Cost-Optimal Algorithms for Planning with Procedural Control Knowledge

Vikas Shivashankar[1] and **Ron Alford**[2] and **Mark Roberts**[3] and **David W. Aha**[4]

1 Motivation and Background

Formalisms for automated planning (to represent and solve planning problems) broadly fall into either *domain-independent planning* or *domain-configurable planning*. Domain-independent planning formalisms, such as *classical planning* requires that the users only provide models of the base actions executable in the domain. In contrast, domain-configurable planning formalisms (e.g., *Hierarchical Task Network (HTN) planning* [1]) allow users to supplement action models with additional domain-specific knowledge structures that increases the expressivity and scalability of planning systems.

An impressive body of work exploring search heuristics has been developed for classical planning that has helped speed up generation of high-quality solutions. More specifically, search heuristics such as the relaxed planning graph heuristic [2], landmark generation algorithms [3, 5], and landmark-based heuristics [5, 4] dramatically improved optimal and anytime planning algorithms by guiding search towards (near-) optimal solutions to planning problems.

Yet relatively little effort has been devoted to develop analogous techniques to guide search towards high-quality solutions in domain-configurable planning systems. In lieu of such search heuristics, domain-configurable planners often require additional domain-specific knowledge to provide the necessary search guidance. This requirement not only imposes a significant burden on the user, but also sometimes leads to brittle or error-prone domain models.

In this paper, we address this gap by developing the ***H**ierarchically-**Op**timal **G**oal **D**ecomposition **P**lanner (*HOpGDP*)*, a hierarchical planning algorithm that uses admissible heuristic estimates to generate *hierarchically-optimal* plans (i.e., plans that are valid and optimal with respect to the given hierarchical knowledge). HOpGDP leverages recent work on a new hierarchical planning formalism called *Hierarchical Goal Network* (HGN) Planning [8, 6], which combines the hierarchical structure of HTN planning with the goal-based nature of classical planning.

In particular, our contributions are as follows:

- **Admissible Heuristic**: We present h_{HL}(**H**GN **L**andmark heuristic), a planning heuristic that extends landmark-based admissible classical planning heuristics to derive admissible cost estimates for HGN planning problems. To the best of our knowledge, h_{HL} is the first non-trivial admissible hierarchical planning heuristic.
- **Optimal Planning Algorithm**: We introduce HOpGDP, an A* search algorithm that uses h_{HL} to generate *hierarchically-optimal* plans.
- **Experimental Evaluation**: We describe an empirical study on three benchmark planning domains in which HOpGDP outperforms optimal classical planners due to its ability to exploit hierarchical knowledge. We also found that h_{HL} provides useful search guidance; despite substantial computational overhead, it compares favorably in terms of runtime and nodes explored to $\mathsf{HOpGDP}_{\mathsf{blind}}$, using the trivial heuristic $h = 0$.

For the full paper, the readers are referred to the online e-Print [7].

2 Preliminaries

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. An *HGN domain* is a pair $D = (D_c, M)$ where D_c is a classical planning domain and M is a set of HGN methods. D_c describes the models of the base actions, while the HGN methods M specifies hierarchical control knowledge the planner needs to respect when generating plans. Finally, a *goal network* is a partially ordered multiset of goals; this is analogous to the central data structure in HTN planning, the *task network* [1].

HGN Methods. An *HGN method* encodes knowledge on how to decompose goals. Each method m consists of a goal goal(m) that m decomposes, the conditions precond(m) under which it is applicable, and the goal network network(m)) that m decomposes into.

Solutions to HGN Planning Problems. The set of *solutions* for $P = (D, s_0, gn_0)$ is inductively defined as follows: (1) if gn_0 is empty, the empty plan is a solution. If not, assuming $g \in gn_0$ is a goal without predecessors, (2) if g is true in s_0, we can remove g from gn_0, or (3) if there is an action/method applicable in s_0 and relevant to g, we can apply it; actions progress the state while methods progress the goal network.

Let us denote $\mathcal{S}(P)$ as the set of solutions to an HGN planning problem P as allowed by the previous definition. Then we can define what it means for a solution π to be *hierarchically optimal* with respect to P as follows:

Definition 1 (Hierarchically Optimal Solutions)**.** *A solution $\pi^{h,*}$ is hierarchically optimal with respect to P if $\pi^{h,*} = \mathit{argmin}_{\pi \in \mathcal{S}(P)} \mathsf{cost}(\pi)$.*

3 The HOpGDP Algorithm and the h_{HL} Heuristic

HOpGDP takes as input an HGN planning problem $P = (D, s_0, gn_0)$. It does a standard A* search using the admissible HGN

[1] Knexus Research Corporation, National Harbor, MD, vikas.shivashankar@knexusresearch.com
[2] MITRE, McLean, VA, ralford@mitre.org
[3] NRC Postdoctoral Fellow, Naval Research Laboratory, Washington DC, mark.roberts.ctr@nrl.navy.mil
[4] Naval Research Laboratory, Washington DC, david.aha@nrl.navy.mil

Full paper: https://arxiv.org/abs/1607.01729

heuristic h_{HL} (described later in this section) to compute a hierarchically optimal solution to the problem; it either returns a plan if it finds one, or failure if the problem is unsolvable. In particular, starting from the search node (s_0, gn_0), HOpGDP (1) generates successors according to the solution definition in Section 2, and (2) evaluates them using using h_{HL}; it repeats this cycle until it either (a) finds a search node with an empty goal network, at which point it can terminate and return the corresponding plan, or (b) it exhausts the entire search space, in which case it returns failure. HOpGDP thus explores an identical search space as previous HGN planners like GDP [8], but unlike them, it explores the space in a best-first manner, allowing it to explicitly optimize for total plan cost.

The h_{HL} Heuristic. As mentioned previously, HOpGDP uses h_{HL} to compute the h-values (and thus, the f-values) of search nodes. The main insight behind the construction of h_{HL} is as follows: *given a problem $P = (D, s, gn)$, every goal in gn must be achieved, and in the order specified in gn. In other words, the elements of gn can be thought of as landmarks enforced by the hierarchical knowledge, with the partial order serving as landmark orderings.* So, one way to develop admissible HGN heuristics is to use goals in gn as starting points for generating an expanded set of landmarks, and then invoke off-the-shelf landmark-based classical planning heuristics on these landmarks to compute admissible estimates.

Concretely, we construct h_{HL} as follows (details in [7]):

1. We define a relaxation of HGN planning that ignores the provided methods and allows unrestricted action chaining as in classical planning, which expands the set of allowed solutions,
2. We extend landmark generation algorithms for classical planning problems to compute sound landmark graphs for the relaxed HGN planning problems, which in turn are sound with respect to the original HGN planning problems as well, and finally
3. We use admissible classical planning heuristics like h_L [4] on these landmark graphs to compute admissible cost estimates for HGN planning problems.

Based on the admissibility of h_L we can prove that h_{HL} generates admissible cost estimates for HGN planning problems:

Theorem 2 (Admissibility of h_{HL})**.** *Given an HGN planning domain D, a search node (s, gn, π) and its cost-optimal solution $\pi^{*,HGN}_{s,gn}$, $h_{HL}(s, gn, \pi) \leq \pi^{*,HGN}_{s,gn}$.*

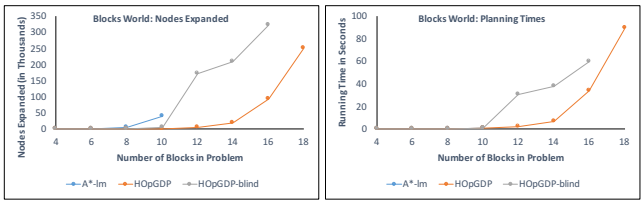
4 Experimental Study

We implemented HOpGDP within the Fast-Downward codebase, and extended LAMA's landmark generation code to develop h_{HL}, our HGN planning heuristic. We tested two hypotheses in our study:

H1. HOpGDP*'s ability to exploit hierarchical planning knowledge enables it to outperform state-of-the-art optimal classical planners.* To test this, we compared the performances of HOpGDP with A*-h_L [4], the optimal classical planner whose heuristic we extended to develop h_{HL}.

H2. *The heuristic used by* HOpGDP*, h_{HL}, provides useful search guidance.* To test this, we compared the performances of HOpGDP with $\text{HOpGDP}_{\text{blind}}$, which is identical to HOpGDP except that it uses the trivial heuristic estimate of $h = 0$.

We evaluated HOpGDP, $\text{HOpGDP}_{\text{blind}}$, and A*-$h_L$ on three well-known planning benchmarks, Logistics, Blocks World and Depots. Due to space constraints, we show results only for Blocks-World; the reader can find the full experimental study in [7].



(a) (b)

Figure 1: Node expansions and running times in Blocks-World. Each data point is the average over 25 randomly generated problems. Data points where all the problems are not solved were discarded.

Figure 1, which shows the results for Blocks-World, shows that A*-h_L could only solve problems up to size 10, while $\text{HOpGDP}_{\text{blind}}$ and HOpGDP could solve problems up to sizes 16 and 18 respectively within the 30 minute timelimit, thus supporting Hypothesis **H1**. In terms of node expansions, Figure 1a indicates that the guidance provided by h_{HL} helped substantially; HOpGDP on average expanded 76% fewer nodes than $\text{HOpGDP}_{\text{blind}}$. This savings far outweighed the heuristic computation overhead (on average about 48% of the total running time), resulting in smaller overall planning times for HOpGDP as shown in Figure 1b, supporting Hypothesis **H2**.

To conclude, our experimental results demonstrate that HOpGDP outperforms optimal classical planners (due to its ability to exploit domain-specific planning knowledge) as well as optimal blind search HGN planners (due to the search guidance provided by h_{HL}).

In the future, we plan to explore extensions of HOpGDP to support *anytime-optimal* planning as well as temporal planning.

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. Ron Alford performed part of this work under an ASEE postdoctoral fellowship at NRL. We also would like to thank the anonymous reviewers at ECAI 2016 and HSDIP 2016 for their insightful comments.

REFERENCES

[1] Kutluhan Erol, James Hendler, and Dana S. Nau, 'UMCP: A sound and complete procedure for hierarchical task-network planning', in *Proc. Int. Conf. on AI Planning Systems*, pp. 249–254, (June 1994).

[2] J. Hoffmann and Bernhard Nebel, 'The FF planning system', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).

[3] Jörg Hoffmann, Julie Porteous, and Laura Sebastia, 'Ordered landmarks in planning', *J. Artif. Intell. Res. (JAIR)*, **22**, 215–278, (2004).

[4] Erez Karpas and Carmel Domshlak, 'Cost-optimal planning with landmarks', in *International Joint Conference on Artificial Intelligence*, pp. 1728–1733, (2009).

[5] Silvia Richter and Matthias Westphal, 'The LAMA planner: Guiding cost-based anytime planning with landmarks', *J. Artif. Intell. Res. (JAIR)*, **39**, 127–177, (2010).

[6] Vikas Shivashankar, Ron Alford, Ugur Kuter, and Dana Nau, 'The GoDeL planning system: a more perfect union of domain-independent and hierarchical planning', in *International Joint Conference on Artificial Intelligence*, pp. 2380–2386. AAAI Press, (2013).

[7] Vikas Shivashankar, Ron Alford, Mark Roberts, and David W. Aha. Cost-optimal algorithms for planning with procedural control knowledge. `https://arxiv.org/abs/1607.01729`, 2016.

[8] Vikas Shivashankar, Ugur Kuter, Dana Nau, and Ron Alford, '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)*, volume 2, pp. 981–988, (June 2012).