On Guiding Search in HTN Planning with Classical Planning Heuristics*

Daniel Höller, **Pascal Bercher**, **Gregor Behnke** and **Susanne Biundo**
Institute of Artificial Intelligence, Ulm University, Germany
{daniel.hoeller, pascal.bercher, gregor.behnke, susanne.biundo}@uni-ulm.de

Abstract

Planning is the task of finding a sequence of actions that achieves the goal(s) of an agent. It is solved based on a model describing the environment and how to change it. There are several approaches to solve planning tasks, two of the most popular are *classical planning* and *hierarchical planning*. Solvers are often based on *heuristic search*, but especially regarding *domain-independent* heuristics, techniques in classical planning are more sophisticated. However, due to the different problem classes, it is difficult to use them in hierarchical planning. In this paper we describe how to use arbitrary classical heuristics in hierarchical planning and show that the resulting system outperforms the state of the art in hierarchical planning.

1 Introduction

Planning is the task of finding a course of action that achieves the goal(s) of an agent when it is executed. In its simplest form, the model underlying the task contains a description of the relevant parts of the environment the agent is planning for and a set of actions that describe how he/she can change it. The environment is usually described by using a (finite) set of propositional variables; the definition of an action contains a description of preconditions that need to be fulfilled to be able to apply it and a set of effects (changes to the system it causes). The objective is to find a sequence of actions transforming the system into a state where certain state features hold. This is often done using *heuristic search*.

Two of the most popular approaches to planning are *classical planning* (that is equal to the abstract description given above) and *hierarchical planning* [Bercher *et al.*, 2019]. In Hierarchical Task Network (HTN) planning, the most common hierarchical formalism, the model as given above is complemented by a hierarchy on the things to do, the *tasks*. The planning process is started with one or more so-called *abstract* tasks. Abstract tasks can not be executed directly (like actions), but need to be decomposed into other tasks until all remaining tasks are actions. Inserting actions apart from this decomposition hierarchy is (usually) not allowed. Interestingly, the objective in HTN planning is not to reach a state with certain properties, but to find a decomposition that can be executed. The process is very similar to the derivation of a word from a formal grammar, but with two differences: the symbols may be partially ordered, and the terminal symbols are actions with preconditions and effects. It has been shown that HTN planning models can describe much more complex behavior than possible in classical planning [Erol *et al.*, 1996; Höller *et al.*, 2014; Höller *et al.*, 2016].

When a domain is modeled, the designer may introduce knowledge about the domain itself as well as advice for the planner on how to find a plan. Having less advice decreases the modeling effort and allows a simple adaptation of planning systems to new domains, but planners need to rely (even more) on their search techniques to find a plan. Systems that are designed to find plans without any advice are called to be *domain-independent*. In HTN planning, such systems have been developed based on heuristic search (e.g. our system or Bercher *et al.*'s [2017]), or on translation techniques, e.g. to propositional logic [Behnke *et al.*, 2019b]. However, compared to classical planning, the solving techniques in HTN planning are less sophisticated, especially domain-independent heuristics to guide search. Therefore hierarchical systems often rely on advice encoded in the problem.

Using the more sophisticated techniques from classical planning in HTN planning seems appealing, but it is not straightforward due to several reasons:

- The hierarchy determines which actions are reachable, i.e. which actions *may be* in a plan.
- All tasks need to be decomposed and the resulting actions need to be integrated into the plan, i.e. the hierarchy also determines which actions *have to be* in a plan.
- The goal in classical planning is defined by a set of state features that shall be achieved. Therefore classical heuristics are designed to estimate the distance from a given state to a goal *state*. In HTN planning the "goal" is an *abstract task* to perform; no state-based goal is given.

In this paper we describe a generic way to use heuristics from classical planning to guide the search in HTN planning that overcomes the challenges given above.

*This paper was invited for the *Best Papers from Sister Conference Track*. It describes content of the paper *"A Generic Method to Guide HTN Progression Search with Classical Heuristics"* [Höller *et al.*, 2018] from the International Conference on Automated Planning and Scheduling (ICAPS).

2 Formal Framework

A classical planning problem is a tuple $P = (L, A, s_0, g, \delta)$, where L is a set of propositional environment variables. A state s of the system is defined by the subset of state variables that is fulfilled in this state, i.e. $s \subseteq L$. s_0 is the initial state that holds in the beginning of the planning process; $g \subseteq L$ is the goal definition that specifies which state features shall be fulfilled in a goal state. A is a set of action names; δ is a triple $(prec, add, del)$ of functions that define the preconditions and effects of the actions, each mapping actions to a set of state variables $f : A \to 2^L$. An action a is applicable when its preconditions are contained in the current state s, i.e. when $prec(a) \subseteq s$. When it is applicable, the state resulting from the application is defined by the function $\gamma : A \times 2^L \to 2^L$ with $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. The objective in classical planning is to find a sequence that transforms the initial state into one that includes the goal conditions g; more formally, a sequence $\langle a_0 a_1 \dots a_l \rangle$ of actions where a_i is applicable in s_i and s_i is defined as $\gamma(a_{i-1}, s_{i-1})$ for $i > 0$.

An HTN planning problem is defined as a tuple $P = (L, C, A, M, s_0, tn_I, \delta)$. L, A, s_0, and δ are defined as before. C is a set of abstract (also called compound) task names. During the planning process, the tasks to accomplish are maintained in so-called *task networks*. A task network combines a (multi-)set of task names with the definition of ordering relations between these tasks. Formally, it is defined as a triple $(T, \prec, \alpha)$, where T is a set of (unique) identifiers, $\alpha : T \to A \cup C$ is a function that maps these ids to task names, and $\prec \subseteq T \times T$ is a strict partial order on the ids.

As given in the introduction, abstract tasks are decomposed until only primitive tasks are left. The rules on how to decompose these tasks are given by the set of *decomposition methods* M. Each method $m \in M$ is defined as a pair (c, tn) of a task name $c \in C$ that may be decomposed by the method and a task network tn that specifies how the task is decomposed. When a task is decomposed, it is removed from the task network, the tasks in the method's task network are added and inherit the ordering relations from the decomposed task. Formally, a method (c, tn) decomposes a task network $tn_1 = (T_1, \prec_1, \alpha_1)$ into a task network $tn_2 = (T_2, \prec_2, \alpha_2)$ if there is a task $t \in T_1$ with $\alpha_1(t) = c$ and a task network $tn' = (T', \prec', \alpha')$ that is equal to tn but using ids not contained in the decomposed network (i.e. $T_1 \cap T' = \emptyset$). The task network tn_2 is defined as follows:

$$tn_2 = ((T_1 \setminus \{t\}) \cup T', \prec' \cup \prec_D, (\alpha_1 \setminus \{t \mapsto c\}) \cup \alpha')$$
$$\prec_D = \{(t_1, t_2) \mid (t_1, t) \in \prec_1, t_2 \in T'\} \cup$$
$$\{(t_1, t_2) \mid (t, t_2) \in \prec_1, t_1 \in T'\} \cup$$
$$\{(t_1, t_2) \mid (t_1, t_2) \in \prec_1, t_1 \neq t \wedge t_2 \neq t\}$$

The planning process starts with the initial task network tn_I and the objective is to find a task network tn such that the following conditions hold.

- tn can be reached by decomposing the initial task network tn_I.
- All task names in tn are primitive.
- There is a sequence of the tasks in tn that is in line with its ordering relations and applicable in s_0.

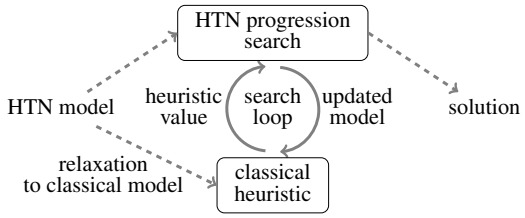

Figure 1: Schema of our overall approach.

3 Using Classical Heuristics in HTN Planning

As given in the introduction, a common way to solve planning problems is heuristic search. In HTN planning, there are two common search algorithms: *plan space search* and *progression search*. The former maintains a partial ordering between the tasks and selects the next modification to do based on the so-called *flaw selector*, a kind of heuristic. The latter searches in a forward manner, processing the tasks next that have no predecessors in the ordering. It commits to a total order of the tasks in the prefix of the generated plan. This makes the representation less compact (because there may be many totally ordered linearizations of a single partial ordering), but the system can apply the state transition during search and is thus informed about the current state. This may be beneficial to calculate heuristics that depend on the state.

We introduced an improved progression algorithm [Höller *et al.*, 2018] not given here due to the limited space. Instead, we want to focus on our method to guide the search with classical heuristics. In progression search, the distance to the next goal node is equal to the sum of decompositions and action applications necessary to transform a search node into a solution. To guide the search we need to estimate this number.

We want to do this using arbitrary and unchanged heuristics from classical planning. Our overall approach is illustrated in Figure 1. Before search, the HTN model is relaxed to a classical model. A *relaxation* is necessary because it has been shown that such a transformation is (in general) impossible [Erol *et al.*, 1996; Geier and Bercher, 2011]. During search, the HTN planning system updates the initial state and the goal definition of the classical model. Based on the updated model, a classical heuristic is calculated and the resulting heuristic value is used to guide the search.

The transformation needs to overcome the challenges discussed in the introduction: 1. the reachability is restricted by the hierarchy, 2. the hierarchy enforces actions to be integrated into the plan, and 3. there is no state-based goal definition that can be used to calculate the heuristic.

From a top-down view the hierarchy represents a tree with *AND* nodes and *OR* nodes: for each task, a single method needs to be selected (an OR node). A method introduces new tasks that all need to be decomposed/executed (an AND node) and thus imply new OR nodes. This is illustrated in a simple transport domain in Figure 2. First consider its right side: The root of the tree is a single *deliver* task: a package p shall be delivered to position d (this is an OR node). There are several methods to decompose it (AND nodes) using different vehicles v_i, the planner has to select one. In the figure, the method using vehicle v_2 is selected. It leads to several new

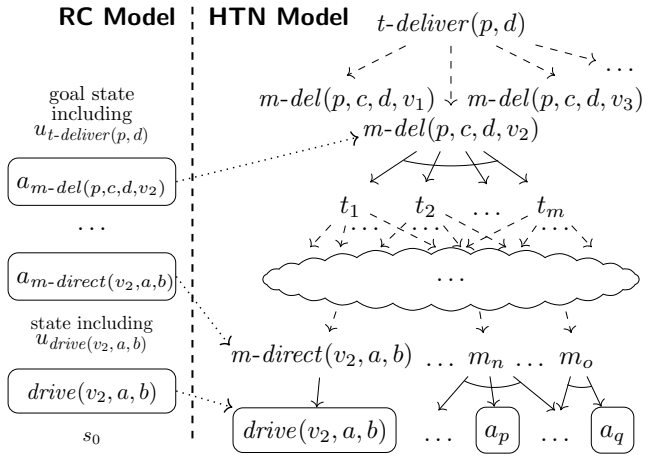

Figure 2: Correspondence of actions in our RC model to nodes in the AND/OR tree defined by the HTN. Task names start with t, methods with m, actions are boxed. The actions in the RC plan belonging to the tree nodes m_n, m_o, a_p, and a_q are omitted due to readability.

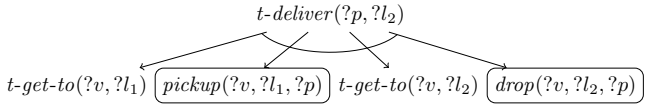

Figure 3: Example method. The abstract task at the top is decomposed into four subtasks. Primitive tasks are boxed.

task nodes (denoted t_1 to t_m). These form new OR nodes. The leaves of the tree are actions (indicated by the boxes) that form the solution. The planner has to select methods that lead to an executable sequence of actions (in line with the ordering definitions induced by the HTN model).

We want a classical planner to do this selection, or, more precisely, a classical *heuristic* to estimate the minimum number of methods and actions necessary to build the tree. Therefore we encode the problem in a bottom-up approach into a classical planning problem: we start with the state and action definition of the HTN problem. For every (abstract and primitive) task n in the problem we add a new state feature u_n indicating whether it is part of the tree. Each action a gets a new add effect u_a that achieves the corresponding state feature. For each method $m = (c, tn)$ of the HTN model a new action a_m is introduced. It has as many preconditions as the method has subtasks: for every subtask n, the newly introduced state feature u_n is part of the precondition set. As single effect, a_m adds the fact u_c (the fact corresponding to the decomposed task). The goal of the transformed problem is to achieve the state features corresponding to the task(s) in the current task network, i.e., the tree is built in a bottom-up manner. Since it thus somehow applies the decomposition methods in a backward manner, *composing* the tasks, we call our heuristics *Relaxed Composition* (RC) heuristics.

Consider the method $m\text{-}del(?p, ?l_1, ?l_2, ?v)$ given in Figure 3 that decomposes the abstract task $t\text{-}deliver(?p, ?l_2)$ that defines how to deliver a package $?p$ to a location $?l_2$. To do so, a vehicle $?v$ needs to get to the current location of the package, pick it up, get to its final location, and drop it. The method is translated into an action $a_{m\text{-}del(?p,?l_1,?l_2,?v)}$ with the precondition set containing the four state features $\{u_{t\text{-}get\text{-}to(?v,?l_1)}, u_{pickup(?v,?l_1,?p)}, u_{t\text{-}get\text{-}to(?v,?l_2)}, u_{drop(?v,?l_2,?p)}\}$ and a single add effect $u_{t\text{-}deliver(?p,?l_2)}$.

A classical *planner* applied to that domain builds a plan representing the tree discussed above. Such a plan is shown on the left side of Figure 2. It starts at the bottom of the figure with the initial state s_0. The actions in the model correspond to 1. actions or 2. methods of the HTN problem. The former are those that also appear in solutions of the HTN problem. The classical system needs to select an executable sequence. The latter represent the decisions at the OR nodes. When we apply a classical *heuristic* (instead of a planner), we can estimate the size of the tree.

This solves challenges 2 and 3: we have a meaningful state-based goal; to reach it, the classical system needs to integrate the actions enforced by the hierarchy into the plan. However, there are several relaxations made:

- The ordering relations given in the HTN are ignored.
- Every task needs to be done only once (regardless how often it is enforced by the HTN).
- To make enforced actions applicable, other actions may be inserted.

The third point is harmful to challenge 1: we should restrict the actions in a plan to those reachable via the hierarchy. Though we need to relax the problem to make it tractable, we can fix this issue to a certain extent. When there are two methods for a certain task, one decomposing it into the action a, the other one into b, a solution can only contain one action out of a and b. We can not encode this easily into the problem. However, we can encode which actions are reachable from the current task network at all and exclude these that are not. The heuristic may, however, still insert actions that can not appear *together* due to the hierarchy, e.g. because they are part of different methods for the same task (as given above). The set of reachable tasks can be preprocessed before search.

To restrict the set of all actions in the RC model to *reachable* actions, we could adapt the heuristic function(s) so check it. However, instead we added an additional precondition to every action that encodes whether it is reachable. Reachability is then set in the initial state and excludes those actions from a solution that can not be reached via the hierarchy. That way we can use unchanged heuristics.

Formally, our transformation is defined as follows:

Definition 1 (Relaxed Composition Model)**.** *Given an HTN planning problem $P = (L, C, A, M, s_0, tn_I, (prec, add, del))$ with $tn_I = (T_I, \prec_I, \alpha_I)$, we define our RC model as the following classical planning problem P':*

$$
\begin{aligned}
P' &= \big(L', A', s_0', g', (prec', add', del')\big) \\
L' &= L \cup L_u \cup L_d \\
&\qquad L_u = \{u_n \mid n \in A \cup C\}, L \cap L_u = \emptyset \\
&\qquad L_d = \{d_n \mid n \in A\}, (L \cup L_u) \cap L_d = \emptyset \\
A' &= A \cup A_M, A_M = \{a_m \mid m \in M\}, A \cap A_M = \emptyset \\
s_0' &= s_0 \cup \{d_n \mid \exists tn' : tn_I \rightarrow^* tn' \\
&\qquad \textit{with } tn' = (T', \prec', \alpha'), t \in T', \alpha'(t) = n\}
\end{aligned}
$$

$$g' = \{u_n \mid t \in T_I, \alpha_I(t) = n\}$$

The functions $prec'$, add', and del' are defined as follows: For actions $a \in A$:

$$prec'(a) = prec(a) \cup \{d_a\}$$
$$add'(a) = add(a) \cup \{u_a\}$$
$$del'(a) = del(a)$$

For actions $a_m \in A_M$ with $m = (c, (T, \prec, \alpha))$:

$$prec'(a_m) = \{u_n \mid t \in T, \alpha(t) = n\}$$
$$add'(a_m) = \{u_c\}$$
$$del'(a_m) = \emptyset$$

An interesting theoretical property of our encoding is that for each HTN solution, there is a classical plan in the transformation that has costs equal to the number of nodes in the decomposition tree of the HTN solution. This especially holds for the optimal plan. This makes it possible to create *safe*, *goal-aware*, and *admissible* HTN heuristics, i.e. those that never return a heuristic value of ∞ if there still is a solution reachable (enabling pruning without becoming incomplete), that return 0 for goal nodes, and that never overestimate goal distance, respectively. The later can be used to find cost-optimal plans (for details see Behnke *et al.* [2019c], Sec. 4).

During search, 1. the current state, 2. the set of reachable tasks, and 3. the tasks in the current search node change. These HTN elements are represented in the initial state and the goal definition of the encoding. This enables an efficient update of the model without a full recreation.

In principle our encoding can be combined with any heuristic from classical planning. However, the used heuristic should allow for a "cheap" change of the goal definition (not all do). We tested our approach with the Add [Bonet and Geffner, 2001], FF [Hoffmann and Nebel, 2001], and LM-Cut [Helmert and Domshlak, 2009] heuristic.

4 Discussion

We first want to discuss related work. Alford *et al.* [2009; 2016] introduced an approach to translate HTN planning problems into classical planning problems to solve them with classical planners. To make this possible, they use a bound to restrict the size of task networks in the translated problem. To solve arbitrary HTN planning problems, this bound must be increased while no solution is found (and in general there is no technique to determine when this can be stopped without potentially getting an incomplete search). In principle, their translation could also be integrated as heuristic model into the overall process depicted in Figure 1. However, when increasing the bound, the model grows very fast (while ours is linear in the size of the input model), and when the bound is chosen too low, the search may become incomplete because it marks search nodes as dead ends that are none.

FAPE [Bit-Monnot *et al.*, 2016] uses blind search and a computation-intensive pruning technique. For pruning, it transforms the hierarchical model – which is essentially an acyclic HTN – into a temporal planning problem. Abstract tasks are represented using temporal actions, following an idea similar to our encoding: preconditions enforcing subtasks, which are temporally arranged according to their ordering relation, and an effect marking the abstract task as fulfilled at the end. Using a temporal reachability analysis, it determines whether a delete-relaxed plan exists, while still incorporating some of the problem's ordering information.

	#instances	RC^{LM-Cut}	RC^{Add}	RC^{FF}	TDG_m	TDG_c	2ADL Jasper	DFS, Alg. 2	DFS, Alg. 1	JSHOP2
UM-TRANSLOG	22	**22**	**22**	**22**	**22**	**22**	19	**22**	**22**	**22**
SATELLITE	25	**24**	23	**24**	**24**	21	23	23	22	22
WOODWORKING	11	**10**	9	**10**	8	**10**	5	6	7	8
SMARTPHONE	7	5	5	5	**6**	5	**6**	4	4	4
PCP	17	9	**12**	11	8	8	3	1	1	0
ENTERTAINMENT	12	**11**	**11**	**11**	9	**11**	5	6	6	5
ROVER	20	4	4	4	4	**6**	5	3	2	3
TRANSPORT	30	11	5	10	1	1	**19**	0	0	0
total	144	96	91	**97**	82	84	85	65	64	64

Figure 4: Coverage results for search-based HTN planning systems.

The maybe most closely related heuristics from the literature are those by Bercher *et al.* [2017] (that are also included in the evaluation). They are based on the so-called *Task Decomposition Graph* (TDG), a finite graph representing the task hierarchy. Heuristic calculation is started by assigning each action its costs as heuristic value. The heuristic value of each method is then set to the sum of the costs of its subtasks, and the costs of abstract tasks are set to the minimum of all applicable methods. The costs of abstract tasks and methods need to be updated until they converged (however, this can be done efficiently as described in the paper). The value has been further improved by including a state-based reachability analysis. The resulting heuristic includes the minimum effort induced by the hierarchy to reach a plan executable using task insertion. There are two main differences between the two approaches: First, our encoding as classical planning problem enables the combination with any classical heuristic (defining a family of heuristics instead of a single one). Second, our approach includes the costs of added actions into the heuristic value (what is not done by the TDG heuristics).

Figure 4 shows coverage results of several search-based HTN planning systems. We included our system using the Add, FF, and LM-Cut heuristic; plan space search with two TDG-based heuristics [Bercher *et al.*, 2017]; and the translation approach as described by Alford *et al.* [2016] in combination with the classical planner JASPER [Xie *et al.*, 2014]. For our system and the one of Bercher *et al.* the results are those for Greedy A* search with a weight of 2 (that has the highest coverage). The DFS data compares the uninformed *Depth First Search* of the original progression algorithm and our improved version (both implemented in our system). The JSHOP2 [Nau *et al.*, 2003] planning system also relies on DFS. The data given here is its combination with our grounding and preprocessing [Behnke *et al.*, 2019a] that increases its performance. Our system has the highest coverage. Our novel progression algorithm (not described in this paper) increases the coverage by 1 instance in DFS.

5 Conclusion

Domain-independent planning systems do not depend on advice modeled into a planning model. This makes models more compact and decreases the effort to create them. However, due to the lack of advice, such systems must rely on their solving techniques to find a solution. Though there has been progress in that direction in hierarchical planning, the used solving techniques – especially search heuristics – are less evolved than in classical planning. Due to the different problem classes, the adaptation of classical techniques to the setting of HTN planning is not straightforward. In this paper, we introduced an approach to use arbitrary classical heuristics to guide search in HTN planning. We have shown that it can be used to create HTN heuristics with interesting theoretical properties and that – when combined with our new progression search algorithm – it outperforms the state of the art in search-based HTN planning.

Acknowledgements

This work was partly funded by the technology transfer project "Do it yourself, but not alone: Companion-Technology for DIY support" of the SFB/TRR 62 funded by the German Research Foundation (DFG). Industrial project partner is the Corporate Research Sector of the Robert Bosch GmbH.

References

[Alford *et al.*, 2009] Ronald Alford, Ugur Kuter, and Dana S. Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1629–1634. AAAI Press, 2009.

[Alford *et al.*, 2016] Ron Alford, Gregor Behnke, Daniel Höller, Pascal Bercher, Susanne Biundo, and David W. Aha. Bound to plan: Exploiting classical heuristics via automatic translations of tail-recursive HTN problems. In *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 20–28. AAAI Press, 2016.

[Behnke *et al.*, 2019a] Gregor Behnke, Daniel Höller, Pascal Bercher, and Susanne Biundo. More succinct grounding of HTN planning problems – Preliminary results. In *Proc. of the 2nd ICAPS Workshop on Hierarchical Planning*, pages 40–48, 2019.

[Behnke *et al.*, 2019b] Gregor Behnke, Daniel Höller, and Susanne Biundo. Bringing order to chaos – A compact representation of partial order in SAT-based HTN planning. In *Proc. of the 33rd AAAI Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 2019.

[Behnke *et al.*, 2019c] Gregor Behnke, Daniel Höller, and Susanne Biundo. Finding optimal solutions in HTN planning – A SAT-based approach. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI, 2019.

[Bercher *et al.*, 2017] Pascal Bercher, Gregor Behnke, Daniel Höller, and Susanne Biundo. An admissible HTN planning heuristic. In *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 480–488. IJCAI, 2017.

[Bercher *et al.*, 2019] Pascal Bercher, Ron Alford, and Daniel Höller. A survey on hierarchical planning – One abstract idea, many concrete realizations. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI, 2019.

[Bit-Monnot *et al.*, 2016] Arthur Bit-Monnot, David E. Smith, and Minh Do. Delete-free reachability analysis for temporal and hierarchical planning. In *Proc. of the 22nd European Conf. on Artificial Intelligence (ECAI)*, pages 1698–1699. IOS Press, 2016.

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

[Erol *et al.*, 1996] Kutluhan Erol, James A. Hendler, and Dana S. Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93, 1996.

[Geier and Bercher, 2011] Thomas Geier and Pascal Bercher. On the decidability of HTN planning with task insertion. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1955–1961. AAAI Press, 2011.

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

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

[Höller *et al.*, 2014] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo. Language classification of hierarchical planning problems. In *Proc. of the 21st European Conf. on Artificial Intelligence (ECAI)*, pages 447–452. IOS Press, 2014.

[Höller *et al.*, 2016] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo. Assessing the expressivity of planning formalisms through the comparison to formal languages. In *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 158–165. AAAI Press, 2016.

[Höller *et al.*, 2018] Daniel Höller, Pascal Bercher, Gregor Behnke, and Susanne Biundo. A generic method to guide HTN progression search with classical heuristics. In *Proc. of the 28th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 114–122. AAAI Press, 2018.

[Nau *et al.*, 2003] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[Xie *et al.*, 2014] Fan Xie, Martin Müller, and Robert Holte. Jasper: The art of exploration in greedy best first search. In *Proc. of the 8th Int. Planning Competition*, pages 39–42, 2014.