Probabilistic Hierarchical Goal Network Planning with UCT

David H. Chan[1,2], **Mark Roberts**[2], **Dana S. Nau**[1]

[1]University of Maryland, College Park, MD, USA
[2]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 prior work has examined nondeterminism for hierarchical planning (specifically, HTNs), scant work studies how HGNs can help in stochastic settings. We introduce a formalism for probabilistic HGN planning with action-insertion semantics, enabling probabilistic planners to incorporate domain knowledge from goal decomposition methods. We design and evaluate two UCT-based algorithms for solving probabilistic HGN planning problems: an asymptotically optimal approach and a compressed, shared-value approach that optimizes separately for each goal within the goal-subgoal hierarchy. We compare our two UCT-based HGN search algorithms experimentally on modified benchmark domains from the FOND HTN literature. Our results demonstrate that on larger problems, the compressed search converges more quickly and outperforms the asymptotically optimal search. This suggests that HGNs can be effective in probabilistic planning, and compression may yield better performance on large problems in anytime settings with stochastic action outcomes.

Code — https://github.com/dhrchan/phgnuct

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 that recursively break down compound tasks into sub-tasks until a sequence of primitive actions is identified. However, HTN planning suffers from the difficulty of designing domain-independent heuristics and the need for a complete set of expert-defined methods. Hierarchical goal networks (HGNs) (Shivashankar et al. 2012) address these challenges by decomposing goals, rather than tasks, which enables the use of classical planning techniques to supplement decomposition while maintaining higher expressivity (Alford et al. 2016b).

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).

To our knowledge, no work has addressed the use of HGNs in a probabilistic setting. Yet, many properties of HGN planning that make it attractive for deterministic planning extend to the probabilistic setting. The goal-based structure of HGNs enables planners to reason over ordered sequences of goals—a form of temporally extended goal—and 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; in such environments, the strict procedural task sequences of HTNs may be too brittle. Furthermore, some HGN formalisms (e.g., work by Shivashankar et al. (2013)) allow executing actions not motivated by a goal, thus support the interleaving of goal decomposition and heuristically-guided action selection. These benefits provide a principled basis for integrating hierarchical domain knowledge with search.

To leverage hierarchical goal decomposition in probabilistic planning, we formalize *probabilistic* HGN planning with action-insertion semantics. We construct two UCT-based algorithms (Kocsis and Szepesvári 2006) that leverage HGN knowledge: (1) $\mathrm{UCT}_{\mathrm{base}}$, a baseline approach that transforms a probabilistic HGN problem into an equivalent stochastic shortest path (SSP) problem and applies UCT directly to solve it, yielding asymptotically optimal behavior; and (2) $\mathrm{UCT}_{\mathrm{comp}}$, a compressed variant of UCT that reduces the size of the search tree by sharing learned value estimates across related goal networks. Results on four benchmarks adapted from FOND HTN domains show that $\mathrm{UCT}_{\mathrm{base}}$ performs well on smaller problems but struggles to scale to larger problems, while the $\mathrm{UCT}_{\mathrm{comp}}$ variant converges more quickly, requires less memory, and consistently finds lower-cost solutions on larger problems. These results suggest that a compressed UCT which abstracts over goal networks offers an effective, scalable approach to probabilistic HGN planning.

2 Background and Related Work

HTN planning uses methods to encode hierarchical structure and domain control knowledge, making it strictly more expressive than classical planning (Erol, Hendler, and Nau 1994). However, HTN tasks are syntactic constructs that do

not directly correspond to world-state goals, but rather represent abstract activities whose semantics are determined by available decomposition methods. This task-goal decoupling creates several practical challenges: reachability analyses become non-trivial since tasks and goals may not align; incomplete or incorrect methods can cause search to fail; and designing domain-independent heuristics is difficult because tasks may not explicitly specify the world conditions they achieve. Recent advances address some of these limitations, developing search mechanisms which include reachability analysis (Bit-Monnot, Smith, and Do 2016; Behnke et al. 2020), heuristic search (Bercher et al. 2017; Höller et al. 2020), compilation to classical planning (Alford, Kuter, and Nau 2009; Alford et al. 2016a; Behnke et al. 2022; Höller 2024) or SAT (Behnke, Höller, and Biundo 2018; Schreiber 2021; Quenard, Pellier, and Fiorino 2024), and Monte-Carlo tree search adaptations (Wichlacz et al. 2020).

HGNs address these limitations by making goals explicit in the hierarchical structure. In doing so, HGN planning simplifies method authoring (Shivashankar et al. 2012) and enables the use of classical planning heuristics (Shivashankar et al. 2016; Shivashankar, Alford, and Aha 2017). Some HGN planners use an action-insertion semantics[1] that allows the planner to fall back on classical search if no method is available for a given goal (Shivashankar et al. 2013). This design enables HGN planning algorithms to gracefully handle incomplete domain knowledge by leveraging hierarchical methods when available while maintaining completeness through classical planning when necessary.

While HGNs address many limitations of traditional HTN planning, both classical HTN and HGN approaches have been primarily limited to deterministic environments. HTN planning has recently been extended to the nondeterministic setting. Notably, work by Chen and Bercher (2021) formalized FOND HTN planning. Subsequent work introduced policy-based variants that allow decomposition decisions to be contingent on outcomes and proposed method for generating strong and weak solutions using deterministic relaxations (Chen and Bercher 2022; Yousefi and Bercher 2024). Concurrent with our work, Yousefi et al. (2025) independently developed a probabilistic HTN formulation with known action outcome distributions. However, this does not leverage the representational advantages of HGNs.

Building on the foundation of HGN planning and recent advances in FOND HTN planning, our work extends hierarchical planning to probabilistic domains with known action outcome distributions. In contrast to the HTN-based approaches, our method leverages the goal-centric structure of HGNs, applying them to stochastic domains with UCT-based search. Probabilistic HGN planning leverages the expressiveness of hierarchical models while maintaining the flexibility required in probabilistic domains.

3 Probabilistic HGN Planning

We adapt partial-order classical HGN planning (Alford et al. 2016b) to incorporate probabilistic actions, drawing inspiration from FOND HTN planning (Yousefi and Bercher 2024).

[1]Essentially an HGN restriction of GTN_I by Alford et al. (2016b).

Definition 1 (Goal Network (Alford et al. 2016b))**.** Let $\mathcal{L}$ be a propositional language with a set of 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 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.

A goal network is a partially ordered multiset of goals, represented using a set I of indices, or labels, for goals. 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 $UC(gn) = \{i \in I \mid i \text{ has no } \prec\text{-predecessors}\}$.

Because the specific symbols used to label goals in a goal network are arbitrary and semantically unimportant, it is standard in the current literature to define an isomorphism relation to formally capture when two goal networks are structurally equivalent (Alford et al. 2016b). To simplify our reasoning about goal networks and avoid label collisions, we fix a canonical set G of all goal networks such that all index sets are disjoint. That is, $G = \{(I_1, \prec_1, \alpha_1), (I_2, \prec_2, \alpha_2), \dots\}$ is a complete set of goal networks up to isomorphism such that for all $i \neq j$, $I_i \cap I_j = \varnothing$. We also fix $\tilde{I} = \bigcup_{(I,\prec,\alpha)\in G} I$ to be the set of all labels across all goal networks in G.

Definition 2 (Method (Alford et al. 2016b))**.** A *(goal) method* m is a tuple $(\text{pre}(m), \text{post}(m), \text{sub}(m))$ where $\text{pre}(m), \text{post}(m) \in \mathcal{F}$ are the pre- and postcondition of m, respectively, and $\text{sub}(m) = (I_m, \prec_m, \alpha_m) \in G$ is a goal network.

A method m 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 $\text{post}(m)$ matches a literal in the NNF of $\alpha(i)$. This ensures that at least part of $\alpha(i)$ is true by accomplishing $\text{post}(m)$. To ensure that m accomplishes its own goal, we require that there exists $i \in I_m$ such that $\alpha_m(i) = \text{post}(m)$ and for all $j \in I_m$, if $j \neq i$ then $j \prec i$.

Definition 3 (Probabilistic HGN Planning Domain)**.** Let $\mathcal{L}$ be a propositional language with sets of atoms $\mathcal{X}$ and propositional formulae $\mathcal{F}$. A *probabilistic HGN planning domain* is a tuple $\mathcal{D} = (\mathcal{L}, M, A, \gamma, \Pr)$, where:

- $S = 2^{\mathcal{X}}$ 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 $a = (\text{pre}(a), \text{eff}(a)) \in A$ where $\text{pre}(a) \in \mathcal{F}$ is the precondition, and $\text{eff}(a) \in 2^{2^{\mathcal{X}} \times 2^{\mathcal{X}}}$ is a set of pairs $(\text{add}(a), \text{del}(a))$ of add effects $\text{add}(a) \subseteq \mathcal{X}$ and delete effects $\text{del}(a) \subseteq \mathcal{X}$;
- $\gamma : S \times A \to 2^S$ is a transition function where $\gamma(s, a)$ is defined iff $s \models \text{pre}(a)$, and $\gamma(s, a) = \{(s \backslash \text{del}(a)) \cup \text{add}(a) \mid (\text{add}(a), \text{del}(a)) \in \text{eff}(a)\}$; and
- $\Pr(\cdot \mid s, a)$ is a probability distribution over $\gamma(s, a)$, where for all $s' \in \gamma(s, a)$, $\Pr(s' \mid s, a)$ 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 $s \models \text{pre}(u)$. For simplicity, we assume all actions have unit cost.

Definition 4 (Node)**.** 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.

We adapt the three forms of node progression from classical HGN planning: goal release, goal decomposition, and action application.

Definition 5 (Node Progression)**.** Let (s, gn) be a node, where $gn = (I, \prec, \alpha)$.

- *Deterministic Goal Release* (P^i_{rel}): Let $i \in UC(gn)$ such that $s \models \alpha(i)$. Release of i removes it from gn 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 $(s, gn') = P^i_{rel}(s, gn)$.
- *Deterministic Goal Decomposition* ($P^{i,m}_{dec}$): Suppose $s \not\models \alpha(j)$ for all $j \in UC(gn)$. Let $i \in UC(gn)$, and let $m \in M$ be relevant to i and applicable in s, where $\mathrm{sub}(m) = (I_m, \prec_m, \alpha_m)$. Decomposition by m prepends gn with $\mathrm{sub}(m)$ 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 $(s, gn') = P^{i,m}_{dec}(s, gn)$.
- *Probabilistic Action Application* (P^a_{app}): Suppose $s \not\models \alpha(i)$ for all $i \in UC(gn)$, and let $a \in A$ be applicable in s. Application of action a yields the node (s', gn) with probability $\Pr(s' \mid s, a)$, for all $s' \in \gamma(s, a)$. We denote this by $(s', gn) \sim P^a_{app}(s, gn)$.

Our model of probabilistic HGN planning with action-insertion semantics allows actions to be executed independently of the hierarchy, as in work by Shivashankar et al. (2013). That is, an action that is applicable in the current state can be executed without requiring "relevance" to an unconstrained goal i. This flexibility is particularly helpful in probabilistic settings where primitive action chaining may be needed to recover from unexpected states caused by stochastic actions, or incomplete hierarchical models.

We also impose an additional constraint: subgoals are immediately released from the goal network once satisfied. That is, goal decomposition and action application are disallowed until all unconstrained subgoals satisfied in the current state are released. This captures our adaptation of the action-insertion semantics defined by Shivashankar et al. (2013).

Definition 6 (Probabilistic HGN Planning Problem)**.** 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 $gn_0 \in G$ is the initial goal network.

In probabilistic HGN planning, the planning objective is specified using an initial goal network rather than a single goal condition. This goal network serves as a temporally extended goal, and planning is complete once all goals in the network have been successfully released. We denote the empty goal network as $gn_\varnothing = (\varnothing, \varnothing, \varnothing)$. 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. In fact, since we allow action insertion, if no HGN methods are provided and the initial goal network contains only a single goal, probabilistic HGN planning reduces to standard (non-hierarchical) probabilistic planning.

While solutions to standard probabilistic planning problems are policies over actions, HGN planning requires selecting both actions and methods. To that end, we introduce *action-method policies* to represent solutions to probabilistic HGN planning problems, where actions and methods can be selected based on the current node. These are analogous to "method-based policies" of Chen and Bercher (2022).

Definition 7 (Action-Method Policy)**.** An *action-method policy* is a partial mapping $\hat{\pi} : S \times G \to (\{\epsilon\} \times A) \cup (\tilde{I} \times M)$, where ϵ is a reserved symbol to indicate action application. $\hat{\pi}$ is *well-formed* if for all (s, gn) in the domain of $\hat{\pi}$ with $\hat{\pi}(s, gn) = (i, u)$, the following conditions hold: $gn = (I, \prec, \alpha) \neq gn_\varnothing$; $s \not\models \alpha(j)$ for all $j \in UC(gn)$; $s \models \mathrm{pre}(u)$; and if $u \in M$ then $i \in UC(gn)$ and u is relevant to i.

To classify the different types of solutions, we define the *reachability graph* of a policy.

Definition 8 (Reachability Graph)**.** Let $\hat{\pi}$ be a well-formed action-method policy for $\mathcal{P}$. Its *reachability graph* is a tuple $\mathrm{Graph}(\hat{\pi}, s_0, gn_0) = (V, E, p)$, with nodes $V \subseteq S \times G$, edges $E \subseteq V \times V$, and a weight function $p : E \to (0, 1]$.

$\mathrm{Graph}(\hat{\pi}, s_0, gn_0)$ is constructed as follows:

- $(s_0, gn_0) \in V$
- [*goal release*] If $(s, gn) \in V$ and there exists $i \in UC(gn)$ such that $s \models \alpha(i)$, then $(s, gn') \in V$ and $e = ((s, gn), (s, gn')) \in E$, where $(s, gn') = P^i_{rel}(s, gn)$ and $p(e) = 1$.
- [*goal decomposition*] If $(s, gn) \in V$ and $\hat{\pi}(s, gn) = (i, m)$ where $(i, m) \in \tilde{I} \times M$, then $(s, gn') \in V$ and $e = ((s, gn), (s, gn')) \in E$, where $(s, gn') = P^{i,m}_{dec}(s, gn)$ and $p(e) = 1$.
- [*action application*] If $(s, gn) \in V$ and $\hat{\pi}(s, gn) = (\epsilon, a)$ where $a \in A$, then for all $s' \in \gamma(s, a)$, $(s', gn) \in V$ and $e = ((s, gn), (s', gn)) \in E$, with $p(e) = \Pr(s' \mid s, a)$.

A node $n \in V$ is *terminal* if it has no outgoing edges. A *goal node* is a terminal node (s, gn) where $gn = gn_\varnothing$.

$\mathrm{Graph}(\pi, s_0, gn_0) = (V, E, p)$ defines the reachability graph—called the "*execution structure*" by Yousefi and Bercher (2024)—induced by following $\hat{\pi}$ from $n_0 = (s_0, gn_0)$ using node progression with edge weights corresponding to probability.

Definition 9 (History)**.** Let $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$, let $\hat{\pi}$ be a well-formed action-method policy for $\mathcal{P}$ with reachability graph $\mathrm{Graph}(\hat{\pi}, s_0, gn_0) = (V, E, p)$, and let $n = (s, gn) \in V$. A *history* σ of $\hat{\pi}$ from n is a finite sequence of nodes $\sigma = \langle n_0, n_1, \ldots, n_h \rangle$ such that $n_0 = n$, $\forall j \in \{1, \ldots, h\}, (n_{j-1}, n_j) \in E$, and n_h is terminal.

We write $|\sigma|$ to denote the length of σ, and σ_{-1} to denote the final node in σ. We write $H(\hat{\pi}, n)$ to denote the set of all histories of policy $\hat{\pi}$ from node n. The probability of a history is defined as $\Pr(\sigma \mid \hat{\pi}, n) = \prod_{j=1}^{|\sigma|} p(n_{j-1}, n_j)$.

Definition 10 (Solution Policy)**.** Let $\hat{\pi}$ be a well-formed action-method policy for $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$. Let $H^* = \{\sigma \in H(\hat{\pi}, (s_0, gn_0)) \mid \sigma \text{ terminates at a goal node}\}$. Let $\rho \in [0, 1]$. $\hat{\pi}$ is a *ρ-policy* if $\sum_{\sigma \in H^*} \Pr(\sigma) \geq \rho$.

$\hat{\pi}$ is *(cyclic)* safe if it is a 1-policy; or equivalently, for all $n \in \mathrm{Graph}(\hat{\pi}, s_0, gn_0)$, there exists $\sigma \in H(\hat{\pi}, n)$ such that σ_{-1} is a goal node. $\hat{\pi}$ is *acyclic safe* if $\hat{\pi}$ is safe and $\mathrm{Graph}(\hat{\pi}, s_0, gn_0)$ contains no cycles. Otherwise, $\hat{\pi}$ is *unsafe*.

4 Probabilistic HGN Planning Algorithm

We propose two HGN-guided probabilistic planning algorithms based on Monte Carlo tree search (MCTS), an 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. Of the many variants of MCTS (e.g., (Gelly and Silver 2011; Feldman and Domshlak 2014; Painter, Lacerda, and Hawes 2020)), we develop an approach based on 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). UCT converges to optimal policies given sufficiently many rollouts, but its main strength lies in providing good anytime performance under limited computational resources. UCT's simplicity makes it a suitable baseline for analyzing the impact of HGNs.

We adapt UCT for goal-directed planning by integrating the GUBS (Goals with Utility-Based Semantics) criterion (Freire and Delgado 2017; Freire, Delgado, and Reis 2019). GUBS jointly maximizes the probability of goal achievement and minimizes expected cost using a utility function

$$U(\sigma) = util(|\sigma|) + K_g \mathbf{1}_{[\sigma_{-1} \text{ is a goal node}]}, \tag{1}$$

where $util$ is a strictly decreasing utility over cost and K_g is a goal utility constant. UCT-GUBS, as described by Crispino, Freire, and Delgado (2024), estimates expected utility using Q-values, preserving the exploration-exploitation balance while effectively handling dead ends.

4.1 Baseline UCT ($\mathrm{UCT}_{\mathrm{base}}$)

We first establish a baseline approach using standard probabilistic planning techniques. A probabilistic HGN planning problem $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$ can be reformulated as an SSP problem over nodes, where each node captures both the current world state and the current goal network.

Construction 1. Let $\mathcal{P} = (\mathcal{D}, s_0, gn_0)$ be a probabilistic HGN planning problem. We construct a (non-hierarchical) SSP $\tilde{\mathcal{P}} = (\tilde{S}, \tilde{A}, \tilde{\Pr}, \tilde{s_0}, \tilde{S_g})$ to represent the planning problem over nodes, where

- $\tilde{S} = S \times G$ is the state space comprising all nodes;
- $\tilde{A} = A_{rel} \cup A_{dec} \cup A_{app}$ where $A_{rel} = \{P^i_{rel} \mid i \in \tilde{I}\}$, $A_{dec} = \{P^{i,m}_{dec} \mid i \in \tilde{I}, m \in M\}$, and $A_{app} = \{P^a_{app} \mid a \in A\}$;
- $\tilde{\Pr}(\tilde{s}' \mid \tilde{s}, \tilde{a}) = \Pr[\tilde{a}(\tilde{s}) = \tilde{s}']$ defines transition probabilities for all $\tilde{a} \in \tilde{A}$ and $\tilde{s} \in \tilde{S}$;
- $\tilde{s_0} = (s_0, gn_0)$ is the initial node; and
- $\tilde{S_g} = S \times \{gn_\varnothing\}$ is the set of goal nodes.

Following our definitions for node progressions, $\tilde{a} \in \tilde{A}$ is applicable in $(s, gn) \in \tilde{S}$ if any of the following hold:

- $\tilde{a} = P^i_{rel} \in A_{rel}$, where $i \in UC(gn)$ and $s \models \alpha(i)$;
- $\tilde{a} = P^{i,m}_{dec} \in A_{dec}$, where $i \in UC(gn), m$ is relevant to i, $s \models \mathrm{pre}(m)$, and $s \not\models \alpha(j)$ for all $j \in UC(gn)$; or
- $\tilde{a} = P^a_{app} \in A_{app}$, $s \models \mathrm{pre}(a)$, and $s \not\models \alpha(i)$ for all $i \in UC(gn)$.

Proposition 1. *Construction 1 maps the probabilistic HGN problem as an equivalent SSP problem.*

Proof Sketch. Construction 1 creates a direct correspondence between solutions to $\tilde{\mathcal{P}}$ and solutions to $\mathcal{P}$. The state space $\tilde{S}$ mirrors the set of nodes, and $\tilde{S_g}$ corresponds to the set of goal nodes. Each SSP action in A_{red}, A_{dec}, and A_{app} directly encodes one of the three types of node progression. The only subtlety is that action-method policies for $\mathcal{P}$ do not explicitly trigger goal releases; instead, goal release is handled automatically by the semantics of node progression whenever a subgoal is satisfied. □

This equivalence allows us to directly apply any SSP algorithm to solve $\mathcal{P}$. As a baseline, we use the UCT algorithm, which provably converges to the optimal value function with probability 1 in the limit of infinite rollouts (Kocsis and Szepesvári 2006). Therefore, UCT applied to the SSP $\tilde{\mathcal{P}}$ will converge to a policy for $\mathcal{P}$ which maximizes the GUBS utility. In our implementation, the SSP is not precompiled, but rather it is expanded at runtime by UCT.

4.2 Compressed UCT ($\mathrm{UCT}_{\mathrm{comp}}$)

While the baseline $\mathrm{UCT}_{\mathrm{base}}$ is theoretically sound, it suffers from scalability limitations. The state space $\tilde{S} = S \times G$ introduces an additional exponential factor, thus can grow prohibitively large in problems with complex goal networks, leading to degraded performance in time- or compute-limited settings. In such cases, UCT may fail to converge to a viable solution within the available computational resources.

The inefficiency stems from treating each node (s, gn) as a distinct state. This overlooks the crucial observation that nodes sharing the same underlying world state s and unconstrained subgoals in gn are likely to benefit from similar actions. In other words, the optimal action in a given world state with respect to a goal network can often be approximated using only the immediate subgoals that need to be achieved, rather than the complete, potentially complex, structure of the entire goal network.

To address this, we propose a "compressed" UCT approach that creates only one tree node per underlying world state s. Rather than a single value estimate per node as in standard UCT, we maintain a separate Q-function for each unconstrained subgoal encountered in that world state. This allows the algorithm to share learned information across nodes with different goal networks, leading to more efficient exploration and faster convergence.

We implement this compressed approach in our algorithm, $\mathrm{UCT}_{\mathrm{comp}}$ (Algorithm 1), a UCT-based probabilistic planner that uses the rollout procedure in Algorithm 2. At each step, a fixed number of rollouts are performed (Lines 6–7). Using collected rollout statistics, the algorithm selects and applies the most promising node progression (Line 11), observes the resulting state, and repeats until the goal network is empty.

$\mathrm{UCT}_{\mathrm{comp}}$ introduces several key modifications to the standard UCT framework. First, it extends action selection to include both primitive actions and methods, reflecting the structure of action-method policies. At a node $n = (s, gn)$, the algorithm considers all actions applicable in s as well as

Algorithm 1: ($\mathrm{UCT}_{\mathrm{comp}}$) A compressed UCT algorithm for probabilistic HGN planning problems. $\mathcal{D} = (\mathcal{L}, M, A, \gamma, \Pr)$ 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 performed at each step, d is the maximum rollout depth, and c is the cumulative cost (initially 0).

1: **procedure** $\mathrm{UCT}_{\mathrm{comp}}(\mathcal{D}, s, gn, n_{ro}, d, c)$
2: **if** $gn = gn_{\varnothing}$ **then return** success
3: **for all** $i \in UC(gn)$ such that $s \models \alpha(i)$ **do**
4: $(s, gn) \leftarrow P^{i}_{rel}(s, gn)$
5: **return** $\mathrm{UCT}_{\mathrm{comp}}(\mathcal{D}, s, gn, n_{ro}, d, c)$
6: **for** n_{ro} times **do**
7: $\textsc{Rollout}_{\mathrm{comp}}(\mathcal{D}, s, gn, d, c)$ ▷ learn $Q_{\alpha(i)}$
8: $U \leftarrow \{(i, m) \in UC(gn) \times M \mid$ m is applicable in s and relevant to $i\}$
9: $U \leftarrow U \cup \{a \in A \mid a \text{ is applicable in } s\}$
10: **if** $U = \varnothing$ **then return** failure
11: $u \leftarrow \operatorname{argmax}_{u \in U} \sum_{i \in UC(gn)} Q_{\alpha(i)}(s, u)$
12: **if** $u \in A$ **then**
13: $s' \leftarrow \textsc{apply}(s, u)$
14: **return** $\mathrm{UCT}_{\mathrm{comp}}(\mathcal{D}, s', gn, n_{ro}, d, c + 1)$
15: **else** ▷ u is a subgoal-method pair
16: $(i, m) \leftarrow u$
17: $(s, gn) \leftarrow P^{i,m}_{dec}(s, gn)$
18: **return** $\mathrm{UCT}_{\mathrm{comp}}(\mathcal{D}, s, gn, n_{ro}, d, c)$

all methods applicable in s that are relevant to some subgoal $i \in UC(gn)$. To capture this, the Q-function is extended to estimate values for both actions and methods:

- for actions a, $Q(s, a)$ is the estimated reward of executing a in state s, corresponding to the node progression P^{a}_{app}.
- for decompositions, $Q(s, (i, m))$ denotes the estimated reward of using method m to decompose subgoal i in state s, corresponding to the node progression $P^{i,m}_{dec}$.

Second, to optimize for multiple goals in the goal network, $\mathrm{UCT}_{\mathrm{comp}}$ learns a separate Q-function for each unconstrained subgoal. That is, for all $g \in \alpha[UC(gn)]$ (the image of $UC(gn)$ under α), $Q_g(s, u)$ estimates the expected reward of applying u in state s with respect to g. Here, u is either a primitive action $a \in A$, or a decomposition represented by a subgoal-method pair (i, m). 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 is in contrast to standard UCT planning, which learns a single Q-function aimed at the final planning goal.

To progress nodes, $\mathrm{UCT}_{\mathrm{comp}}$ chooses the action or method that maximizes the sum of these Q-values (Algorithm 1 Line 11), jointly optimizing for multiple subgoals. During rollouts, UCB1-values are used instead of Q-values, where

$$\mathrm{UCB1}(Q, N, s, u) = Q(s, u) + C\sqrt{\tfrac{\log(N(s))}{N(s,u)}} \qquad (2)$$

and C is the exploration constant (Algorithm 2 Line 10). Our UCT algorithm uses random selection to break ties.

The compressed approach offers computational advantages over $\mathrm{UCT}_{\mathrm{base}}$ by using a smaller state space, but it compromises optimality. By maintaining separate Q-functions for each subgoal, $\mathrm{UCT}_{\mathrm{comp}}$ may fail to capture interdependencies among subgoals within a goal network. The selection process—which selects actions and methods based on Q-functions for each individual subgoal—implicitly assumes that subgoals can be optimized independently, treating them as separable objectives. However, this assumption breaks down in problems where the optimal strategy for achieving one subgoal depends on the configuration of other subgoals in the goal network. In such cases, the compressed representation may lead the algorithm toward suboptimal solutions, as it cannot reason about future, constrained subgoals in the full goal network. This efficiency-optimality trade-off implies that $\mathrm{UCT}_{\mathrm{comp}}$, unlike $\mathrm{UCT}_{\mathrm{base}}$, cannot guarantee convergence to an optimal policy even with unlimited compute.

Algorithm 2: ($\textsc{Rollout}_{\mathrm{comp}}$) The Monte Carlo rollout procedure for $\mathrm{UCT}_{\mathrm{comp}}$. $\mathcal{D}$, s, gn, and c are as defined in Algorithm 1. d is the remaining rollout depth, $util$ is a strictly decreasing utility function over cost, and K_g is the goal-utility constant. Q_* and N_* are global maps with default value 0. $\textsc{Rollout}_{\mathrm{comp}}$ returns a pair of maps $\lambda : I \to \mathbb{R}^{\geq 0}$ and $\beta : I \to \{\mathsf{true}, \mathsf{false}\}$ where $\lambda(i)$ is the rollout cost for subgoal i, and $\beta(i)$ indicates whether subgoal i was achieved.

1: **procedure** $\textsc{Rollout}_{\mathrm{comp}}(\mathcal{D}, s, gn, d, c)$
2: **if** $gn = gn_{\varnothing}$ **then return** $(\varnothing, \varnothing)$
3: **for all** $i \in UC(gn)$ such that $s \models \alpha(i)$ **do**
4: $(s, gn) \leftarrow P^{i}_{rel}(s, gn)$
5: $(\lambda, \beta) \leftarrow \textsc{Rollout}_{\mathrm{comp}}(\mathcal{D}, s, gn, d, c)$
6: **return** $(\lambda \cup \{(i, 0)\}, \beta \cup \{(i, \mathsf{true})\})$
7: $U \leftarrow \{(i, m) \in UC(gn) \times M \mid$ m is applicable in s and relevant to $i\}$
8: $U \leftarrow U \cup \{a \in A \mid a \text{ is applicable in } s\}$
9: **if** $d = 0$ or $U = \varnothing$ **then return** $(I \times \{d\}, I \times \{\mathsf{false}\})$
10: $u \leftarrow \operatorname{argmax}_{u \in U} \sum_{i \in UC(gn)} \mathrm{UCB1}(Q_{\alpha(i)}, N_{\alpha(i)}, s, u)$
11: **if** $u \in A$ **then**
12: sample $(s', gn) \sim P^{u}_{app}(s, gn)$
13: $(\lambda, \beta) \leftarrow \textsc{Rollout}_{\mathrm{comp}}(\mathcal{D}, s', gn, d - 1, c + 1)$
14: $\lambda \leftarrow \{(i, \lambda(i) + 1) \mid i \in I\}$
15: **else** ▷ u is a subgoal-method pair
16: $(i, m) \leftarrow u$
17: $(s, gn) \leftarrow P^{i,m}_{dec}(s, gn)$
18: $(\lambda, \beta) \leftarrow \textsc{Rollout}_{\mathrm{comp}}(\mathcal{D}, s, gn, d, c)$
19: **for all** $i \in I$ **do**
20: $Q_{\alpha(i)}(s, u) \leftarrow \frac{N_{\alpha(i)}(s,u) Q_{\alpha(i)}(s,u) + util(\lambda(i)+c) + K_g \mathbf{1}_{[\beta(i)]}}{1 + N_{\alpha(i)}(s,u)}$
21: $N_{\alpha(i)}(s) \leftarrow N_{\alpha(i)}(s) + 1$
22: $N_{\alpha(i)}(s, u) \leftarrow N_{\alpha(i)}(s, u) + 1$
23: **return** (λ, β)

5 Experiments

We compared the performance of $\mathrm{UCT}_{\mathrm{base}}$ and $\mathrm{UCT}_{\mathrm{comp}}$ on a collection of benchmark domains adapted for probabilistic HGN planning. These domains were originally developed

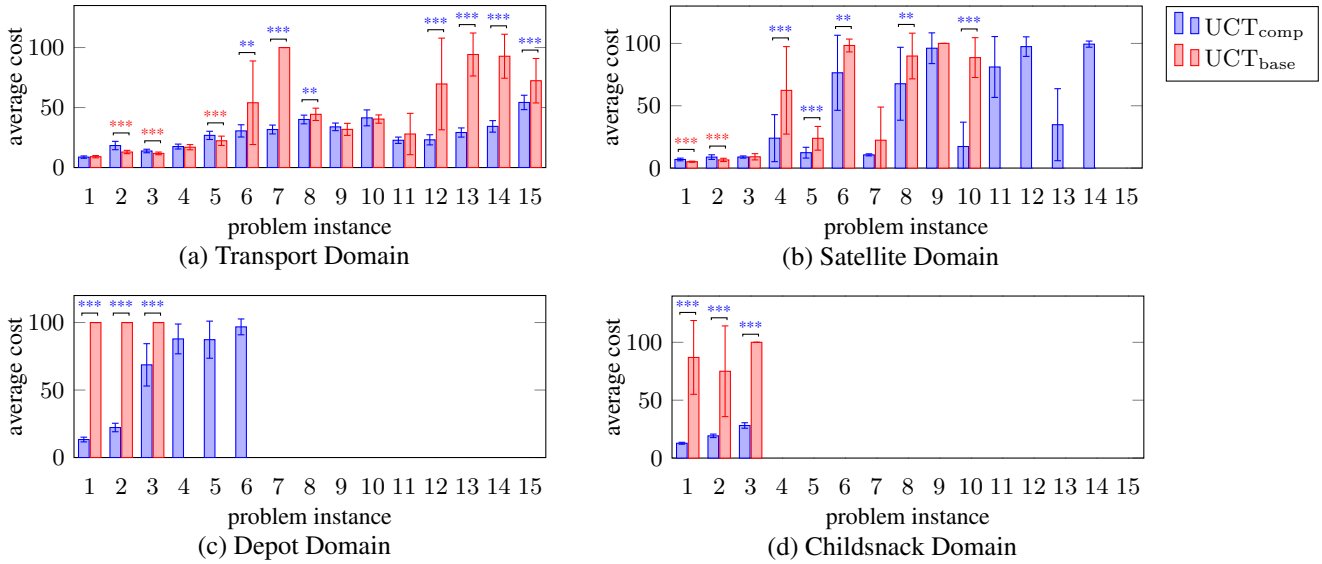

Figure 1: Average cost (lower is better) of runs of $\mathrm{UCT}_{\mathrm{comp}}$ and $\mathrm{UCT}_{\mathrm{base}}$ on problem instances of each domain. Missing bars indicate runs did not finish due to memory constraints. Error bars show ± 1 standard deviation. Statistical significance was determined using a two-tailed t-test. ** indicates $p < 0.01$ and *** indicates $p < 0.001$; stars are colored by the better variant.

for FOND HTN planning (Yousefi and Bercher 2024), so it was necessary to modify them to support the probabilistic action outcomes and hierarchical goal structures within our framework. First, nondeterministic actions—which in FOND planning specify multiple possible outcomes without associated probabilities—were converted into probabilistic actions by assigning uniform probabilities over the original outcome set. This allowed us to preserve the branching behavior of nondeterministic effects in a manner compatible with probabilistic planning. Second, we replaced each domain's HTN methods with a new set of HGN methods tailored to our formalism. While there is no direct one-to-one mapping from HTN to HGN methods, we manually constructed the HGN methods to preserve the structure and intent of the original decompositions wherever possible; when HTN methods decomposed tasks into specific sequences of primitive actions, the corresponding HGN methods were designed to produce equivalent primitive solutions.

Each domain consists of 15 problems. Every problem defines an initial state and an initial task network, which was manually translated into a corresponding initial goal network for use in our HGN planning framework. These domain adaptations preserve the essential planning challenges of the original benchmarks while enabling evaluation under the probabilistic HGN framework. Both UCT variants were tested on the same adapted domains and problems.

For each benchmark problem, we evaluated $\mathrm{UCT}_{\mathrm{base}}$ and $\mathrm{UCT}_{\mathrm{comp}}$ by running 20 independent trials per algorithm and averaging results (Figure 1). Both UCT variants commit to actions during search. Execution for both algorithms was bounded by 100 action executions per trial; runs that exceeded the 100 action budget were halted and contributed 100 to the average. During each trial, each planner performed 1000 rollouts at each decision step. Both UCT variants used a non-heuristic implementation in which all Q-values were initialized to 0, and a maximum rollout depth of 20 was enforced. The UCT exploration constant was set to $\sqrt{2}$ following Kocsis and Szepesvári (2006). Planning was evaluated under the GUBS criterion, with the goal utility constant K_g set to 1 and the utility function defined as $util(cost) = \exp\left(-\frac{1}{10} \cdot cost\right)$, consistent with the default settings used by Crispino, Freire, and Delgado (2024). All trials were independently run on compute nodes with 32 GB memory.

Our experimental results highlight key trade-offs between $\mathrm{UCT}_{\mathrm{base}}$ and $\mathrm{UCT}_{\mathrm{comp}}$ across problem sizes and domains. In smaller problems—such as Satellite problems 1–3 and Transport problems 1–5—both planning approaches consistently converged within the allotted budget of 1000 rollouts per decision step. In these cases, $\mathrm{UCT}_{\mathrm{base}}$ outperformed the compressed variant in terms of solution quality. This is expected: $\mathrm{UCT}_{\mathrm{base}}$ operates on the full SSP formulation, which guarantees convergence to an optimal solution given sufficient rollouts. For smaller problems, this optimality can often be achieved within the given computational budget.

However, for larger, more complex problems, $\mathrm{UCT}_{\mathrm{base}}$ struggled to converge within the computational limits. In particular, for problems with larger state spaces—such as Depot problems 4–6 and Satellite problems 11–14—the baseline planner often failed to return any solution due to exceeding memory constraints. In contrast, the compressed $\mathrm{UCT}_{\mathrm{comp}}$ variant was able to solve many of these larger problems within the same computational environment, demonstrating greater memory efficiency and broader scalability.

On these larger problems, $\mathrm{UCT}_{\mathrm{comp}}$ consistently outperformed $\mathrm{UCT}_{\mathrm{base}}$ in terms of average solution cost, and was the only variant that could solve the most memory-intensive instances. This behavior stems from the compressed state

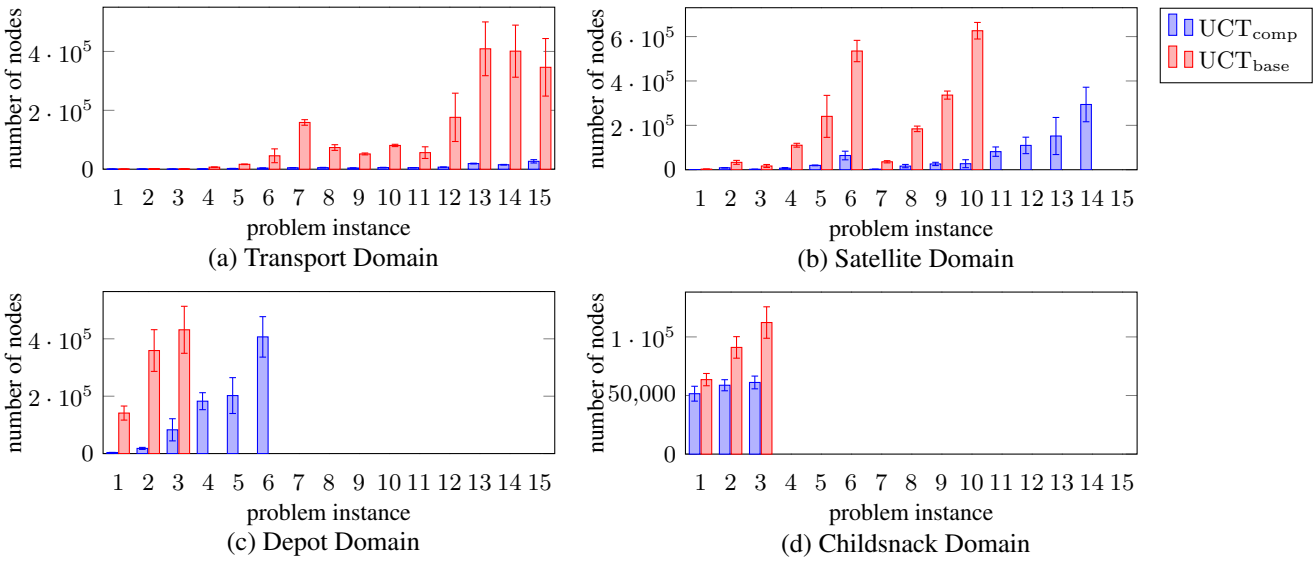

Figure 2: Average number of nodes in the search tree for $\mathrm{UCT}_{\mathrm{base}}$ and $\mathrm{UCT}_{\mathrm{comp}}$ on problem instances of each domain. Missing bars indicate runs did not finish due to memory constraints. Error bars show ± 1 standard deviation.

representation used by $\mathrm{UCT}_{\mathrm{comp}}$ and the ability to generalize across goal networks that share the same underlying world state. While this sacrifices UCT's asymptotic optimality guarantees—meaning it may converge to suboptimal policies even with unlimited computation—it has significantly improved performance in compute-constrained settings.

To further understand the efficiency gains of the compressed variant, we analyzed the size of the search trees generated by both algorithms (Figure 2). Our measurements showed that $\mathrm{UCT}_{\mathrm{base}}$ consistently produced larger search trees, with node counts significantly exceeding those of the compressed $\mathrm{UCT}_{\mathrm{comp}}$ variant. However, we note that node count does not directly translate to memory footprint, as the two algorithms employ fundamentally different node structures. In $\mathrm{UCT}_{\mathrm{base}}$, each node represents a specific state–goal network pair and stores a single value estimate for the node. In contrast, each node in the compressed $\mathrm{UCT}_{\mathrm{comp}}$ tree represents a underlying world state and maintains value estimates for multiple goals simultaneously.

Overall, the compressed $\mathrm{UCT}_{\mathrm{comp}}$ variant demonstrates a favorable balance between solution quality and scalability. It retains comparable performance to $\mathrm{UCT}_{\mathrm{base}}$ on small problems, while offering considerable advantages on larger domains where memory and computation are limiting factors.

6 Conclusion and Future Work

We formalized probabilistic HGN planning to extend classical HGN planning to probabilistic domains and we evaluated two UCT planning algorithms: $\mathrm{UCT}_{\mathrm{base}}$, a baseline which transforms the probabilistic HGN problem into an SSP problem, preserving asymptotic optimality, and $\mathrm{UCT}_{\mathrm{comp}}$, a compressed variant of UCT that reduces the size of the search tree by sharing learned Q-values across similar goal network configurations within the same world state.

Experimental results on four adapted FOND HTN planning problems indicate that, although $\mathrm{UCT}_{\mathrm{base}}$ performs well on small problems, it struggles to scale under limited rollout and memory budgets. By contrast, $\mathrm{UCT}_{\mathrm{comp}}$ consistently handles larger problems and produces lower-cost solutions within the same limited rollouts and memory. It achieves faster convergence and reduced memory usage, particularly in larger problems. These findings suggest that the compressed variant is well-suited for anytime planning, where quick, high-quality decisions are required under computational constraints. Moreover, its efficiency and low overhead make it a strong candidate for use as a heuristic or policy prior to guide more computationally intensive optimal planners.

The current implementation of $\mathrm{UCT}_{\mathrm{comp}}$ uses a greedy action selection strategy that focuses only on the immediate unconstrained subgoals in the goal network. While this design choice is more efficient, it can be myopic—leading to locally optimal behaviors that interfere with future subgoals, especially in the presence of misleading or unsafe methods. Future work could explore ways to incorporate broader goal network context into decision-making, potentially improving long-horizon performance. Additionally, although our focus in this work was not on incomplete domain models, the use of action insertion naturally supports planning with an incomplete set of HGN methods by allowing planners to interleave hierarchical decomposition with primitive action chaining. Further investigation could reveal how partial domain knowledge provided in the form of HGN methods could be effectively leveraged to guide search.

Acknowledgments

David Chan and Mark Roberts thank the U.S. Naval Research Laboratory for supporting parts of this research.

References

Alford, R.; Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; and Aha, D. W. 2016a. Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems. In *ICAPS 2016*, 20–28.

Alford, R.; Kuter, U.; and Nau, D. S. 2009. Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way. In *IJCAI 2009*, 1629–1634.

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016b. Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing. In *IJCAI 2016*, 3022–3029.

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

Behnke, G.; Höller, D.; and Biundo, S. 2018. totSAT - Totally-Ordered Hierarchical Planning Through SAT. In *AAAI 2018*, 6110–6118.

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *AAAI 2020*, 9775–9784.

Behnke, G.; Pollitt, F.; Höller, D.; Bercher, P.; and Alford, R. 2022. Making Translations to Classical Planning Competitive with Other HTN Planners. In *AAAI 2022*, 9687–9697.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *IJCAI 2017*, 480–488.

Bit-Monnot, A.; Smith, D. E.; and Do, M. 2016. Delete-Free Reachability Analysis for Temporal and Hierarchical Planning. In *ECAI 2016*, volume 285, 1698–1699.

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

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

Crispino, G. N.; Freire, V.; and Delgado, K. V. 2024. Monte Carlo Tree Search Algorithm for SSPs Under the GUBS Criterion. *CLEI Electronic Journal*, 27(3): 5:1–5:18.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. HTN Planning: Complexity and Expressivity. In *AAAI 1994*, 1123–1128.

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

Freire, V.; and Delgado, K. V. 2017. GUBS: a Utility-Based Semantic for Goal-Directed Markov Decision Processes. In *AAMAS 2017*, 741–749.

Freire, V.; Delgado, K. V.; and Reis, W. A. S. 2019. An Exact Algorithm to Make a Trade-Off between Cost and Probability in SSPs. In *ICAPS 2019*, 146–154.

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.

Höller, D. 2024. The TOAD System for Totally Ordered HTN Planning. *J. Artif. Intell. Res.*, 80: 613–663.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN Planning as Heuristic Progression Search. *J. Artif. Intell. Res.*, 67: 835–880.

Keller, T.; and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *ICAPS 2012*, 119–127.

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

Painter, M.; Lacerda, B.; and Hawes, N. 2020. Convex Hull Monte-Carlo Tree-Search. In *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 *ICAPS 2020*, 478–487.

Quenard, G.; Pellier, D.; and Fiorino, H. 2024. SibylSat: Using SAT as an Oracle to Perform a Greedy Search on TOHTN Planning. In *ECAI 2024*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, 4157–4164.

Schreiber, D. 2021. Lilotane: A Lifted SAT-based Approach to Hierarchical Planning. *J. Artif. Intell. Res.*, 70: 1117–1181.

Shivashankar, V.; Alford, R.; and Aha, D. W. 2017. Incorporating Domain-Independent Planning Heuristics in Hierarchical Planning. In *AAAI 2017*, 3658–3664.

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 *IJCAI 2013*, 2380–2386.

Shivashankar, V.; Alford, R.; Roberts, M.; and Aha, D. W. 2016. Cost-Optimal Algorithms for Planning with Procedural Control Knowledge. In *ECAI 2016*, volume 285, 1702–1703.

Shivashankar, V.; Kuter, U.; Nau, D. S.; and Alford, R. 2012. A hierarchical goal-based formalism and algorithm for single-agent planning. In *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. *Nature*, 529(7587): 484–489.

Wichlacz, J.; Höller, D.; Torralba, Á.; and Hoffmann, J. 2020. Applying Monte-Carlo Tree Search in HTN Planning. In *SoCS 2020*, 82–90.

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

Yousefi, M.; Schmalz, J.; Haslum, P.; and Bercher, P. 2025. Probabilistic HTN Planning: Formalization and Computational Complexity Analysis. In *KR 2025*, 869–879.