Hierarchical Goal Networks for Probabilistic Planning: Preliminary Results

David H. Chan[1,3], **Mark Roberts**[3], **Dana S. Nau**[1,2]
[1]Department of Computer Science and [2]Institute for Systems Research, University of Maryland, College Park, MD, USA
[3]Navy Center for Applied Research in Artificial Intelligence, U.S. Naval Research Laboratory, Washington, DC, USA
dhchan@cs.umd.edu, mark.c.roberts20.civ@us.navy.mil, nau@umd.edu

Abstract

Hierarchical goal networks (HGNs) provide a framework for goal-directed planning by decomposing high-level goals into ordered subgoals. While effective in deterministic settings, HGN planning has not been extended to handle stochastic domains. We introduce a new formalism for probabilistic HGN planning with task-insertion semantics, enabling probabilistic planners to incorporate domain knowledge from goal decomposition methods. This formalism retains the efficiency and scalability of classical HGN planning while supporting probabilistic reasoning and online search. We present PHGN-UCT, a UCT-based online planner that leverages hierarchical methods when available and falls back on UCT search when they are not, enabling planning under partial domain models. Preliminary experiments demonstrate that PHGN-UCT can effectively exploit even incomplete domain knowledge to outperform standard UCT planning, suggesting that probabilistic HGN methods are an effective way to incorporate domain knowledge in probabilistic planning.

1 Introduction

Hierarchical planning formalisms introduce a task hierarchy to allow rich domain-specific guidance and facilitate reasoning at multiple levels of abstraction (Ghallab, Nau, and Traverso 2016, 2025). Hierarchical task network (HTN) planning achieves this through methods which recursively break down compound tasks into sub-tasks until a sequence of primitive actions is identified. However, HTN planning suffers from several key limitations, including the difficulty of designing domain-independent heuristics and the need for a complete set of expert-defined methods. Hierarchical goal networks (HGNs) address these challenges by decomposing goals—rather than tasks—into subgoals (Shivashankar et al. 2012, 2013). This enables the ability to specify goals declaratively and leverage reasoning techniques from classical planning to augment goal decomposition, all while maintaining the expressivity of HTN planning (Alford et al. 2016).

Real-world domains often involve uncertainty, where actions can produce nondeterministic outcomes (Patra et al. 2020, 2021). Fully observable nondeterministic (FOND) models capture such behavior, and recent work has extended HTN planning to operate within the FOND framework (Chen and Bercher 2021, 2022; Yousefi and Bercher 2024). However, many of the properties of HGN planning that make it an attractive formalism for deterministic planning also extend to the probabilistic setting. In particular, the goal-based structure of HGNs enables planners to reason over ordered sequences of goals, a form of temporally extended goal. This makes HGNs especially well-suited for integration with probabilistic planners that rely on online search, where flexibility and adaptability to dynamic environments and stochastic action outcomes are critical, and strict procedural task sequences may be too brittle. Moreover, task-insertion semantics in HGN planning allows executing actions that do not immediately contribute to a goal, thus supporting the interleaving of goal decomposition and heuristic-guided action selection. This provides a principled basis for planning approaches that integrate hierarchical domain knowledge with search.

In this paper, we introduce probabilistic HGN planning, a new formalism that extends HGN planning to probabilistic domains—analogous to how FOND HTN planning extends classical HTN planning. To leverage the new formalism, we also develop an online probabilistic planning algorithm, PHGN-UCT, which embeds the hierarchical guidance and domain knowledge of HGNs into UCT search. We adopt task-insertion semantics, which provides a crucial mechanism for planning with incomplete domain models. The planner takes a hybrid approach that can leverage methods when available, but will otherwise fall back on standard search.

We present preliminary results on a hand-crafted domain suggesting that PHGN-UCT can effectively leverage hierarchical domain knowledge. In our tests, providing more methods consistently improves performance over a standard UCT planner with no methods, indicating that even imperfect or incomplete domain knowledge can significantly accelerate probabilistic planning compared to standard UCT. In summary, our contributions are a novel probabilistic HGN planning formalism and a UCT-based algorithm, demonstrating the benefit of combining hierarchical structure and domain knowledge with search in stochastic domains.

2 Probabilistic HGN Planning

Following the definitions of partial-order classical HGNs from Alford et al. (2016), let $\mathcal{L}$ be a propositional language with a set atoms $\mathcal{X}$ and propositional formulae $\mathcal{F}$. A **goal network** is a tuple $gn = (I, \prec, \alpha)$, where I is a set of symbols which are indices, or labels, for goals, $\prec \subseteq I \times I$ is a partial order on I, and $\alpha : I \to \mathcal{F}$ maps each symbol in I to a goal formula. This labeling of goals is necessary to differentiate multiple occurrences of the same goal in the goal network—the labels will be unique while the goal formulae they map to under α may be equivalent. We denote the set of *unconstrained goals* as $UC(gn) = \{i \in I \mid i \text{ has no } \prec\text{-predecessors}\}$.

Two goal-networks $(I, \prec, \alpha)$ and $(I', \prec', \alpha')$ are *isomorphic* if there exists a bijection $f : I \to I'$ such that $i \prec i' \iff f(i) \prec f(i')$ and $\alpha(i) \equiv \alpha'(f(i))$ for all $i \in I$. This isomorphism relation induces a quotient set of equivalence classes of all goal networks. Going forward, we fix $G = \{(I_1, \prec_1, \alpha_1), (I_2, \prec_2, \alpha_2), \dots\}$ to be a complete set of representatives of this quotient set of all goal networks such that for all $i \neq j$, $I_i \cap I_j = \varnothing$.

A **method** m is a tuple $(\mathrm{pre}(m), \mathrm{post}(m), \mathrm{sub}(m))$, where $\mathrm{pre}(m), \mathrm{post}(m) \in \mathcal{F}$ are the pre- and postcondition of m, respectively, and $\mathrm{sub}(m) = (I_m, \prec_m, \alpha_m) \in G$ is a goal network. A method is *relevant* to a subgoal $i \in I$ in a goal network $gn = (I, \prec, \alpha)$ if at least one literal in the negation-normal form (NNF) of $\mathrm{post}(m)$ matches a literal in the NNF of $\alpha(i)$. This ensures that at least part of $\alpha(i)$ is true by accomplishing $\mathrm{post}(m)$. To ensure that m accomplishes its own goal, we require that there exists $i \in I_m$ such that $\alpha_m(i) \equiv \mathrm{post}(m)$ and for all $j \in I_m$, $j \preceq i$.

A **probabilistic HGN planning domain** is a tuple $\mathcal{D} = (S, M, A, \gamma, \mathrm{Pr})$, where

- $S \subseteq 2^{\mathcal{X}}$ is a finite set of states;
- $M \subseteq \mathcal{F} \times \mathcal{F} \times G$ is a finite set of methods;
- $A \subseteq \mathcal{F} \times 2^{2^{\mathcal{X}} \times 2^{\mathcal{X}}}$ is a finite set of nondeterministic actions $a = (\mathrm{pre}(a), \mathrm{eff}(a)) \in A$ where $\mathrm{pre}(a) \in \mathcal{F}$ is the precondition, and $\mathrm{eff}(a) \in 2^{2^{\mathcal{X}} \times 2^{\mathcal{X}}}$ is a set of pairs $(\mathrm{add}(a), \mathrm{del}(a))$ of add effects $\mathrm{add}(a) \subseteq \mathcal{X}$ and delete effects $\mathrm{del}(a) \subseteq \mathcal{X}$;
- $\gamma : S \times A \to 2^S$ is a transition function where $\gamma(s, a)$ is defined iff $s \models \mathrm{pre}(a)$, and $\gamma(s, a) = \{(s \backslash \mathrm{del}(a)) \cup \mathrm{add}(a) \mid (\mathrm{add}(a), \mathrm{del}(a)) \in \mathrm{eff}(a)\}$; and
- $\mathrm{Pr}(\,\cdot \mid s, a)$ is a probability distribution over $\gamma(s, a)$, where for all $s' \in \gamma(s, a)$, $\mathrm{Pr}(s' \mid s, a)$ is the probability of reaching state s' when action a is executed in state s.

An action or method u is *applicable* in state s if $s \models \mathrm{pre}(u)$.

A **node** is a pair $n = (s, gn)$, where $s \in S$ is a state, and $gn = (I, \prec, \alpha) \in G$ is a goal network. In classical HGN planning, there are three forms of node progression: action application, goal decomposition, and goal release. Analogous forms of node progression extend to probabilistic HGN planning as follows: Let (s, gn) be a node, where $gn = (I, \prec, \alpha)$.

- **action application:** Let $a \in A$ be an action applicable in s. Application of a probabilistic action a yields the node (s', gn) with probability $\mathrm{Pr}(s' \mid s, a)$, for all $s' \in \gamma(s, a)$. We denote this by $(s', gn) \sim P^a_{app}(s, gn)$. Note that under task-insertion semantics, actions need not be "relevant" to an unconstrained subgoal to be applied.
- **goal decomposition:** Let $i \in UC(gn)$, and let m be a method relevant to i and applicable in s, where $\mathrm{sub}(m) = gn_m = (I_m, \prec_m, \alpha_m)$. Goal decomposition by m prepends gn with gn_m to yield the node (s, gn'), where $gn' = (I \cup I_m, \prec \cup \prec_m \cup (I_m \times \{i\}), \alpha \cup \alpha_m)$. We denote this by $(s, gn') = P^{i,m}_{dec}(s, gn)$.
- **goal release:** Let $i \in UC(gn)$, and suppose $s \models \alpha(i)$. Release of subgoal i removes it from gn if it is satisfied in the current state, to yield the node (s, gn'), where $gn' = (I \backslash \{i\}, \{(i_1, i_2) \in \prec \mid i_1 \neq i)\}, \{(i', g) \in \alpha \mid i' \neq i\})$. We denote this by $(s, gn') = P^i_{rel}(s, gn)$.

This model of probabilistic HGN planning operates under task-insertion semantics, which allows an action to be applied whenever an action's preconditions are supported by the current state. This is an extension of the typical HGN planning semantics where actions must be motivated by the hierarchy, i.e. relevant to an unconstrained goal in the goal network, to be executed. This more flexible formalism is helpful in probabilistic settings and with incomplete domain models.

A **probabilistic HGN planning problem** is a tuple $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$, where $\mathcal{D}$ is a probabilistic HGN planning domain, $s_0 \in S$ is the initial state, and gn_0 is the initial goal network. Note that the standard planning paradigm with a standalone goal g can be represented as a goal network $(\{\mathfrak{g}\}, \varnothing, \{(\mathfrak{g}, g)\})$, where $\mathfrak{g}$ is any symbol for g. We use $gn_\varnothing$ to denote the empty goal network $gn_\varnothing = (\varnothing, \varnothing, \varnothing)$.

A solution to a probabilistic HGN planning problem takes the form of an **action-method policy**, which is a partial mapping $\pi : S \times G \to A \cup M$ from a node to either an action $a \in A$ or a method $m \in M$. For the policy to be well-formed, we require that for all (s, gn) in the domain of π the following conditions hold: $gn = (I, \prec, \alpha) \neq gn_\varnothing$; if $\pi(s, gn) = a \in A$ then $s \models \mathrm{pre}(a)$; and if $\pi(s, gn) = m \in M$ then $s \models \mathrm{pre}(m)$ and m is relevant to some $i \in UC(gn)$.

To classify the different types of solutions to a probabilistic HGN planning problem, let $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$ and let π be an action-method policy for $\mathcal{P}$. We define $\mathrm{Graph}(\pi, s_0, gn_0) = (V, E, p)$, where $V \subseteq S \times G$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, and $p : E \to (0, 1]$ is a weight function. We construct $\mathrm{Graph}(\pi, s_0, gn_0)$ as follows:

- $(s_0, gn_0) \in V$
- If $(s, gn) \in V$ and $\pi(s, gn) = a \in A$, then for all $s' \in \gamma(s, a)$, $(s', gn) \in V$ and $e = ((s, gn), (s', gn)) \in E$, with $p(e) = \mathrm{Pr}(s' \mid s, a)$.
- If $(s, gn) \in V$, and $\pi(s, gn) = m \in M$ is relevant to $i \in UC(gn)$, then $(s, gn') \in V$ and $e = ((s, gn), (s, gn')) \in E$ with $p(e) = 1$, where $(s, gn') = P^{i,m}_{dec}(s, gn)$.
- If $(s, gn) \in V$, and there exists $i \in UC(gn)$ s.t. $s \models \alpha(i)$, then $(s, gn') \in V$ and $e = ((s, gn), (s, gn')) \in E$ with $p(e) = 1$, where $(s, gn') = P^i_{rel}(s, gn)$.

$\mathrm{Graph}(\pi, s_0, gn_0) = (V, E, p)$ defines the reachability graph, or "*execution structure*" (Yousefi and Bercher 2024), induced by policy π from initial node $n_0 = (s_0, gn_0)$, where E is a set of directed edges connecting nodes in V via node progression by π, and p are edge weights corresponding to the probability of the node progression. Note that goal decomposition and goal release are deterministic.

Let $n = (s, gn) \in V$. n is *terminal* if there are no outgoing edges from n, and n is a *goal node* if $gn = gn_\varnothing$; note that all goal nodes are necessarily terminal. A *history* σ from n is a finite sequence of nodes $\sigma = \langle n_0, n_1, \dots, n_h \rangle$ such that $n_0 = n$, $\forall j \in \{1, \dots, h\}, (n_{j-1}, n_j) \in E$, and n_h is terminal. We write $H(\pi, n)$ to denote the set of all histories of

Algorithm 1: A UCT planning algorithm for probabilistic HGN planning problems. $\mathcal{D} = (S, M, A, \gamma, \Pr)$ is a probabilistic HGN planning domain, s is the current state, $gn = (I, \prec, \alpha)$ is the current goal network, n_{ro} is the number of rollouts to perform at each step, d is the maximum rollout depth, $w : I \to \mathbb{R}^{\geq 0}$ is a weighting function, and $\Omega : S \to 2^M$ is the current map from states to the sets of methods that have been used in those states.

```
 1: procedure PHGN-UCT($\mathcal{D}, s, gn, n_{ro}, d, w, \Omega$)
 2:   if $gn = gn_\varnothing$ then return success
 3:   for all $i \in UC(gn)$ do
 4:     if $s \models \alpha(i)$ then
 5:       $(s, gn) \leftarrow P^i_{rel}(s, gn)$
 6:       return PHGN-UCT($\mathcal{D}, s, gn, n_{ro}, d, w, \Omega$)
 7:   for $n_{ro}$ times do
 8:     ROLLOUT($\mathcal{D}, s, gn, d, w, \Omega$)                ▷ learn $Q_{\alpha(i)}$
 9:   $U \leftarrow \{m \in M \mid m$ is applicable in $s$, relevant to
               some $i \in UC(gn)$, and not in $\Omega(s)\}$
10:   $U \leftarrow U \cup \{a \in A \mid a$ is applicable in $s\}$
11:   if $U = \varnothing$ then return failure
12:   $u \leftarrow \operatorname{argmax}_{u \in U} \sum_{i \in I} w(i) Q_{\alpha(i)}(s, u)$
13:   if $u$ is a method then
14:     $\Omega(s) \leftarrow \Omega(s) \cup \{u\}$
15:     for all $i \in UC(gn)$ s.t. $u$ is relevant to $i$ do
16:       $(s, gn) \leftarrow P^{i,u}_{dec}(s, gn)$
17:     return PHGN-UCT($\mathcal{D}, s, gn, n_{ro}, d, w, \Omega$)
18:   else                                            ▷ $u$ is an action
19:     $s' \leftarrow$ APPLY($s, u$)
20:     return PHGN-UCT($\mathcal{D}, s', gn, n_{ro}, d, w, \Omega$)
```

Algorithm 2: The Monte Carlo rollout procedure for PHGN-UCT. $\mathcal{D}$, s, gn, w, and Ω are as defined in Algorithm 1. d is the remaining rollout depth, and $util$ is a strictly decreasing utility function over cost. Q_* and N_* are global maps with default value 0. ROLLOUT returns a map $\lambda : I \to \mathbb{R}^{\geq 0}$ where $\lambda(i)$ is the rollout cost for subgoal i.

```
 1: procedure ROLLOUT($\mathcal{D}, s, gn, d, w, \Omega$)
 2:   if $gn = gn_\varnothing$ then return $\varnothing$
 3:   for all $i \in UC(gn)$ do
 4:     if $s \models \alpha(i)$ then
 5:       $(s, gn) \leftarrow P^i_{rel}(s, gn)$
 6:       return ROLLOUT($\mathcal{D}, s, gn, d, w, \Omega$) $\cup \{(i, 0)\}$
 7:   $U \leftarrow \{m \in M \mid m$ is applicable in $s$, relevant to
               some $i \in UC(gn)$, and not in $\Omega(s)\}$
 8:   $U \leftarrow U \cup \{a \in A \mid a$ is applicable in $s\}$
 9:   if $d = 0$ or $U = \varnothing$ then return $I \times \{d\}$
10:   $u \leftarrow \operatorname{argmax}_{u \in U} \sum_{i \in I} w(i) \mathrm{UCB1}(Q_{\alpha(i)}, N_{\alpha(i)}, s, u)$
11:   if $u$ is a method then
12:     $\Omega(s) \leftarrow \Omega(s) \cup \{u\}$
13:     for all $i \in UC(gn)$ s.t. $u$ is relevant to $i$ do
14:       $(s, gn) \leftarrow P^{i,u}_{dec}(s, gn)$
15:     $\lambda \leftarrow$ ROLLOUT($\mathcal{D}, s, gn, d, w, \Omega$)
16:   else                                            ▷ $u$ is an action
17:     sample $(s', gn) \sim P^u_{app}(s, gn)$
18:     $\lambda \leftarrow$ ROLLOUT($\mathcal{D}, s', gn, d - 1, w, \Omega$)
19:     $\lambda \leftarrow \{(i, \lambda(i) + 1) \mid i \in I\}$
20:   for all $i \in I$ do
21:     $Q_{\alpha(i)}(s, u) \leftarrow \frac{N_{\alpha(i)}(s,u) Q_{\alpha(i)}(s,u) + util(\lambda(i))}{1 + N_{\alpha(i)}(s,u)}$
22:     $N_{\alpha(i)}(s) \leftarrow N_{\alpha(i)}(s) + 1$
23:     $N_{\alpha(i)}(s, u) \leftarrow N_{\alpha(i)}(s, u) + 1$
24:   return $\lambda$
```

policy π from node n. The probability of history is defined as $\Pr(\sigma \mid \pi, n) = \prod_{j=1}^{|\sigma|} p(n_{j-1}, n_j)$. A policy π is a *solution policy* for $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$ if $\mathrm{Graph}(\pi, s_0, gn_0)$ contains a goal node. A solution policy π is

- *safe* if for all nodes n in $\mathrm{Graph}(\pi, s_0, gn_0)$, there exists $\sigma \in H(\pi, n)$ that terminates at a goal node; or equivalently, $\sum_{\sigma \in H^*} \Pr(\sigma) = 1$, where $H^* = \{\sigma \in H(\pi, (s_0, gn_0)) \mid \sigma$ terminates at a goal node $\}$. Otherwise, π is *unsafe*.
- *cyclic safe* if π is safe and $\mathrm{Graph}(\pi, s_0, gn_0)$ contains a cycle; it is *acyclic safe* if it is safe and $\mathrm{Graph}(\pi, s_0, gn_0)$ contains no cycles.

3 Probabilistic HGN Planning Algorithm

We propose an online HGN-guided probabilistic planning algorithm based on Monte Carlo tree search (MCTS). MCTS is a general-purpose MDP search algorithm, well-known for its success in computer Go (Silver et al. 2016). It is well-suited to online probabilistic planning tasks (Keller and Eyerich 2012), where it incrementally builds a search tree using Monte Carlo rollouts to estimate the values of states. Many variants of MCTS exist (Gelly and Silver 2011; Feldman and Domshlak 2014; Painter, Lacerda, and Hawes 2020), but the traditional and most popular is Upper Confidence bounds applied to Trees (UCT) (Kocsis and Szepesvári 2006), which leverages principles from the multi-armed bandit problem to strike a balance between exploration and exploitation during planning (Auer, Cesa-Bianchi, and Fischer 2002). We choose UCT because its simplicity and well-understood behavior make it a suitable baseline for analyzing the impact of HGNs on probabilistic planning performance.

Our PHGN-UCT algorithm (Algorithm 1), is an online probabilistic planner which uses the rollout procedure shown in Algorithm 2. At each planning step, a fixed number of rollouts are performed (Lines 7–8), after which the most promising node progression is selected (Line 12) and applied. The resulting state is then observed, and the process repeats until a goal node is reached. Importantly, PHGN-UCT operates under our formalism of HGNs with task-insertion semantics, allowing any action applicable in the current state to be executed, regardless of its relevance to any subgoal.

PHGN-UCT makes several modifications to the standard UCT planning procedure. First, UCT must be adapted to select both actions and methods, mirroring the form of an action-method policy. At a node $n = (s, gn)$, UCT must search among all actions applicable in s, as well as all methods applicable in s and relevant to some $i \in UC(gn)$. To capture this, we extend the standard UCT action-value function Q to estimate values for both actions and methods. That is, $Q(s, u)$ is the estimated reward of progressing the current node n using u, where u is either an action (corresponding to action application P^{u}_{app}) or method (corresponding to goal decomposition $P^{i,u}_{dec}$). We also restrict the set of applicable methods to those which have not been used in s, precluding the same method from being used repeatedly in the same state. Whenever a subgoal $i \in UC(gn)$ is satisfied in s, it is immediately released from gn.

Additionally, to manage multiple goals in the goal network, PHGN-UCT learns a separate Q-function for each subgoal. That is, for all $g \in \alpha[I]$ (the image of I under α), $Q_g(s, u)$ estimates the expected reward of applying action or method u in state s with respect to g. It also maintains corresponding values for $N_{\alpha(i)}$, where $N_{\alpha(i)}(s, u)$ is the number of times u was selected in s, and $N_{\alpha(i)}(s)$ is the number of times s was visited. This contrasts with standard UCT planning, which learns a single Q-function aimed at the final planning goal.

To progress nodes, PHGN-UCT chooses the action or method that maximizes a weighted combination of these Q-values (Algorithm 1 Line 12). During rollouts, UCB1-values are used instead of Q-values, where

$$\text{UCB1}(Q, N, s, u) = Q(s, u) + C\sqrt{\frac{\log(N(s))}{N(s,u)}} \qquad (1)$$

and C is the exploration constant (Algorithm 2 Line 10). The weighting is governed by a function $w : I \to \mathbb{R}^{\geq 0}$, which assigns the importance to each subgoal. A simple greedy implementation would define $w(i) = \mathbf{1}_{UC(gn)}(i)$, prioritizing immediate subgoals. However, such a strategy can be myopic, potentially leading to states that make future subgoals harder or impossible to achieve. More sophisticated weighting schemes that balance short-term and long-term goal achievement may yield better results, but we leave the exploration of such approaches to future work.

4 Preliminary Results

We ran some preliminary tests of PHGN-UCT on a small warehouse planning problem. In a grid world, a robot must move a box from a locked room to a target location by first retrieving a key. Three HGN methods were created for the domain, and our experiments varied the number of methods made available to the planner, from 0 methods (corresponding to standard UCT) to all 3. Our experiments only captured the special case of totally-ordered probabilistic HGNs, where in all goal networks $gn = (I, \prec, \alpha)$, $\prec$ is a total order. In this case, $UC(gn)$ is always a singleton set, so the weighting function $w = \mathbf{1}_{UC(gn)}(i)$ guides the planner to exclusively search for the next immediate subgoal.

We used a non-heuristic implementation of UCT with exploration constant $C = \sqrt{2}$, a maximum rollout depth $d = 20$, and a utility function $util : cost \mapsto \exp(-cost)$. The number of rollouts was varied in $n_{ro} \in \{5, 10, 20, 50, \ldots, 10000\}$. The cost of a run of PHGN-UCT is measured as the number of actions executed (Algorithm 1 Line 19). A maximum cost budget of 100 was imposed. Table 1 reports the average costs of running PHGN-UCT across 100 trials on the warehouse problem, varying the number of methods provided to the planner and the number of rollouts performed. When no methods are provided, PHGN-UCT reduces to standard UCT, which serves as the baseline. Runs which exceeded the cost budget of 100 were halted, and contributed 100 to the average.

Table 1: Average cost $\overline{x}$ (lower is better) and standard deviation σ of runs of the PHGN-UCT algorithm in the warehouse planning problem over 100 trials.

	0 methods		1 method		2 methods		3 methods	
n_{ro}	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
5	100.0	0.0	41.6	18.7	13.1	1.6	13.4	1.7
10	100.0	0.0	29.7	10.4	13.8	1.8	13.8	1.6
20	100.0	0.0	21.1	4.8	13.7	1.5	13.3	1.4
50	100.0	0.0	15.5	2.5	12.9	1.3	12.8	1.3
100	100.0	0.0	13.3	1.3	12.8	1.5	12.5	1.2
200	100.0	0.0	12.9	1.3	13.2	1.6	12.5	1.3
500	100.0	0.0	13.2	1.6	12.7	1.5	12.8	1.2
1000	100.0	0.0	13.0	1.4	13.0	1.4	12.7	1.5
2000	96.6	14.2	12.8	1.2	12.9	1.5	12.6	1.4
5000	68.9	36.1	13.1	1.3	12.9	1.5	12.4	1.2
10000	65.6	40.0	12.7	1.1	12.6	1.3	12.3	1.0

Standard UCT without any HGN methods provided struggles to find a solution in the warehouse planning problem. However, even a single HGN method boosts performance significantly, immediately outperforming standard UCT at 10000 rollouts while only using 5 rollouts, and converging to a near-optimal solution at only 100 rollouts per step. Performance consistently improves as more methods are provided, demonstrating that PHGN-UCT can effectively leverage hierarchical domain knowledge—even when it is incomplete—to achieve significantly better performance than standard UCT. While preliminary, these results suggest that HGNs may be useful for capturing domain knowledge for probabilistic planning, and future work should characterize the kinds of problems where these results hold.

5 Conclusion and Future Work

We introduced a new formalism for partial-order, probabilistic HGN planning with task-insertion semantics. This formalism preserves many of the key benefits of classical HGN planning while extending its applicability to stochastic domains. To leverage this formalism, we developed PHGN-UCT, a UCT-based planner that dynamically interleaves method-based decomposition and action-level search. Our approach balances structured hierarchical guidance with adaptive online search, enabling effective planning even in the presence of incomplete or imperfect domain models.

Preliminary results suggest PHGN-UCT effectively utilizes domain knowledge captured in HGN methods, but future work should expand the empirical evaluation across a broader range of benchmarks. Additionally, future work should refine the greedy weight function that prioritizes immediate goals, exploring strategies that more effectively trade off short- and long-term goals. Finally, a formal analysis of the expressiveness and complexity of probabilistic HGN planning, particularly in relation to FOND HTN planning (Chen and Bercher 2022), will help to understand the theoretical foundations, capabilities, and limitations of this framework.

6 Acknowledgments

This work was supported in part by the U.S. Naval Reseach Laboratory.

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 *Proc. IJCAI 2016*, 3022–3029.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, 47(2-3): 235–256.

Chen, D. Z.; and Bercher, P. 2021. Fully Observable Nondeterministic HTN Planning - Formalisation and Complexity Results. In *Proc. ICAPS 2021*, 74–84.

Chen, D. Z.; and Bercher, P. 2022. Flexible FOND HTN Planning: A Complexity Analysis. In *Proc. ICAPS 2022*, 26–34.

Feldman, Z.; and Domshlak, C. 2014. Simple Regret Optimization in Online Planning for Markov Decision Processes. *J. Artif. Intell. Res.*, 51: 165–205.

Gelly, S.; and Silver, D. 2011. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artif. Intell.*, 175(11): 1856–1875.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press. ISBN 978-1-107-03727-4.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2025. *Acting, Planning, and Learning*. Cambridge University Press. ISBN 9781009579346.

Keller, T.; and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proc. ICAPS 2012*.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *Proc. ECML 2006*, 282–293.

Painter, M.; Lacerda, B.; and Hawes, N. 2020. Convex Hull Monte-Carlo Tree-Search. In *Proc. ICAPS 2020*, 217–225.

Patra, S.; Mason, J.; Ghallab, M.; Nau, D. S.; and Traverso, P. 2021. Deliberative acting, planning and learning with hierarchical operational models. *Artif. Intell.*, 299: 103523.

Patra, S.; Mason, J.; Kumar, A.; Ghallab, M.; Traverso, P.; and Nau, D. S. 2020. Integrating Acting, Planning, and Learning in Hierarchical Operational Models. In *Proc. ICAPS 2020*, 478–487.

Shivashankar, V.; Alford, R.; Kuter, U.; and Nau, D. S. 2013. The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning. In *Proc. IJCAI 2013*, 2380–2386.

Shivashankar, V.; Kuter, U.; Nau, D. S.; and Alford, R. 2012. A hierarchical goal-based formalism and algorithm for single-agent planning. In *Proc. AAMAS 2012*, 981–988.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587): 484–489.

Yousefi, M.; and Bercher, P. 2024. Laying the Foundations for Solving FOND HTN Problems: Grounding, Search, Heuristics (and Benchmark Problems). In *Proc. IJCAI 2024*, 6796–6804.