Landmark Generation in HTN Planning

Daniel Höller,[1] **Pascal Bercher**[2]
[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2] The Australian National University, College of Engineering and Computer Science, Canberra, Australia
hoeller@cs.uni-saarland.de, pascal.bercher@anu.edu.au

Abstract

Landmarks (LMs) are state features that need to be made true or tasks that need to be contained in every solution of a planning problem. They are a valuable source of information in planning and can be exploited in various ways. LMs have been used both in classical and hierarchical planning, but while there is much work in classical planning, the techniques in hierarchical planning are less evolved. We introduce a novel LM generation method for Hierarchical Task Network (HTN) planning and show that it is sound and *in*complete. We show that every complete approach is as hard as the **co**-class of the underlying HTN problem, i.e. **coNP**-hard for our setting (while our approach is in **P**). On a widely used benchmark set, our approach finds more than twice the number of landmarks than the approach from the literature. Though our focus is on LM *generation*, we show that the newly discovered landmarks bear information beneficial for solvers.

Introduction

Planning is the task of finding a sequence of actions that change the environment to fulfill a certain objective. Two widely used approaches to planning are *classical planning* and *Hierarchical Task Network (HTN) planning*. In classical planning, the environment is described using a set of (propositional) state features that are modified by *actions*, which define valid state transitions. The objective is to find a sequence of actions transforming the initial state of the system into one in which certain goal features hold.

In HTN planning there are two kinds of tasks: actions like in classical planning (also *primitive tasks*) and *abstract tasks*, which are not applicable directly, but are decomposed into other (primitive or abstract) tasks by using *decomposition methods*. The objective in HTN planning is not to fulfill a state-based goal condition, but to find an applicable decomposition of a given abstract task. Since there is (usually) more than one method for an abstract task, the hierarchy implies a second combinatorial problem because a planner has to choose the "right" one for a certain task. This makes HTN planning more expressive (Erol, Hendler, and Nau 1994, 1996; Höller et al. 2014, 2016). The process can be seen as AND/OR tree (Kambhampati, Mali, and Srivastava 1998; Ghallab, Nau, and Traverso 2004). Starting with the initial task, a planner chooses a single method (i.e. abstract tasks form OR nodes) and has to include all subtasks into the plan (i.e. methods form AND nodes), and so on.

A concept that has been successful especially in classical planning is that of *landmarks* (LMs). LMs are state features (or actions) that are made true (contained) in every solution. It was first used for problem decomposition (Porteous, Sebastia, and Hoffmann 2001; Hoffmann, Porteous, and Sebastia 2004) and later for creating non-admissible (see e.g. Zhu and Givan (2003), Richter, Helmert, and Westphal (2008), and Richter and Westphal (2010)) and admissible heuristics (see e.g. Karpas and Domshlak (2009) or Helmert and Domshlak (2009)) for heuristic search. LMs have also been introduced in hierarchical planning. First in form of *task* LMs in hybrid planning (Elkawkagy, Schattenberg, and Biundo 2010) (an extension of HTN planning), later in form of *fact* LMs in HGN planning, a formalism where the hierarchy is defined on *goals*, not on tasks. The former can directly be applied to HTN planning and will be the baseline for our approach. While the latter can apply LM generation techniques from classical planning directly (Shivashankar et al. 2013, 2016a,b; Shivashankar, Alford, and Aha 2017), the presented techniques are not applicable to HTN planning.

Work on LMs can be divided into two orthogonal categories (Keyder, Richter, and Helmert 2010): *LM utilization* showing how to exploit LM information, and *LM generation*, showing how to find LMs. We focus on the latter.

Based on techniques from classical planning, we introduce a novel approach for LM generation in HTN planning that elegantly combines the generation of fact, action, and method LMs. It dominates the existing work (finding at least the same LMs). Our approach is sound and *in*complete. We show that every (sound and) complete approach is as hard as the **co**-class of the underlying plan existence problem, i.e., in our setting – delete-effects and ordering-relations of the HTN model are ignored during generation – **coNP**-hard (while our approach is in **P**). On a widely used benchmark set we find more than twice the number of LMs as related work. We further show that the additional LMs bear valuable information for search guidance.

Formal Framework

A classical planning problem P is a tuple (F, A, s_0, g, δ). F is a set of propositional *state features* (or *facts*) used to describe the environment. A *state* $s \in 2^F$ is given by those state features that hold in it, all others are supposed to be false. $s_0 \in 2^F$ is called the *initial state* and $g \subseteq F$ is the goal condition. A is a set of *action names*. The functions $\delta = (prec, add, del)$ with $prec, add, del : A \to 2^F$ map them to state features defining their *precondition*, *add effects*, and *delete effects*, respectively. An action a is *applicable in a state* $s \in 2^F$ if and only if its precondition is contained in the current state, $prec(a) \subseteq s$. When a is applicable in s, the state s' resulting from its application is defined as $s' = (s \setminus del(a)) \cup add(a)$. A sequence of actions $(a_1, a_2, \dots, a_n)$ is applicable in a state s when action a_i with $1 \leq i \leq n$ is applicable in state s_{i-1}, where s_i for $1 \leq i \leq n$ results from applying the sequence up to action i. The state s_i is called the *state resulting from the application*. All states $s \supseteq g$ are called *goal states*. A *plan* (or *solution*) is an action sequence applicable in s_0 resulting in a goal state.

We extend classical problems to HTN problems based on the formalism of Geier and Bercher (2011). An *HTN planning problem* $\mathcal{P} = (F, A, C, M, s_0, tn_I, g, \delta)$ extends a classical problem by a decomposition hierarchy on the things to do, the *tasks*. Let A and C be the sets of primitive and abstract (also compound) tasks. We assume that their intersection is empty and call the set of all task names $N = A \cup C$.

Tasks are organized in *task networks* (TNs). A TN is a triple $tn = (T, \prec, \alpha)$, where T is a set of identifiers (ids), $\prec$ a strict partial order on the ids, and α a mapping from ids to actual tasks $\alpha : T \to N$. This definition allows having a certain task more than once in a TN.

Planning starts with a special TN defining the objective of the problem called *initial task network* tn_I.

The decomposition rules are called *(decomposition) methods* M. They map a task $c \in C$ to a TN, i.e. they are pairs (c, tn). When a method (c, tn) is applied to a task t with $\alpha(t) = c$ in a TN, the task is deleted from the network, the tasks defined in tn are added and inherit the ordering relations previously present for t. A TN $tn_1 = (T_1, \prec_1, \alpha_1)$ is decomposed into a TN $tn_2 = (T_2, \prec_2, \alpha_2)$ by a method (c, tn), if it contains a task $t \in T_1$ with $\alpha_1(t) = c$ and there is a TN $tn' = (T', \prec', \alpha')$ equal to tn but using different ids (i.e. $T_1 \cap T' = \emptyset$). tn_2 is defined as follows:

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

We write $tn_1 \xrightarrow[t,m]{} tn_2$ to denote that tn_1 can be transformed into tn_2 by decomposing a task t contained in tn_1 using the method m. We write $tn_1 \to^* tn_2$ to denote that a TN tn_1 can be decomposed into a TN tn_2 by using a (possibly empty) sequence of methods.

The elements s_0, g, and δ are defined as before. A *solution* is a TN $tn_S = (T_S, \prec_S, \alpha_S)$ such that

1. $tn_I \to^* tn_S$, i.e. tn_S can be created by decomposing tn_I,
2. all tasks are primitive, and
3. there is a sequence of all tasks in line with the ordering constraints $\prec_S$ applicable in s_0 resulting in a goal state.

An HTN planning system is not allowed to add tasks apart from the decomposition process. Since we defined the HTN problem as an extension of a classical problem, it contains a state-based goal definition. Usually, this definition is empty in HTN planning. Our LM generation works fine without it and most instances in the benchmark set do not contain one.

The landmarks of a given problem are defined as follows:

Definition 1 (Task Landmark)**.** *A task landmark is a task name $n \in N$ such that every sequence of decompositions leading to some solution tn_S contains a task network including the landmark. Thus, each decomposition sequence from tn_I to tn_S has the form $tn_I \to^* tn \to^* tn_S$, where $tn = (T, \prec, \alpha)$ with $t \in T$ and $\alpha(t) = n$.*

Definition 2 (Method Landmark)**.** *A method landmark is a method $m \in M$ such that every sequence of decompositions leading to some solution tn_S contains two task networks $tn_1 = (T_1, \prec_1, \alpha_1)$ and tn_2 such that there is a task $t \in T_1$ and it holds that $tn_I \to^* tn_1 \xrightarrow[t,m]{} tn_2 \to^* tn_S$.*

Definition 3 (Fact Landmark)**.** *A fact landmark is a fact $f \in F$ such that for every solution tn_S, every linearization applicable in s_0 in line with the ordering and resulting in a goal state there is an intermediate state s_i with $f \in s_i$.*

The definition for *task landmarks* (Def. 1) is essentially equivalent to that of Elkawkagy et al. (2012, Def. 3), but adapted to our formalism. The definition of *fact landmarks* is a canonical adaptation of fact landmarks from classical planning (Porteous, Sebastia, and Hoffmann 2001).

Landmark Generation in HTN Planning

The concept of landmarks in HTN-like planning has first been studied by Elkawkagy, Schattenberg, and Biundo (2010). They introduced a technique to identify tasks that are contained in all methods $(c, tn) \in M$ decomposing a certain task c by computing the intersection of their subtasks. These tasks are called *mandatory tasks*. However, the gain reported in the empirical evaluation was mainly caused by a presented model reduction, not by the mandatory tasks. In follow-up work Elkawkagy et al. (2012) tested how the computed landmark information can be exploited by introducing and evaluating landmark-based search strategies. These search strategies are tailored to the deployed search algorithm, which prioritizes different methods that belong to the same abstract task similar to SHOP (Nau et al. 2003; Goldman and Kuter 2019). However, whereas the SHOP systems rely on depth-first search and the order of the methods is specified in the model, Elkawkagy et al.'s system uses informed search strategies and computes the methods' order based on the mandatory tasks. The core idea is to prioritize methods with fewer tasks, whereas only *non*-mandatory tasks are considered as the latter have to be achieved anyway. So, their work did not yet define a landmark heuristic that can be exploited by standard heuristic HTN planners.

Bercher, Keen, and Biundo (2014) then introduced these ideas to standard heuristic search, using mandatory task landmarks for an admissible LM counting heuristic. To show in which way we extend their LM heuristic (Bercher, Keen, and Biundo 2014, Def. 1), we reproduce their definition, but simplified and adapted to our notation:

Definition 4 (Mandatory Task Landmarks)**.** *Let $\mathcal{P} = (F, A, C, M, s_0, tn_I, g, \delta)$ and $tn_I = (T_I, \prec_I, \alpha_I)$ be an HTN planning problem. For a primitive task $a \in A$, we define the set of mandatory tasks as $MT(a) = \emptyset$ and for an abstract task $c \in C$ it is defined as follows:*

$$MT(c) = \bigcap_{(c,(T,\prec,\alpha)) \in M} \quad \bigcup_{t \in T} \{\alpha(t)\}$$

A set of MT landmarks LM^{mt} for $\mathcal{P}$ can be computed by:

```
1 $LM^{mt} \leftarrow \bigcup_{t \in T_I} \{\alpha_I(t)\}$
2 while $LM^{mt}$ changes do
3     $LM^{mt} \leftarrow LM^{mt} \cup \bigcup_{n \in LM^{mt}} MT(n)$
```

The generation method collects tasks contained in all methods belonging to the same abstract task. Thus all tasks introduced at deeper levels cannot always be found. This, however, could be improved by lookahead techniques as done in early approaches in classical planning.

AND/OR Landmarks in HTN Planning

We now introduce HTN LM generation based on AND/OR graphs, adapting a technique from classical planning. Since the objective in classical planning is given in terms of a state-based goal, techniques like LM generation usually rely on it. When an HTN problem also includes one, techniques could be directly applied on it, but it is usually not present. This is a main problem when applying techniques from classical planning in HTN planning. One approach would be to extract task LMs for the HTN model and calculate the state-based LMs of the preconditions of primitive task LMs.

However, we introduce a more elegant approach that smoothly combines the generation of task, method, and fact LMs in HTN models based on the approach of Keyder, Richter, and Helmert (2010). Their technique extracts LMs from an AND/OR graph representation for delete-relaxed classical planning problems that was introduced by Mirkis and Domshlak (2007). We first introduce the approach of Keyder, Richter, and Helmert and then show that it can nicely be adapted to HTN planning.

Extracting Landmarks in Classical Planning Using AND/OR Graphs

We use the definition of AND/OR graphs by Keyder, Richter, and Helmert (2010, p. 2):

Definition 5 (AND/OR Graph)**.** *An AND/OR graph $G = (V_I, V_{and}, V_{or}, E)$ is a directed graph with vertices $V = V_I \cup V_{and} \cup V_{or}$ and edges E, where V_I, V_{and} and V_{or} are disjoint sets called initial nodes, AND nodes, and OR nodes, respectively. A subgraph $J = (V^J, E^J)$ of G is said to justify $V_G \subseteq V$ if and only if the following conditions holds:*

1. $V_G \subseteq V^J$
2. $\forall a \in V^J \cap V_{and} : \forall (v, a) \in E : v \in V^J \wedge (v, a) \in E^J$
3. $\forall o \in V^J \cap V_{or} : \exists (v, o) \in E : v \in V^J \wedge (v, o) \in E^J$
4. *J is acyclic*

Let $P = (F, A, s_0, g, \delta)$ with $\delta = (prec, add, del)$ be a delete-relaxed classical problem ($\forall a \in A : del(a) = \emptyset$). It can be represented as the following AND/OR graph (Mirkis and Domshlak 2007; Keyder, Richter, and Helmert 2010):

Definition 6 (AND/OR representation of delete-relaxed classical problems)**.** *Let $G = (V_I, V_{and}, V_{or}, E)$ with $V_I = s_0$, $V_{and} = A$, and $V_{or} = F \setminus s_0$. The set of edges is defined as $E = \{(a, f) \mid a \in A, f \in add(a)\} \cup \{(f, a) \mid a \in A, f \in prec(a)\}$.*

Landmarks in these graphs are characterized by the following definition (Keyder, Richter, and Helmert 2010):

Definition 7 (Landmarks in AND/OR graphs)**.**

$$LM(v) = \{v\} \text{ for } v \in V_I,$$

$$LM(v) = \{v\} \cup \bigcap_{u \in pred(v)} LM(u) \text{ for } v \in V_{or},$$

$$LM(v) = \{v\} \cup \bigcup_{u \in pred(v)} LM(u) \text{ for } v \in V_{and},$$

where $pred(v)$ is the set of predecessors of v in G, i.e. $pred(v) = \{u \mid (u, v) \in E\}$.

The set of landmarks for a problem is then defined as the set of landmarks for the nodes representing the goal definition g, i.e. $V_G = g$ and we are looking for $\bigcup_{n \in V_G} LM(n)$.

Keyder, Richter, and Helmert calculate the maximal set fulfilling these equations in **P** by initializing the LM sets of all nodes apart from V_I with the full LM set. Nodes in V_I are initialized with its own value. Then the sets are updated using the given rules until a fixpoint is reached.

Extracting Landmarks in HTN Planning Using AND/OR Graphs

From a high-level perspective, what is encoded in the AND/OR graph is that for every state feature in the goal condition, there must be (at least) one action that has it as an add effect. When an action is in the graph, its preconditions must be fulfilled, i.e. there must be at least one action (for each precondition) with this state feature as add effect, and so on (until the state features in the initial state are reached).

In HTN planning we find a similar structure: for each abstract task in the initial task network, there must be a method decomposing it. When a method is in the graph, all its subtasks must be in the graph, and so on (until all tasks are primitive). This similarity to AND/OR graphs has been pointed out before (Kambhampati, Mali, and Srivastava 1998; Ghallab, Nau, and Traverso 2004, Chapter 11).

However, we do not need to stop at this point: when we have reached an action, we know that its preconditions need to be fulfilled. So there must be actions that have those state features as add effects. To reflect this in landmark generation, we do not replace the definition of the AND/OR graph given before, but extend it in the following way.

Definition 8 (AND/OR representation of delete-relaxed HTN problems)**.** *Let $P = (F, A, C, M, s_0, tn_I, g, \delta)$ be an HTN planning problem. We define the corresponding AND/OR graph as follows:*

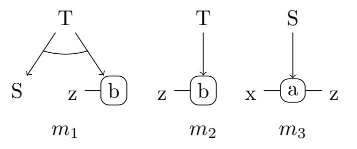


Figure 1: Simple HTN domain. S and T are abstract tasks, m_1 to m_3 methods, a and b actions, and x and z state features. T might be decomposed by m_1 into S and b, or by m_2 into b. S can be decomposed by m_3 into a.

$G = (V_I, V_{and}, V_{or}, E)$ with $V_I = s_0$, $V_{and} = A \cup M$[1] *and $V_{or} = F \setminus s_0 \cup C$. The set of edges is defined as*

$$\begin{aligned} E = &\{(a, f) \mid a \in A, f \in add(a)\} \cup \\ &\{(f, a) \mid a \in A, f \in prec(a)\} \cup \\ &\{(m, c) \mid m = (c, tn) \in M\} \cup \\ &\{(n, m) \mid m = (c, (T, \prec, \alpha)) \in M, t \in T, \alpha(t) = n\} \end{aligned}$$

We generate the LMs as described Keyder, Richter, and Helmert. Since the size of the graph is linear in the size of the model, the following proposition holds:

Proposition 1. *LM generation can be done in **P**.*

The overall set of LMs is then defined based on the hierarchy and (if present) state-based goal:

Definition 9 (HTN Landmarks)**.** *Let $tn_I = (T_I, \prec_I, \alpha_I)$ be the problem's initial task network. The overall set of HTN and/or landmarks LM^{ao} is defined as*

$$LM^{ao} = \bigcup_{v \in V_G} LM(v) \text{ with } V_G = \bigcup_{t \in T_I} \{\alpha_I(t)\} \cup \bigcup_{f \in g} \{f\}$$

Figure 1 illustrates the interplay of hierarchy and state during LM generation. The initial task network contains a single abstract task T that may be decomposed using the methods m_1 or m_2, both introducing an action b. The abstract task S can be decomposed into an action a using m_3. There are two state features x and z. The former is included in the initial state ($s_0 = \{x\}$) and precondition of a. The latter is the precondition of b. When we apply the *mandatory task* LM generation, we end up with the LM set $\{T, b\}$.

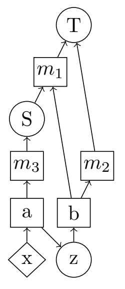
The AND/OR graph resulting from the problem is given in Figure 2. The resulting landmark sets are given at the right. Notably, though it is not even reachable when using m_2, we end up with a inside our landmark set, since it is the only action that fulfills the precondition of the landmark b.

Theoretical Properties

Before coming to our approach, we have a look at the theoretical properties of LM generation *in general*. Similar to classical planning, we see that deciding whether a task, method, or fact is a LM falls in the **co**-class of the plan existence problem of the respective problem class.

Theorem 1 (Complexity of LM generation)**.** *Let $\mathcal{P}$ be an HTN planning problem and **C** be the complexity class of the respective plan existence problem. Deciding whether a task, or a method, or a fact is a LM is **coC**-complete.*

[1]Wlog., we assume that $A \cap M = \emptyset$ and $F \cap C = \emptyset$.



$$\begin{aligned} LM(x) &= \{x\} \\ LM(z) &= \{a, x, z\} \\ LM(a) &= \{a, x\} \\ LM(b) &= \{a, b, x, z\} \\ LM(m_1) &= \{a, m_3, S, x\} \cup \{a, b, x, z\} \cup \{m_1\} \\ &= \{a, b, m_1, m_3, S, x, z\} \\ LM(m_2) &= \{a, b, m_2, x, z\} \\ LM(m_3) &= \{a, m_3, x\} \\ LM(S) &= \{a, m_3, S, x\} \\ LM(T) &= (\{a, b, m_1, m_3, S, x, z\} \cap \{a, b, m_2, x, z\}) \cup \{T\} \\ &= \{T, a, b, x, z\} \end{aligned}$$

Figure 2: AND/OR graph of our example given in Fig. 1. Circles are OR nodes, boxes are AND nodes, and the diamond-shaped node labeled x is the only initial node.

Proof. *Hardness.* Our proof is a straight-forward adaptation of the corresponding proof by Hoffmann, Porteous, and Sebastia (2004, Thm. 1) for classical planning landmarks.

We introduce a new initial abstract task c_I with two corresponding methods m_1 and m_2. m_1 decomposes c_I into the original tn_I. m_2 decomposes it into a new TN that solves the problem. It introduces an abstract task t, which can only be decomposed into an action t'. t' as an empty precondition, a new fact f as effect, and g (the original goal) as further effects. t' does not use negative effects[2]. Clearly, m_2, t (abstract), t' (primitive), and f are landmarks if and only if $\mathcal{P}$ is *un*solvable.

Membership. Similar to Hoffmann, Porteous, and Sebastia (2004, Thm. 1), we test whether the problem gets unsolvable if we ignore all parts of the model that "relate" to the LM(s) in question. I.e., in case of a method we just remove it. In case of a task (abstract or primitive), we remove all TNs and methods that contain them. And in case of a fact, we remove all actions (i.e., again removing all methods that introduce it) that add it. The this renders the task unsolvable, the task, method, or fact is a LM. □

Since deterministic complexity classes are closed under complement, our landmark decision in those classes can also be decided in **C**. This includes many known restrictions such as regular and tail-recursive problems, or (non-relaxed, ground) totally ordered problems (Erol, Hendler, and Nau 1994, 1996; Alford, Bercher, and Aha 2015a).

Using Thm. 1 we can deduce the computational hardness of determining whether a task, method, or fact is a landmark for many classes of HTN planning problems (Bercher, Alford, and Höller 2019; Erol, Hendler, and Nau 1996; Alford et al. 2014; Alford, Bercher, and Aha 2015a; Höller, Bercher, and Behnke 2020), e.g. the following ones:

Corollary 1. *Let $\mathcal{P}$ be an HTN planning problem. Deciding whether a task, method, or fact is a landmark of $\mathcal{P}$ is **undecidable**. If $\mathcal{P}$ is totally ordered, its complexity is **EXPTIME**-complete. If it is delete-relaxed it is **coNP**-complete.*

[2]We could put t and t' into the same TN, saving a "decomposition level" and method, but we wanted to keep the size of new TNs limited to 1 so we do not change properties of the problem we reduce from (like tasks per TN, ordering constraints, or the "position" of the task, which may all influence the computational complexity).

We can conclude that any *complete* LM generation technique for delete- and ordering-relaxed HTN problems cannot run in polynomial time unless **P**=**coNP**.

For totally ordered HTN planning, Olz, Biundo, and Bercher (2021) studied the complexity of proving whether a fact is a precondition/effect of *every* refinement of an abstract task, making such facts LMs of the task. They show the problem is in the same complexity class as the underlying plan existence problem. This is in line with our results, since it is a deterministic complexity class.

In the following, we use the so-called *Decomposition Tree* (Geier and Bercher 2011) in our proofs. This is a formal representation of a TN and its derivation from the initial task[3], i.e., it is a witness proving that the TN can be produced via decomposing the initial task.

Definition 10 (Decomposition Tree)**.** *Given an HTN planning problem, a Decomposition Tree (DT) is a tuple* $g = (V, E, \prec, \alpha, \beta)$*.* V *and* E *are the vertices and edges of a directed tree.* $\prec$ *is a strict partial ordering on* V*.* $\alpha : V \to N$ *maps the vertices to (primitive or abstract) tasks from the problem. Vertices that are labeled with abstract tasks are mapped to methods by* $\beta : V \to M$*.*

A DT is valid *if its root is labeled with the initial task of the problem and for every vertex* v *labeled with an abstract task* c*, the following conditions hold:*

1. *It is labeled with a method applicable to* c*, i.e.* $\beta(v) = (c, tn_m)$*.*
2. *The task network induced by the children of* v *in* g *differs from* tn_m *only in the task identifiers.*
3. *For all vertices* $v' \in V$*, the ordering with respect to the children of* v *is like defined for HTN planning, i.e. for each child* v'' *the following conditions hold:*
   - *(a) if* $(v, v') \in \prec$ *then* $(v'', v') \in \prec$
   - *(b) if* $(v', v) \in \prec$ *then* $(v', v'') \in \prec$
4. $\prec$ *does only contain ordering relations enforced by the conditions 2 and 3.*

Note that there is a valid DT for every solution of an HTN problem (Geier and Bercher 2011, Prop. 1), since it simply represents the decomposition that led to the respective TN.

Now we first show that for a given solution to the HTN problem, there is a corresponding justification in the AND/OR graph representing the model. This lemma is then used in the proof of the soundness of our LMs.

Lemma 1. *Let* $\mathcal{P}$ *be an HTN planning problem,* tn *a solution, and* dt *its decomposition tree. Then, there exists a justification for the initial task of the AND/OR graph (given in Definition 8) representing the* dt*.*

Proof. Consider the following observations:

1. Task Insertion – Assume we have a justification for an AND/OR graph representation of a classical problem. Assume we want to add additional actions. This results in more AND nodes, but as long as we support their preconditions by other action nodes or the initial state, we get another valid justification.
2. Eliminating Cycles – Geier and Bercher (2011, Sec. 4.1 and 4.2) have shown that – when allowing an HTN planner to insert tasks apart from the decomposition process – cycles in the decomposition structure are not necessary and can be removed. When removed actions have been needed to make the resulting sequence applicable, they can be reintroduced via task insertion. While Geier and Bercher use this result to show an upper bound of the size of TNs for this special class of HTN planning, we need it to show the existence of justifications without cycles.

Given a decomposition tree, we know by Obs. 2 that (a) there is a modified tree that (a) contains a subset of the tasks of g, that (b) does not contain cyclic decompositions and that (c) the contained actions can be made applicable by task insertion. Now consider the basic structure of a DT: We start by adding all tasks contained in the (acyclic) DT to the justification. Therefore we know that condition 1 for the justifications is fulfilled. For every abstract task, it explicitly contains the method used for decomposition, i.e., we can use this method to add the edges from the abstract task node to the method node, and from the method node to the subtask nodes. For the hierarchical part of the graph, the latter fulfills condition (2) and the former condition (3) of the justification definition. Since our decomposition structure is acyclic, we know that the new graph is.

What is left to show is that there is a justification for the part of the graph representing the state transition system. By Obs. 1 we know we can "add actions" to fulfill the conditions for a justification. Since we started with a valid decomposition tree before we removed cycles, we know that there is a set of actions that makes the sequence applicable, thus there is a valid justification for the state transition system. ☐

Theorem 2 (Soundness)**.** LM^{ao} *landmarks are landmarks for the underlying HTN planning problem.*

Proof. The approach by Keyder, Richter, and Helmert extracts landmarks for AND/OR graphs, i.e., nodes that have to be in *every* justification. By Lemma 1, there is a justification corresponding to every DT. Since every justification includes the nodes, this holds for every one that represents a DT and every DT includes the nodes. ☐

Informally, this means that we calculate landmarks for a superset of the solutions. Since the LMs are contained in all of them, they are also contained in the actual solutions.

We know from the complexity results that our approach is not complete (unless **coNP = P**). We now have a closer look at why it is not. Obviously, delete effects and ordering relations are not represented in the graph. Thus all LMs depending on these cannot be found, but we will see from the proof for the following theorem that there is another relaxation made when constructing the AND/OR graph.

Theorem 3 (Completeness)**.** LM^{ao} *does not find all LMs in Delete- and Ordering-Free HTN planning problems.*

Proof. Consider the HTN domain given in Figure 3. The initial task network (tn_I) contains the abstract tasks S and T, and the action e. The initial state is $s_0 = \{x\}$. All tasks

[3]Problems with an initial task *network* can trivially be compiled into one with just an initial *task* (Geier and Bercher 2011).

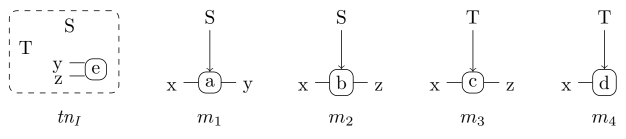


Figure 3: HTN domain without delete-effects and ordering relations. S and T are abstract tasks, a–e are actions, m_1–m_4 are methods, and x–z are state features.

$$
\begin{array}{ll}
LM(x) = \{x\} & LM(m_1) = \{m_1, a, x\} \\
LM(a) = \{a, x\} & LM(m_2) = \{m_2, b, x\} \\
LM(b) = \{b, x\} & LM(m_3) = \{m_3, c, x\} \\
LM(c) = \{c, x\} & LM(m_4) = \{m_4, d, x\} \\
LM(d) = \{d, x\} & LM(S) = \{S, x\} = \{S\} \cup \\
LM(e) = \{a, e, x, y, z\} & \quad (\{m_1, a, x\} \cap \{m_2, b, x\}) \\
LM(y) = \{y, a, x\} & LM(T) = \{T, x\} = \{T\} \cup \\
LM(z) = \{z, x\} = \{z\} \cup & \quad (\{m_3, c, x\} \cap \{m_4, d, x\}) \\
\quad (\{b, x\} \cap \{c, x\}) &
\end{array}
$$

Figure 4: Landmarks found on the domain given in Figure 3.

are unordered. S can be decomposed by m_1 into the action a, or by m_2 into the action b. T can be decomposed by m_3 and m_4 into the actions c and d, respectively.

Since e is in tn_I, it is necessarily in every solution. To make it applicable, y needs to be fulfilled, thus S needs to be decomposed using m_1 to include a in the plan, which is the only way to make y true. z is also precondition of e, i.e. b or c must be contained in every plan. However, since S needs to be decomposed into a, c is the only option to fulfill z. I.e. c must be contained in the set of LMs.

The LMs found by LM^{ao} are given in Fig. 4. The LMs for the overall problem includes $LM(S) \cup LM(T) \cup LM(e) = \{S, T, e, x, y, z, a\}$. The landmark c is not included. □

The reason for the incompleteness can be found in our AND/OR encoding of the hierarchy. Besides delete-relaxation and ordering-relaxation, we have already seen that a third relaxation is made: task insertion. This relaxation is often used in HTN heuristics to make computation feasible[4] (Alford et al. 2014), e.g. by Bercher et al. (2017) or Höller et al. (2018; 2019; 2020).

This raises the question whether there is a complete algorithm that is feasible (i.e. that can be computed in **P**). However, due to Cor. 1 we know that his is unlikely (as it would require **P**=**coNP**). There might, of course, be incomplete methods finding more LMs than ours. However, when we compare our method with the one from the literature, we see that the following theorem holds:

Theorem 4 (Dominance)**.** *Let L_1 and L_2 be the* task *landmarks generated by LM^{mt} and LM^{ao}, respectively. Then it holds that $L_1 \subseteq L_2$.*

Proof. In our generation, a task c is represented by an OR node. When its LM set is updated, it is set to the intersection of its predecessors. These predecessors are nodes resulting from the methods m_1 to m_k applicable to c. The LM sets of m_1 to m_k are set to the union of the sets of their subtasks. Since a LM set of a node n contains n by definition, the subtasks of m_1 to m_k contain themselves, i.e. that the sets of m_1 to m_k contain at least all their subtasks (but might contain more), and c the intersection of all these sets. This is exactly the definition of MT LMs. In Fig. 1 and 2 we have given an example for a LM found by LM^{ao} but not by LM^{mt}, so we might find a proper superset of LMs. □

[4]See Geier and Bercher (2011) and Alford, Bercher, and Aha (2015b) for an investigation of its impact on the comp. complexity.

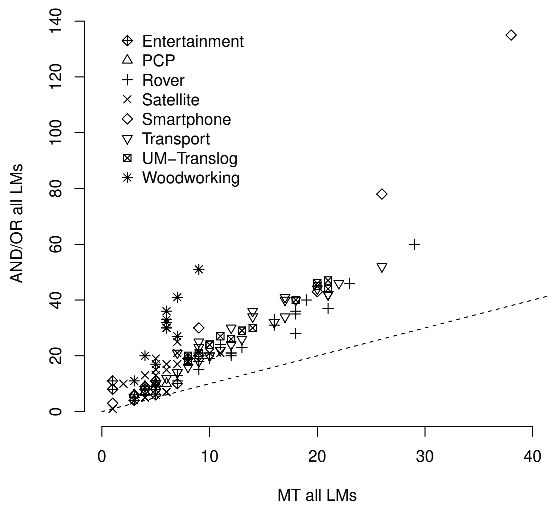


Figure 5: Number of all LMs extracted by MT (on the x axis) and AND/OR (on the y axis) generation split by domain. Please be aware the different scaling of the axis.

Evaluation

We evaluate our new LM generation on a widely-used HTN benchmark set. It has e.g. been used by Höller et al. (2018) and Behnke, Höller, and Biundo (2019a,b). It contains 144 problem instances from 8 domains. Experiments ran on Xeon E5-2660 v3 CPUs, 4 GB RAM and 10 min time.

Landmark Generation. Task LMs are extracted by both generation procedures. Over all instances, our generation finds 13% more task LMs than MT. Besides task LMs, our approach also extracts fact and method LMs. However, we find only very few method LMs (0 to 1 per instance)[5]. When we compare the full sets of LMs that are found (Figure 5), we extract more than twice the number LMs over the entire instance set.

Generation time is not an issue for both methods: MT LM generation needs 0.03 ms on average, we need 1.3 ms.

Landmark-guided Search. Though the focus of this paper is on LM *generation*, we want to show that the newly

[5]This is caused by the grounding procedure of PANDA (see Behnke et al. (2020)). Whenever there is only a single method m for a task c, occurrences of c in other methods (or the initial task network) are replaced by the subtasks of m. At most a single method LM is left that is caused by a second compilation step that replaces an initial task *network* by an initial *task*.

	#instances	LMC-AND/OR-R WA*5	LMC-AND/OR WA*5	LMC-MT WA*6	RC FF WA*2	SAT'19	SAT'18	TDG$_m$ WA*2	TDG$_c$ WA*2	2ADL Jasper
ENTERTAINMENT	12	9	9	9	**12**	**12**	**12**	9	9	5
PCP	17	13	13	13	**14**	12	12	9	8	3
SATELLITE	25	21	21	21	**25**	**25**	**25**	24	21	23
SMARTPHONE	7	4	4	4	5	**7**	6	6	5	6
UM-TRANSLOG	22	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	19
WOODWORKING	11	5	6	6	10	**11**	**11**	9	9	5
ROVER	20	4	4	3	4	**10**	4	5	5	5
TRANSPORT	30	7	1	1	15	**22**	**22**	2	1	19
total	144	85	80	79	107	**121**	114	86	80	85

Table 1: Coverage table for different systems.

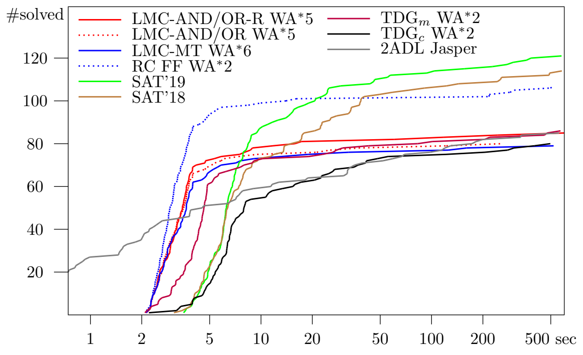


Figure 6: Number of solved instances over time.

found LMs bear information that helps guiding the search. We therefore integrated the generation mechanisms into the PANDA framework[6] (Höller et al. 2021) and combined them with the progression search algorithm described by Höller et al. (2020, Alg. 3). We realized the following heuristics:

- **LMC-MT** – Landmark count heuristic using MT LMs. Landmarks are extracted once for the initial task network. During search, reached LMs are tracked and the number of unfulfilled LMs is used as heuristic value.
- **LMC-AND/OR** – Same as before, but using our LM generation (which also includes fact and method LMs).
- **LMC-AND/OR-R** – As before with additional analysis checking whether all unfulfilled LMs are still reachable.

Be aware that a configuration with reachability analysis is not reasonable for MT LM generation. Here, all LMs are reached *by definition*, there is no chance to prevent this. Have a second look at Fig. 1 and 2. After applying m_2, a is not reachable anymore and the search node can be pruned. For the MT LM set $\{T, b\}$, however, pruning is not possible.

Figure 1 shows the coverage of several HTN planning systems. It contains the configuration with the highest coverage for each of our LM heuristics; the Relaxed Composition heuristic (Höller et al. 2018) with FF (Hoffmann and

[6]PANDA is available online under *panda.hierarchical-task.net*

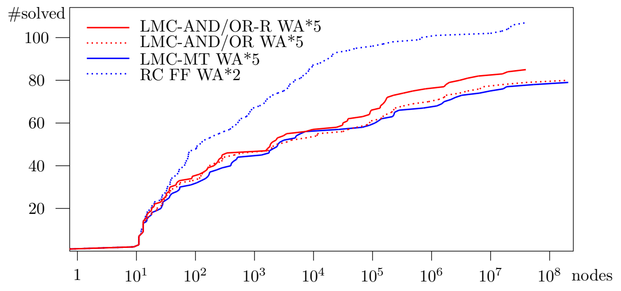


Figure 7: Solved instances for given number of search nodes.

Nebel 2001) as inner heuristic (RC FF); TDG$_m$ and TDG$_c$ heuristics (Bercher et al. 2017), and compilation-based systems. Two of the latter bound the problem and translate it to propositional logic (see Behnke, Höller, and Biundo (2018, 2019a)). When no solution is found, the bound is increased. The third compilation (Alford et al. 2016) translates the (also bounded) problem to classical planning and uses the Jasper planner (Xie, Müller, and Holte 2014) to solve it.

It can be seen that the LMC heuristic benefits from the new LMs. When only looking at coverage, the possibility to integrate a reachability analysis has a larger impact than the increased LM set. However, as can be seen in Figure 6 (showing solved instances after a given time), the increased LM set also speeds up search considerably compared to the MT-based system. When we compare instance per instance, we need 13% less search nodes (median) with a maximum of 99,81%. Summed over all instances, we need 71,22% less nodes. Figure 7 shows the number of instances (on the y-axis) solved after a certain number of search nodes (on the x-axis). It can be seen that the additional LMs cause the search to take less nodes to find plans (i.e., it is more informed).

While the SAT-based systems perform best, our new LM generation makes LMC competitive with all search-based systems apart from the RC FF heuristic. However, having the sophisticated search techniques of successful LM planners in classical planning like LAMA (Richter and Westphal 2010) in mind, it is not surprising that a simple LM count heuristic is not competitive with the RC heuristic.

Conclusion

We introduced a novel LM generation technique for HTN planning that is based on AND/OR graphs. Our approach finds fact, task, and method LMs in a single generation process. It dominates the approach on HTN LMs from the literature. We have shown that the approach is sound, incomplete, runs in **P**, and that every complete technique must solve an **coNP**-hard problem. Our evaluation shows that our approach also finds more LMs in practice and that the new LMs bear information valuable to guide the search. As next steps we consider the realization of more elaborated LM-based search techniques and consider a LAMA-like system or the integration into IP/LP-based heuristics as done in classical planning (Pommerening et al. 2014) most promising.

Acknowledgments

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 232722074 – SFB 1102/Funded by the Deutsche Forschungsgemeinschaft (DFG, German-Research Foundation) – Project-ID 232722074 – SFB 1102.

References

Alford, R.; Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; and Aha, D. W. 2016. Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 20–28. AAAI Press.

Alford, R.; Bercher, P.; and Aha, D. 2015a. Tight Bounds for HTN Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*, 7–15. AAAI Press.

Alford, R.; Bercher, P.; and Aha, D. W. 2015b. Tight Bounds for HTN Planning with Task Insertion. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 1502–1508. AAAI Press.

Alford, R.; Shivashankar, V.; Kuter, U.; and Nau, D. S. 2014. On the Feasibility of Planning Graph Style Heuristics for HTN Planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2018. Tracking Branches in Trees – A Propositional Encoding for Solving Partially-Ordered HTN Planning Problems. In *Proceedings of the 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 73–80. IEEE Press.

Behnke, G.; Höller, D.; and Biundo, S. 2019a. Bringing Order to Chaos – A Compact Representation of Partial Order in SAT-Based HTN Planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 7520–7529. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2019b. Finding Optimal Solutions in HTN Planning – A SAT-based Approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 5500–5508. IJCAI Organization.

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9775–9784. AAAI Press.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 6267–6275. IJCAI Organization.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 480–488. IJCAI Organization.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid Planning Heuristics Based on Task Decomposition Graphs. In *Proceedings of the 7th Annual Symposium on Combinatorial Search (SoCS)*, 35–43. AAAI Press.

Elkawkagy, M.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2012. Improving Hierarchical Planning Performance by the Use of Landmarks. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 1763–1769. AAAI Press.

Elkawkagy, M.; Schattenberg, B.; and Biundo, S. 2010. Landmarks in Hierarchical Planning. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, 229–234. IOS Press.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. HTN Planning: Complexity and Expressivity. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*, 1123–1128. AAAI Press / The MIT 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 *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 1955–1961. AAAI Press.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning – Theory and Practice*. Elsevier.

Goldman, R. P.; and Kuter, U. 2019. Hierarchical Task Network Planning in Common Lisp: the case of SHOP3. In *Proceedings of the 12th European Lisp Symposium (ELS)*, 73–80. ACM.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 162–169. AAAI.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research* 14: 253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research* 22: 215–278.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language Classification of Hierarchical Planning Problems. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the Expressivity of Planning Formalisms through the Comparison to Formal Languages. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 158–165. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *Künstliche Intelligenz* doi:10.1007/s13218-020-00699-y.

Höller, D.; Bercher, P.; and Behnke, G. 2020. Delete- and Ordering-Relaxation Heuristics for HTN Planning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 4076–4083. IJCAI Organization.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2018. A Generic Method to Guide HTN Progression Search with Classical Heuristics. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, 114–122. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6171–6175. IJCAI Organization.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN Planning as Heuristic Progression Search. *Journal of Artificial Intelligence Research* 67: 835–880.

Kambhampati, S.; Mali, A.; and Srivastava, B. 1998. Hybrid Planning for Partially Hierarchical Domains. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, 882–888. AAAI Press.

Karpas, E.; and Domshlak, C. 2009. Cost-Optimal Planning with Landmarks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1728–1733. AAAI Press.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, 335–340. IOS Press.

Mirkis, V.; and Domshlak, C. 2007. Cost-Sharing Approximations for h+. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, 240–247. AAAI.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20: 379–404.

Olz, C.; Biundo, S.; and Bercher, P. 2021. Revealing Hidden Preconditions and Effects of Compound HTN Planning Tasks – A Complexity Analysis. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-Based Heuristics for Cost-Optimal Planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 226–234. AAAI Press.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering, and Usage of Landmarks in Planning. In *Proceedings of the 6th European Conference on Planning (ECP)*, 174–182. AAAI Press.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 975–982. AAAI Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal Artificial Intelligence Research* 39: 127–177.

Shivashankar, V.; Alford, R.; and Aha, D. 2017. Incorporating Domain-Independent Planning Heuristics in Hierarchical Planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 3658–3664. AAAI Press.

Shivashankar, V.; Alford, R.; Kuter, U.; and Nau, D. 2013. The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2380–2386. AAAI Press.

Shivashankar, V.; Alford, R.; Roberts, M.; and Aha, D. W. 2016a. Cost-Optimal Algorithms for Hierarchical Goal Network Planning: A Preliminary Report. In *Proceedings of the 8th Workshop on Heuristics and Search for Domain-independent Planning (HSDIP)*, 102–111.

Shivashankar, V.; Alford, R.; Roberts, M.; and Aha, D. W. 2016b. Cost-Optimal Algorithms for Planning with Procedural Control Knowledge. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, 1702–1703. IOS Press.

Xie, F.; Müller, M.; and Holte, R. 2014. Jasper: The art of Exploration in Greedy Best First Search. In *Proceedings of the 8th International Planning Competition (IPC)*, 39–42.

Zhu, L.; and Givan, R. 2003. Landmark extraction via planning graph propagation. In *Doctoral Consortium of the International Conference on Automated Planning and Scheduling (ICAPS DC)*, 156–160.