Landmark Generation in HTN Planning Revisited

Victor Scherer Putrich[1,2], **Felipe Meneguzzi**[3,4], **André Grahl Pereira**[1]

[1]Federal University of Rio Grande do Sul, Brazil
[2]Saarland University, Germany
[3]University of Aberdeen, Scotland
[4]Pontifical Catholic University of Rio Grande do Sul, Brazil
victor.sputrich@inf.ufrgs.br, felipe.meneguzzi@abdn.ac.uk, agpereira@inf.ufrgs.br

Abstract

In Hierarchical Task Network (HTN) planning, *landmarks* are facts that must hold true, and tasks or methods that must be included in every solution. Existing landmark generation techniques for HTN planning rely on the Delete and Ordering Free (DOF) relaxation and are known to be sound but incomplete, primarily due to the limitations introduced by Task Insertion. This paper presents a new landmark generation method that builds on a previous AND/OR graph-based approach, extending it to capture additional hierarchical dependencies among tasks and methods. We prove that our approach is sound and dominates existing techniques, though it remains incomplete under the DOF relaxation. Experimental results on IPC benchmarks for totally ordered problems show that our method identifies significantly more task and method landmarks across most domains, improving coverage with minimal computational overhead.

1 Introduction

Planning is finding a sequence of *actions* that transforms an initial state into one that satisfies the goal condition. *Classical planning* models the environment as a set of propositions or facts that change due to deterministic actions. *Hierarchical Task Network* (HTN) extends classical planning by organizing objectives into a hierarchical structure of tasks and enforcing tasks to comply with predefined ordering constraints. High-level *compound tasks* decompose into simpler subtasks through predefined *methods* until only low-level (*primitive*) tasks remain. Primitive tasks are actions that are applied to states to modify them. Besides defining how high-level tasks can be decomposed, methods impose ordering constraints among subtasks. This hierarchical structure enables modeling more expressive planning problems. However, it also increases computational complexity: while classical planning is PSPACE-complete, plan existence in unrestricted HTN planning is undecidable (Erol, Hendler, and Nau 1994).

Early work on landmark generation in HTN planning used an AND/OR graph representation known as the *Task Decomposition Graph* (TDG). The TDG models the hierarchical structure of the HTN planning problem to identify task landmarks, referred to as *mandatory tasks* (Elkawkagy et al. 2012; Bercher, Keen, and Biundo 2014). Up to that point, the only source of information for landmark identification was the task hierarchy, disregarding the executability of primitive tasks. Höller and Bercher (2021) addressed this limitation by adapting an AND/OR encoding to HTN based on the formulation from (Keyder, Richter, and Helmert 2010). Their approach incorporates both facts and methods as potential landmarks and introduces two key relaxations. First, it ignores delete effects and ordering constraints, a relaxation known as Delete- and Ordering-Free (DOF) (Höller, Bercher, and Behnke 2020). Second, it encodes the Task Insertion relaxation (Geier and Bercher 2011), which allows primitive tasks to be reached independently of the task hierarchy. This can result in overlooking hierarchical dependencies and potentially missing landmarks.

In this paper, we enhance landmark generation by modifying the encoding by Höller and Bercher 2021. Our approach has three steps. First, it generates landmarks using the previously introduced AND/OR graph. Second, we modify the AND/OR graph to capture hierarchy dependencies previously ignored by task insertion. Finally, we use a fix-point computation to generate more landmarks using the two previous steps as sources of landmarks. We prove the soundness of our approach and establish that it dominates existing techniques by identifying all landmarks found by earlier methods and possibly additional ones; however, our approach remains incomplete under DOF relaxation.

Experiments on IPC-2023 benchmarks for total-order problems show that our method generates more landmarks in 12 out of 22 domains. Specifically, we detect 8% more fact landmarks, 87% more task landmarks, and $5,094\%$ more method landmarks. Beyond landmark generation, the additional landmarks increase coverage across the tested benchmarks and reduce the number of expanded nodes in most problems.

2 Background

Classical planning. A classical planning problem is a tuple $\mathcal{P} = (F, A, \gamma, s_0, \mathcal{G})$ where F is a finite set of propositional facts, A a finite set of actions, and $\gamma : A \to (2^F, 2^F, 2^F)$ assigns to each action a its *preconditions* $\text{pre}(a)$, *add effects* $\text{add}(a)$ and *delete effects* $\text{del}(a)$.

A (world) state is any subset $s \subseteq F$. The problem starts in the *initial state* s_0 and aims to reach a state satisfying the *goal condition* $\mathcal{G} \subseteq F$. Action a is *applicable* in s iff $\text{pre}(a) \subseteq s$; its execution yields the successor $s' = (s \setminus \text{del}(a)) \cup \text{add}(a)$. A plan is a sequence $\pi = \langle a_1, \ldots, a_n \rangle$ such that each a_i is applicable in the state produced by its predecessor and the final state satisfies $\mathcal{G}$.

HTN planning. An *HTN planning* problem is $P = (s_0, \mathcal{G}, tn_I, F, A, C, M)$ [1] and extend a classical planning with a task hierarchy: C is a set of *compound tasks*, they abstractly describe a task that needs refinement; A is reused for *primitive tasks* (actions). M is a set of *methods* $m = \langle c, tn_m \rangle$ that refine a compound task $c \in C$ into a *task network* tn_m.

A task network is a triple $tn = (T, \prec, \alpha)$ where T is a finite set of task identifiers, $\prec \subseteq T \times T$ a strict partial order, and $\alpha : T \to A \cup C$ labels each identifier with an action or a compound task. Refining $tn_1 = (T_1, \prec_1, \alpha_1)$ with method $m = (c, tn_m)$ at identifier $t \in T_1, \alpha_1(t) = c$ (denoted $tn_1 \xrightarrow{t,m} tn_2$) removes t and inserts a (renamed) copy of tn_m, while progressing ordering constraints and label functions (cf. Geier and Bercher 2011 for full details). We write $tn_I \to^* tn$ for a sequence of such refinements. A *solution* is a primitive task network tn_S reachable from tn_I whose linearization can be executed from s_0 to reach the goal (if defined) and comply with $\prec_S$.

To establish key landmark properties, we review the concept of *decomposition trees*. For a complete formalization, refer to Geier and Bercher (2011). Decomposition trees (DT) serve as witnesses that the initial task network derives a primitive task network through a series of valid decompositions.

Definition 2.1 (Decomposition tree)**.** Let P be an HTN planning problem. A *decomposition tree* for P is a quintuple $g = (T, E, \prec, \alpha, \beta)$ where

- (T, E) is a rooted tree (edges oriented towards the leaves);
- $\prec$ is a strict partial order on T that extends the ordering constraints of method's application.
- $\alpha : T \to A \cup C$ assign each node with a task name;
- $\beta : T \to M$ assign each *inner* node with the method used for its decomposition.

g is *valid* iff (i) the root is $\alpha(t_{\text{root}}) = c_I$ with c_I an (artificial) initial task that decomposes into tn_I, and (ii) for every inner node t with $\beta(t) = (\alpha(t), tn_m)$, the children of t, together with $\prec$, reproduce a copy of tn_m. The *yield* of g, denoted $\text{yield}(g)$, is the task network formed by its leaves and the induced order $\prec$.

A task network tn is reachable from tn_I iff there exists a valid DT g with $\text{yield}(g) = tn$ (Geier and Bercher 2011, Prop. 1). DTs are trees, hence cycle-free, even though cycles might occur over compound task names.

Landmarks A *landmark* is any component that must appear or become true in every solution to the underlying planning problem. More generally, one can also speak of disjunctive landmarks (i.e., one of its elements must be achieved in any solution). Our work only considers unary landmarks, or disjunctions containing a single element.

Definition 2.2 (Landmarks in HTN planning)**.** (Höller and Bercher 2021) Let $P = (s_0, \mathcal{G}, tn_I, F, A, C, M)$.

1. **Task-landmark:** $n \in A \cup C$ if every solution derivation contains some identifier t with $\alpha(t) = n$.
2. **Method-landmark:** $m \in M$ if every solution derivation applies m.
3. **Fact-landmark:** $f \in F$ if every executable linearisation of every solution passes through a state where f holds.

Landmark membership is co-class of the plan existence problem in the underlying planning formalism. That is PSPACE-complete in classical and undecidable in HTN planning (Hoffmann, Porteous, and Sebastia 2004; Höller and Bercher 2021), hence, practical methods rely on problem relaxations.

Encoding HTN problems with AND/OR Graphs. The *AND/OR graph* of Keyder, Richter, and Helmert (2010) offers a complete and polynomial-time method for identifying landmarks in delete-relaxed classical planning. Höller and Bercher (2021) extend this approach to HTN planning by adapting the graph structure to capture hierarchical task relations. The graph encodes a relaxed planning problem using an AND/OR graph. The edges capture dependency relations among the components (i.e, for HTN the components are tasks,facts and methods). In this graph, a fixed-point procedure finds landmark information within each node and extracts landmarks to solve the problem.

For HTN planning, the AND/OR graph described by Höller and Bercher (2021) works as follows: every primitive action is an AND-node whose predecessor facts must hold before the action is applied; every method is an AND-node whose incoming edges are the subtasks that must all be achieved for the method to hold. Compound tasks and facts are OR-nodes that can be supported by *any* of their producers (that is, methods and primitive tasks, respectively). Initial nodes are the starting point (don't need supporters), and goal nodes are goal facts from $\mathcal{G}$ along with the tasks from the initial task network tn_I.

We adopt the same notation but call it the *Bottom-up Graph* (BU), defined by G^{bu}, to emphasize that landmarks propagate upward through the task hierarchy.

Definition 2.3 (Bottom-up Graph)**.** For an HTN problem $P = (s_0, tn_I, \mathcal{G}, F, A, C, M)$, the BU graph is a directed graph $G^{bu} = (V, E)$ with

$$V_{\text{and}} = A \cup M, \quad V_{\text{or}} = C \cup (F \setminus s_0), \\ V_I = s_0, \qquad V_{\mathcal{G}} = tn_I \cup \mathcal{G}.$$

$V = V_{\text{and}} \cup V_{\text{or}} \cup V_I \cup V_{\mathcal{G}}$ and edges

$$\begin{aligned} E = &\{(f, a) \mid a \in A,\ f \in \text{pre}(a)\} \\ &\cup \{(a, f) \mid a \in A,\ f \in \text{add}(a)\} \\ &\cup \{(n, m) \mid m = (c, tn) \in M,\ t \in T_{tn},\ \alpha_{tn}(t) = n\} \\ &\cup \{(m, c) \mid m = (c, tn) \in M\}. \end{aligned}$$

According to Keyder, Richter, and Helmert (2010) the original AND/OR graph can be understood as specifying a

[1]For brevity, we omit γ from P.

delete-relaxed classical problem, while a *justification* from V_I to $V_\mathcal{G}$ corresponds to a solution of the problem; Höller and Bercher (2021) establish the same result for HTN planning under some additional relaxations; which will be presented at the end of this section. The appropriate notion of justification is formalized below as an acyclic *subgraph* J.

Definition 2.4 (Justification Graph)**.** A subgraph $J = (V^J, E^J)$ of an AND/OR graph G *justify* the goal nodes $V_\mathcal{G} \subseteq V^J$ if and only if the following conditions hold:

1. $V_\mathcal{G} \subseteq V^J$
2. $\forall a \in V^J \cap V_{\text{AND}} : \forall (v, a) \in E : v \in V^J \wedge (v, a) \in E^J$
3. $\forall o \in V^J \cap V_{\text{OR}} : \exists (v, o) \in E : v \in V^J \wedge (v, o) \in E^J$.
4. J is acyclic.

In other words, $V_\mathcal{G}$ must be part of J, AND–nodes are justified if *all* its predecessors are selected, whereas an OR–node requires at least one predecessor (except for nodes in V_I which are trivially justified). J must be acyclic to ensure non-circular dependencies.

Computing landmarks. A simple, but intractable, way to find landmarks is to enumerate all justifications and then take the intersection of these sets of nodes. A set containing the intersection nodes yields exactly those nodes common to all solutions (i.e., the landmarks by definition). Instead, Keyder, Richter, and Helmert (2010) introduce a fix-point equation whose unique maximal solution computes the same set in polynomial time via iterative set updates:

Definition 2.5 (Landmark-table update rules)**.** For $v \in V$ let $\text{pred}(v) = \{u \mid (u, v) \in E\}$. Initially $LM(v) = V$ for every $v \notin V_I$ and $LM(v) = \{v\}$ for $v \in V_I$. Iteratively replace each entry according to

$$LM(V_\mathcal{G}) = \bigcup_{v \in V_\mathcal{G}} LM(v),$$

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

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

The landmark table LM initially maps each node to an over-approximation of the landmarks reachable from it, and this table is updated until convergence. The procedure terminates after at most $|V|$ iterations and runs in $O(|V| + |E|)$ (Keyder, Richter, and Helmert 2010). The landmarks to the problem are exactly the union set of landmarks in $V_\mathcal{G}$.

We illustrate this with a simple HTN planning problem $\mathsf{Problem}_1$ with the following components: T is a compound task $T \in C$ that appears in the initial task network $T \in tn_I$, methods m_1,$m_2 \in M$ decomposes T. There are two primitive actions a,$b \in A$, and facts x,$y \in F$, where $x \in \text{pre}(a)$, $y \in \text{add}(a)$, and $y \in \text{pre}(b)$. The initial state is given by $\{x\} \subseteq s_i$, and the goal is empty $\mathcal{G} = \emptyset$, meaning the problem solution depends only on successfully decomposing T.

Figure 1 illustrates the BU graph for $\mathsf{Problem}_1$: circles denote OR-nodes; squares denote AND-nodes. The landmark set computed at the goal is $LM^{bu}(V_\mathcal{G}) = \{T, x, a, y, b\}$. In this example, the BU graph misses a method landmark m_1,

$$\begin{tikzcd}
 & T & \\
m_1 \arrow[ur] & & m_2 \arrow[ul] \\
a \arrow[u] \arrow[drr] & & b \arrow[u] \arrow[ull] \\
x \arrow[u] & & y \arrow[u]
\end{tikzcd}$$

$$LM(V_\mathcal{G})^{bu} = \{T, x, a, y, b\}$$

$$\begin{aligned}
LM^{bu}(T) &= \{T, x, a, y, b\} \\
LM^{bu}(m_1) &= \{x, a, y, b, m_1\} \\
LM^{bu}(m_2) &= \{x, a, y, b, m_2\} \\
LM^{bu}(b) &= \{x, a, y, b\} \\
LM^{bu}(y) &= \{x, a, y\} \\
LM^{bu}(a) &= \{x, a\} \\
LM^{bu}(x) &= \{x\}
\end{aligned}$$

Figure 1: BU graph for $\mathsf{Problem}_1$.

which is easily detectable upon human inspection: since a is a landmark and m_1 is the only method that introduces a through decomposition, m_1 must also be a landmark.

Figure 1 also illustrates the relaxations inherent to the BU graphs. (1) The only successors of primitive tasks are their add effects set, so the problem is *Delete relaxed.* (2) For justified methods, it suffices that subtasks are reached without considering any ordering constraints. Together (1) and (2) characterize the *Delete- Ordering Free relaxation* (DOF) (Höller, Bercher, and Behnke 2020). (3) For a justified primitive task a it suffices to have its predecessors $\text{pred}(a)$ fulfilled. This means primitive tasks don't need to be reachable from the task hierarchy; they can be included to justify other tasks. This HTN relaxation is called *Task Insertion* (TI) (Geier and Bercher 2011).

The main properties that Höller and Bercher (2021) show for landmark computation using BU are the following:

- *Polynomial complexity.* The BU graph encodes a relaxed HTN planning problem under DOF+TI semantics, for which plan existence can be decided in polynomial time. As shown by Höller and Bercher (2021, Thm. 1), determining whether a given HTN element is a landmark lies in the co-class of the plan existence problem. Since both plan existence and its complement can be decided in polynomial time under this relaxation, landmark identification in P is also solvable in polynomial time.

- *Sound Landmarks.* For a DOF+TI HTN problem P, let dt be a decomposition tree whose yield is a solution tn for P. Then, exist a justification of $V_\mathcal{G}$ representing dt and tn; and since LM stores the intersection over *all* justifications, every node it returns is guaranteed to occur in every solution (Höller and Bercher 2021, Thm. 2).

- *Incompleteness under DOF.* For a DOF+TI HTN problem P, building G^{bu} ignores delete effects and ordering constraints and, more subtly, admits *task insertion*. Consequently, exist DOF-relaxed problems with landmarks that the BU procedure never finds (Höller and Bercher 2021, Thm. 3).

3 Filling the Blanks with the Top-down Graph

This section introduces a novel AND/OR graph structure called the *Top-down Graph* (TD) to mitigate dependency relations missed due to Task Insertion. The TD graph enforces landmarks identified by the BU graph to be hierarchically justified.

Intuition Top-down Graph. Whereas the landmarks in the BU graph start at actions and flows *upwards*, the landmarks in the TD graph start at compound tasks and flows *downwards*. Each compound task propagates all its landmarks to the methods that refine it, and methods propagate some of their landmarks to their subtasks. The landmarks of an action include all common landmarks of all methods that produce it. To implement this "intersection" over methods that produce the same action, we add an OR-node, called a *merge node*. We create a *merge node* for each action, the predecessors of the *merge node* are all the methods that produce the action; and the successor of the *merge node* is the action. The connections between actions and facts remains the same. Therefore, the top-down graph represents dependencies that the upward view could not capture. Since we want to fill gaps left by the BU propagation, the landmark set already discovered by the BU procedure becomes the goal set of the new graph.

Definition 3.1 (Top-down Graph)**.** Let $P = (s_0, \mathcal{G}, tn_I, F, A, C, M)$ be an HTN task and LM^{bu} the landmark table computed on the BU graph G^{bu}. Let $X = \{x_a \mid a \in A\}$ be the set of *merge nodes* and $\rho \colon A \to X$ the function that maps actions $a \in A$ to merge nodes $x_a \in X$. The TD graph is $G^{td} = (V', E')$ with

$$V'_{\text{and}} = M \cup A, \qquad V'_{\text{or}} = (C \setminus tn_I) \cup (F \setminus s_0) \cup X,$$
$$V'_I = s_0 \cup tn_I, \qquad V'_{\mathcal{G}} = LM^{\text{bu}}(V_{\mathcal{G}}),$$

and edges

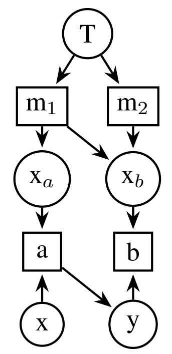
$$\begin{aligned} E' = \{&(c, m) \mid m = (c, tn) \in M\} \\ &\cup \{(m, x_a) \mid m \in M,\ t \in T_m,\ \alpha_m(t) = a; \rho(a) = x_a\} \\ &\cup \{(m, c') \mid m \in M,\ t \in T_m,\ \alpha_m(t) = c'\} \\ &\cup \{(x_a, a) \mid a \in A; \rho(a) = x_a\} \\ &\cup \{(a, f) \mid a \in A,\ f \in \text{add}(a)\} \\ &\cup \{(f, a) \mid a \in A,\ f \in \text{pre}(a)\}. \end{aligned}$$

Landmark propagation. The landmark-table update rules are identical to those of Definition 2.5 except that merge nodes are discarded during goal set updates.

Definition 3.2 (Goal-set update)**.**

$$LM^{\text{td}}(V'_{\mathcal{G}}) \ = \ \bigcup_{v \in V'_{\mathcal{G}}} LM^{\text{td}}(v) \setminus X.$$

Figure 2 depicts a TD graph G^{td} for PROBLEM$_1$. The node a is a landmark detected by the BU graph (Figure 1), therefore a belongs to $V'_{\mathcal{G}}$. In the TD graph, m_1 must appear in the justification of a. Consequently, m_1 is added to LM^{td}



$$LM^{td}(V'_{\mathcal{G}}) = \{T, m_1, x, a, y, b\}$$

$$\begin{aligned} LM^{td}(T) &= \{T\} \\ LM^{td}(m_1) &= \{T, m_1\} \\ LM^{td}(m_2) &= \{T, m_2\} \\ LM^{td}(x_a) &= \{T, m_1, x_a\} \\ LM^{td}(a) &= \{T, x, a, x_a, m_1\} \\ LM^{td}(y) &= \{T, x, y, a, x_a, m_1\} \\ LM^{td}(x_b) &= \{T, x_b\} \\ LM^{td}(b) &= \{T, x, y, a, x_a, m_1, b, x_b\} \\ LM^{td}(x) &= \{x\} \end{aligned}$$

Figure 2: TD graph for PROBLEM$_1$.

as a new method landmark. Note that the merge node x_b ensures that only the intersection of the landmarks of methods m_1 and m_2 are propagated to the action node b.

The number of edges and nodes in BU and TD is the same, except for construction related to action nodes. In TD, action nodes have one predecessor besides its fact preconditions, the merge node, and the merge node inherits all the predecessor edges of methods of the action node. Therefore, the TD graph has more $|A|$ nodes and $|A|$ edges. Since the equations converge in polynomial time (Keyder, Richter, and Helmert 2010), the overall computation of LM^{td} is still polynomial in the size of P.

We discuss top-down graph properties in three steps. (1) We show that every valid solution to an HTN DOF problem induces a TD justification; hence, any vertex contained in *all* such justifications is a landmark. (2) We compare the landmark set returned by LM^{td} with the BU landmarks LM^{bu}. (3) We observe that LM^{td} is still *incomplete* under the DOF relaxation, paralleling the result of Höller and Bercher (2021) for BU graphs G^{bu}.

First, observe that every compound task appears *once* in a top-down graph, and thus in a top-down justification. Each compound task can simultaneously justify *all* methods that refine it (akin to task sharing Alford et al. 2016); admitting task name duplicates would only create cycles without contributing to the set of solutions.

Lemma 3.1. Let g be a valid decomposition tree whose yield is a solution tn for the DOF problem P. There exists a justification J_{td} for $V'_{\mathcal{G}}$ such that J_{td} and g share exactly the same task names, method labels and facts.

Sketch. We build the justification $J_{td} = (V_{td}, E_{td})$ by a single depth–first traversal of the decomposition tree $g = (T, E, \prec, \alpha, \beta)$:

(C) *Tasks.* Whenever a task $\alpha(t) = A \cup C$ is encountered for the *first* time, add (or revisit) vertex $\alpha(t) \in V_{td}$.

(M) *Methods.* At an inner node t with $\beta(t) = m$ add m and the edge $(\alpha(t), m)$.

(S) *Subtasks of* m. For each subtask u of m:

(S1) if $\alpha(u) = c'$ is compound and c' is not justified yet add (m, c');

(S2) if $\alpha(u) = a \in A$ let x_a be the merge node of a, insert (m, x_a), (x_a, a)

Because tn is a delete-relaxed solution there is a justification for the part of the graph representing the state transition system (Keyder, Richter, and Helmert 2010).

Why J_{td} satisfies Definition 2.4.

1. *Goal inclusion.* Every vertex in $V'_{\mathcal{G}} = LM^{\text{bu}}(V_{\mathcal{G}})$ is a landmark, hence occurs either in g (task or method) or in the state sequence induced by its yield tn (facts). All of those vertices are added by the procedure.
2. *AND-nodes justified.* Each method m has its compound task as its single predecessor (step M). Each action a has its merge node x_a and all precondition facts $f \in \text{pre}(a)$ as predecessors since a is a task in the solution tn.
3. *OR-nodes justified.* A compound task c gains at most one incoming edge (m, c) from the method chosen in g (step S1). Merge nodes x_a obtain at least one incoming edge (m, x_a) from a method that actually produces a (step S2). Because tn is a solution, each fact $f \notin s_0$ is reached by some producer action.
4. *No cyclic dependencies.* A justification J^{td} won't have cyclic justifications for two reasons. (1) each compound task c receives one incoming edge, except for the initial vertices $V'_I = s_0 \cup tn_I$, which doesn't require incoming edges at all, (2) each node visited when traversing g had all their ancestors justified by the hierarchy (including itself).

hence J_{td} is a valid justification for $V'_{\mathcal{G}}$.

□

Theorem 3.1 (Soundness)**.** Every vertex in $LM^{\text{td}}(V'_{\mathcal{G}})$ is a landmark for the underlying planning problem.

Sketch. Lemma 3.1 maps every decomposition tree g and solution tn to a justification; Definition 2.5 computes the same set as the intersection of all such justifications. Hence, every element of $LM^{\text{td}}(V'_{\mathcal{G}})$ appears in every solution. □

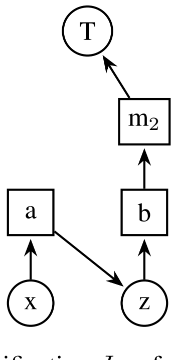


Figure 3: Justification J_{bu_1} for PROBLEM$_1$.

Not every BU justification J_{bu} can be mapped into a TD justification. Figure 3 exemplifies one justification J_{bu_1} for PROBLEM$_1$, in which action a is justified solely by its fact preconditions. In contrast, the TD graph construction contains a merge node x_a that connects a to the methods producing it. Since m_1 is the only method that introduces a, the justification for a must include m_1. Thus, m_1 appears in LM^{td} but not in LM^{bu}, showing that $LM^{\text{td}}(V'_{\mathcal{G}}) \nsubseteq LM^{\text{bu}}(V_{\mathcal{G}})$ in this case.

Theorem 3.2 (DOMINANCE)**.** $LM^{\text{bu}}(V_{\mathcal{G}}) \subseteq LM^{\text{td}}(V'_{\mathcal{G}})$.

Sketch. The TD graph G^{td} is initialized with the landmark set $LM^{\text{bu}}(V_{\mathcal{G}})$. Therefore, every landmark $v \in LM^{\text{bu}}(V_{\mathcal{G}})$ is justified by construction. By soundness, all elements of $LM^{\text{td}}(V'_{\mathcal{G}})$ must appear in every DOF solution. Thus, $LM^{\text{bu}}(V_{\mathcal{G}}) \subseteq LM^{\text{td}}(V'_{\mathcal{G}})$. The inclusion is strict in *some* problems. For example, in PROBLEM$_1$ (Fig. 2). □

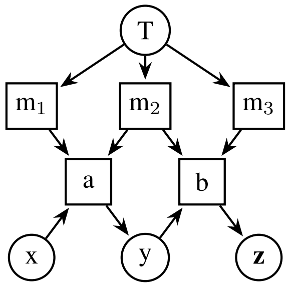


Figure 4: TD graph for PROBLEM$_2$. Method m_2 is a landmark in every DOF solution but is missed by LM^{td}.

Dominance does not necessarily imply completeness:

Theorem 3.3 (INCOMPLETENESS)**.** $LM^{\text{td}}(V'_{\mathcal{G}})$ is incomplete under the DOF relaxation.

Proof. In PROBLEM$_2$ (Fig. 4) fact x is in s_0 and z in goal. Fact z is achievable only via action b; b requires y, which in turn is produced by action a. The *only* hierarchical path to $\{a, b\}$ begins with method m_2; hence m_2 is a landmark. Yet the sub-graph $J_{td} = (\{T, m_1, m_3, a, b, x, y, z\}, E^J_{td})$ satisfies Definition 2.4 without containing m_2, so $m_2 \notin LM^{\text{td}}(V'_{\mathcal{G}})$. □

The example illustrates the incompleteness of TD under DOF relaxation. Because a single compound task can justify any of its methods, the TD graph may overlook some landmarks.

4 Bidirectional Landmarks

The TD graph adds *hierarchical dependencies* omitted by the BU graph by enforcing every landmark to be rooted in a valid decomposition of the initial task network tn_I. However, it does not replace the BU approach: each direction reveals information the other misses. Therefore, we combine the two tables in an inexpensive fix-point that alternates between them until no new landmarks are found.

Figure 5 shows PROBLEM$_3$, which extends PROBLEM$_1$ with an additional compound task S yielding to an arbitrary sized subgraph Δ. Computing landmarks for PROBLEM$_3$ in BU graph G^{bu} and TD graph G^{td} initially reproduce the landmarks of PROBLEM$_1$: $\{x, a, z, b, T\}$ and

Domain	Fact Landmarks			Task Landmarks			Method Landmarks		
	LM^{bu}	LM^{td}	$\mathcal{L}^{\text{bid}}$	LM^{bu}	LM^{td}	$\mathcal{L}^{\text{bid}}$	LM^{bu}	LM^{td}	$\mathcal{L}^{\text{bid}}$
Barman-BDI (20)	1,007	1,007	1,007	226	651	651	0	0	0
Blocksworld-GTOHP (30)	16,039	16,039	16,066	12,268	16,157	24,119	0	4,116	4,116
Blocksworld-HPDDL (30)	29,923	29,923	36,183	17,976	23,975	35,869	30	12,259	12,259
Depots (30)	7,764	7,764	7,764	2,621	2,844	2,844	0	0	0
Factories-simple (20)	323	323	343	303	488	508	0	20	20
Monroe-Fully-Obs. (20)	347	347	350	154	423	534	0	248	248
Monroe-Partially-Obs. (20)	315	315	320	133	380	474	0	209	209
Multiarm-Blocksworld (74)	15,728	15,728	15,739	4,401	8,253	8,277	326	346	346
Robot (20)	2,261	2,261	2,261	1,576	2,197	2,749	20	2,105	2,105
Satellite-GTOHP (17)	1,046	1,046	1,049	1,026	1,035	1,035	1	3	3
Transport (40)	1,147	1,147	1,147	1,146	1,201	1,203	0	49	49
Woodworking (7)	120	120	121	37	42	47	3	5	5
TOTAL	76,020	76,020	**82,350**	41,867	57,646	**78,310**	380	**19,360**	**19,360**

Table 1: Comparison of the number of landmarks generated by BU, TD, and BID landmarks.

$\{x, a, z, b, m_1, T\}$. However, the entry $LM^{\text{bu}}(m_1)$ contains any landmarks that appear in $LM^{bu}(S)$ which are missed by the landmarks found by TD. This means that the landmark set could be further expanded with $LM^{\text{bu}}(m_1)$. These additional landmarks, however, are not necessarily hierarchically justified, since they originate from the BU graph. Each such landmark can then be refined by consulting the TD table once more to uncover any hierarchical dependencies that were overlooked.

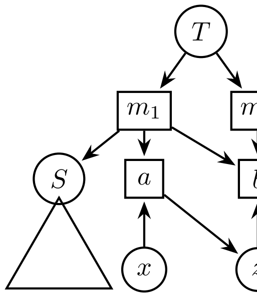


Figure 5: An illustration representing the HTN planning problem PROBLEM$_3$. The problem is similar to PROBLEM$_1$ besides the additional compound task S connected to a subtree of arbitrary size Δ.

Algorithm 1 formalizes this intuition. Starting from the BU landmarks of the whole task, it repeatedly *expands* the current set through the landmark tables of both graphs until a fix-point is reached. The result is a single landmark *set* $\mathcal{L}^{\text{bid}}$, not a table.

Algorithm 1: Bidirectional Landmarks

```
1: $\mathcal{L}^{\text{bid}} \leftarrow LM^{\text{bu}}(V_{\mathcal{G}})$
2: **while** $\mathcal{L}^{\text{bid}}$ changes **do**
3:     $\mathcal{L}^{\text{bid}} \leftarrow \bigcup_{l \in \mathcal{L}^{\text{bid}}} LM^{\text{bu}}(l) \cup LM^{\text{td}}(l)$
4: **end while**
```

We then obtain the following properties regarding bidirectional landmarks:

Corollary 4.1 (SOUNDNESS). $\mathcal{L}^{\text{bid}}$ contains only landmarks of the original problem P.

Proof. Both LM^{bu} (Höller and Bercher 2021, Thm. 2) and LM^{td} (Theorem 3.1) are sound. Algorithm 1 combines subsets of these tables for components l that are guaranteed landmarks. ☐

Corollary 4.2 (DOMINANCE). $LM^{\text{td}}(V_{\mathcal{G}}) \subseteq \mathcal{L}^{\text{bid}}$.

Proof. $\mathcal{L}^{\text{bid}}$ is initialized with $LM^{\text{bu}}(V_{\mathcal{G}})$. Each iteration augments $\mathcal{L}^{\text{bid}}$ by $LM^{\text{td}}(l)$ or $LM^{\text{bu}}(l)$, so every entry of LM^{td} reachable is eventually absorbed, in particular $LM^{\text{td}}(V'_{\mathcal{G}})$. The inclusion is strict in *some* problems. For example, in PROBLEM$_3$ ☐

Corollary 4.3 (INCOMPLETENESS). $\mathcal{L}^{\text{bid}}$ is incomplete under the DOF relaxation.

Proof. The counter-example from Theorem 3.3 also applies, since method m_2 is missing from *both* LM^{bu} and LM^{td}, hence Algorithm 1 can never add it. ☐

5 Experiments

We evaluated our approach using total-order benchmarks from the 2023 IPC[2] and conducted experiments on an Intel i7-14700F CPU, using a 30-minute timeout and an 8 GB RAM limit. We implemented our techniques within the PANDA planner (Höller et al. 2021). Out of the 22 domains in the total-order track, we report results for 12 domains where LM^{td} generates more landmarks than LM^{bu}. Both strategies identified the same landmarks in the other domains, providing no additional insights.

Landmark Generation: Table 1 provides a comparison of three landmark generation strategies: landmark generated by the BU graph (LM^{bu}), landmark generated by the top-down graph (LM^{td}), and bidirectional landmarks ($\mathcal{L}^{\text{bid}}$). The increase in the time required to compute the LM^{td} and $\mathcal{L}^{\text{bid}}$ landmarks is minimal. The average total time to generate

[2]https://ipc2023-htn.github.io

landmarks for all problems in Table 1 for LM^{bu}, LM^{td}, and $\mathcal{L}^{\text{bid}}$ is 4.23 seconds, 4.94 seconds, and 4.95 seconds, respectively. Note that the three techniques complete the landmark generation step for all tasks of the domains we report.

DOMAIN	**2WA***		**5WA***	
	bid	*bu*	*bid*	*bu*
Barman-BDI	**7**	**7**	**7**	**7**
Blocks.-GTOHP	**17**	15	**30**	28
Blocks.-HPDDL	**26**	14	**26**	**26**
Depots	**18**	**18**	**23**	**23**
Factories-simple	**5**	**5**	**5**	**5**
Monroe-Fully-Obs.	**15**	14	**17**	16
Monroe-Partially-Obs.	**10**	3	5	**8**
Multiarm-Blocks.	**16**	13	**66**	62
Robot	**11**	**11**	**16**	11
Satellite-GTOHP	**6**	**6**	**12**	**12**
Transport	**10**	4	**10**	9
Woodworking	**7**	**7**	**7**	**7**
TOTAL	**148**	117	**224**	214

Table 2: Coverage of Weighted A* with weights of 2 (2WA*) and 5 (5WA*) for landmark count, for $\mathcal{L}^{\text{bid}}$ and LM^{bu} (denoted by *bid* and *bu* respectively).

Table 1 shows that the LM^{td} and $\mathcal{L}^{\text{bid}}$ methods generate more landmarks than LM^{bu}. LM^{td} identified 37% more task landmarks than LM^{bu}, while $\mathcal{L}^{\text{bid}}$ found 87% more task landmarks than LM^{bu}. Additionally, the number of facts landmarks increased by 8%, mainly due to the differences in landmarks generated for the *Blocksworld-HPPDL* domain. LM^{bu} detects a low number of method landmarks in partial-order domains (Höller and Bercher 2021), and this limitation persists in total-order domains, where it identifies landmarks in only five domains. LM^{td} improves on this limitation by significantly increasing the detection of method landmarks, identifying landmarks for ten domains.

Landmark-guided Search: Although this paper primarily focuses on landmark generation, we also evaluated the impact of our bidirectional landmarks within a search framework using the *landmark count* heuristic (lm-count). In our experiments, we generate landmarks during preprocessing, and the heuristic value is the number of unreached landmarks. Given the minimal overhead in computing bidirectional landmarks compared to the top-down approach (primarily table lookups), we directly compared the coverage achieved by the bidirectional and bottom-up methods.

We conducted experiments using the Weighted A* algorithm with 2 (2WA*) and 5 (5WA*) weights. As shown in Table 2, the additional landmarks identified by the bidirectional method generally enhance coverage. Figure 6 and 7 illustrates the impact on node expansions across various domains, each data point represents a problem instance, with axes comparing bottom-up (BU) and bidirectional (BID) landmark generation techniques using the landmark count heuristic. In many cases, bidirectional landmarks result in fewer expanded nodes than the bottom-up approach. Although there are a few instances where LM^{bu} shows a slight

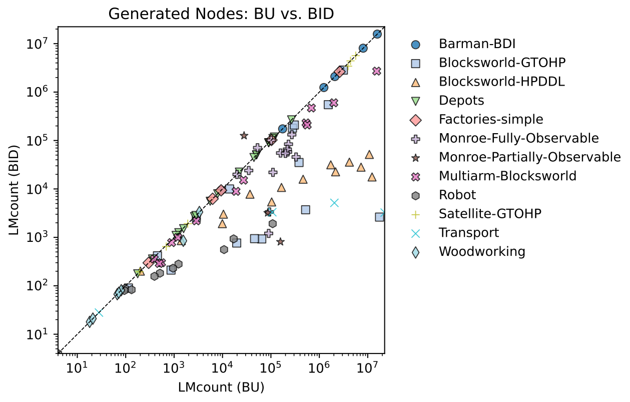


Figure 6: Comparison of the number of nodes expanded for solved problems on a logarithmic scale using Weighted A* with weights of 2 (2WA*).

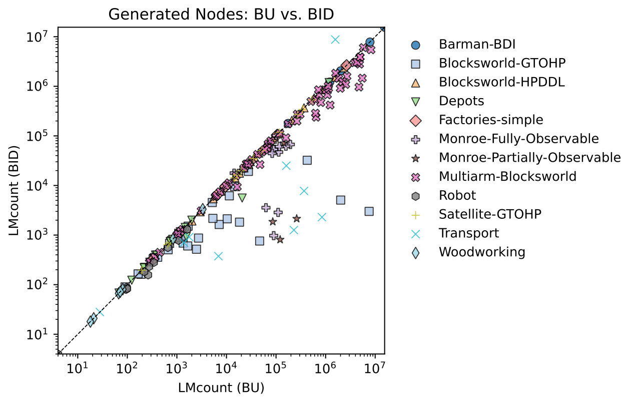


Figure 7: Comparison of the number of nodes expanded for solved problems on a logarithmic scale using Weighted A* with weights of 5 (5WA*).

advantage, the overall trend favors $\mathcal{L}^{\text{bid}}$ in reducing node expansions, reinforcing its value in heuristic-guided search.

6 Discussion

Research on landmarks for Hierarchical Task Network planning has progressed through three main stages. The earliest accounts focused on *mandatory tasks*, i.e., task names that occur in every decomposition of the initial task network (Bercher, Keen, and Biundo 2014). Although inexpensive to compute, this view ignores the executability of primitive tasks and misses state-dependent requirements. Höller and Bercher (2021) address this limitation by adapting the AND/OR graph of Keyder, Richter, and Helmert (2010) to HTN planning. Their AND/OR graph combines hierarchical structure with delete-relaxed state transitions, yielding sound but incomplete sets of fact, task, and method landmarks under the Delete- and Ordering-Free (DOF) relaxation. The third stage leverages the classical planning heuristic *LM-Cut* (Helmert and Domshlak 2009) to find *disjunctive landmarks*. The *Relaxed Composition* (RC) translation converts a DOF with Task Insertion HTN problem into a delete-relaxed classical problem so that LM-Cut and any other classical heuristic can be applied directly (Höller et al. 2019). This version of LM-Cut landmarks was used by the *PANDADealer* planner (Olz, Höller, and Bercher 2024), the winner of the IPC-2023 total-order.

This paper revisited landmark generation for Hierarchical Task Network planning and introduced two complementary contributions. First, the *top-down graph* complements the bottom-up encoding by enforcing that every landmark is justified along at least one valid decomposition of the initial task network. We prove that the resulting landmark table is sound, polynomial to compute, and dominates the bottom-up set while remaining incomplete under the same DOF relaxation. Second, an iterative *bidirectional* procedure combines both graphs until a fix-point is reached, yielding the most extensive landmark set currently obtainable in polynomial time. Empirical results on the IPC-2023 total-order benchmarks confirm the practical value of our approach. Bidirectional propagation increases the number of task landmarks by 87%.

Acknowledgments

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil* (CAPES) – Finance Code 001. We acknowledge support from FAPERGS with project 21/2551-0000741-9. Victor was additionally funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – GRK 2853/1 “Neuroexplicit Models of Language, Vision, and Action” - project number 471607914.

References

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016. Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing. In *IJCAI*.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid Planning Heuristics based on Task Decomposition Graphs. In *SoCS*.

Elkawkagy, M.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2012. Improving Hierarchical Planning Performance by the Use of Landmarks. In *AAAI*.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. HTN Planning: Complexity and Expressivity. In *AAAI*.

Geier, T.; and Bercher, P. 2011. On the Decidability of HTN Planning with Task Insertion. In *IJCAI*.

Helmert, M.; and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *JAIR*.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *KI-Künstliche Intelligenz*.

Höller, D.; and Bercher, P. 2021. Landmark Generation in HTN Planning. In *AAAI*.

Höller, D.; Bercher, P.; and Behnke, G. 2020. Delete-and Ordering-Relaxation Heuristics for HTN Planning. In *IJCAI*.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. *IJCAI*.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *ECAI*.

Olz, C.; Höller, D.; and Bercher, P. 2024. The PANDADealer System for Totally Ordered HTN Planning in the 2023 IPC. *IPC 2023*.