Incorporating Domain-Independent Planning Heuristics in Hierarchical Planning

Vikas Shivashankar
Knexus Research Corporation
National Harbor, MD
vikas.shivashankar@knexusresearch.com

Ron Alford
MITRE
McLean, VA
ralford@mitre.org

David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory
Washington, DC
david.aha@nrl.navy.mil

Abstract

Heuristics serve as a powerful tool in modern domain-independent planning (DIP) systems by providing critical guidance during the search for high-quality solutions. However, they have not been broadly used with hierarchical planning techniques, which are more expressive and tend to scale better in complex domains by exploiting additional domain-specific knowledge. Complicating matters, we show that for Hierarchical Goal Network (HGN) planning, a goal-based hierarchical planning formalism that we focus on in this paper, any poly-time heuristic that is derived from a delete-relaxation DIP heuristic has to make some relaxation of the hierarchical semantics. To address this, we present a principled framework for incorporating DIP heuristics into HGN planning using a simple relaxation of the HGN semantics we call Hierarchy-Relaxation. This framework allows for computing heuristic estimates of HGN problems using any DIP heuristic in an admissibility-preserving manner. We demonstrate the feasibility of this approach by using the LMCut heuristic to guide an optimal HGN planner. Our empirical results with three benchmark domains demonstrate that simultaneously leveraging hierarchical knowledge and heuristic guidance substantially improves planning performance.

Introduction

The primary goal of AI planning research has been to develop planning systems that can efficiently generate high-quality plans. Search heuristics serve as a powerful tool in achieving this by guiding the search algorithms towards high-quality solutions. While domain-independent planning (DIP) algorithms have greatly benefited in this regard by the development of informative search heuristics that permit efficient optimal/anytime plan generation, this is less true in hierarchical planning formalisms, where planners typically resort to brute-force search algorithms and rely on sophisticated domain-specific knowledge to control the search.

The main reason for the capability gap is that while DIP heuristics are typically developed by devising a principled relaxation of the original planning problem that is easier to solve (usually in \( P \)), analogous relaxations do not work in hierarchical planning formalisms. For example, Alford et al. (2014) showed that when delete-relaxation, a problem relaxation technique in DIP that is the source of many powerful heuristics, is applied to Hierarchical Task Network (HTN) planning, the resulting decision problem is NP-hard. Therefore, if we want to leverage DIP problem relaxation techniques to develop efficiently computable heuristics for hierarchical planning formalisms, we need to not only relax the problem, but also relax the hierarchical semantics.

In this paper, we focus on the problem of incorporating DIP heuristics into another hierarchical planning formalism called Hierarchical Goal Network (HGN) planning (Shivashankar et al. 2012) via a systematic relaxation of its semantics. We chose HGN planning because it is possible to represent hierarchical knowledge similar to HTN planning, and at the same time work with goals instead of tasks, which makes it easier to incorporate techniques from DIP. In particular, our main contributions are as follows:

Hardness of Delete-Relaxed HGN Planning. We show that, like HTN planning, delete-relaxed HGN planning is also NP-hard, meaning that we need to relax HGN semantics to develop efficient HGN analogues of arbitrary DIP heuristics.

Hierarchy-Relaxed HGN Planning. Next, we propose one such relaxation called Hierarchy-Relaxation, and prove that Hierarchy-Relaxed HGN (HRHGN) Planning can be compiled into DIP. This provides a principled framework to incorporate any DIP heuristic \( h \) into HGN planning: given an HGN planning problem \( P \), we can hierarchy-relax it to the HRHGN problem \( P_{hr} \), compile \( P_{hr} \) into the DIP problem \( P_{dip} \) and use \( h \) to compute a heuristic estimate \( h(P_{dip}) \) for \( P \). We show that this framework is admissibility-preserving: admissible DIP heuristics will translate into admissible HGN heuristics.

Optimal HGN Planner Guided by the LMCut Heuristic. We demonstrate the feasibility of this approach by incorporating the admissible DIP heuristic LMCut (Helmert and Domshlak 2009) into an optimal HGN planning algorithm, which we call Hierarchically-Optimal Goal Decomposition Planner using LMCut (HOGL). Our experiments on three benchmark domains demonstrate that HOGL’s ability to simultaneously leverage hierarchical knowledge and DIP heuristics allows it to outperform state-of-the-art optimal DIP algorithms and blind search HGN planning algorithms. Furthermore, they also show that the use of LMCut, a state-of-the-art heuristic, allows
HOGL to outperform HOpGDP (Shivashankar et al. 2016), a recently proposed optimal HGN planner that uses a version of the h_L heuristic (Karpas and Domshlak 2009) customized for HGN planning.

**Preliminaries**

Here we detail the DIP model and formalize HGN planning, borrowing heavily from Shivashankar et al. (2016).

**Domain-Independent Planning**

We define a **DIP domain** $D_{dip}$ as a finite-state transition system in which each state $s$ is a finite set of ground atoms of a first-order language $L$, and each action $a$ is a ground instance of a planning operator $o$. A planning operator is a 4-tuple $o = (\text{head}(o), \text{pre}(o), \text{eff}(o), \text{cost}(o))$, where $\text{pre}(o)$ and $\text{eff}(o)$ are conjuncts of literals called $o$’s preconditions and effects, and $\text{head}(o)$ includes $o$’s name and argument list (a list of the variables in $\text{pre}(o)$ and $\text{eff}(o)$). $\text{cost}(o)$ represents the non-negative cost of applying operator $o$.

**Actions.** An action $a$ is applicable in a state $s$ if $s \models \text{pre}(a)$, in which case the resulting state is $\gamma(s, a) = (s - \text{eff}^-(a)) \cup \text{eff}^+(a)$, where $\text{eff}^-(a)$ and $\text{eff}^+(a)$ are the atoms and negated atoms, respectively, in $\text{eff}(a)$. A plan $\pi = (a_1, \ldots, a_n)$ is executable in $s$ if each $a_i$ is applicable in the state produced by $a_{i-1}$, and in this case $\gamma(s, \pi)$ is the state produced by executing $\pi$. If $\pi$ and $\pi'$ are plans or actions, then their concatenation is $\pi \circ \pi'$.

We define the cost of $\pi = (a_1, \ldots, a_n)$ as the sum of the costs of the actions in the plan, i.e. $\text{cost}(\pi) = \sum_{i \in \{1, \ldots, n\}} \text{cost}(a_i)$.

**Goal Networks and HGN Methods**

A goal network is a way to represent the objective of satisfying a partially ordered multiset of goals. Formally, it is a pair $gn = (T, \prec)$ such that:

- $T$ is a finite nonempty set of nodes;
- Each node $t \in T$ contains a goal $g_t$ that is a DNF (disjunctive normal form) formula over ground literals;
- $\prec$ is a partial order over $T$.

**HGN Methods.** An HGN method $m$ is a 4-tuple $(\text{head}(m), \text{goal}(m), \text{pre}(m), \text{network}(m))$ where the head $\text{head}(m)$ and preconditions $\text{pre}(m)$ are similar to those of a planning operator. $\text{goal}(m)$ is a conjunct of literals representing the goal $m$ can decompose. $\text{network}(m)$ is the goal network that $m$ decomposes into. By convention, $\text{network}(m)$ has a last node $t_g$ containing $\text{goal}(m)$ to ensure that $m$ accomplishes its own goal.

An action $a$ (or method instance $m$) is relevant to a goal formula $g$ if $\text{eff}(a)$ or $\text{post}(m)$, respectively, entails at least one literal in $g$ and does not entail the negation of any literal in $g$.

Whether a node has predecessors impacts the kinds of operations we allow. We refer to any node in a goal network $gn$ having no predecessors as an unconstrained node of $gn$, otherwise the node is constrained. We define the following operations over any goal network $gn = (T, \prec)$:

1. **Goal Release:** Let $t \in T$ be an unconstrained node. Then the removal of $t$ from $gn$, denoted by $gn - t$, results in the goal network $gn' = (T', \prec')$ where $T' = T \setminus \{t\}$ and $\prec'$ is the restriction of $\prec$ to $T'$.

2. **Method Application:** Let $t \in T$ be an unconstrained node. Also, let $m$ be a method applicable to $t$ with $\text{network}(m) = (T_m, \prec_m)$. Then the application of $m$ to $gn$ via $t$, denoted by $gn \circ_o m$, results in the goal network $gn' = (T', \prec')$ where $T' = T \cup T_m$ and $\prec' = \prec \cup \prec_m \cup \{(t_m, t) \mid m \in T_m\}$. Informally, this operation adds the elements of $\text{network}(m)$ to $gn$, preserving the order specified by $\text{subgoals}(m)$ and setting all nodes of $m$ as a predecessors of $t$.

**HGN Domains, Problems and Solutions**

An HGN domain is a pair $D = (D_{dip}, M)$ where $D_{dip}$ is a DIP domain and $M$ is a set of HGN methods.

An HGN planning problem is a triple $P = (D, s_0, gn_0)$, where $D$ is an HGN domain, $s_0$ is the initial state, and $gn_0 = (T, \prec)$ is the initial goal network.

**Definition 1** (Solutions to HGN Planning Problems). The set of solutions for $P$ is defined as follows:

**Base Case.** If $T$ is empty, the empty plan is a solution.

**In the following cases, let $t \in T$ be an unconstrained node.**

**Unconstrained Goal Satisfaction.** If $s_0 \models g_t$, then any solution for $P' = (D, s_0, gn_0 - t)$ is also a solution for $P$.

**Action Application.** If action $a$ is applicable in $s_0$ and $a$ is relevant to $g_t$, and $\pi$ is a solution for $P' = (D, \gamma(s_0, a), gn_0)$, then $a \circ \pi$ is a solution for $P$.

**Method Decomposition.** If $m$ is a method applicable in $s_0$ and relevant to $g_t$, then any solution to $P' = (D, s_0, gn_0 \circ_m m)$ is also a solution to $P$.

**Note** in Definition 1 that the HGN methods pose constraints on solution validity; even if a plan $\pi$ is executable in $s_0$ and accomplishes all the goals in $gn_0$ in the right order, it is not a valid solution unless it can be shown to be derived from the given methods. In this sense, HGN planning is similar to HTN planning.

Let us denote $S(P)$ as the set of solutions to an HGN planning problem $P$ as allowed by Definition 1. Then we can define what it means for a solution $\pi$ to be hierarchically optimal with respect to $P$ as follows:

**Definition 2** (Hierarchically Optimal Solutions). A solution $\pi^{h,*}$ is hierarchically optimal with respect to $P$ if $\pi^{h,*} = \arg\min_{\pi \in S(P)} \text{cost}(\pi)$.

**Limits to Deriving HGN Heuristics from DIP Problem Relaxations**

The decompositional structure of HGNs provides significant pruning power during search, and, ideally, we would incorporate this structure into HGN heuristics. Below, we show that the delete-relaxation technique (planning using only positive preconditions and effects) popularized by Hoffmann and Nebel (2001) is not sufficient to make HGN problems...
solvable in polynomial time. Thus, we cannot incorporate strict HGN semantics into our adaptations of DIP heuristics.

**Theorem 3.** Let HGN$^+$ be the class of ground (propositional) HGN problems with positive preconditions, goals, and effects. Then plan-existence for HGN$^+$ is NP-complete.

**Proof.** **Membership:** HGN problems can be translated to HTN problems in a plan preserving manner, and the translation does not require negative preconditions (Alford et al. 2016b). Since HTN planning with positive preconditions and effects is NP-complete, HGN$^+$ is in NP.

**Hardness:** Let $v_1, \ldots, v_n$ be the variables of a CNF-SAT formula $F = c_1 \land \ldots \land c_m$. Define an HGN problem $P = (D, s, g)$ as follows.

Let the initial state $s = \emptyset$ and goal $g = (\text{chosen } v_1)$.

For each $v_i$ ($i \leq n$), let $D$ have an action $(\text{set } v_i \text{ false})$ with the precondition $(\text{can\_choose\_false } v_i)$ and effect $(\text{is\_false } v_i) \land (\text{chosen } v_i)$ and an action with no preconditions that asserts $(\text{can\_choose\_false } v_i)$. For $i < n$, let $D$ also have a method with the goal $(\text{chosen } v_i)$ and the totally-ordered sequence of subgoals $\langle (\text{can\_choose\_false } v_i), (\text{is\_false } v_i), (\text{chosen } v_{i+1}) \rangle$. Let $D$ have analogous actions and methods for $(\text{set } v_i \text{ true})$. For $v_n$, both the true and false methods replace their last subgoal with $(\text{checked } c_1)$.

For each conjunction $c_k$ and negative literal $\neg v_i$ of $c_k$, there is an action $(\text{check } c_k)$ with precondition $(\text{can\_check\_false } c_k) \land (\text{is\_false } v_i)$ and effect $(\text{checked } c_k)$, as well as a precondition-less action that sets $(\text{can\_check } c_k)$.

For each $k < m$, there is a goal with method $(\text{checked } c_k)$ and a totally-ordered sequence of subgoals $\langle (\text{can\_check } c_k), (\text{checked } c_k), (\text{checked } c_{k+1}) \rangle$. For $c_m$, the method for $(\text{checked } c_m)$ omits the final task.

From the initial state, both the $(\text{set } v_1 x)$ actions and the $(\text{chosen } v_1)$ methods are relevant, but only the methods are applicable. After either method is applied, the $(\text{can\_choose\_x } v_1)$ subgoal is now the only unconstrained subgoal, and the $(\text{set } v_1 x)$ action can be applied. Once that goal is released, only the $(\text{set } v_1 x)$ action is relevant and applicable to $(\text{is\_x } v_1)$, which can then be released. Then the last subgoal $(\text{chosen } v_2)$ starts the process for the next variable. Once $v_1$ is chosen, the only unconstrained subgoal is $(\text{checked } c_1)$, and a similar process ensues to ensure that each $c_k$ has at least one true literal. After $c_m$ is checked, the entire subgoal graph can be released, and the chosen values for each $v_i$ form a witness to the solvability of $F$. Since solving $P$ implies a solution to $F$ and $P$ contains only positive effects, preconditions, and goals, plan-existence for HGN$^+$ is NP-hard. □

**Deriving HGN Planning Heuristics from Domain-Independent Planning Heuristics**

The previous section demonstrated the need for relaxations of HGN semantics to allow for efficient HGN analogues of existing DIP heuristics. We shall now define Hierarchy-Relaxation, one such relaxation of the HGN semantics.

**Hierarchy-Relaxed HGN (HRHGN) Planning**

Hierarchy-relaxation, as the name suggests, relaxes the semantics of HGN planning problems to allow any executable sequence of actions that achieve the goals in the goal network in the right order, whether or not they are consistent with the hierarchical knowledge.

More formally, the hierarchy-relaxation of an HGN problem $P = ((D_{\text{dip}}, M), s_0, g_{\text{dip}})$ removes the hierarchical decomposition knowledge (i.e. the set of methods $M$), resulting in the HRHGN problem $P_{hr} = (D_{\text{dip}}, s_0, g_{\text{dip}})$.

**Solutions to HRHGN Problems.** The set of solutions $S_{HR}(P_{hr})$ to an HRHGN problem $P_{hr}$ can be defined as the set of all action sequences that are (1) executable in $s_0$, and (2) achieve all the goals in $g_{\text{dip}}$ in the order specified.

Given the HGN problem $P$ and its hierarchy-relaxed version $P_{hr}$, it is easy to see that the set of solutions to $P$ $S(P) \subseteq S_{HR}(P_{hr})$, the intuition being that every solution to $P$ trivially satisfies the definition for $P_{hr}$, while there could be solutions to $P_{hr}$ that are not solutions to $P$ because they don’t obey the constraints imposed by the methods. This is equivalent to task-insertion from the HTN literature, which (in general) reduces computational complexity by allowing the planner to insert arbitrary tasks into the task network (Geier and Bercher 2011; Alford et al. 2014; Alford, Bercher, and Aha 2015).

Looking at the definition of HRHGN planning, it looks very similar to domain-independent planning, the only difference being the objective is not a single goal formula, but is instead a goal network. In fact, we can show that an HRHGN planning problem can be translated into a DIP problem having an “equivalent” set of solutions:

**Theorem 4.** Given an HRHGN problem $P_{hr} = (D_{\text{dip}}, s_0, g_{\text{dip}})$, we can construct a DIP problem $P'_{\text{dip}} = (D'_{\text{dip}}, s'_{\text{dip}}, g'_{\text{dip}})$ in polynomial time such that there exists a bijection between the solution sets for $P_{hr}$ and $P'_{\text{dip}}$ that matches solutions having identical costs.

**Proof Sketch.** We can construct the DIP problem $P'_{\text{dip}}$ as follows. We begin by creating a dummy predicate $\text{achieved}_{x}$ and a dummy action $\text{dummy}_{x}$ for each node $x$ in $g_{\text{dip}}$. The preconditions of $\text{dummy}_{x}$ ensure that every parent of $x$ in $g_{\text{dip}}$ has already been achieved, and that the goal $g_x$ in $x$ is satisfied. If these conditions are true, it inserts $\text{achieved}_{x}$ into the state. All dummy actions are of zero cost. More formally, the model of $\text{dummy}_{x}$ is:

- $\text{pre}(\text{dummy}_{x}) = \wedge_{x' \text{ is a parent of } x \text{ in } g_{\text{dip}}} \text{achieved}_{x'} \land g_x$
- $\text{eff}(\text{dummy}_{x}) = \text{achieved}_{x}$
- $\text{cost}(\text{dummy}_{x}) = 0$

$D'_{\text{dip}}$ is the original $D_{\text{dip}}$ combined with the new dummy predicates and actions. $s'_{\text{dip}}$ is identical to $s_0$. Finally, $g'_{\text{dip}} = \wedge_{x \in g_{\text{dip}}} \text{achieved}_{x}$. This represents the objective that all the goals in the goal network must be achieved; the requirement that they must be achieved in the correct order is encoded in the action models of the dummy actions. Note that this compilation can be done in $O(|gn|)$.
By virtue of the construction, it can be shown that every solution \( \pi \) to \( P_{hr} \) can be mapped to an equivalent solution \( \pi' \) to \( P'_{disp} \) which contains all of \( \pi \) and some additional dummy actions (which are of zero-cost, and hence don’t change plan cost). Furthermore, we can show that \( P'_{disp} \) does not have additional solutions that don’t map back to a solution to \( P_{hr} \). Hence the theorem follows.

An immediate consequence of the above theorem is that the optimal solution costs of \( P_{hr} \) and \( P'_{disp} \) must be equal:

**Corollary 5.** The costs of the optimal solutions to an HRHGN problem \( P_{hr} \) and the corresponding compiled DIP problem \( P_{disp} \) are equal.

**Heuristics for HGN Planning via HRHGN Planning**

Using Thm. 4, we can formulate a procedure \( h_{hgn}(P, h_{disp}) \) that can use an arbitrary DIP heuristic \( h_{disp} \) to compute heuristic estimates for a HGN planning problem \( P \):

1. **Input:** HGN planning problem \( P \) and DIP heuristic \( h_{disp} \).
2. Let \( P_{hr} \) be the hierarchy-relaxed version of \( P \), and \( P_{disp} \) the compilation of \( P_{hr} \) into DIP as specified in Thm. 4.
3. **Return** \( h_{disp}(P_{disp}) \).

If \( h_{disp} \) is an admissible heuristic, then we can show that \( h_{hgn} \) as constructed above is an admissible HGN heuristic:

**Theorem 6** (From admissible DIP to admissible HGN heuristics). Given an HGN problem \( P = ((D_{disp}, M), s_0, g_n) \) and an admissible DIP heuristic \( h_{disp}, h_{hgn}(P, h_{disp}) \) is an admissible cost estimate of \( P \).

**Proof Sketch.** Let \( OPT_P, OPT_{P_{hr}}, \) and \( OPT_{P_{disp}} \) be the optimal solution costs respectively of \( P \), its hierarchy-relaxation \( P_{hr} \) and the DIP compilation \( P_{disp} \) of \( P_{hr} \). We then know that \( OPT_{P_{hr}} \leq OPT_P \) due to the hierarchy-relaxation. Furthermore, from Corollary 5, we know that \( OPT_{P_{hr}} = OPT_{P_{disp}} \). Finally, we know from the admissibility of \( h_{disp} \) that \( h_{disp}(P_{disp}) \leq OPT_{P_{disp}} \). From these, we can show that \( h_{hgn}(P, h_{disp}) = h_{disp}(P_{disp}) \leq OPT_P \).

**Discussion**

HRHGN planning is a strong relaxation of HGN planning because it effectively ignores the hierarchical constraints from the HGN methods. Therefore, if a planning domain models its critical constraints in the methods, then the heuristic computation ignores them, thus leading to potentially uninformative heuristic estimates. Despite this, we believe that **Hierarchy-Relaxation** is a useful concept for the following reasons.

From a theoretical standpoint, Theorem 3 shows that some relaxation of the HGN semantics is necessary to efficiently compute heuristic estimates using DIP heuristics; we show that hierarchy-relaxation is a candidate relaxation for this. From a practical standpoint, hierarchy-relaxation helps advance the state-of-the-art in admissible heuristics for hierarchical planning. Not only does it trivially dominate blind search algorithms, but we can also show that it is a generalization of \( h_{HL} \) (Shivashankar et al. 2016), the only other non-trivial admissible heuristic for HGN planning that we are aware of. Shivashankar et al. proposed this admissible heuristic that extended the admissible landmark-based DIP heuristic \( h_L \) (Karpas and Domshlak 2009) to compute heuristic estimates for HGN planning problems. \( h_{HL} \) proceeds by extending DIP landmark generation techniques (in this case, LAMA’s (Richter and Westphal 2010)) to compute sound landmark graphs for HGN planning problems. \( h_L \) takes these landmark graphs as input to compute admissible HGN estimates. It can be shown that \( h_{HL} \) is a specific instantiation of our HRHGN framework using \( h_L \) as the DIP heuristic. In our experimental study, we compare HOGL, our planner, against HOGL, an optimal HGN planner that uses \( h_{HL} \) for search guidance.

**Experiments**

Theorem 6 allows us to take any admissible DIP heuristic and transform it into the corresponding admissible HGN heuristic that can then be used to guide optimal HGN planners. To test the feasibility of this approach, we implemented HOGL, an optimal HGN planner that executes an \( A^* \) search in accordance with Definition 1. It computes heuristic estimates using our framework instantiated with the LMCut heuristic (Helmert and Domshlak 2009).

Our main experimental hypotheses are:

**H1.** Due to its ability to simultaneously leverage state-of-the-art DIP heuristics and sophisticated hierarchical knowledge, HOGL can outperform both optimal DIP algorithms and blind HGN search algorithms, which respectively do not leverage hierarchical knowledge and search heuristics. To test this, we compared HOGL to the optimal classical planner \( A^*-LMCut^2 \) and \( HOGL_{blind} \), a blind version of HOGL that uses the trivial heuristic \( h = 0 \).

**H2.** The use of LMCut, one of the most informative DIP heuristics, in HOGL results in fewer node expansions than other optimal heuristic HGN planners. To test this hypothesis, we compared HOGL to HOGL, which uses the \( h_L \) heuristic (Karpas and Domshlak 2009), one that Helmert and Domshlak (2009) have shown to be less informative than LMCut.

For each domain, we randomly generated 25 problem instances per problem size. We ran all problems on a Xeon E5-2639 with a per problem limit of 8 GB of RAM and 25 minutes of planning time.

<table>
<thead>
<tr>
<th></th>
<th>( A^*-LMCut )</th>
<th>( HOGL_{blind} )</th>
<th>HOGL</th>
<th>( HOGL_{blind} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_{log} (175) )</td>
<td>44</td>
<td>66</td>
<td>140</td>
<td>163</td>
</tr>
<tr>
<td>( bw (225) )</td>
<td>118</td>
<td>208</td>
<td>216</td>
<td>219</td>
</tr>
<tr>
<td>( depots (175) )</td>
<td>79</td>
<td>131</td>
<td>148</td>
<td>155</td>
</tr>
<tr>
<td>( total (575) )</td>
<td>241</td>
<td>405</td>
<td>504</td>
<td>537</td>
</tr>
</tbody>
</table>

Table 1: Number of problems solved by the various planners.

Table 1 provides an overview of our study. As shown, HOGL solves more problems than the other planners across
all three domains, solving 537 out of a possible 575 problems. Both A*-LMCut and HOGL-blind solve far fewer problems than HOGL; in fact, the only other planner that is somewhat competitive with HOGL is HOpGDP, which solves 504 problems within the time/memory limit.

Results. Figure 2 shows the plots for the three test domains. Note that all plots are log-linear. We evaluated the planners both in terms of planning times (indicating the planning performance) and number of node expansions (indicating the search guidance). Data points were discarded for a problem size if the planner did not solve all of the corresponding problem instances within the time/memory limit. Across all three domains, A*-LMCut performs the worst according to both measures due to lack of domain-specific knowledge. HOGL-blind performs slightly better in all three domains, even though it does a blind A* search, due to the additional knowledge it can leverage. However, its performance is heavily dependent on the quality of the knowledge. For example, it performs poorly in Logistics due to the looseness of the hierarchical knowledge with respect to the multiple loads/unloads. In contrast, it solves lots of problems in Blocks-World since there is sophisticated knowledge available to control the search.

HOGl and HOpGDP both perform better than the first two planners due to their ability to exploit both DIP heuristic guidance as well as domain-specific knowledge, thus supporting Hypothesis H1. HOGL also consistently outperforms HOpGDP due to the use of the more-informative LMCut heuristic; this can be seen in terms of lower planning times and lower node expansions across all three domains. This supports Hypothesis H2.

Takeaways from the Experimental Study

The main takeaway from this study is that due to its ability to leverage both search heuristics and hierarchical knowledge, HOGL is the top-performing planner in this experiment across the test domains. This result can be viewed in two different ways. First, it provides a generic mechanism to incorporate search heuristics into hierarchical goal-based planners in domains that are naturally expressed in HGN planning but not DIP. The current approaches to compute optimal plans using hierarchical planners in such domains use brute-force search algorithms similar to HOGl-blind or the HTN planner SHOP2 (Nau et al. 2003) run in breadth-first search/depth-first branch-and-bound mode.

Second, this can be flipped around and be viewed as a generic mechanism to inject hierarchical knowledge into DIP. This would be applicable in domains that are naturally expressed in PDDL, but also have domain-specific strategies that can be leveraged. In such cases, search heuristics are no longer divorced from the use of hierarchical knowledge; our approach instead provides a principled way to combine the power of the two modes of search guidance.

We also noted that using search heuristics made writing hierarchical knowledge easier. This was because in the absence of heuristics, search guidance is provided by only the knowledge. Hence, there is more pressure on the domain author to write sophisticated knowledge that minimizes the amount of backtracking; with heuristics, we can afford to write much simpler (but complete) models, knowing that the
heuristic will provide additional guidance. An example of this was in the Logistics and Depots domains where choosing the right set of packages to load onto a vehicle at a given location can be tricky if we want to ensure optimality. In our HGN model, we modeled this using a simple recursive method that can load as many packages as the planner wants at that location; the planner can choose to terminate the loading process at any point with a dummy zero-cost action. As the difference in performance between HOGL and HOGL blind in these domains indicates, the heuristic provided useful guidance to the planner at these decision points, which HOGL blind sorely missed.

Related Work

The most relevant literature on incorporating search heuristics into hierarchical planning is the prior work on HGN planning, in particular the Goal Decomposition Planner (GDP) (Shivashankar et al. 2012), the Goal Decomposition with Landmarks planner (GoDeL) (Shivashankar et al. 2013) and HOpGDP, the last of which we have already discussed. As mentioned previously, HGN planning allows for hierarchical modeling of planning domains like HTN planning, but allows for more compact planning models than HTN planning (Shivashankar et al. 2012) as well as greater compatibility with DIP techniques owing to the use of goals in the hierarchy as opposed to tasks. GDP and GoDeL both adapt the Relaxed Planning Graph heuristic (Hoffmann and Nebel 2001) for use with a depth-first search algorithm, and so provide no optimality guarantees. There have been attempts to incorporate heuristics into HTN planning as well.

The H₂O planner (Waibel, Kuter, and Konik 2008) extended SHOP2 with DIP heuristics to estimate how close a method’s goal was to the current state. However, it retained the depth-first nature of SHOP2, and thus also could not provide optimality guarantees.

Alford, Kuter, and Nau (2009) provided a translation of a subset of HTN planning into PDDL (extended further in (Alford et al. 2016a)), and therefore could use DIP algorithms to solve the translated problem. They later proved a negative result stating that delete-relaxation heuristics are not feasible for HTN planning in general by proving that delete-relaxed HTN planning is NP-hard (Alford et al. 2014). However, they showed that if task-insertion is allowed, delete-relaxed HTN planning is in P. This corresponds well with our results; delete-relaxed HRHGN planning (which roughly is HGN planning with task insertion) is in P, while delete-relaxed HGN planning is still NP-hard. The main distinction with the task insertion results is that we further show that HRHGN planning can be compiled into DIP, which allows us to leverage DIP heuristics directly. An analogous polynomial compilation is not possible in the case of HTN planning with task insertion since it is NEXPTIME-complete (Alford, Bercher, and Aha 2015).

Finally, there has been some work on developing search heuristics into POCL-based HTN planning (Elkawkagy et al. 2012; Bercher, Keen, and Biundo 2014) that do not estimate plan cost, but instead estimate the number of flaws needed to be resolved before reaching a solution.
Conclusions and Future Work

In this paper, we have developed a principled framework to incorporate arbitrary DIP heuristics into HGN planning in an admissibility-preserving manner by relaxing the HGN semantics appropriately. We used this approach to develop HGN, an optimal HGN planner, and our experiments demonstrated that simultaneously leveraging hierarchical knowledge and DIP heuristic guidance results in substantially improved planning performance. Therefore, an important takeaway from this paper is that search heuristics in hierarchical planning, despite being a relatively underexplored topic in AI Planning, can play an important role in computing high-quality solutions more efficiently and scalably.

There are several avenues for future work, including using HGN heuristics to develop anytime planners that continue searching after the first solution is found, and extending our framework to incorporate numeric and temporal DIP heuristics to guide numeric and temporal HGN planners. In addition, an interesting theoretical question is whether we can tighten Hierarchy-Relaxation by re-introducing “calculated” amounts of knowledge from the methods in such a way so as to compute more informative heuristic estimates while remaining in \( P \).

Acknowledgements. This work is sponsored in part by OSD ASD (R&E). The information in this paper does not necessarily reflect the position or policy of the sponsors, and no official endorsement should be inferred. We would also like to thank Mark Roberts for many useful discussions, and the anonymous reviewers for their insightful comments.

References


Helmer, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What’s the difference anyway? In AIPS.


