Hybrid Planning with Preferences Using a Heuristic for Partially Ordered Plans

Pascal Bercher and Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany

Abstract. This paper is concerned with the problem of finding preferred plans in a hybrid planning setting, which is the fusion of classical and hierarchical planning. Here, we define preferences as weighted *soft goals* — facts one would like to see satisfied in a goal state, but which do not have to hold necessarily.
We present a branch-and-bound algorithm that allows a broad variety of search strategies, as opposed to the majority of existing planning systems which usually perform progression. The algorithm prunes task networks from the search space which will never lead to a better solution than the best solution found so far. To this end, we developed an admissible heuristic, based on a combination of the h^2 heuristic and delete relaxation, which takes as input a task network and estimates the best quality of any solution that can be developed from it.

Keywords: preferences, soft goals, hybrid planning, POCL planning, h^2 heuristic, delete relaxation, reachability analysis

1 Introduction

In real-world planning, for example, when assisting human users in their everyday life [3], plans are often of different quality depending on the specific user carrying out the plan. One user may like to go by bus, whereas the other may prefer to travel by taxi. In many application contexts, there is therefore the need to specify a quality measure that reflects the different needs and preferences of different human users. In this setting, which is called *planning with preferences*, a planning problem is augmented with a set of preference formulas. The goal is to find a solution to the planning problem that satisfies the preferences to the largest possible extent.

Planning with preferences has attracted increased attention with the development of PDDL3 [11], the language for the fifth International Planning Competition (IPC-5). In PDDL3, preferences are either *soft goals* (also called *simple preferences*) or *plan constraints*. The former are certain conditions that should hold in the final state produced by a solution plan; i.e., they are not mandatory, but should be achieved if possible. The latter are soft (i.e., non-mandatory) constraints on the state trajectories induced by a plan. They can be expressed by a special form of Linear Temporal Logic (LTL) formulas. These plan constraints enable the expression of rich user preferences, like “never drive by bus”

or "before calling taxi, check whether there is enough money to pay it". Each preference is associated with a numerical value (we call it *weight* in this paper), which represents the importance of the respective preference.

This paper is concerned with solving hybrid planning problems favoring those solutions which satisfy the weighted preferences — given in terms of soft goals — to a larger extent. The hybrid planning paradigm is particularly well suited for solving real-world planning problems, as it fuses ideas from classical planning with those of hierarchical task network (HTN) planning: many real-world problems are inherently hierarchical and can more easily and adequately be encoded in the HTN planning paradigm. However, parts of the domain might be non-hierarchical and could be modeled more adequately in the classical state-based paradigm. Hybrid planning fuses both, in that it allows for the specification of an initial task network and of compound tasks as in HTN planning, but also enables the arbitrary insertion of tasks to support open preconditions as in classical planning.

The paper is organized as follows. In Section 2, we explain the concepts of hybrid planning and introduce our notion of preferences. In Section 3, we introduce our novel preference-based branch-and-bound algorithm for hybrid planning. This algorithm uses a heuristic function to estimate the final quality of a current task network in order to prune it from the search space if it leads to a suboptimal solution. In Section 4, we introduce such a heuristic function, which is based on a variant of the h^2 heuristic [12] using delete relaxation. After that, we give a brief review of related work followed by a summary and outlook.

2 Planning with Preferences

We are interested in solving hybrid planning problems [4,7,10,13] with preferences. Hybrid planning relies on the concept of HTN planning and classical planning; the former of which will be explained next.

HTN planning is based on the concepts of tasks and methods [6]. Compound tasks represent high-level activities like making a business trip or transporting certain goods to a specific location. Primitive tasks correspond to classical planning operators. Hierarchical domain models hold a number of methods for each compound task. Each method provides a task network which specifies a predefined abstract/partial solution of the corresponding compound task. The decomposition of a compound task by an appropriate method replaces this task by the task network specified by the respective method. A solution to a planning problem is an executable decomposition of the initial task network. Thus, a planning problem is solved by incrementally decomposing the compound tasks in the initial task network until it contains only primitive tasks and is consistent w. r. t. their ordering and causal structure. Such a task network is a solution to the given problem and called a *plan*.

Hybrid planning extends HTN planning in the following way:

- Tasks may be inserted into any task network without the need of being introduced via decomposition.[1] This allows to plan for partially hierarchical domain models [13], because missing tasks may be inserted ad hoc.
- Also compound tasks show pre- and postconditions. Hence, they can be inserted into task networks thereby improving the search efficiency. The performance increase results from the fact that the decomposition methods specify predefined standard solutions for the compound tasks' postconditions.
- A goal description can be specified like in classical planning.

In our formalization of hybrid planning, we use the concepts of partial order causal link (POCL) planning, as they allow explicit causal reasoning and a broad variety of search strategies in addition to the standard progression and regression search. In hybrid planning, a task network $TN = (T, \prec, V, C)$ consists of a set of tasks T, where each task $l{:}t \in T$ consists of a (partially) instantiated primitive or compound task schema t and a label l to differentiate between multiple occurrences of t. The set $\prec$ of ordering constraints imposes a partial order on the tasks in T. V is a set of variable constraints, each of the form $(l, v) \circ x$, where l is the label of the task schema to which the variable v belongs, $\circ \in \{=, \neq\}$, and x is either another variable of the form (l', v'), or a constant c. C is a set of causal links; a causal link $l' \rightarrow_\phi l$ indicates that the precondition literal ϕ of $l{:}t \in T$ is an effect of $l'{:}t' \in T$ and is *supported* this way. $\mathcal{T}$ denotes the set of available task schemata, whereas $\mathcal{M}$ denotes the set of available decomposition methods. A primitive or compound task schema $t = (\text{prec}, \text{eff})$ is a tuple consisting of sets of preconditions and effects; both are sets of literals and depend on the parameters $\bar{v}(t)$. For the sake of simplicity, we write $\text{prec}(t)$ and $\text{eff}(t)$ to refer to the preconditions and effects of t. A method $m = (t, TN)$ relates a compound task schema t to a task network TN, which represents a predefined abstract/partial solution or "implementation" of the task, or, more precisely, of the task's effects. In general, a number of different methods is provided for each compound task. Given a set of available constants, we denote by $\mathcal{P}$ the set of all possible ground atomic propositions. Then, a state is defined as usual as a set of (positive) facts from $\mathcal{P}$. Thus, given a set of constants, a **hybrid planning problem** $\Pi = (\mathcal{T}, \mathcal{M}, s_{init}, TN_{init}, g)$ includes the task schemata $\mathcal{T}$, the methods $\mathcal{M}$, an initial state $s_{init} \in 2^{\mathcal{P}}$, an initial task network TN_{init} that needs to be decomposed, and a goal description $g \subseteq \mathcal{P}$ which must be achieved by any solution. Then, a **solution to** Π, also called a plan, is a task network $TN = (T, \prec, V, C)$ that satisfies the following criteria:

- TN is a refinement of TN_{init} w. r. t. decomposition of compound tasks and the insertion of tasks, causal links, ordering- and variable constraints,
- no causal links are threatened, i.e., for each causal link $l' \rightarrow_\phi l \in C$ the ordering constraints ensure that no task $l''{:}t''$ with $t'' \in \mathcal{T}$ and effect $\neg\psi$ that can be unified with $\neg\phi$, can be ordered between $l'{:}t' \in T$ and $l{:}t \in T$,

[1] In recent work [9] we showed that this feature makes hybrid planning decidable as opposed to standard HTN planning in which tasks may not be inserted arbitrarily.

- the ordering and variable constraints $\prec$ and V are consistent, i.e., the ordering does not induce cycles on the tasks and the (in-) equations of variable constraints are free of contradiction,
- all tasks in T are primitive, and
- TN is executable in s_{init} and all ground linearizations of tasks of TN generate a state s' that satisfies the goal description, i.e., $s' \supseteq g$.

In addition to a planning problem Π, we are given a set of weighted ground facts. Thus, $Pref \subseteq \mathcal{P} \times \mathbb{N}$ is a set of *preferences* or *soft goals*. For a preference $(p, n) \in Pref$, the weight is interpreted as a violation value which depreciates a given plan TN by n if p does not necessarily hold in the final state produced by TN (which we denote by $TN \not\models p$). For a plan TN, the metric m is hence defined as $m(TN) := \sum_{(p,n) \in Pref,\, TN \not\models p} n$. Then, a plan TN_1 is preferred over a plan TN_2, written $TN_1 \leq TN_2$, if and only if $m(TN_1) \leq m(TN_2)$.

Please note that this kind of preference and metric can also be expressed in PDDL3. There, each preference is given a specific name, which can then be used by the metric function. For example, let $Pref := \{(\text{accepted}(\text{PAPER1}), 10), (\text{accepted}(\text{PAPER2}), 5)\}$ denote the user's preference that it is twice as important to him to have PAPER1 accepted as PAPER2. In PDDL3, this would look as follows:

```
(:constraints (and (preference P1 (at-end (accepted PAPER1)))
                   (preference P2 (at-end (accepted PAPER2)))))

(:metric minimize (+ (* 10 (is-violated P1))
                     (*  5 (is-violated P2))))
```

Although PDDL3 can also express preferences on state trajectories, from a theoretical point of view, it is not a real restriction to design a planning algorithm, that is "only" capable of searching for plans which are optimal w. r. t. soft goals: Baier and McIlraith [2] have developed a technique which compiles a classical planning problem with preferences on state trajectories into an equivalent planning problem which only makes use of soft goals. Hence, any planner capable of maximizing plan quality w. r. t. soft goals is in principle capable of handling (the seemingly more expressive) state trajectory constraints.

3 Search Algorithm

In this section, we present our preference-based branch-and-bound algorithm, which allows for arbitrary partial order search strategies. To enable the flexibility of making use of various search strategies, we provide an explicit representation of *flaws* and *modifications*. Flaws are deficiencies which are contradicting a solution criterion or a preference. Modifications are refinements to address these deficiencies like the decomposition of a compound task or the insertion of an ordering constraint.

Algorithm 1 shows our planning procedure. It always keeps track of the best solution found so far (`best-tn`) and its metric value `best-m`. The fringe is a sequence of task networks; initially, it contains only the initial task network.

Algorithm 1: Preference-based Branch-and-Bound Planning Algorithm

Input : The sequence `Fringe` $= \langle TN_{init} \rangle$.
Output : A plan or *fail*.

`best-m` $:= \infty$
`best-tn` $:=$ *fail*
while *Fringe* $= \langle TN_1 \dots TN_n \rangle \neq \varepsilon$ **do**
  `Fringe` $:= \langle TN_2 \dots TN_n \rangle$
  F_H $:= f^{\text{HardFlaw-Det}}(TN_1)$
  F_S $:= f^{\text{SoftFlaw-Det}}(TN_1)$
  `// Store best plan and its metric value`
  if $F_H = \emptyset$ **and** $m(TN_1) <$ *best-m* **then**
    `best-m` $:= m(TN_1)$
    `best-tn` $:= TN_1$
  `// Discard current task network or refine it further?`
  if $h(TN_1) <$ *best-m* **then**
    $\langle m_1 \dots m_k \rangle := f^{\text{Mod-Ord}}(\bigcup_{\mathtt{f} \in F_H \cup F_S} f^{\text{Mod-Gen}}(\mathtt{f}))$
    `Fringe` $:= f^{\text{TN-Ord}}(\langle \mathtt{app}(m_1, TN_1) \dots \mathtt{app}(m_k, TN_1) \rangle \circ \mathtt{Fringe})$
return *best-tn*

The closer a task network is to the front of the fringe, the faster it is expected to lead to a solution. Inside the loop, the first — and thus most promising — task network is removed from the fringe and all its hard flaws and soft flaws are detected and stored into the sets F_H and F_S, respectively. A flaw is a set of syntactical elements of a task network, which are involved in the violation of a solution criterion in the case of a hard flaw, and in the violation of a soft goal in the case of a soft flaw. For example, if a task network contains a compound task $l{:}t$, F_H contains the set $\{l{:}t\}$, as any plan must only contain primitive tasks. In the case of an open/unsupported precondition literal ϕ of a task $l{:}t$, F_H contains the set $\{l{:}t, \phi\}$. If the set of hard flaws is empty, the current task network is a solution; it is hence tested, whether it has a better metric value than the best solution found so far. If this is the case, `best-tn` and `best-m` are updated accordingly. After that, the current task network is tested for sub optimality: if the heuristic function h estimates a higher value than the best metric value achieved so far, the current task network can be discarded. Obviously, all task networks pruned by this step are suboptimal if h is admissible. If the current task network can not be pruned, all modifications that resolve some flaw are generated. For example, a hard flaw consisting of a compound task can only be resolved by decomposing it. In this case, the modifications would be the appropriate methods. In the case of an open/unsupported precondition literal ϕ of a task $l{:}t$, the modifications would be the insertion of a causal link $l' \rightarrow_\phi l$ from a task $l'{:}t'$, which either may already exist or which would be inserted together with the causal link. All modifications collected this way are ordered according to the modification ordering function $f^{\text{Mod-Ord}}$. The task networks resulting from applying the modifications, denoted by $\mathtt{app}(m, TN)$ for some task network TN

and some modification m, are inserted into the fringe as its new head and then rearranged according to the task network ordering function $f^{\text{TN-Ord}}$.

The chosen functions $f^{\text{TN-Ord}}$ and $f^{\text{Mod-Ord}}$define the planning strategy. For instance, if $f^{\text{TN-Ord}}$ is the identity function, then the strategy is a depth-first search, where $f^{\text{Mod-Ord}}$ decides, which branches to visit first; $f^{\text{Mod-Ord}}$ could then be defined to prioritize those modifications higher, which are associated with hard flaws thus directing the search to task networks which can be completed to valid solutions faster.

After the fringe has been rearranged, the loop enters another cycle. The search finally terminates when the fringe becomes empty. Because only those task networks are discarded for which sub optimality can be proved, the last plan returned (if any) is an optimal solution to the given problem. However, in the general case, the fringe may never become empty although solutions have been found. This is a well-known issue for POCL planners, as they perform search in the space of plans rather than in the space of states. In the former, cycle-detection is non-trivial which causes the lack of a simple pruning criterion. Thus, in order to guarantee termination, one may specify a time-out criterion to leave the loop in cases when the fringe does not become empty. In that case the algorithm returns the best solution found so far.

4 Preference-based Heuristic for Partially Ordered Plans

In this section we describe a variant of the h^2 heuristic [12] — the heuristic implicitly used by the planning system GRAPHPLAN [5]. GRAPHPLAN builds a directed, layered planning graph containing fact and task nodes. Each layer contains only nodes of one of those types; the layers alternate between fact and task layers, starting with fact layer 0 containing exactly the facts of the initial state followed by task layer 0, which contains all tasks applicable in that state. More generally, a task layer at level i contains all tasks applicable to the fact layer at level i. GRAPHPLAN calculates binary symmetric mutex relations between facts inside the same layer and between tasks inside the same layer. The former indicates that two facts cannot be true at the same time, whereas the latter indicates that two tasks can not be executed in an arbitrary order leading to the same successor state. Blum and Furst have shown that the fact layers monotonically increase, whereas the mutex relations monotonically decrease. Thus, the planning graph construction of GRAPHPLAN eventually terminates with a final fact layer containing some mutex relations. Given a partially ordered task network *TN*, our heuristic uses this final fact layer to estimate the best quality of any solution obtainable by refining *TN*.

Although the h^2 heuristic can be calculated in polynomial time, it is known to be too expensive to calculate it in each state. Hence, we build the last fact layer based on a *relaxed* domain model, in which negative effects of tasks are ignored. Our heuristic differs from h^2 in the following way: (1) it takes into account the metric function m which is purely based on the final state produced by a plan, whereas h^2 estimates the cost of a plan based on action costs. Hence,

our heuristic uses only the final fact layer for the estimate of m, whereas h^2 takes also the previous layers into account, (2) the planning graph construction is based on delete relaxation, i.e., tasks do not show negative effects, and (3) it takes as input a partially ordered task network rather than a state. Our heuristic can hence be regarded as a relaxed reachability analysis for task networks, which takes the preference-based metric into account.

Our heuristic consists of two steps: First, we transform a given *hybrid* planning problem $\Pi = (\mathcal{T}, \mathcal{M}, s_{init}, TN_{init}, g)$ and a task network TN (for which the heuristic value is calculated) into a relaxed *classical* planning problem $\Pi' = (\mathcal{T}', \emptyset, s'_{init}, \varepsilon, g')$, such that Π' has a solution if Π has one. Furthermore, any solution of Π' contains the tasks of TN and respects its constraints. This transformation enables the use of any heuristic that is defined for a state rather than for a task network. Section 4.1 explains this transformation. The second step is the actual heuristic calculation; it is described in Section 4.2.

4.1 Domain Transformation

In this section we will see how a *hybrid* planning problem Π and a current task network TN can be transformed into a relaxed *classical* planning problem Π', such that any solution of Π' contains the tasks of TN and respects its constraints.

The domain transformation consists of the following three steps: (1) all tasks are represented in STRIPS notation using the transformation technique described by Gazen and Knoblock [8], (2) no tasks show negative effects, i.e., delete lists are ignored, and (3) for each task used in the task network TN, an additional non-relaxed task schema is introduced which is modified in such a way that each solution of Π' contains one task for each of these schemata thus ensuring that each solution to Π' is a refinement of TN.

Before we formally describe the complete domain transformation, we briefly sketch the details of step (3): we include an additional task schema in $\mathcal{T}'$ for any task that occurs in TN. To ensure that these additional task schemata are used by any solution of Π', we simply modify the goal description as follows. Let l:t be a task in TN. To encode that the task l:t occurs in a plan, we augment the postcondition of the (additional) task schema t by the fact *occ-l*. To ensure that this task schema is used at most once in any solution, we augment the precondition of t by the fact *not-occ-l*, which encodes $\neg$*occ-l*. Finally, to enforce the occurrence of TN's tasks, we simply add a fact *occ-l* for any task l:t of TN to the goal description. The ordering constraints are also encoded by means of the occurrence facts: let (l, l') be an ordering constraint in TN and l:t, l':t' the respective tasks. Then, this ordering is encoded by including the fact *occ-l* in the precondition of the (additional) task schema t'.

Now, we formally define the domain transformation. As our heuristic is based on GRAPHPLAN which uses the STRIPS specification to describe tasks, we also transform the domain into the respective representation in which tasks do not have negative preconditions and effects are explicitly represented using add and delete lists. To this end, we introduce a fact *not-p* for each fact $p \in \mathcal{P}$, which occurs negatively in some task's precondition and alter the schemata, such that

in each state s, either *not-p* $\in s$ or $p \in s$. Let $neg(\mathcal{T}) := \{p \mid \exists t \in \mathcal{T}, \text{ s.t. } \neg p \in \text{prec}(t)\}$. For $t = (\text{prec}(t), \text{eff}(t)) \in \mathcal{T}$, let

$$
\begin{aligned}
&\mathit{STRIPS}(t) = (\text{pre}(t), \text{add}(t), \text{del}(t)) := \\
&\qquad (\{p \mid p \in \text{prec}(t), p \text{ fact}\} \cup \{\textit{not-p} \mid \neg p \in \text{prec}(t), p \text{ fact}\}, \\
&\qquad\ \{p \mid p \in \text{eff}(t), p \text{ fact}\} \cup \{\textit{not-p} \mid \neg p \in \text{eff}(t), p \text{ fact}, p \in \mathit{neg}(\mathcal{T})\}, \\
&\qquad\ \{p \mid \neg p \in \text{eff}(t), p \text{ fact}\} \cup \{\textit{not-p} \mid p \in \text{eff}(t), p \text{ fact}, p \in \mathit{neg}(\mathcal{T})\}).
\end{aligned}
$$

Now, let $\Pi = (\mathcal{T}, \mathcal{M}, s_{init}, TN_{init}, g)$ and the current task network to refine be $TN = (T, \prec, V, C)$. The transformed classical planning problem in STRIPS notation is then given by $\Pi' = (\mathcal{T}', \emptyset, s'_{init}, \varepsilon, g')$ with:

$$
\begin{aligned}
\mathcal{T}' := \{&(\text{pre}(t), \text{add}(t), \emptyset) \mid \exists t \in \mathcal{T}, \text{ s.t. } t \text{ is a } \textit{primitive} \text{ task schema} && (1.1)\\
&\qquad \text{and } \mathit{STRIPS}(t) = (\text{pre}(t), \text{add}(t), \text{del}(t))\} \cup \\
&\{(\{\textit{not-occ-l}\} \cup \{\textit{occ-l'} \mid (l', l) \in \prec \text{ or } l' \rightarrow_\phi l \in C\}, && (1.2)\\
&\ \{\textit{occ-l}\}, \\
&\ \{\textit{not-occ-l}\}) \mid \exists l{:}t \in T, \ t \text{ is a } \textit{compound} \text{ task schema}\} \cup \\
&\{(\text{pre}(t) \cup \{\textit{not-occ-l}\} \cup \{\textit{occ-l'} \mid (l', l) \in \prec \text{ or } l' \rightarrow_\phi l \in C\}, && (1.3)\\
&\ \text{add}(t) \cup \{\textit{occ-l}\}, \\
&\ \text{del}(t) \cup \{\textit{not-occ-l}\})[v_1 \leftarrow c_1, \ldots, v_n \leftarrow c_n] \\
&\ \mid \exists l{:}t \in T, \text{ s.t. } t \text{ is a } \textit{primitive} \text{ task schema}, \\
&\ \ \mathit{STRIPS}(t) = (\text{pre}(t), \text{add}(t), \text{del}(t)), \\
&\ \ V \models (l, v_i) = c_i \text{ for all } 1 \leq i \leq n, \text{ and there is no } c' \text{ and no} \\
&\ \ v' \notin \{v_1, \ldots, v_n\} \text{ s.t. } V \models (l, v') = c'\}
\end{aligned}
$$

$$
s'_{init} := s_{init} \cup \{\textit{not-p} \mid p \in \mathit{neg}(\mathcal{T}) \text{ and } p \notin s_{init}\} \cup \{\textit{not-occ-l} \mid \exists l{:}t \in T\} \qquad (2)
$$

$$
g' := g \cup \{\textit{occ-l} \mid \exists l{:}t \in T\} \qquad (3)
$$

The set defined in (1.1) contains all primitive task schemata of $\mathcal{T}$ in a relaxed form in which delete lists are ignored. This enables an efficient construction of the planning graph, as the relaxed tasks cannot introduce any mutex relations.

The sets defined in (1.2) and (1.3) contain additional task schemata for all tasks contained in the task network TN. Their preconditions as well as add- and delete lists use the facts *occ-l* and *not-occ-l* to encode that a task in TN with label l occurs in a task network; they are also used to encode the ordering constraints and causal links of TN (cf. first line of (1.2) and (1.3), respectively).

The set defined in (1.2) adds schemata of *compound* tasks to $\mathcal{T}'$, which can thereby be used by our heuristic like primitive tasks. However, because our heuristic does not directly handle decomposition, they are only included to represent the ordering constraints in TN, which would be lost if the compound tasks would not be included in $\mathcal{T}'$. The preconditions and effects of the compound tasks are ignored because their semantics are different from the ones of

primitive tasks. They are abstract specifications of their subtasks, but they do not specify state transitions as is the case for primitive tasks.

(1.3) defines the task schemata for all *primitive* tasks occurring in *TN*. They do not show any relaxation, because these tasks are the only information currently available. Planning systems that perform progression update the initial state as the generated sequence of tasks increases. In our case, the initial state remains always the same — it is just the current task network that is becoming larger. Hence, the task schemata defined in this step encode the information about the progression of the search and consequently should not be relaxed: relaxing these task schemata would correspond to relaxing the current state in a progression search. Please also note that it is only the tasks in this step that might introduce mutex relations into the planning graph. To respect the variable bindings, we simply substitute any bound variable in a task by the constant it is bound to (denoted by $[v_i \leftarrow c_i]$). Thus, we do not encode variable constraints of the form $(l, v) \neq x$, where x is either a constant or another variable. We also do not encode variable constraints of the form $(l, v) = (l', v')$ when neither $V \models (l, v) = c$ nor $V \models (l', v') = c$ for some constant c; both cases remain future work.

The new initial state is defined in (2). This definition belongs to the procedure of transforming a planning domain into an equivalent STRIPS representation.

Finally, (3) defines the new goal description. It contains the original goal description, but is extended by the occurrence facts for all tasks which are contained in the current task network *TN*.

Our domain transformation clearly runs in polynomial time w. r. t. the size of Π and *TN*. Please note that the delete-relaxation part of our transformation has to be performed only *once*, whereas the construction of s'_{init}, g', and the insertion of the additional task schemata (cf. (1.2) and (1.3)) into $\mathcal{T}'$ has to be done for each task network *TN*. Thus, ignoring preprocessing, the transformation can be done in $O(|TN|)$. However, an incremental domain transformation will clearly reduce the necessary effort.

4.2 Heuristic Calculation

In this section, we describe how our heuristic calculates an admissible estimate of the metric function m (cf. Section 2) based on the transformed domain model Π' described in the last section. We call our heuristic h^2_{dr}, since it fuses ideas from the h^2 heuristic with delete relaxation.

Given a task network *TN*, we build the planning graph starting in the new initial state $s := s'_{init}$ of Π'. Let *layer* be the fix point layer produced by calling GRAPHPLAN in s and let *mutex* be the set of its symmetric mutex relations.

First of all, we can define $h^2_{dr}(s) := \infty$, if $g \not\subseteq$ *layer* or if there is a mutex relation $\{p, p'\} \in$ *mutex* with $p, p' \in g$, since in these cases the goal formula can not be satisfied. Otherwise, the heuristic value is calculated as follows.

We can add all the weights of preference facts to $h^2_{dr}(s)$ that do not appear in *layer*, as they can not be made true even by relaxed tasks.

Furthermore, all preference facts, for which there is a mutex to g are also violated: let $layer_{\bar{g}} := layer \setminus g$ and $\{p, p'\} \in mutex$, $p \in layer_{\bar{g}}$ being a preference fact and $p' \in g$. Then, the weight of p can be used to increase the value of $h^2_{dr}(s)$, as p directly contradicts a goal fact.

The more interesting case is the handling of mutexes for the remaining facts $layer'_{\bar{g}} := layer_{\bar{g}} \setminus \{p \mid \exists p' \in g, \text{ s.t. } \{p, p'\} \in mutex\}$. Let $b : layer'_{\bar{g}} \to \{\top, \bot\}$ be a truth assignment of the facts in $layer'_{\bar{g}}$ which respects the mutex relations; i.e., if $\{p, p'\} \in mutex$, then either $b(p) = \neg b(p')$ or $b(p) = b(p') = \bot$. Since we are going to use this assignment to calculate a non-overestimation of the metric m, b needs to minimize the sum $\sum_{(p,n) \in Pref, b(p)=\bot} n$.

Putting it all together, we get:

$$h^2_{dr}(s) := \sum_{\substack{(p,n) \in Pref \text{ and} \\ (p \notin layer \text{ or } \{p,p'\} \in mutex, p' \in g)}} n + \min_b \sum_{(p,n) \in Pref, b(p)=\bot} n$$

Whereas the first summation term can clearly be calculated in linear time w. r. t. the size of *Pref*, the (optimal) calculation of the second term turns out to be NP hard. We can prove this by a reduction from the weighted minimum vertex cover problem: let $\mathcal{G} = (V, E)$ be a graph with a weight $w(v)$ for each $v \in V$. Then, the minimal weighted vertex cover is a set $V' \subseteq V$, such that for each edge $(v, v') \in E$, at least one of the vertices v, v' is in V' and the weighted sum $\sum_{v \in V'} w(v)$ is minimal. If we set $layer'_{\bar{g}} := V$, $mutex := E$, and $Pref = \{(p, n) | p \in V,\ w(p) = n\}$, it is easy to see that the value of $\min_b \sum_{(p,n) \in Pref, b(p)=\bot} n$ is also the value of the minimal weighted vertex cover. As the minimal *weighted* vertex cover problem is a generalization of the NP complete vertex cover problem [14], in which there are no weights and the decision problem is whether there is a vertex cover V' of size at most k, we have shown the NP hardness of the calculation of the second term in our heuristic function.

Please note that the input size of this NP hard subproblem is bounded by the size of the intersection of the preference facts with $layer'_{\bar{g}}$. Because only the tasks in *TN* can introduce mutex relations, we do not expect this set to be overly large. Hence, the calculation will probably be tractable in practice. As a second observation, please note that the NP hardness of our heuristic comes with the prize of admissibility together with high accuracy. One could easily drop this term and still achieve an admissible, but less accurate, heuristic. A second possibility to achieve tractability is to use polynomial-time bounded approximations for this term thus sacrificing admissibility.

5 Related Work

Our work is closely related to that of Baier et al. [1]. Using their compilation technique [2], they transform a (classical) planning problem with preferences on state trajectories into an equivalent planning problem that does only contain soft goals. They solve this problem via heuristic search using a branch-and-

bound algorithm that performs progression in the space of states (whereas our algorithm is a POCL algorithm which performs search in the space of plans).

They propose several heuristics for the search guidance and for pruning. The heuristics they propose for pruning are the *Optimistic Metric Function (O)* and the *Best Relaxed Metric Function (B)*.

We do not give details about O, as its calculation relies on their compilation technique to represent state trajectories constraints and because B dominates O, anyway.

B basically reduces to the same idea like our h^2_{dr} heuristic, but it is much easer to formulate and calculate, as it takes a state as input rather than a task network: starting in the current state, it builds the relaxed planning graph using tasks with delete relaxation until it reaches the final fact layer. Note that due to the absence of negative effects this graph does not show any mutex relations. B is then the metric value m evaluated in the last fact layer[2].

6 Summary and Future Work

We presented a branch-and-bound algorithm for solving hybrid planning problems with soft goals. Its explicit representation of flaws and modifications enables it to use arbitrary partial order search strategies rather than just progression as most of the other preference-based search algorithms do.

Additionally, we have introduced h^2_{dr}, a heuristic based on the h^2 heuristic in combination with a relaxed reachability analysis, for handling soft goals in the context of hybrid and POCL planning. So far, all of the work in the field of heuristics for POCL planners does either not incorporate the given metric function at all, which is true for heuristics that only use the (number of) tasks, open preconditions, ordering- and variable constraints and causal links and do thus not perform any well-informed goal distance estimate, or the metric they do estimate is based on actions costs [15,16]. Thus, h^2_{dr} is — to the knowledge of the authors — the first heuristic for partial order planners that estimates the quality of a task network w. r. t. preferences, rather than action costs. h^2_{dr} works as follows: first, we perform a domain transformation that reduces the problem of calculating a heuristic value for a *task network* to the problem of calculating a heuristic value for a *state*. Starting in that state, we build a partially relaxed planning graph to perform a relaxed reachability analysis. Afterwards, we calculate the heuristic value based on the last fact layer of this planning graph, taking into account the mutex relations present in that layer.

Beside an empirical evaluation of our search algorithm and heuristic, future work will also include the adaptation of our heuristic to handle state trajectory constraints. Their evaluation in POCL planning seems straight-forward as the trajectories are implicitly given in the current task network.

[2] To be precise, B is defined as the minimum of the metric value m evaluated in *each* fact layer. These two definitions are different from each other only in the case where violating a preference can increase the plan quality which is not the case in our setting.

Acknowledgements

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Baier, J.A., Bacchus, F., McIlraith, S.A.: A heuristic search approach to planning with temporally extended preferences. Artificial Intelligence 173, 593–618 (2009)
2. Baier, J.A., McIlraith, S.A.: Planning with first-order temporally extended goals using heuristic search. In: Proc. of the 21st National Conference on AI (AAAI 2006). pp. 788–795 (2006)
3. Biundo, S., Bercher, P., Geier, T., Müller, F., Schattenberg, B.: Advanced user assistance based on AI planning. Cognitive Systems Research 12(3-4), 219–236 (2011), special Issue on Complex Cognition
4. Biundo, S., Schattenberg, B.: From abstract crisis to concrete relief (a preliminary report on combining state abstraction and HTN planning). In: Proc. of the 6th European Conference on Planning (ECP 2001). pp. 157–168 (2001)
5. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300 (1997)
6. Erol, K., Hendler, J., Nau, D.S.: UMCP: A sound and complete procedure for hierarchical task-network planning. In: Proc. of the 2nd International Conference on AI Planning Systems (AIPS 1994). pp. 249–254 (1994)
7. Estlin, T.A., Chien, S.A., Wang, X.: An argument for a hybrid HTN/operator-based approach to planning. In: Proc. of the 4th European Conference on Planning: Recent Advances in AI Planning (ECP 1997). pp. 182–194 (1997)
8. Gazen, B.C., Knoblock, C.A.: Combining the expressivity of ucpop with the efficiency of graphplan. In: Proc. of the 4th European Conference on Planning: Recent Advances in AI Planning (ECP 1997). pp. 221–233 (1997)
9. Geier, T., Bercher, P.: On the decidability of HTN planning with task insertion. In: Proc. of the 22nd International Joint Conference on AI (IJCAI 2011). pp. 1955–1961 (2011)
10. Gerevini, A., Kuter, U., Nau, D.S., Saetti, A., Waisbrot, N.: Combining domain-independent planning and HTN planning: The duet planner. In: Proc. of the 18th European Conference on AI (ECAI 2008). pp. 573–577 (2008)
11. Gerevini, A., Long, D.: Plan constraints and preferences in PDDL3. Tech. rep., Department of Electronics for Automation, University of Brescia, Italy (2005)
12. Haslum, P., Geffner, H.: Admissible heuristics for optimal planning. In: The Fifth International Conference on AI Planning Systems (AIPS 2000). pp. 140–149 (2000)
13. Kambhampati, S., Mali, A., Srivastava, B.: Hybrid planning for partially hierarchical domains. In: Proc. of the 15th National Conference on AI (AAAI 1998). pp. 882–888 (1998)
14. Karp, R.M.: Complexity of Computer Computations, chap. Reducibility Among Combinatorial Problems, pp. 85–103 (1972)
15. Nguyen, X., Kambhampati, S.: Reviving partial order planning. In: Proc. of the 17th International Joint Conference on AI (IJCAI 2001). pp. 459–466 (2001)
16. Younes, H.L.S., Simmons, R.G.: VHPOP: Versatile heuristic partial order planner. Journal of Artificial Intelligence Research 20, 405–430 (2003)