states visited, sometimes by several orders of magnitude. Also, it's often the case that the strategies that make the goal the first priority expand more nodes, and sometimes generate longer plans. A plausible explanation is that these strategies tend to be "goal obsessive" in the sense that whenever a plan is found, any action that violates the goal will have a low priority, even if it helps to satisfy a preference.

Finally, it is important to note that PPLAN is an optimal planner. It uses an admissible heuristic to guarantee that it always finds the optimal plan first. The benefit of our approach is that the heuristic is more informative, the draw-

Prob#	PPLAN	goal-easy	goal-value	easy-goal	value-goal
1	7	3	3	3	3
7	29	34	20	27	8
8	42	12	12	4	4
9	55	13	13	4	4
10	57	22	22	10	9
11*	57	107	45	102	5
12	92	33	33	6	6
13	171	11617	11617	24	24
14	194	4	4	4	4
15	257	178	32	174	26
17	13787	12	12	7562	7
19*	>20000	3	3	3	3
20	>20000	554	22	554	9
21	>20000	71	71	8	8
22*	>20000	85	30	7	145
23*	>20000	4	4	4	6
24*	>20000	49	22	7	8

Table 2: Nodes expanded by PPLAN and our 4 strategies from Table 1.

back that optimality cannot be guaranteed unless we search the entire search space. The number of expanded nodes reported in the table for HPlan-QP is the number required to find the plan; usually HPlan-QP needs to expand more nodes in order to prove that the plan found is optimal.

Summary and Related Work

In this paper we explored computational issues associated with planning with QTEPs expressed in the LPP preference language. Poor performance of existing QTEP planners provided motivation for our approach, which was to develop domain-independent heuristic search techniques for QTEP planners. To this end, we proposed a suite of heuristic functions and associated strategies that can be used for planning with QTEPs expressed in LPP. We also proposed a planning algorithm that is k-optimal. We were able to employ more informative inadmissible heuristics while still guaranteeing optimality by developing sound pruning techniques that enabled us to vastly reduce the plan search space. While focused on LPP our results are amenable to a variety of QTEP languages.

Key to our approach is the simplification of the original qualitative preference formula: first, by simplifying it syntax, and then by incorporating additional predicates in the domain to eliminate their temporal formulae. Preliminary experimental results suggest that our planner performs up to orders of magnitude better than PPLAN, a planner designed for the same language.

For obvious reasons, we did not compare our planner to a variety of related work on planning with *quantitative* preferences. Most notable among them are the participants of IPC-5, which handle the PDDL3 language. *Yochan*^{PS} [4] is a heuristic planner for finite-state preferences. MIPS-XXL [9] and MIPS-BDD [8] both use Büchi automata to plan with temporally extended preferences. SGPlan$_5$ [14] uses a completely different approach, partitioning the planning problem into several subproblems. Finally, HPLAN-P [2], a heuristic planner that we developed, exploits the same compilation used in this paper to simplify temporal formulae in PDDL3. However, it cannot handle qualitative preferences.

Other planners for problems with preferences include the following. [16] proposes a planner for QTEPs based on answer set programming. This planner was not designed to be efficient. The planning strategy described in [10] employs the heuristic planner Metric-FF [12] to plan for *prioritized goals*. A plan for a high-priority goal is found by iteratively planning for goals with increasing priority. Prioritized goals only refer to final states.

References

[1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Annals of Math and AI*, 22(1-2):5–27, 1998.

[2] J. A. Baier, F. Bacchus, and S. McIlraith. A heuristic search approach to planning with temporally extended preferences. In *IJCAI-07*, pp. 1805–1818, Hyderabad, India, January 2007.

[3] J. A. Baier and S. A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *AAAI-06*, pp. 788–795, Boston, MA, 2006.

[4] J. Benton, S. Kambhampati, and M. B. Do. YochanPS: PDDL3 simple preferences and partial satisfaction planning. In *IPC-2006*, pp. 54–57, Lake District, England, July 2006.

[5] M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *KR-06*, pp. 134–144, Lake District, England, 2006.

[6] B. Bonet and H. Geffner. Planning as heuristic search. *AIJ*, 129(1-2):5–33, 2001.

[7] J. P. Delgrande, T. Schaub, and H. Tompits. Domain-specific preferences for causal reasoning and planning. In *ICAPS-04*, pp. 63–72, Whistler, Canada, June 2004.

[8] S. Edelkamp. Optimal symbolic PDDL3 planning with MIPS-BDD. In *IPC-2006*, pp. 31–33, Lake District, England, July 2006.

[9] S. Edelkamp, S. Jabbar, and M. Naizih. Large-scale optimal PDDL3 planning with MIPS-XXL. In *IPC-2006*, pp. 28–30, Lake District, England, July 2006.

[10] R. Feldmann, G. Brewka, and S. Wenzel. Planning with prioritized goals. In *KR-06*, pp. 503–514, Lake District, England, July 2006.

[11] A. Gerevini and D. Long. Plan constraints and preferences for PDDL3. Tech. Report 2005-08-07, Univ. of Brescia, 2005.

[12] J. Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR*, 20:291–341, 2003.

[13] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[14] C.-W. Hsu, B. Wah, R. Huang, and Y. Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *IJCAI-07*, pp. 1924–1929, Hyderabad, India, January 2007.

[15] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[16] T. C. Son and E. Pontelli. Planning with preferences using logic programming. *TPLP*, 6(5):559–608, 2006.