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, often by orders of magnitude.

1 Introduction

Standard goals only distinguish between plans that satisfy the goal and those that do not but they provide no way of differentiating between successful plans. Preferences, on the other hand, express information about how "good" a plan is, thus enabling a planner to identify successful plans that are more, or less desirable.

The problem of planning with temporally extended preferences (TEPs), i.e., preferences that refer to the whole execution of the plan, was popularized by the 2006 International Planning Competition (IPC-5). Nevertheless, IPC-5 focused effort on planning with preferences specified in PDDL3 [Gerevini and Long, 2005], 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., [Bienvenu *et al.*, 2006; Son and Pontelli, 2004; Delgrande *et al.*, 2004]). 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 [Bienvenu *et al.*, 2006] and Son and Pontelli [2004]'s planner 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, its heuristic 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 Bienvenu *et al.* [2006] 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 planning for 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.

2 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.

2.1 The Situation Calculus

The situation calculus is a logical language for specifying and reasoning about dynamical systems [Reiter, 2001]. In the situation calculus, the *state* of the world is expressed in terms

of functions and relations (fluents) relativized to a particular *situation s*, e.g., $F(\vec{x},s)$. A situation s is a sequence of the primitive actions $a \in \mathcal{A}$ performed from an initial, distinguished situation S_0. The function $do(a,s)$ maps a situation and an action into a new situation. The distinguished binary predicate *Poss* is such that $Poss(a,s)$ is true iff action a can be performed in situation s.

A basic action theory in the situation calculus, $\mathcal{D}$, comprises *domain-independent foundational axioms* and a set of *domain-dependent axioms*. The foundational axioms, Σ, define the situations, their branching (tree) structure, and the situation predecessor relation $\sqsubset$, such that $s \sqsubset s'$ states that situation s precedes situation s' in the situation tree. The domain-dependent axioms describe: the dynamics of fluents (through *successor state axioms*), action preconditions (a set of axioms defining *Poss*), and the initial situation. Details of the form of these axioms can be found in [Reiter, 2001].

A *planning problem* is a tuple $\langle \mathcal{D}, G \rangle$ where $\mathcal{D}$ is a basic 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] such that $\mathcal{D} \models (\exists s).executable(s) \wedge G(s)$ where $executable(s) \stackrel{\text{def}}{=} (\forall a,s').do(a,s') \sqsubseteq s \supset Poss(a,s')$.

2.2 The Preference Language LPP

In this section, we describe the syntax of LPP [Bienvenu *et al.*, 2006] [2], a first-order language for specifying user preferences. We provide an informal description of LPP here, directing the reader to [Bienvenu *et al.*, 2006] for further details. LPP is richly expressive, enabling 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 composition of preferences over constituent properties of situations. The basic building block of our preference formula is a *Basic Desire Formula* which describes properties of (partial) 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,

$$(\exists x).hasIngrnts(x) \wedge knowsHowToMake(x), \quad \text{(P1)}$$

$$(\exists x).\mathbf{eventually}(\mathbf{occ}(cook(x))), \quad \text{(P2)}$$

$$(\exists x).(\exists y).\mathbf{eventually}(\mathbf{occ}(orderTakeout(x,y))), \quad \text{(P3)}$$

$$(\exists x).(\exists y).\mathbf{eventually}(\mathbf{occ}(orderRestaurant(x,y))), \quad \text{(P4)}$$

P1 expresses that in the initial situation Claire has the ingredients to cook something she knows how to make. Observe that fluent formulae that are not inside temporal connectives refer only to the initial situation. P2 – P4 tell us respectively that at some point Claire cooked something, ordered something from take-out, or ordered something at a restaurant.

To define preference orderings over alternative properties of situations, we define *Atomic Preference Formulae* (APFs). 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 $\varphi_0[v_0] \gg \varphi_1[v_1] \gg \ldots \gg \varphi_n[v_n]$, 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.*

An APF expresses a preference over alternatives. 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, followed by crêpes):

$$\begin{array}{l}\mathbf{eventually}(\mathbf{occ}(eat(pizza)))[0] \gg \\ \quad \mathbf{eventually}(\mathbf{occ}(eat(spag)))[0.4] \gg \\ \qquad \mathbf{eventually}(\mathbf{occ}(eat(crêpes)))[0.5]\end{array} \quad \text{(P5)}$$

Moreover, Claire can use the following APF:

$$P3[0] \gg (P1 \wedge P2)[0.2] \gg P4[0.7], \quad \text{(P6)}$$

to say that her first choice is take-out, followed by cooking if she has the ingredients for something she knows how to make, followed by going to a restaurant.

To allow the user to specify more complex preferences and 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*

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

[2] The name LPP was not coined until after publication of this paper.

- 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.*

Here are some example general preference formulae:

$$P1 : P2 \quad (P7) \qquad P5 \mid P6 \quad (P8) \qquad P5 \,\&\, P6 \quad (P9)$$

P7 states that if Claire initially has the ingredients for something she can make, then she prefers to cook. Preferences P9 and P8 show two ways we can combine Claire's food and time preferences into a single complex preference. P9 tries to maximize the satisfaction of both of her preferences, whereas P8 is appropriate if she would be content if either of the two were satisfied.

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. Returning to our example above, for P5 if a situation included the action of eating spaghetti, but not pizza, it would get a weight of 0.4. 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—as in standard planning—$\mathcal{D}$ 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* $\langle \mathcal{D}, G \rangle$*,* $pref(do(\vec{a}, S_0), do(\vec{b}, S_0), \Phi)$.

3 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. This simplified form makes the problem more amenable to exploiting heuristic search. We start by motivating the need for heuristics in 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.

3.1 The Need for Heuristics and Simplification

A common property of existing planners for QTEPs like PPLAN and Son and Pontelli [2004]'s planner 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, the more efficient of the two planners, 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. 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,

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

Here, φ_1 might be $\mathbf{occ}(clean(kitchen))$, φ_2 might be $\mathbf{occ}(eat(pizza))$ 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 $\mathbf{eventually}(\varphi_1) \wedge \mathbf{eventually}(\varphi_2)$ will be true whether or not either of φ_1 or φ_2 have actually been satisfied. $\mathbf{eventually}(\varphi_i)$ can never be falsified, since there is always hope that φ_i 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 $\mathbf{always}(\varphi_3)$ 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 $\mathbf{eventually}(\varphi_1) \wedge \mathbf{eventually}(\varphi_2)$ is always optimistically satisfied, our example Φ is always evaluated to weight v_1.

In this case, as in many, the optimistic evaluation function provides poor guidance for QTEP planning. First, as illustrated above, the optimistic evaluation function used in PPLAN cannot, in many cases, distinguish between partial plans that make progress towards satisfying preferences and those that do not. Second, and more importantly, the evaluation function provides no estimate of the number of actions required to satisfy BDF $\mathbf{eventually}(\varphi_1) \wedge \mathbf{eventually}(\varphi_2)$ nor does it have a way of determining actions to select that will make progress towards satisfaction of preferences. These two limitations motivated us to explore the use of lookahead heuristics from classical planning for guiding search.

In classical planning (where there is just one goal to satisfy), heuristic approaches have proved to be quite successful. All winners of recent international planning competitions (in the non-optimal tracks) use heuristics to guide their

search (e.g. FF [Hoffmann and Nebel, 2001], Fast Downward [Helmert, 2006], SGPlan$_5$ [Hsu *et al.*, 2007]). Unfortunately, there are several barriers to immediate application of these techniques to planning with QTEPs. First, these techniques have been developed for single goals. In our case, we may have multiple different preferences that we wish to satisfy. 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, so notions such as distance to a goal are not defined for formulae such as **eventually**(ϕ).

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. In Section 3.2, we simplify the syntax of LPP by transforming GPFs into an equivalent APF representation. In this way, we eliminate the problem of dealing with complex interactions among preferences. Then, in Section 3.3 we use techniques proposed by Baier and McIlraith [2006] to transform temporal first-order preference formulae into equivalent atemporal formula. In so doing, we transform the problem of planning with QTEPs into an equivalent problem in which temporal preferences are expressed as final-state preferences. In Section 4 we are then able to propose a set of heuristics, tailored to QTEP planning, that we exploit for planning with LPP preferences.

3.2 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. (Note that ϕ_ψ contains *all* values in $\mathcal{V}$; however, it can be often simplified when their BDFs are equivalent to FALSE.) 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 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.

3.3 Simplifying Temporal Formulae

We use techniques presented by Baier and McIlraith [2006] to represent the achievement of first-order temporally extended formulae within a classical planning domain. This results in a new augmented classical planning domain in which each

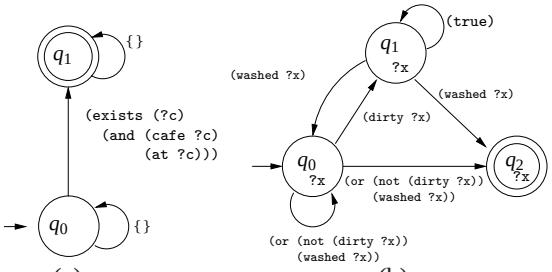

Figure 1: PNFA for (a) **eventually**$((\exists c)\, cafe(c) \wedge at(c))$ and (b) $\forall x\,[\textbf{always}(dirty(x) \supset \textbf{eventually}(washed(x)))]$

temporally extended BDF φ, is replaced by 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 parameterized 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 "monitor" the progress towards satisfying the original temporal formula. A PNFA A_φ accepts a sequence of domain states iff such sequence satisfies φ. Figure 1 shows some examples of PNFA for first-order LTL formulae.

Parameters in the automata appear when the LTL formula is externally quantified (e.g. Figure 1(b)). The intuition is that different *objects* (or tuples of objects) can be in different states of the automata. As an example, consider that in the dinner domain, the dishes A and B are clean. Focusing on the formula of Figure 1(b), both objects start off in states q_0 and q_2 of the automata because they are not *dirty* in the initial state. This means that initially both objects satisfy the temporal formula, since both are in the automaton's accepting state q_2. That is, the null plan satisfies the formula (b) of Figure 1. Now, assume we perform the action $serve(A, Pasta)$ (which makes A dirty). In the resulting state, B stays in q_0 and q_2 while A now moves to q_1. Hence, A no longer satisfies the formula; it will satisfy it only if the plan reaches a state where $washed(A)$ is true.

To represent the automata within the domain, for each automaton, we define a predicate specifying the automaton's current set of states. When the automaton is parameterized, the predicate has arguments, representing the current set of automaton states for a particular *tuple of objects*. In our example, the fact `(aut-state q0 A)` represents that object `A` is in `q0`. Moreover, for each automaton we define an *accepting predicate*. The accepting predicate is true of a tuple of objects if the plan has satisfied the temporal formula for such a tuple.

For further details of the compilation, we refer the reader to [Baier and McIlraith, 2006]. We however present three of its results now.

[3] The construction works for an expressive a subset of LTL, i.e. those formulae in extended prenex normal form. Refer to [Baier and McIlraith, 2006] for more details.

Proposition 1 (Correctness) *Let A_φ be the automaton constructed by the compilation algorithm from an LTL formula φ . Then A_φ accepts exactly the models of φ.*

Proposition 2 (Size of the Automaton) *Let φ be in negated normal form, then the number of states of A_φ is $2^{O(|\varphi|)}$.*

Proposition 3 (Size of output planning problem) *The size of the resulting planning domain is $O(n|Q|\ell)$ where ℓ is the maximum size of a transition in $A_\mathcal{G}$, n is the number of action terms in the domain, and $|Q|$ is the number of states of the automaton.*

Although in theory, the number of states of the automaton can blowup exponentially, we have developed techniques to reduce its final size. We have observed that in practice, the number of states of the resulting automata is comparable to the size of the formula [Baier and McIlraith, 2006].

4 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 proposes a planning algorithm for planning with preferences.

4.1 Guiding the Search

In the new compiled domain, to determine whether a TEP is satisfied we just need to check whether the corresponding accepting predicate is satisfied in the last state of the plan. This enables us to use heuristics that have been proposed for classical planning.

In particular, our heuristics for preferences and goals utilize the additive heuristic proposed for classical planning by Bonet and Geffner [2001]. Although Bonet and Geffner [2001]'s heuristic was defined for STRIPS operators, in this paper we lift it to the more general case of ADL operators [Pednault, 1989].

To compute the heuristic, we use a well-known artifact for classical planning: the *relaxed planning graph* [Hoffmann and Nebel, 2001]. 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. Relaxed states can simultaneously contain both a fact f, and its negation, $\neg f$. Thus, if an executable action has the effect of making fact p true and fact q false, then $\{p, \neg q\}$ is added to the successive relaxed state.

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. In other words, $\Gamma_{a,f}$ is a minimal set of facts, that makes true both the precondition of a and any formula on which the effect f of a was conditioned. On the other hand, if fact f was copied from layer n to $n+1$ then $h(f,n+1) = h(f,n)$.

To compute heuristics for a state s, we expand the relaxed graph starting from state s. The relaxed graph is expanded until a fixed point is found or until the goal and all preferences are satisfied.

Intuitively, any mechanism for guiding search when planning with preferences should guide the search towards (1) satisfying the goal, and (2) generating good-quality (low-weight) 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. Each function addresses some aspect of these intuitions.

Heuristic functions

Goal distance function (G) This function is a measure of how hard it is to reach the goal. Formally, 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) An estimation of the preference weight of any successor of the current state. The best relaxed preference weight is a lowerbound on the preference weight that a successor of the current state can achieve when completed to satisfy the goal. Although this function is similar in spirit to the optimistic weight by Bienvenu *et al.* [2006], now, 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. Thus, the B function is an evaluation of the original APF which only regards such unreachable predicates as being false.

Strategies for Guiding Search

With the heuristic functions defined above, we are ready to propose strategies to heuristically guide search for planning with QTEPs. Each of these strategies 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.

[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.

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 (with a lower heuristic cost) 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 $best_{\mathbf{P}}$ corresponds to the index of the best-weighted preference that is also estimated to be the easiest among all the preferences in the APF. Therefore, 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 s_1 is better than 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 of s_1 is less than that of 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. We want to achieve the goal and therefore we consider progress towards its satisfaction as important in all the defined strategies. 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 achievable. 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.

4.2 The Planning Algorithm

Our planning algorithm, depicted in Figure 2, 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***.

```
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 2: HPLAN-QP's search algorithm.

4.3 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.*

Proof sketch: The result follows by first proving that if there is a fact f (resp. a negative fact $\neg f$) that does not appear in the deepest state layer of a relaxed plan graph constructed from s, then f (resp. $\neg f$) is not true in any successor of s. Now if our APF is $\varphi_0[v_0] \gg \cdots \gg \varphi_n[v_n]$, when we evaluate each Acc_{φ_i} in the deepest relaxed state, we obtain that Acc_{φ_i} is false iff φ_i is false in every successor of s. It is easy to see that when we evaluate the APF in the deepest relaxed state we obtain an optimistic estimation of the preference weight that can be reached by any successor of s. □

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 visiting 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 2 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.

5 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 [Bacchus and Kabanza, 1998] which is able to compute the heuristic functions and implements the search algorithm of Section 4.

We performed a preliminary evaluation of the different strategies we proposed over a *dinner domain* originally introduced in [Bienvenu *et al.*, 2006]. 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, the preference states that she would like to eat several types of food and/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 2 was executed) and the length of the final plan. We also show the same metrics for the PPLAN planner. Problems marked with a star (*) are those where the weight of the optimal plan is greater than 0, i.e., the preferences cannot be fully satisfied.

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.[5] 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 this happens because 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.

[5] Note however that the development of PPLAN did not focus on optimizing the efficiency of the implementation.

	PPLAN		goal-easy		goal-value		easy-goal		value-goal	
Prob#	#ExpN	ℓ	#ExpN	ℓ	#ExpN	ℓ	#ExpN	ℓ	#ExpN	ℓ
1	7	2	3	2	3	2	3	2	3	2
2	7	2	3	2	3	2	3	2	3	2
3	8	2	3	2	3	2	3	2	3	2
4	9	2	3	2	3	2	3	2	8	7
5	15	4	3	2	3	2	3	2	3	2
6	23	3	3	2	7	4	3	2	3	2
7	29	5	34	5	20	5	27	5	8	7
8	42	3	12	3	12	3	4	3	4	3
9	55	3	13	5	13	5	4	3	4	3
10	57	8	22	8	22	8	10	8	9	8
11*	57	6	107	6	45	7	102	6	5	4
12	92	5	33	5	33	5	6	5	6	5
13	171	6	11617	7	11617	7	24	7	24	7
14	194	3	4	3	4	3	4	3	4	2
15	257	7	178	7	32	7	174	7	26	7
16	313	7	58	7	58	7	8	7	8	7
17	13787	6	12	6	12	6	7562	6	7	6
18	17606	4	5	2	5	4	2948	7	5	4
19*	>20000	-	3	2	3	2	3	2	3	2
20	>20000	-	554	8	22	8	554	8	9	8
21	>20000	-	71	7	71	7	8	7	8	7
22*	>20000	-	85	7	30	7	7	6	145	7
23*	>20000	-	4	3	4	3	4	3	6	5
24*	>20000	-	49	6	22	2	7	6	8	7

Table 2: Nodes expanded (#ExpN) and plan length (ℓ) obtained by PPLAN and our 4 strategies from Table 1.

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 is that the heuristic is more informative, the drawback is that optimality cannot be guaranteed unless we search the entire search space. To guarantee optimality, we search the whole search space cleverly by exploiting sound pruning techniques discussed in Section 4.3 that enable us to vastly reduced the space that must be searched. While we prove that our planner finds the most preferred plan, we make no guarantees about the length of that plan. Nevertheless, experimental results show that in the dinner domain, lengths of plans are comparable to the optimal found by PPLAN.

6 Summary and Related Work

In this paper we explored computational issues associated with planning with temporally extended preferences expressed in the LPP preference language. The 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 heuristics 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 effective search strategies that do not guide the search to optimal solutions, while still guaranteeing optimality by developing sound pruning techniques that enabled us to vastly reduce the plan search space. While focussed 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. We proved bounds on the size of this transformation.

Preliminary experimental results suggest that our planner performs up to orders of magnitude better than PPLAN, a planner designed for the same language. Nevertheless, we believe that there is still room for improvement. First, the heuristics we used for preferences are simple; we believe that exploiting more complex heurisitcs such as, for example, the length of *relaxed plans* [Hoffmann and Nebel, 2001] may provide even better guidance. Further, we think that strategies that better combine goal-directed heuristics and preference-directed heuristics still need to be explored.

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*$^{\mathcal{PS}}$ [Benton *et al.*, 2006] is a heuristic planner for finite-state preferences. MIPS-XXL [Edelkamp *et al.*, 2006] and MIPS-BDD [Edelkamp, 2006] both use Büchi automata to plan with temporally extended preferences by invoking the heuristic planner FF [Hoffmann and Nebel, 2001]. SGPlan$_5$ [Hsu *et al.*, 2007] uses a completely different approach. It partitions the planning problem into several subproblems. It then solves them heuristically and integrates their solutions. Finally, HPLAN-P [Baier *et al.*, 2007] is a heuristic planner that exploits the same compilation shown in this paper to simplify temporal formulae in PDDL3. However, it cannot handle qualitative preferences.

Other planners for problems with preferences include the following. Son and Pontelli [2004] propose a planner for qualitative temporally extended preferences based on answer set programming. This planner was not designed to be efficient; its performance degrades significantly as the length of the plan increases. The planning strategy by Feldmann *et al.* [2006] employs the heuristic planner Metric-FF [Hoffmann, 2003] to plan for *prioritized goals*. A plan for a high-priority goal is found by interatively planning for goals with increasing priority. Prioritized goals only refer to final states. Finally, less related is the work by Brafman and Chernyavsky [2005], who proposed a CSP approach to planning with final-state qualitative preferences specified using TCP-nets.

References

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

[Baier and McIlraith, 2006] J. A. Baier and S. A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pp. 788–795, 2006.

[Baier *et al.*, 2007] J. Baier, F. Bacchus, and S. McIlraith. A heuristic search approach to planning with temporally extended preferences. In *Proc. of the 20th Int'l Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007. To appear.

[Benton *et al.*, 2006] J. Benton, S. Kambhampati, and M. B. Do. YochanPS: PDDL3 simple preferences and partial satisfaction planning. In *5th International Planning Competition Booklet (IPC-2006)*, pp. 54–57, 2006.

[Bienvenu *et al.*, 2006] M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *Proc. of the 10th Int'l Conference on Knowledge Representation and Reasoning (KR-06)*, pp. 134–144, 2006.

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

[Brafman and Chernyavsky, 2005] R. Brafman and Y. Chernyavsky. Planning with goal preferences and constraints. In *Proc. of the 15th Int'l Conference on Automated Planning and Scheduling (ICAPS-05)*, pp. 182–191, 2005.

[Delgrande *et al.*, 2004] J. P. Delgrande, T. Schaub, and H. Tompits. Domain-specific preferences for causal reasoning and planning. In *Proc. of the 14th Int'l Conference on Automated Planning and Scheduling (ICAPS-04)*, pp. 63–72, 2004.

[Edelkamp *et al.*, 2006] S. Edelkamp, S. Jabbar, and M. Naizih. Large-scale optimal PDDL3 planning with MIPS-XXL. In *5th International Planning Competition Booklet (IPC-2006)*, pp. 28–30, 2006.

[Edelkamp, 2006] S. Edelkamp. Optimal symbolic PDDL3 planning with MIPS-BDD. In *5th International Planning Competition Booklet (IPC-2006)*, pp. 31–33, 2006.

[Feldmann *et al.*, 2006] R. Feldmann, G. Brewka, and S. Wenzel. Planning with prioritized goals. In *Proc. of the 10th Int'l Conference on Knowledge Representation and Reasoning (KR-06)*, pp. 503–514, 2006.

[Gerevini and Long, 2005] A. Gerevini and D. Long. Plan constraints and preferences for PDDL3. Tech. Rep. 2005-08-07, Department of Electronics for Automation, University of Brescia, Brescia, Italy, 2005.

[Helmert, 2006] M. Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

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

[Hsu *et al.*, 2007] C.-W. Hsu, B. Wah, R. Huang, and Y. Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proc. of the 20th Int'l Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007. To appear.

[Pednault, 1989] E. P. D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. of the 1st Int'l Conference of Knowledge Representation and Reasoning (KR-89)*, pp. 324–332, 1989.

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

[Son and Pontelli, 2004] T. C. Son and E. Pontelli. Planning with preferences using logic programming. In *Proc. of the 7th Int'l Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-04)*, number 2923 in LNCS, pp. 247–260. Springer, 2004.