Hybrid Planning
Theoretical Foundations and Practical Applications

Pascal Bercher

supervisor: Susanne Biundo
Institute of Artificial Intelligence,
Ulm University, Germany,
email: forename.surname@uni-ulm.de

Abstract

The thesis presents a novel set-theoretic formalization of (propositional) *hybrid planning* – a planning framework that fuses *Hierarchical Task Network* (HTN) planning with *Partial-Order Causal-Link* (POCL) planning. Several sub classes thereof are identified that capture well-known problems such as HTN planning and POCL planning. For these problem classes, the complexity of the plan-existence problem is investigated, i.e., the problem of deciding whether there exists a solution for a given planning problem.

For solving the problems of the respective problem classes, a hybrid planning algorithm is presented. Its search is guided by informed heuristics. Several such heuristics are introduced, both for POCL planning problems (i.e., problems without task hierarchy) and for hybrid planning problems (i.e., heuristics that are "hierarchy-aware").

1 Introduction

Hybrid planning (Biundo and Schattenberg 2001) combines *Hierarchical Task Network* (HTN) planning with concepts known from *Partial-Order Causal-Link* (POCL) planning. The smooth integration of hierarchical planning with causal reasoning enables a planning process that is similar to the way humans solve their tasks: on the one hand, it is done in a top-down manner, where initially abstract tasks become refined into more concrete courses of action; on the other hand, it is driven by causality: actions are inserted into plans based on causal reasoning. That similarity has several advantages when planning for humans: it enables the realization of mixed-initiative systems that allow humans to make decisions concerning the planning process. The hierarchy further allows to incorporate expert knowledge into the domain model, as such knowledge is often structured in a hierarchical manner. That is one of the main reasons why many real-world planning applications are solved via hierarchical planning systems (Nau et al. 2005; Lin, Kuter, and Sirin 2008; Biundo et al. 2011).

The explicit representation of causal dependencies between actions enables to repair plans in case of execution failures (Bidot, Schattenberg, and Biundo 2008), but also to generate so-called plan explanations – a chain of arguments explaining what purpose a specific action serves for the given problem. The action hierarchy further allows for more flexible explanations w.r.t. the level of abstraction (Seegebarth et al. 2012). The integration of these capabilities enables the realization of intelligent assistance systems (Biundo et al. 2011) in various contexts of daily life, as demonstrated by a system that assists a user in the task of setting up a complex home theater (Honold et al. 2014; Bercher et al. 2014; 2015).

The next section formally introduces hybrid planning and its sub problem classes resulting from various restrictions on the domain and problem description. Section 3 presents the complexity results for the presented problem classes and further restrictions thereof. Section 4 presents the hybrid planning algorithm PANDA that can solve planning problems of all presented problem classes. Section 5 presents heuristics for hybrid planning before Section 6 gives a short summary.

2 Hybrid Planning

Hybrid planning (Kambhampati, Mali, and Srivastava 1998; Biundo and Schattenberg 2001) fuses Hierarchical Task Network (HTN) planning (Erol, Hendler, and Nau 1996) with Partial-Order Causal-Link (POCL) planning (McAllester and Rosenblitt 1991; Penberthy and Weld 1992).

In hybrid planning, there are *primitive* and *abstract* (or *compound*) tasks. Both primitive and compound tasks are tuples $t = \langle \text{prec}^+, \text{prec}^-, \mathit{eff}^+, \mathit{eff}^- \rangle$ specifying the positive and negative preconditions and effects. Preconditions and effects are represented as sets of ground atoms. As usual, states are sets of ground atoms assuming the closed world assumption. Primitive tasks correspond to standard STRIPS actions[1]. Action and action sequence applicability are defined as usual. Plans are tuples $(PS, \prec, CL)$ consisting of the following elements. The set of plan steps PS is a set of uniquely labeled tasks $l : t$. Unique labeling is required to allow multiple occurrences of the same task within a plan. The set $\prec$ of ordering constraints induces a partial order on the plan steps in PS. The set CL contains the causal links of the plan. A causal link $l : t \rightarrow_{\varphi} l' : t'$ denotes that the precondition φ of the plan step $l' : t'$ is sup-

[1] A STRIPS action is typically given as a tuple $(\text{prec}, \text{add}, \text{del})$ consisting of a precondition, an add and a delete list. Here, we additionally feature a set of negative preconditions to allow more concise complexity analyses.

ported by the plan step $l : t$. If there is no causal link for a precondition φ we call it an *open* (or *unprotected*) *precondition*. Plans may also contain abstract tasks. These cannot be executed directly. Instead, they need to be decomposed into more specific plans using so-called (decomposition) methods. A method $m = \langle t, P \rangle$ maps an abstract task $t = \langle \text{prec}^+, \text{prec}^-, \mathit{eff}^+, \mathit{eff}^- \rangle$ to a plan P that is a predefined standard solution of t that "implements" that task (Biundo and Schattenberg 2001). Decomposition of t within a plan P' leads to a successor plan P'' in which t is replaced by P. Ordering constraints imposed on t are passed down to its sub tasks within P, as well as the causal links involving t.

Now, a *planning domain* $\mathcal{D}$ is given by the tuple $\langle T_a, T_p, M \rangle$ consisting of a finite set of abstract and primitive tasks T_a and T_p, respectively, and a set of methods M. A *planning problem* is given by a planning domain and an initial plan P_{init}. As it is the case in standard POCL planning, P_{init} contains two special primitive tasks that encode an initial state and a goal description. The task t_0 has no precondition and the initial state as effect. The task t_∞ has the goal description as precondition and no effects.

A plan P_{sol} is a solution to a hybrid planning problem if and only if the following criteria are met:

1. P_{sol} is a refinement of P_{init} w.r.t. the decomposition of abstract tasks and the insertion of (primitive or abstract) tasks, ordering constraints, and causal links.
2. P_{sol} needs to be executable in the initial state. Thus,
   (a) all tasks are primitive,
   (b) there are no open preconditions, and
   (c) there are no causal threats. That is, given a causal link $l : t \rightarrow_\varphi l' : t'$, we call the task $l'' : t''$ a threat to that link if and only if the ordering constraints allow it to be ordered between the tasks of the causal link and it has an effect $\neg\varphi$. So, if $\varphi \in \text{prec}^+$ of t', then $\varphi \in \mathit{eff}^-$ of t'' threatens that link and, for the other case, if $\varphi \in \text{prec}^-$ of t', then $\varphi \in \mathit{eff}^+$ of t'' threatens that link.

Note that solution criterion 1. is inherited from hierarchical planning. Enforcing any plan to be a refinement of the initial plan restricts the set of solutions to those that lie in the "decomposition" hierarchy that is implicitly defined using the decomposition methods[2]. Solution criterion 2. lists the the standard POCL solution criteria (McAllester and Rosenblitt 1991; Penberthy and Weld 1992). These ensure that every linearization compatible with the given ordering constraints is an executable action sequence thus satisfying the goal description.

Hybrid planning problems Hybrid planning problems are given by means of an initial plan consisting of primitive and/or abstract tasks that may be partially ordered and that plan as well as those referenced by the decomposition methods in the domain may contain causal links. Further, the solution criteria explicitly allow the insertion of tasks into plans. One could also define that criterion to not allow task insertions. In both cases, we refer to the respective problem class as hybrid planning. If restricting one or more of all these properties, one can express several problem classes already known to the literature:

POCL planning problems Given the domain does not contain abstract tasks, then the problem description is given in terms of an initial primitive plan that may be partially ordered and that may contain causal links. We refer to such a problem as a POCL planning problem. The literature ordinarily does not explicitly mention such "POCL planning problems"; instead, *POCL planning* is used to refer to a technique for solving classical planning problems, i.e., problems that are given in terms of an initial state and a goal description. Formally defining the set of all POCL planning problems enables to study the plan-existence problem for such problems – this is essential for generating heuristics for both hybrid and POCL planning, as each search node during planning constitutes such a planning problem.

PO planning problems When further restricting POCL planning problems in such a way that the initial plan may not contain causal links, we refer to the respective planning problem as a Partial-Order (PO) planning problem. Analogously to POCL problems, "PO problems" are not explicitly mentioned in the literature. Instead, *PO planning* (or *POP*) is used to refer to a planning technique solving classical planning problem via search in the space of plans (that do not show causal links). Kambhampati (1995) also refers to such planners as non-causal link planners.

HTN planning problems In typical hierarchical planning approaches, as opposed to hybrid planning, the insertion of tasks is not allowed. So, we define HTN problems as a special case of hybrid planning problems given (1) the solution criterion 1. is altered in such a way that task insertion is prohibited, and (2) neither the initial plan, nor any plan used by any method in the domain uses causal links.

TIHTN planning problems To study the theoretical impact of allowing tasks to be inserted into plans, we introduced a new problem class, called HTN planning with task insertion (TIHTN planning) (Geier and Bercher 2011). Then, the only difference between HTN problems and TIHTN problems is whether task insertion is allowed.

3 Complexity Results

The complexity of the plan-existence problem was studied for various of the presented problem classes.

3.1 HTN and TIHTN planning

The set-theoretic propositional formalization of hybrid planning given in the last section is an extension of the formalization given by Geier and Bercher (2011) to additionally feature causal links in the domain. Their formalization of

[2]There is hence a strong relationship between the solutions of hierarchical planning problems and the words within the language of formal grammars (Höller et al. 2014).

HTN and TIHTN planning further does not use causal links as a means to guarantee the executability of plans. Instead, they require the existence of an applicable action sequence (without explicitly representing their causal dependencies). For the thesis and in ongoing work, we alter that solution criterion s.t. a plan is regarded a solution if all its linearizations are applicable (as it is the defined for hybrid planning). Since the domain does not feature causal links, abstract tasks do not specify preconditions or effects as in hybrid planning; they are merely names. For this formalization we showed that HTN planning is still undecidable (Geier and Bercher 2011, Thm. 1), despite its severe simplifications of other hierarchical planning approaches (Erol, Hendler, and Nau 1996; Biundo and Schattenberg 2001). Further, we proved that allowing the insertion of tasks lowers to complexity to at most EXPSPACE (Geier and Bercher 2011, Cor. 1), making the plan-existence problem decidable. While PSPACE-hardness follows from classical planning, tight bounds are yet unknown. For our formalization of HTN planning, however, tight complexity bounds are already known (Alford, Bercher, and Aha 2015).

3.2 Hybrid Planning

The plan-existence results proved for HTN and TIHTN planning can likely be transferred to hybrid planning, as the theoretical impact of causal links seems to be very limited for the plan-existence problem.

3.3 PO and POCL planning

Both PO and POCL planning problems can be proved to be PSPACE-complete – so, they are exactly as hard as classical planning problems. Thus, the problem of transforming an initial state into a goal state does not become harder by additionally specifying an initial plan that needs to be refined into such a solution. Since classical planning problems and PO and POCL planning problems are all PSPACE-complete, we can conclude that there is an encoding of the latter problems into classical planning problems without a space increase more than polynomial. This idea is exploited by a technique that makes state-based planning heuristics applicable to the PO and POCL setting (Bercher, Geier, and Biundo 2013). That technique is presented in the heuristics section.

To obtain a heuristic that is well-informed on the one hand, but tractable to calculate on the other, one needs to investigate how much a given plan needs to be relaxed in order to obtain a tractable problem class. Otherwise, a heuristic could ignore information that does not contribute to the hardness of the heuristic calculation, but could improve its estimates. In POCL planning, there are basically two well-informed heuristics available in the literature: the *Relax heuristic* (Nguyen and Kambhampati 2001) and *the additive heuristic for POCL planning* (Younes and Simmons 2003). Both heuristics rely on delete-relaxation – that is, they ignore negative effects of all actions within the domain, but also those of the current plan for which a heuristic is to be calculated. This is contrast to state-based planning, where only the actions in the domain become delete-relaxed (since the current search node is a state rather than a plan). Hence, the question arises how hard it is to refine a given POCL plan into a solution if that plan is not altered, but only the actions of the domain are. It turns out that delete-relaxing only the actions in the domain is not sufficient to obtain a tractable problem class – it is still NP-complete (Bercher et al. 2013). The source of the hardness lies in the interplay of the partial order on the actions already present in the plan and their negative effects. The result gives theoretical insights required to design heuristics for POCL planning, but we were also able to exploit it by means of an implemented heuristic that is presented in the heuristics section.

4 Search

Planning is done by searching the space of partial plans. To that end, the initial partial plan P_{init} gets refined until it satisfies all solution criteria. The respective generic hybrid planning algorithm is depicted in Alg. 1 (Bercher, Keen, and Biundo 2014). The corresponding planning system PANDA (Planning and Acting in a Network Decomposition Architecture) is based on earlier work (Schattenberg 2009).

Algorithm 1: Hierarchical Refinement Planning

```
1 F ← {P_init}
2 while F ≠ ∅ do
3 |   P ← planSel(F)
4 |   if Flaws(P) = ∅ then return P
5 |   f ← flawSel(Flaws(P))
6 |   F ← (F \ {P}) ∪ { applyMod(m, P)
  |                     | m ∈ Mods(f, P) }
7 return fail
```

The algorithm maintains a set of candidate plans that have been created via refining the initial partial plan P_{init} and that have not yet been chosen for refinement. We refer to that set as fringe F. Initially, it contains only the initial partial plan P_{init}. While this fringe is not empty and no solution has been generated, one of the partial plans in the fringe is chosen for refinement (line 3). The strategy that determines the selected partial plan constitutes the deployed search strategy. In case of an informed search, such as A^* or *greedy search*, heuristic functions are required that estimate the quality or goal distance of partial plans.

For the selected partial plan P, all its *flaws* $Flaws(P)$ are calculated. Flaws are syntactical representations of violations of solution criteria. Since only primitive plans are regarded executable (cf. solution criterion 2.(a)), any abstract task induces a so-called *abstract task flaw*. The flaw classes *open precondition flaws* and *causal threat flaws* (cf. solution criteria 2.(b) and 2.(c), respectively) are inherited from standard POCL planning. Solution criterion 1 is always fulfilled when using the PANDA algorithm, since it mimics the allowed refinement options.

Plans with no flaws are solutions and returned (line 4). In case there are flaws, one of them is picked to be resolved (line 5). For a single flaw f, there may be several possibilities how to resolve it. For instance, for an open precondition, there might be several actions that can serve as a producer

for the respective causal link. Each possibility – called a modification to the partial plan P – is calculated and applied to P leading to a set of successor plans. Then, P is removed from the fringe and its successors are inserted instead (cf. line 6).

Now, the cycle starts over. In case the fringe becomes empty, there is no solution to the problem; hence, *fail* is returned (line 7).

Note that the algorithm is capable to solve all problem classes mentioned in Section 2. Depending on the problem class, termination might not be guaranteed. For instance, since HTN planning is undecidable, termination may not be guaranteed in case no solution exists.

5 Heuristics

The first proposed heuristics are designed for POCL planning problems (that is, for plans with a partial order on primitive tasks that may contain causal links), the last heuristic for hybrid planning can additionally cope with abstract tasks within a plan.

5.1 Sample-FF heuristic for POCL problems

The membership part of the NP-completeness proof for partially delete-relaxed POCL planning problems directly constitutes a new heuristic that we call Sample-FF (Bercher et al. 2013). That proof reveals that given a total order of the actions within a plan, that plan can be refined into a solution using only delete-relaxed actions within polynomial time. Finding such a total order that actually admits a solution is the source of the NP-hardness, however. We simulated that NP-hard guessing part via sampling a fixed number of linearizations. The solution extraction for a given linearization of the actions is done using the FF heuristic (Hoffmann and Nebel 2001). Hence, the proposed heuristic, Sample-FF, first samples n total orders of the plan's actions, then solves the resulting problem using the FF heuristic, and uses the number of actions of the solution as heuristic estimate. In case several linearizations lead to a solution, the minimal cost is chosen. In case all n linearizations turned out to be unsolvable, we used the number of flaws as heuristic estimates in order to prevent being blind. If the number of existing linearizations is smaller than n, then the non-existence of a solution to these $m \leq n$ linearizations proves that the plan cannot be refined to a solution and can hence be discarded.

We have done an empirical evaluation using PANDA with A^* search and the planning problems taken from the IPC 1 to IPC 5. It revealed the following results: The additive heuristic for POCL planning solved 292 out of 446 problems. The Relax heuristic was able to solve 194 problems. For the Sample-FF heuristic, we evaluated 12 different versions that distinguish from each other by the number of used samples (1, 3, 10, or 30) and the way in which present causal links are handled. The best-performing variant solved 187 problems – slightly less than the Relax heuristic. However, for one of the unsolvable planning problems, the Sample-FF heuristic was the only one that was able to prove its unsolvability, while the other heuristics all incurred timeouts.

5.2 State-based heuristics for POCL problems

Since there are basically only two heuristics for POCL planning, but a variety of well-informed heuristics for state-based planning (Helmert and Domshlak 2009), we studied whether these state-based heuristics can *directly* be made applicable to the POCL setting. The idea is to encode a given plan P as a new *classical* planning problem, i.e., a problem without initial plan, but an initial state s (Bercher and Biundo 2013; Bercher, Geier, and Biundo 2013). The proposed encoding can be done in polynomial time and has the property that any solution that can be obtained from P can also be obtained from s and vice versa. Thus, rather than developing a new heuristic h estimating the goal distance for plans, we can use the encoding transforming P into a state s and then defining $h(P) := h'(s)$ for an already existing heuristic h' that is defined for states.

We used PANDA with A^* search for the empirical evaluation. Again, we used the planning benchmarks from the IPC 1 to IPC 5. We have evaluated the proposed technique using several state-based heuristics from the literature. We compared them with the Relax heuristic and the additive heuristic for POCL planning. To evaluate the state-based planning heuristics we chose to use existing implementations from the Fast Downward planning system (Helmert 2006). To make them applicable within PANDA, we automatically created PDDL domain and problem files for each search node encoding the corresponding partial plan. These files were passed to a modified version of Fast Downward that exits after calculating the heuristic value for the initial state. Due to the overhead of starting Fast Downward in each search node, we did not evaluate run times. Instead, we focused our evaluation on the number of solved problem instances given a maximum number of created search nodes. The evaluation reveals that the encoding using the LM-Cut heuristic (Helmert and Domshlak 2009) acts more informed than both the Relax and the additive heuristic, as it always solved more problems within the same search space bounds. Since the LM-cut heuristic is admissible (and the Relax and the additive heuristic are not), our transformation together with the LM-cut heuristic constitutes the first admissible heuristic for POCL planning.

5.3 Task Decomposition Trees as Basis for Heuristics in Hybrid Planning

The previous heuristics were capable to estimate the goal distance for primitive plans, i.e., for a plan in which all tasks are primitive. In hybrid planning, such plans may additionally contain *abstract* tasks. Any well-informed heuristic judging the goal distance for such plans thus has to consider how abstract tasks may be decomposed. For that purpose, we use a Task Decomposition Graph (TDG) (Elkawkagy, Schattenberg, and Biundo 2010; Elkawkagy et al. 2012; Bercher, Keen, and Biundo 2014) that represents the task hierarchy: A TDG contains all primitive and abstract tasks of the planning domain. Any abstract task in the graph is connected to any of its decomposition methods. Finally, any method is connected with all tasks that are contained in the plan referenced by that method. More formally, a TDG is

a 4-tuple $\langle V_T, V_M, E_{T\to M}, E_{M\to T}\rangle$ consisting of the set of task vertices V_T, method vertices V_M, edges $E_{T\to M}$ connecting task to method vertices, and edges $E_{M\to T}$ connecting method vertices with its sub tasks V_T. Fig. 1 depicts an example TDG.

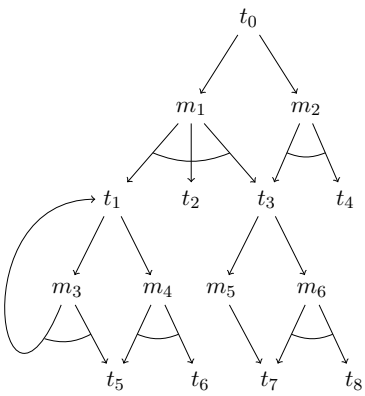

Figure 1: Example TDG (Bercher, Keen, and Biundo 2014) depicted as AND/OR graph. The symbols t_0, t_1, t_3 and $t_2, t_4, \ldots, t_8$ represent abstract and primitive task vertices, respectively. The symbols m_1 through m_6 depict method vertices for the abstract tasks t_0, t_1, and t_3.

These TDGs can be used to calculate informed heuristics that incorporate the task hierarchy. The heuristic judges the number of remaining modifications to turn the given plan into a solution, but it can be easily modified to judge the *cost* of the tasks that still need to be inserted via decomposition. Since the resulting heuristic admissible, we call it *minimal modification effort (MME) heuristic*[3] (Bercher, Keen, and Biundo 2014).

Definition 1 (MME Heuristic)**.**
Let $\langle V_T, V_M, E_{T\to M}, E_{M\to T}\rangle$ be a TDG.

$$h_T(v_t) := \begin{cases} |\text{prec}^+(v_t)| + |\text{prec}^-(v_t)| & \text{if } v_t \text{ is primitive} \\ 1 + \min\limits_{(v_t, v_m)\in E_{T\to M}} h_M(v_m) & \text{else} \end{cases}$$

For a method vertex $v_m = \langle PS, \prec, CL\rangle$, we set:

$$h_M(v_m) := \sum_{(v_m, v_t)\in E_{M\to T}} h_T(v_t) - |CL|$$

Then, for a partial plan $P = \langle PS', \prec', CL'\rangle$ that was generated during search, we define:

$$h_{MME}(P) := h_M(P)$$

The estimated number of modifications for making a primitive task t executable is the number of its preconditions, since for every precondition a causal link has to be inserted and each causal link insertion is done using a single modification. In the case of an abstract task (the "else" case of h_T), we need to apply exactly one modification for its decomposition plus the estimate for the sub plan that is introduced via decomposing that task. To obtain admissible estimates, we minimize over all possible decompositions.

The effort to refine a plan into a solution (heuristics h_M and h_{MME}, respectively) is given by the sum of all the heuristics for the task that are contained within that plan. Since such plans may already contain causal links, we subtract their number in order to be admissible.

In the empirical evaluation we used PANDA with greedy search and the proposed MME heuristic. We compared it with other heuristics for hybrid planning, such as the number of flaws. As a further baseline, we employed uninformed search strategies such as breadth first and depth first search. We also simulated the search behavior of the hierarchical planning systems SHOP2 (Nau et al. 2003) and UMCP (Erol, Hendler, and Nau 1994). We evaluated how many problems were solved given a certain size of the explored search space. The evaluation reveals that MME is the best-performing heuristic among all evaluated configurations.

6 Summary

The thesis introduces hybrid planning as a means to provide intelligent user assistance. The integration of various planning techniques, such as plan generation, repair, and explanation is demonstrated in an intelligent assistance system that helps a user with his task of setting up a complex home theater. Several sub problem classes of hybrid planning are investigated that arise under several restrictions, such as (dis)allowing task insertion or (dis)allowing causal links in the domain model. The complexity of the respective plan-existence problem is investigated including syntactical restrictions, such as problems with no negative preconditions or effects (delete-relaxation). The thesis introduces a hybrid planning algorithm, called PANDA, that is capable of solving problems of all introduced problem classes. Finally, for each of these classes, heuristics are introduced and empirically evaluated. Some of these heuristics are the first admissible heuristics for the respective problem class.

Acknowledgment

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.; Bercher, P.; and Aha, D. 2015. Tight bounds for HTN planning. In *Proc. of the 25th Intl. Conf. on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Bercher, P., and Biundo, S. 2013. Encoding partial plans for heuristic search. In *Proc. of the 4th Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 11–15.

Bercher, P.; Geier, T.; Richter, F.; and Biundo, S. 2013. On delete relaxation in partial-order causal-link planning. In *Proc. of the 2013 IEEE 25th Intl. Conf. on Tools with Artificial Intelligence (ICTAI)*, 674–681. IEEE Computer Society.

[3]The heuristic presented here is a slight modification of the original MME heuristic.

Bercher, P.; Biundo, S.; Geier, T.; Hoernle, T.; Nothdurft, F.; Richter, F.; and Schattenberg, B. 2014. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *Proc. of the 24th Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, 386–394. AAAI Press.

Bercher, P.; Richter, F.; Hörnle, T.; Geier, T.; Höller, D.; Behnke, G.; Nothdurft, F.; Honold, F.; Minker, W.; Weber, M.; and Biundo, S. 2015. A planning-based assistance system for setting up a home theater. In *Proc. of the 29th National Conf. on Artificial Intelligence (AAAI)*, 4264–4265. AAAI Press.

Bercher, P.; Geier, T.; and Biundo, S. 2013. Using state-based planning heuristics for partial-order causal-link planning. In *Advances in Artificial Intelligence, Proc. of the 36nd German Conf. on Artificial Intelligence (KI)*, 1–12. Springer.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid planning heuristics based on task decomposition graphs. In *Proc. of the 7th Annual Symposium on Combinatorial Search (SoCS)*, 35–43. AAAI Press.

Bidot, J.; Schattenberg, B.; and Biundo, S. 2008. Plan repair in hybrid planning. In *Advances in Artificial Intelligence, Proc. of the 31st German Conf. on Artificial Intelligence (KI)*, 169–176. Springer.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief (a preliminary report on combining state abstraction and HTN planning). In *Proc. of the 6th European Conf. on Planning (ECP)*, 157–168. AAAI Press.

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.

Elkawkagy, M.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2012. Improving hierarchical planning performance by the use of landmarks. In *Proc. of the 26th National Conf. on Artificial Intelligence (AAAI)*, 1763–1769. AAAI Press.

Elkawkagy, M.; Schattenberg, B.; and Biundo, S. 2010. Landmarks in hierarchical planning. In *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI)*, 229–234. IOS Press.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the 2nd Intl. Conf. on Artificial Intelligence Planning Systems (AIPS)*, 249–254. AAAI Press.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence* 18(1):69–93.

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *Proc. of the 22nd Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 1955–1961.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. of the 19th Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, 162–169. AAAI Press.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)* 26:191–246.

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

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language classification of hierarchical planning problems. In *Proc. of the 21st European Conf. on Artificial Intelligence (ECAI)*, volume 263, 447–452. IOS Press.

Honold, F.; Bercher, P.; Richter, F.; Nothdurft, F.; Geier, T.; Barth, R.; Hörnle, T.; Schüssel, F.; Reuter, S.; Rau, M.; Bertrand, G.; Seegebarth, B.; Kurzok, P.; Schattenberg, B.; Minker, W.; Weber, M.; and Biundo, S. 2014. Companion-technology: Towards user- and situation-adaptive functionality of technical systems. In *10th Intl. Conf. on Intelligent Environments (IE)*, 378–381. IEEE.

Kambhampati, S.; Mali, A.; and Srivastava, B. 1998. Hybrid planning for partially hierarchical domains. In *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI)*, 882–888. AAAI Press.

Kambhampati, S. 1995. Admissible pruning strategies based on plan minimality for plan-space planning. In *Proc. of the 14th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 1627–1633. Morgan Kaufmann.

Lin, N.; Kuter, U.; and Sirin, E. 2008. Web service composition with user preferences. In *ESWC'08: Proc. of the 5th European Semantic Web Conf.*, 629–643. Springer.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. of the 9th National Conf. on Artificial Intelligence (AAAI)*, 634–639. AAAI Press.

Nau, D. S.; Ilghami, T.-C. A. O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research (JAIR)* 20:379–404.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Wu, D.; Yaman, F.; Muñoz-Avila, H.; and Murdock, J. W. 2005. Applications of SHOP and SHOP2. *Intelligent Systems, IEEE* 20:34–41.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, volume 17, 459–466.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 103–114. Morgan Kaufmann.

Schattenberg, B. 2009. *Hybrid Planning & Scheduling*. Ph.D. Dissertation, University of Ulm, Germany.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In *Proc. of the 22nd Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, 225–233. AAAI Press.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research (JAIR)* 20:405–430.