A Heuristic for Hybrid Planning with Preferences

Pascal Bercher and **Susanne Biundo**
Institute of Artificial Intelligence
Ulm University, D-89069 Ulm, Germany
firstName.lastName@uni-ulm.de

Abstract

In this paper, we introduce an admissible heuristic for hybrid planning with preferences. Hybrid planning is the fusion of hierarchical task network (HTN) planning with partial order causal link (POCL) planning. We consider preferences to be *soft goals* — facts one would like to see satisfied in a goal state, but which do not have to hold necessarily.

Our heuristic estimates the best quality of any solution that can be developed from the current plan under consideration. It can thus be used by any branch-and-bound algorithm that performs search in the space of plans to prune suboptimal plans from the search space.

Introduction

In real-world planning, for example, when assisting human users in their everyday life (Biundo et al. 2011), plans are often of different quality depending on the specific user carrying out the plan. 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, 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 (Gerevini and Long 2005), the language for the fifth International Planning Competition (IPC-5). In PDDL3, preferences are either *soft goals* or *plan constraints*. The former are non-mandatory conditions that should hold in the final state produced by a solution, whereas the latter are non-mandatory constraints on the state trajectories induced by a solution. In this paper, we focus on solving planning problems with *soft goals*.

We are concerned with solving hybrid planning problems (Kambhampati, Mali, and Srivastava 1998; Gerevini et al. 2008; Biundo et al. 2011; Geier and Bercher 2011) favoring those solutions which satisfy the preferences to the largest 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 planning: many real-world problems are inherently hierarchical. 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 hierarchical planning, but also enables the arbitrary insertion of tasks to support open preconditions as in classical planning.

Problem Setting

Our formalization of hybrid planning fuses hierarchical task network (HTN) planning (Erol, Hendler, and Nau 1994) with partial order causal link (POCL) planning (McAllester and Rosenblitt 1991).

A hybrid planning problem is given in terms of an initial task network. Task networks can contain *primitive tasks*, which correspond to classical planning operators, and *compound tasks*, which represent high-level activities. Both primitive and compound tasks show preconditions and effects. However, only primitive tasks specify state transitions. Compound tasks are abstract specifications of standard solutions for their preconditions and effects. For every compound task the domain model holds several decomposition methods mapping that task to some pre-defined task network which "implements" its desired effect and requires the specified precondition. A solution is then a refinement of the initial task network which consists only of primitive tasks and is executable in the classical sense.

More formally, a hybrid planning problem Π is a tuple $(\mathcal{T}_p, \mathcal{T}_c, \mathcal{M}, C, s_{init}, TN_{init}, g)$ consisting of the following components. The set $\mathcal{T}_p$ contains the primitive task schemata and $\mathcal{T}_c$ contains the compound ones. Task schemata are tuples $t = (\text{prec}, \text{add}, \text{del})$ consisting of a precondition, an add-, and a delete list; the latter two are also referred to as effects. The precondition and effects are sets of positive literals and depend on the parameters $\bar{v}(t)$. $\mathcal{M}$ is the set of available decomposition methods, each being a tuple (t, TN) mapping a compound task schema $t \in \mathcal{T}_c$ to a task network TN. C is the set of available constants and $\mathcal{P}$ denotes the set of all possible ground atomic propositions using constants from C. Then, $s_{init} \in 2^{\mathcal{P}}$ is the initial state, $g \subseteq \mathcal{P}$ is the goal description and TN_{init} is the initial task network. Please note that we encode the initial state and the goal description as the effect and the precondition of an artificial start and end task, respectively. A task network is a tuple $(T, \prec, V, CL)$ and consists of the following components. T is a set of tasks, 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. $\prec$ is a partial order on T, V is a set of variable constraints, and CL is a set of causal links. A causal link $l' \rightarrow_{\phi} l \in CL$ indicates that the precondition literal ϕ of $l{:}t \in T$ is an effect of $l'{:}t' \in T$ and is *supported* this way.

Every solution $TN' = (T', \prec', V', CL')$ of Π must fulfill the standard POCL solution criteria (Younes and Simmons 2003) of having no unsupported preconditions and causal threats. These criteria ensure that every linearization of T' that respects $\prec'$, V', and CL' is an executable task sequence that transforms the initial state into a state satisfying the goal description. In addition, all tasks in T' must be primitive and TN' must be a refinement of TN_{init} w.r.t. decomposition of compound tasks and the insertion of tasks, causal links, ordering- and variable constraints.

In addition to the planning problem, we are given *preferences* as a set of ground facts $Pref \subseteq \mathcal{P}$ which denote *optional* planning goals. We can hence assume $Pref \cap g = \emptyset$. Each preference p has an associated weight $w(p)$, which is interpreted as a "violation value" and depreciates a given solution *TN* by the respective weight if p does not necessarily hold in all final states produced by *TN* (which we denote by $TN \not\models p$). More formally, the quality (metric) of a solution *TN* is defined by $m(TN) := \sum_{p \in Pref \text{ with } TN \not\models p} w(p)$. Then, a solution TN_1 is preferred over a solution TN_2, if and only if $m(TN_1) \leq m(TN_2)$.

Preference Heuristic for POCL Planning

In this section we describe our heuristic that can be used to estimate the final quality of the current task network under consideration.

In a first step, we transform a given *hybrid* planning problem Π with the current task network $TN = (T, \prec, V, CL)$ under consideration into a relaxed *classical* planning problem Π', such that Π' has a solution if *TN* can be developed to a solution of Π. Furthermore, any solution of Π' contains all primitive tasks of *TN* and respects its constraints. Using this reduction, we reduce the problem of calculating a heuristic value for a *task network* to the problem of calculating a heuristic value for the initial *state* of the transformed planning problem. In the second step, we build a planning graph until is has "leveled off" (i.e., until a fixed point is reached) and use the mutex relations still present in the last fact layer. The heuristic thus shows close relationship to the h^2 heuristic (Haslum and Geffner 2000), the heuristic implicitly used by the planning system GRAPHPLAN (Blum and Furst 1997).

For the sake of simplicity, we assume that V binds every variable to a constant $c \in C$, i.e., *TN* is ground.

Domain Transformation

In this section, we show how to transform a hybrid planning problem Π with a current task network *TN* under consideration into a classical planning problem Π', s.t. each solution of Π' is a refinement of *TN* w.r.t. its primitive tasks. Although we are going to use the transformed *classical* planning problem Π' for calculating an estimate of the best solution quality of the original *hybrid* planning problem Π, we can ignore the hierarchical components of Π (i.e., compound tasks and decomposition methods) without "loosing solutions" or sacrificing admissibility. This is due to the fact that compound tasks do not directly contribute to the satisfaction of facts like the primitive tasks do — they can be regarded as additional constraints, since they need to be decomposed using only the available decomposition methods. Ignoring the hierarchical components of Π is hence just a relaxation.

The main idea behind the domain transformation is to augment the set of available task schemata by an additional copy of all tasks that occur in *TN* and to alter the initial and goal state description, s.t. any solution of the transformed problem must contain these tasks with exactly the same ordering as present in *TN*. For the remaining, i.e., non-additional, task schemata we perform a delete-relaxation for efficiency reasons: ignoring their delete lists improves the speed of building the planning graph, which is done in the second step (see next section) based on the transformed planning problem. However, the *additional* task schemata cannot show any relaxation, because their effects are the only available information about the current task network. 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 changes. Hence, the additional task schemata 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. Thus, let $\Pi = (\mathcal{T}_p, \mathcal{T}_c, \mathcal{M}, C, s_{init}, TN_{init}, g)$ and the current task network to refine be $TN = (T, \prec, V, CL)$. The transformed classical planning problem is then given by $\Pi' = (\mathcal{T}', C, s'_{init}, g')$ and defined as follows.

$\mathcal{T}'$ contains additional copies of the task schemata used by *TN* as well as all task schemata from Π, but with ignored delete lists. More formally, $\mathcal{T}' := \textit{delete-relax}(\mathcal{T}_p) \cup \textit{encode}(TN)$ with $\textit{delete-relax}(\mathcal{T}_p) := \{(\text{prec}, \text{add}, \emptyset) \mid (\text{prec}, \text{add}, \text{del}) \in \mathcal{T}_p\}$ and $\textit{encode}(TN)$ containing the fresh copies of the task schemata used by *TN*. For every task $l{:}t$ in T, $\textit{encode}(TN)$ contains a modified version of its task schema t, s.t.:

- every task schema t' in $\textit{encode}(TN)$ is used by exactly one task of every solution of Π', and
- the tasks of every solution of Π' which use the task schemata in $\textit{encode}(TN)$ are ordered in the same way like their corresponding tasks in *TN*.

For the first property, let $l{:}t \in T$ be a task in the task network *TN*. The additional task schema t' is an extension of t s.t. it contains the additional precondition $\neg l$ and the additional effect l, where l is a new nullary predicate symbol. Since we use the STRIPS formalization, in which no negative preconditions are allowed, we use two mutually exclusive facts l and *not-l* to encode l and $\neg l$. Now, every solution can contain every task that uses a task schema from *encode*(*TN*) *at most* once. To ensure that it is contained *at least* once, we also alter the goal description to contain the fact l for every primitive task $l{:}t \in T$. For the second property, let $l_i{:}t_i \in T$ and t'_i be the corresponding encoding in *encode*(*TN*). Then, t'_i has an additional precondition l_j for every primitive task $l_j{:}t_j \in T$ that has to be ordered before $l_i{:}t_i$ w.r.t. $\prec$ and CL. Thus, we get:

$$\begin{aligned}
&\mathit{encode}(\mathit{TN}) := \{\mathit{encode}(l{:}t) \mid \exists l{:}t \in T,\ \text{s.t. } t \in \mathcal{T}_p\},\\
&\text{with } \mathit{encode}(l{:}(\text{pre}, \text{add}, \text{del})) :=\\
&\quad(\text{pre} \cup \{\mathit{not}\text{-}l\} \cup \{l' \mid l' \prec l \text{ or } l' \rightarrow_\phi l \in CL\},\\
&\quad\ \text{add} \cup \{l\},\\
&\quad\ \text{del} \cup \{\mathit{not}\text{-}l\})
\end{aligned}$$

The new initial state is defined by $s'_{init} := s_{init} \cup \{\mathit{not}\text{-}l \mid \exists l{:}t \in T,\ \text{s.t. } t \in \mathcal{T}_p\}$; it contains the information that no task of *TN* was inserted, yet.

The new goal description is defined by $g' := g \cup \{l \mid \exists l{:}t \in T,\ \text{s.t. } t \in \mathcal{T}_p\}$ and encodes that every solution must contain the tasks in T.

Concerning the computational complexity of the transformation, note that the delete-relaxation of our transformation has to be performed only *once* for the planning problem Π, whereas the construction of s'_{init}, g', and the calculation of *encode*(*TN*) has to be done for each current task network *TN*. This transformation runs in $O(|\mathit{TN}|)$; however, an incremental domain transformation will clearly reduce the necessary effort.

Heuristic Calculation

In this section, we describe how our heuristic uses the transformed domain model Π' described in the last section to calculate an admissible estimate of the metric function m. We call our heuristic h^2_{dr}, as it fuses ideas from the h^2 heuristic with delete relaxation.

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, which is 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 task network *TN*, our heuristic uses this final fact layer to estimate the best quality of any solution obtainable by refining *TN*.

Given a hybrid planning problem Π and a task network *TN*, let $\Pi' = (\mathcal{T}, C, s_{init}, g)$ be the transformed classical planning problem. Then, we build the planning graph starting in $s := s_{init}$. Let $layer \subseteq \mathcal{P}$ be the fix point layer produced by calling GRAPHPLAN in s and let $\mathit{mutex} \subseteq \mathcal{P} \times \mathcal{P}$ be the set of its symmetric mutex relations. For the sake of readability, we divide the final fact layer into four pairwise disjunctive sets: $layer := l_g \dot\cup l_{p,m} \dot\cup l_{p,\neg m} \dot\cup l_{\neg p}$. $l_g := layer \cap g$ is the set of goal facts in the final layer, $l_{p,m} := layer \cap \{p \in \mathit{Pref} \mid \exists m \in \mathit{mutex}, p \in m, m \cap l_g \neq \emptyset\}$ is the set of all preferences in the final layer which are mutex so some goal fact. Analogously, $l_{p,\neg m} := layer \cap \{p \in \mathit{Pref} \mid \nexists m \in \mathit{mutex}, p \in m, m \cap l_g \neq \emptyset\}$ is the set of all preferences for which there is no goal fact which it is mutex with. Finally, $l_{\neg p} := layer \setminus \mathit{Pref} \setminus g$ is the set of all remaining facts in that layer.

First of all, we can define $h^2_{dr}(s) := \infty$ if $g \neq l_g$ (or, equivalently, $g \not\subseteq layer$) or if there is a mutex relation $m \in \mathit{mutex}$ with $m \subseteq l_g$, since in these cases the goal formula can not be satisfied. Preferences which do not appear in $layer$ can be used to increase $h^2_{dr}(s)$, as they can not be satisfied even by delete-relaxed tasks. Furthermore, all preferences in $l_{p,m}$ can never be satisfied, as they are mutex to goal facts. The only non-trivial case is handling the mutexes between preferences in $l_{p,\neg m}$. We know that certain pairs of preferences cannot be true at the same time, but we do not know which set of preferences do – or do not – necessarily hold in a final state. The idea is to calculate all subsets of preferences in $l_{p,\neg m}$ which cannot be true at the same time and choose the one which leads to the best plan quality. Formally, let $b : l_{p,\neg m} \rightarrow \{\top, \bot\}$ be a truth assignment of the preferences in $l_{p,\neg m}$ which respects the mutex relations; i.e., if $\{p, p'\} \in \mathit{mutex}$, then either $b(p) = \neg b(p')$ or $b(p) = b(p') = \bot$. Since we use this assignment to calculate a non-overestimation of the metric m, b needs to minimize the sum $\sum_{p \in l_{p,\neg m}, b(p)=\bot} w(p)$. Putting it all together, we get:

$$h^2_{dr}(s) := \underbrace{\sum_{\substack{p \in \mathit{Pref} \text{ and} \\ (p \notin layer \text{ or } p \in l_{p,m})}} w(p)}_{(1)} + \underbrace{\min_b \sum_{\substack{p \in l_{p,\neg m} \text{ and} \\ b(p)=\bot}} w(p)}_{(2)}$$

Whereas the summation term (1) can clearly be calculated in linear time w. r. t. the size of Pref, term (2) turns out to be much harder.

Theorem 1. *The calculation of summation term (2) is NP hard w.r.t. the size of* $l_{p,\neg m}$.

We prove this by a reduction from the weighted minimum vertex cover problem:

Proof. Let $\mathcal{G} = (V, E)$ be a graph with a weight $w(v)$ for each $v \in V$. Then, the minimum 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 $l_{p,\neg m} := V$ and $\mathit{mutex} := E$, it is easy to see that the value of $\min_b \sum_{p \in l_{p,\neg m}, b(p)=\bot} w(p)$ is also the value of the minimum weighted vertex cover. As the minimum *weighted* vertex cover problem is a generalization of the NP complete vertex cover problem (Karp 1972), in which there are no weights and the decision problem is whether there is a vertex cover V' of size at most k, NP hardness follows. $\square$

Please note that the size of this NP hard subproblem is bounded by the number of preferences. Hence, solving it will probably not dominate the runtime of the heuristic calculation. 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 or use approximations to achieve an admissible, but less accurate, heuristic.

Related Work

Our work is closely related to that of Baier et al. (2009). They solve classical planning problems with soft goals via heuristic search using a state-based branch-and-bound algorithm. Among others, they propose the *Best Relaxed Metric Function (B)* for pruning. *B* basically reduces to the same idea like our h^2_{dr} heuristic; the main difference is that 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. 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.[1]

The idea of encoding a task network into a planning problem was also already addressed (Ramirez and Geffner 2009; Alford, Kuter, and Nau 2009), but neither for *hybrid* planning problems, nor for *partially ordered* task networks, nor was it used for heuristic calculation in the context of POCL planning, which is one of our main contributions.

Summary

We have proposed an admissible heuristic for hybrid planning with preferences that allows to estimate the final quality of the current task network under consideration. Its core idea is to reduce the problem of calculating a heuristic value for a *task network* to the problem of calculating a heuristic value for a *state*. The actual heuristic calculation is based on a planning graph which is used for a relaxed reachability analysis.

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

Alford, R.; Kuter, U.; and Nau, D. S. 2009. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *IJCAI 2009*, 1629–1634.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *AI* 173:593–618.

Biundo, S.; Bercher, P.; Geier, T.; Müller, F.; and Schattenberg, B. 2011. Advanced user assistance based on AI planning. *Cognitive Systems Research* 12(3-4):219–236. Special Issue on Complex Cognition.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *AI* 90:281–300.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *AIPS 1994*, 249–254.

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *IJCAI 2011*, 1955–1961.

Gerevini, A., and Long, D. 2005. Plan constraints and preferences in PDDL3. Technical report, Dep. of Electronics for Automation, University of Brescia, Italy.

Gerevini, A.; Kuter, U.; Nau, D. S.; Saetti, A.; and Waisbrot, N. 2008. Combining domain-independent planning and HTN planning: The duet planner. In *ECAI 2008*, 573–577.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS 2000*, 140–149.

Kambhampati, S.; Mali, A.; and Srivastava, B. 1998. Hybrid planning for partially hierarchical domains. In *AAAI 1998*, 882–888.

Karp, R. M. 1972. *Complexity of Computer Computations*. chapter Reducibility Among Combinatorial Problems, 85–103.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *AAAI 1991*, 634–639.

Ramirez, M., and Geffner, H. 2009. Plan recognition as planning. In *IJCAI 2009*, 1778–1783.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *JAIR* 20:405–430.

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