Proceedings of the 3^{rd} ICAPS Workshop on Hierarchical Planning (HPlan 2020)

Landmark Extraction in HTN Planning

Daniel Höller
Saarland Informatics Campus,
Saarland University,
Saarbrücken, Germany
hoeller@cs.uni-saarland.de

Pascal Bercher
College of Engineering and Computer Science,
The Australian National University,
Canberra, Australia
pascal.bercher@anu.edu.au

Abstract

Landmarks 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. Landmarks 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. In this paper we introduce a novel landmark generation method for Hierarchical Task Network (HTN) planning and show that it is sound and incomplete. We show that every complete approach is as hard as the underlying HTN problem. Since we make relaxations during landmark generation, this means **NP**-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 landmark *generation*, we show that the newly discovered landmarks bear information beneficial for solvers.

1 Introduction

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 called *primitive tasks*) and *abstract tasks*. The latter 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 executable decomposition of given abstract tasks. 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" method for a certain task. This makes HTN planning more expressive (Erol, Hendler, and Nau 1996; Geier and Bercher 2011; Höller et al. 2014; 2016). The decomposition process can be seen as an AND/OR tree (Ghallab, Nau, and Traverso 2004; Kambhampati, Mali, and Srivastava 1998). 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 for heuristic search (see e.g. Karpas and Domshlak (2009) or Helmert and Domshlak (2009)). 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; 2016b; Shivashankar, Alford, and Aha 2017), the presented techniques are not applicable to HTN planning. We summarize landmark-related work in the context of HGN planning in Section 5.

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

2 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*) that is 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 action names to a set of state features defining the action's *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 $(a_1, a_2, \ldots, a_n)$ 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 a sequence of actions applicable in s_0 that results in a goal state.

We now extend classical problems to HTN problems based on the formalism by 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*. Tasks are divided into *primitive tasks* equal to actions in classical planning, and *abstract tasks* that can not be applied directly but need to be decomposed first. Let A and C[1] be the sets of primitive and abstract tasks, respectively. 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*. A task network 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 task network.

Planning starts with a special task network 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 task network, i.e. they are pairs (c, tn). When a method (c, tn) is applied to a task t with $\alpha(t) = c$ in a task network, the task is deleted from the network, the tasks defined in tn are added and they inherit the ordering relations that have been present for t. A task network $tn_1 = (T_1, \prec_1, \alpha_1)$ is decomposed into a task network $tn_2 = (T_2, \prec_2, \alpha_2)$ by a method (c, tn), if tn_1 contains a task $t \in T_1$ with $\alpha_1(t) = c$ and there is a task network $tn' = (T', \prec', \alpha')$ equal to tn but using different ids (i.e. $T_1 \cap T' = \emptyset$). tn_2 is defined as follows:

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

We will 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 will write $tn_1 \to^* tn_2$ to denote that a task network tn_1 can be decomposed into a task network tn_2 by using a (possibly empty) sequence of methods.

The elements s_0, g, and δ are defined as before. The objective of the problem is to find an executable decomposition of $tn_I = (T_I, \prec_I, \alpha_I)$ by adhering the decomposition methods resulting in a state satisfying g. More formally, a task network $tn_S = (T_S, \prec_S, \alpha_S)$ is a *solution* if and only if (1) $tn_I \to^* tn_S$, i.e. tn_S can be created by decomposing tn_I, (2) All tasks in tn_S are primitive, and (3) There is a sequence of all tasks satisfying the ordering constraints $\prec_S$, applicable in s_0 that results 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 as well. Specifying such a goal is optional, and is indeed not required from a theoretical perspective as it can easily be compiled away (Geier and Bercher 2011). Our LM generation procedure works fine without a state-based goal definition and most problem instances in the used benchmark set do not contain one.

We now define various types of landmarks. Our definition for *task landmarks* (Def. 1) is essentially equivalent to Def. 3 of Elkawkagy et al. (2012), but adapted to our formalism. Most notably, the original formalization is based on a lifted formalization (whereas the respective landmarks are still required to be ground). Def. 2 is new.

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 decomposition sequence to every 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 $tn_I \to^* tn_1 \xrightarrow[t,m]{} tn_2 \to^* tn_S$.*

Our definition of *fact landmarks* is a canonical adaptation of fact landmarks from classical planning (Porteous, Sebastia, and Hoffmann 2001).

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

3 Landmark Generation in HTN Planning

The concept of landmarks in HTN-like planning has first been studied by Elkawkagy, Schattenberg, and Biundo (2010). Here, landmarks have not been extracted from states, but from the task hierarchy. 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 empirical evaluation in their paper evaluates the impact of a domain model reduction that is done simultaneously to the mandatory task generation.

[1] C is short for *compound*, a common synonym for *abstract*.

In follow-up work (Elkawkagy et al. 2012) they 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 mandatory tasks 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. They showed how landmarks of a planning task can be computed based on mandatory tasks and used these landmarks for an admissible landmark counting heuristic. To show in which way we extend their landmark heuristic (Bercher, Keen, and Biundo 2014, Def. 1), we reproduce their definition, but simplified and adapted to our notation:

Definition 4 (Mandatory Task-based 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:*

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

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

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

The generation method collects the tasks contained in all methods that belong to the same abstract task. Thus all tasks that get introduced at deeper levels of abstraction cannot be found unless all methods share some abstract task(s). This, however, could be improved by lookahead techniques as done in early approaches in classical planning.

4 AND/OR Landmarks in HTN Planning

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

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 (Sec. 4.1), and then show that it can nicely be adapted to HTN planning (Sec. 4.2).

4.1 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 (DR) classical planning problem (i.e., for all $a \in A$ holds $del(a) = \emptyset$). It can be understood 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 landmark sets of all nodes apart from V_I with all vertices of the graph, i.e. the full landmark set. Nodes in V_I are initialized with its own value. Then the equations given before are used as update rules for the sets until a fixpoint is reached.

4.2 Extracting Landmarks in HTN Planning Using AND/OR Graphs

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

When we now have a look at HTN planning, we see 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 also 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:*

$G = (V_I, V_{and}, V_{or}, E)$ with $V_I = s_0$, $V_{and} = A \cup M$[2] and $V_{or} = F \setminus s_0 \cup C$. The set of edges is defined as $E =$

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

Now we generate the landmarks by using the same generation mechanism as Keyder, Richter, and Helmert. Since the size of the graph is linear in the size of the model, it trivially follows that this computation is in **P**. The overall set of LMs is then based on 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*

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

The example given in Figure 1 illustrates the interplay of hierarchy and state during landmark 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 landmark generation, we end up with the landmark set $\{T, b\}$.

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

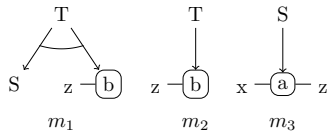
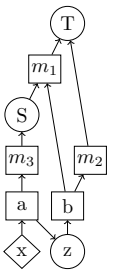


Figure 1: Simple HTN domain. S and T are abstract tasks, m_1 to m_3 methods, a and b actions, and x and z state features. T might be decomposed by m_1 into S and b, or by m_2 into b. S can be decomposed by m_3 into a.



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

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.

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.

4.3 Theoretical Properties

Before we state the properties of *our approach*, let us state theoretical properties of landmarks *in general*. Similar to classical planning, we will see that deciding whether a task, method, or fact is a landmark is as hard as planning itself.

Theorem 1. *Let $\mathcal{P}$ be an HTN planning problem. Let t be a task, m be a method, and f a fact. Deciding whether t, m, or f is a landmark is exactly as hard (with matching upper and lower bounds of the respective complexity class) as deciding the plan existence problem for $\mathcal{P}$.*

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

Hardness. We will introduce a new artificial initial abstract task c_I with two decomposition methods. The first, m_1, decomposes c_I into the original initial task network of $\mathcal{P}$, whereas the other, m_2, decomposes it into a new task network that solves the problem. For this, m_2 decomposes into an abstract task t, which in turn decomposes (i.e., with yet another method) into a primitive task t'. t' uses an empty precondition, a new "dummy" fact f as effect, as well as g as further effects. t' does not use negative effects. Note that we could have put t and t' into the same task network, thus saving another "decomposition level" and method, but we

wanted to keep the size of new task networks limited to 1 so we don not potentially change properties of the problem we reduce from (like number of tasks per task network, their form of ordering constraints, or the "position" of said task, which may all influence the computational complexity).

Clearly, m_2, t (abstract), t' (primitive), and f are landmarks if and only if $\mathcal{P}$ is *un*solvable. Note that determining the unsolvability is as hard as determining the solvability, as those two problems are complimentary to each other.

Membership. Similar to Hoffmann, Porteous, and Sebastia (2004, Thm. 1), we test whether the problem remains solvable if we ignore all parts of the model that "relate" to the landmark(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 task networks and methods that contain them. And in case of a fact landmark, we remove all actions (i.e., again removing all methods that introduce it) that add it. Decide the resulting problem. The task, method, or fact is a landmark if and only if the respective problem is not solvable. □

Using that theorem we can thus deduce the computational hardness of determining whether a task, method, or fact is a landmark for many standard HTN planning problems. We only mention the most important ones here, but as mentioned above our theorem is more general.

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 **NP**-complete.*

Proof. Follows from Thm. 1 in conjunction with the results for the general case (Erol, Hendler, and Nau 1996; Geier and Bercher 2011), totally ordered problems (Alford, Bercher, and Aha 2015a), and delete-relaxed problems (Alford et al. 2014; Höller, Bercher, and Behnke 2020). □

We can conclude that any *complete* landmark extraction technique for delete- and ordering-relaxed HTN problems cannot run in polynomial time unless **P**=**NP** (Höller, Bercher, and Behnke 2020).

Some of our further results (i.e., their proofs) rely on the so-called *Decomposition Tree* (Geier and Bercher 2011), which we formally introduce next. It is a formal representation of a task network and it deviation from the initial task.[3]

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 exists a valid decomposition tree for every solution of an HTN problem (Geier and Bercher 2011, Prop. 1), since they simply represent the underlying hierarchy of the respective task network. We can now discuss some properties of the new approach.

Lemma 1. *Let $\mathcal{P}$ be an HTN planning problem, tn a solution task network, and dt its decomposition tree. Then, there exists a justification for the initial task of the AND/OR graph (given in Definition 8) representing 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 executable, they can be reintroduced via task insertion. While Geier and Bercher use this result to show an upper bound of the size of task networks (for this special class of HTN planning problems), 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 executable 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, DT it explicitly contains the method used for its 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

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

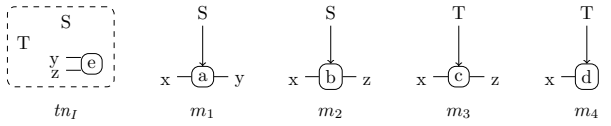


Figure 3: HTN domain without delete-effects and ordering relations. S and T are abstract tasks, a–e are actions, m_1–m_4 are methods, and x–z are state features.

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

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

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. □

A second question is whether the approach is complete. Obviously, delete effects and ordering relations are not represented in the graph. Thus all LMs depending on delete-effects and/or the ordering can not be found. This leaves the question whether all LMs apart from these are found for delete- and ordering-free (DOF) HTN problems.

Theorem 3 (Completeness)**.** *LM^{ao} does not find all LMs in DOF 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 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 executable, 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 seen in the AND/OR encoding. Besides the two obvious relaxations given above (delete-relaxation and ordering-relaxation), a third relaxation is made that further increases the set of solutions: A certain abstract task may be decomposed more than once. This can be seen as task insertion (cf. Geier and Bercher (2011) and Alford, Bercher, and Aha (2015b) for an investigation of its impact on the computational complexity). This relaxation is often used in HTN heuristics to make computation feasible (Alford et al. 2014), e.g. by Bercher et al. (2017) or Höller et al. (2018; 2019; 2020).

The incompleteness result 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**=**NP**). 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 *LMs generated by LM^{mt} & LM^{ao}, respectively. Then, $L_1 \subseteq L_2$.*

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, 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. □

5 Landmarks Apart from HTN Planning

There are many hierarchical planning formalisms in the literature, some of them differ severely in their formalization, semantics, and computational properties (Bercher, Alford, and Höller 2019). Landmarks have also been used in HGN planning, which is concerned with the refinement and achievement of state-based goals rather than tasks (Shivashankar et al. 2012). In a nutshell, there are no *task* networks in HGN planning, but *goal* networks instead, which are partially ordered formulae over state-variables. Methods now refine these goals into further goal networks. There is just one sort of task: the actions known from classical planning. The objective is to find an executable action sequence that achieves all goals and satisfies the given order (possibly by refining goals using the methods), whereas each action can only be applied to a state if it achieves some goal, thereby making it as expressive as HTN planning (Shivashankar et al. 2012; Alford et al. 2016b). When the HGN formalism was first described, Shivashankar et al. (2012) already proposed how an HGN planner could incorporate heuristics, but no landmark information was used.

Their follow-up planner *GoDeL* (Goal Decomposition with Landmarks) (Shivashankar et al. 2013) uses standard classical landmark generation (Hoffmann, Porteous, and Sebastia 2004; Richter and Westphal 2010) to obtain a partially

ordered set of landmarks (called a *landmark graph*) that can be used to guide the search to the next goal.

Their next system *HOpGDP* (Hierarchically-Optimal Goal Decomposition Planner) is guided by a landmark heuristic called h_{HL} (HGN Landmark heuristic) (Shivashankar et al. 2016a; 2016b). h_{HL} builds upon existing landmark techniques and extends them to work on a partially ordered set of goals rather than on a single "at end" goal as in classical planning: Given a current goal network, i.e., a partially ordered set of goals, they can regard this network as a landmark graph and extend it by further landmarks found by techniques from the literature (Richter and Westphal 2010). They then use this extended landmark graph as input to the technique of Karpas and Domshlak (2009) to obtain an admissible heuristic.

The newest HGN system is called *HOGL* (Hierarchically-Optimal Goal Decomposition Planner using LMCut) and again relies on landmarks to guide search (Shivashankar, Alford, and Aha 2017). In this work, they obtain heuristic estimates by first performing a problem relaxation that ignores the hierarchy and action applicability restrictions and then compiling that problem into a classical planning problem (per search node) to use standard classical heuristics. Whereas this approach is agnostic towards the classical heuristic actually used, the approach was evaluated using the LM-cut heuristic (Helmert and Domshlak 2009).

To summarize this line of research: In HGN planning, information about landmarks was used successfully to a large extent, but all approaches use the procedures from classical planning as black box procedures without extending them (at all or) by the information provided by the hierarchy.

6 Evaluation

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

Landmark Extraction. 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)[4]. When we compare the full sets of LMs that are found (Figure 5), we extract 2.3 as many LMs over the entire instance set.

Extraction 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 found LMs bear information that helps guiding the search. We therefore integrated the generation mechanisms into the PANDA framework 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 & 2. After applying m_2, a is not

[4]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*.

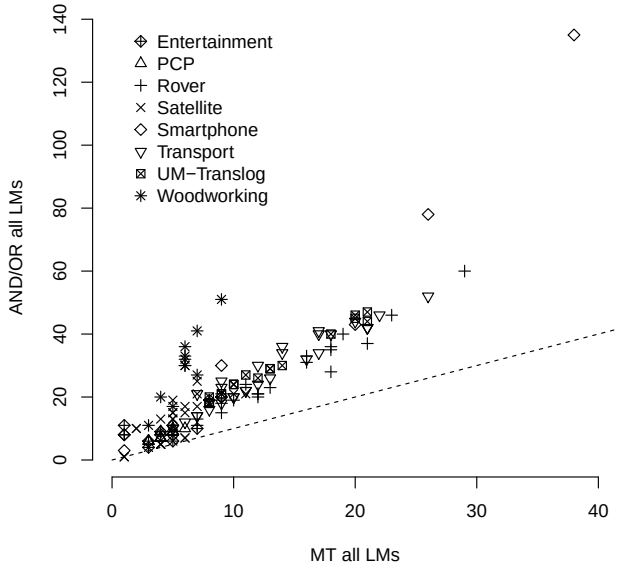


Figure 5: Number of all LMs extracted by MT (on the x axis) and AND/OR (on the y axis) generation split by domain. Please be aware the different scaling of the axis.

	#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.

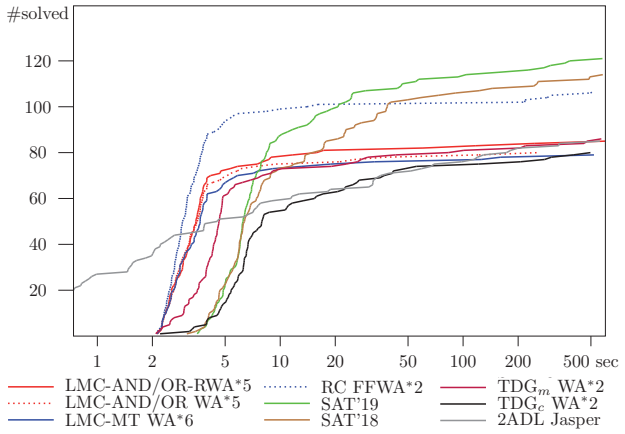


Figure 6: Number of solved instances over time.

reachable anymore and the search node can be pruned. For the MT LM set $\{T, b\}$, however, pruning is not possible.

Table 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 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. 2016a) 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 LM generation process. 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. 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 and rather complex 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.

7 Conclusion

We introduced a novel LM generation technique for HTN planning that is based on AND/OR graphs. Notably, we do not depend on a state-based goal definition, which is often not present in HTN planning though we can also extract LMs from this definition if there is one. Our approach finds fact, task, and method LMs in a single generation process. It dominates the approach on HTN LMs from the literature, even when restricted to just task LMs (no other kinds of LMs could be extracted before). We have shown that the approach is sound, incomplete, runs in **P**, and that every complete technique must be **NP**-hard. We tested our approach on a widely-used benchmark set and showed that it also finds more LMs in practice. Though the simple LM count heuristic we used is not competitive with state-of-the-art solving techniques, we showed that the new LMs bear information valuable to guide the search.

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.; Bercher, P.; and Aha, D. 2015a. Tight bounds for HTN planning. In *Proc. of the 25th Int. Conf. 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 *Proc. of the 24th Int. Joint Conf. on AI (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 *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Alford, R.; Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; and Aha, D. W. 2016a. 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)*, 20–28. AAAI Press.

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016b. Hierarchical planning: relating task and goal decomposition with task sharing. In *Proc. of the 25th Int. Joint Conf. on AI (IJCAI)*, 3022–3029. AAAI Press.

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On succinct groundings of HTN planning problems. In *Proc. of the 34th AAAI Conf. on AI (AAAI)*, 9775–9784. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2018. Tracking branches in trees – A propositional encoding for solving partially-ordered HTN planning problems. In *Proc. of the 30th Int. Conf. on Tools with AI (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 *Proc. of the 33rd AAAI Conf. on AI (AAAI)*, 7520–7529. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2019b. Finding optimal solutions in HTN planning – A SAT-based approach. In *Proc. of the 28th Int. Joint Conf. on AI (IJCAI)*, 5500–5508. IJCAI Organization.

Bercher, P.; Alford, R.; and Höller, D. 2019. A survey on hierarchical planning – One abstract idea, many concrete realizations. In *Proc. of the 29th Int. Joint Conf. on AI (IJCAI)*, 6267–6275. IJCAI Organization.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An admissible HTN planning heuristic. In *Proc. of the 26th Int. Joint Conf. on AI (IJCAI)*, 480–488. IJCAI Organization.

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.

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

Elkawkagy, M.; Schattenberg, B.; and Biundo, S. 2010. Landmarks in hierarchical planning. In *Proc. of the 19th European Conf. on AI (ECAI)*, 229–234. IOS 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 Int. Joint Conf. on AI (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 *Proc. 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 *Proc. of the 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 162–169. AAAI Press.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22:215–278.

Höller, D.; Bercher, P.; and Behnke, G. 2020. Delete- and ordering-relaxation heuristics for HTN planning. In *Proc. of the 29th Int. Joint Conf. on AI (IJCAI)*, 4076–4083. IJCAI Organization.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language classification of hierarchical planning problems. In *Proc. of the 21st European Conf. on AI (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 *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 158–165. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2018. A generic method to guide HTN progression search with classical heuristics. In *Proc. of the 28th Int. Conf. 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 *Proc. of the 28th Int. Joint Conf. on AI (IJCAI)*, 6171–6175. IJCAI Organization.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN planning as heuristic progression search. *JAIR* 67:835–880.

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

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *Proc. of the 21st Int. Joint Conf. on AI (IJCAI)*, 1728–1733. AAAI Press.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for And/Or graphs. In *Proc. of the 19th European Conf. on AI (ECAI)*, 335–340. IOS Press.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h+. In *Proc. of the 17th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 240–247. AAAI Press.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *JAIR* 20:379–404.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc. of the 6th European Conf. on Planning (ECP)*, 174–182. AAAI Press.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. of the 23rd AAAI Conf. on AI (AAAI)*, 975–982. AAAI Press.

Shivashankar, V.; Alford, R.; and Aha, D. 2017. Incorporating domain-independent planning heuristics in hierarchical planning. In *Proc. of the 31st AAAI Conf. on AI (AAAI)*, 3658–3664. AAAI Press.

Shivashankar, V.; Kuter, U.; Nau, D.; and Alford, R. 2012. A hierarchical goal-based formalism and algorithm for single-agent planning. In *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 981–988. IFAAMAS.

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 *Proc. of the 23rd Int. Joint Conf. on AI (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 *Proc. 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 *Proc. of the 22nd European Conf. on AI (ECAI)*, 1702–1703. IOS Press.

Xie, F.; Müller, M.; and Holte, R. 2014. Jasper: The art of exploration in greedy best first search. In *Proc. of the 8th Int. Planning Competition (IPC)*, 39–42.

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